Fastly Help Guides Archive

Generated: Thu, 31 Mar 2022 20:00:16 +0000

Getting started

These articles provide basic instructions for getting started with Fastly services.

https://docs.fastly.com/en/guides/getting-started

§

These articles provide basic information and instructions for configuring Fastly services after getting started.

https://docs.fastly.com/en/guides/getting-started#_basics

Adding CNAME records

Last updated: 2021-09-30

https://docs.fastly.com/en/guides/adding-cname-records

This guide describes how to choose the right hostname and how to update the CNAME record for your domain with your DNS provider. Choosing the appropriate CNAME record is the final step required before Fastly can start acting as a reverse proxy and begin routing client traffic through Fastly services instead of directly to your origin.

Before you begin

Before you add a DNS CNAME record, keep in mind the following:

- To make the changes suggested here you must have access privileges to modify DNS records for your domain.
- If you plan to use Fastly on your apex domain (e.g., `example.com` rather than `www.example.com`), you can't use a CNAME record. See our guide to using Fastly with apex domains for more details.
- You can't add a CNAME record for a free TLS hostname.

Choosing the right Fastly hostname for your CNAME record

To successfully update your DNS CNAME record, you must choose the right Fastly hostname to use. The hostname you choose will differ based on:

- the standard HTTPS (TLS) support requirements for your domain, including whether or not HTTP/2 is enabled.
- any custom TLS options purchased for your domain.
- whether or not you choose to limit your traffic to the North American and EU network or use Fastly's global network.

We've provided recommendations below based on these criteria.

Non-TLS hostnames and limiting traffic

If you don't require TLS support and only need to accept HTTP (Port 80) connections, use one of the following hostnames:

- Use `nonssl.global.fastly.net.` to route traffic through Fastly's entire global network.
- Use `nonssl.us-eu.fastly.net.` to route traffic through Fastly's North American and EU POPs only.

IMPORTANT

Fastly's non-TLS hostnames refuse HTTPS connections (port 443) to prevent TLS certificate mismatch errors.

TLS-enabled hostnames

If you've purchased Fastly TLS, use one of the following HTTP/1.1 and HTTP/2 enabled hostnames:

- Use `[letter].sni.global.fastly.net` to route traffic through Fastly's entire global network.
- Use `j.sni.global.fastly.net` for TLS 1.3 and TLS 1.2 and `k.sni.global.fastly.net` for TLS 1.3 + 0RTT and TLS 1.2 support

IMPORTANT

You must use the assigned Fastly TLS hostname provided in the Fastly web interface. Using the incorrect Fastly hostname will cause a TLS Certificate mismatch error for HTTPS (Port 443) traffic.

Updating the CNAME record with your DNS provider

Once you've determined the appropriate Fastly hostname for your domain, the next step is to create a CNAME record for your domain. The steps you follow will vary depending on your DNS provider's control panel interfaces. Refer to your DNS provider's documentation for exact instructions on how to create or update a CNAME record.

TIP

If you can't find your provider's CNAME configuration instructions, Google maintains instructions for most major providers. Keep in mind that these instructions are maintained by Google, not Fastly, and are tailored specifically for Google enterprise services.

If you run your own DNS server or are familiar with the format of BIND zone files, the CNAME record would look similar to this:

```
www.example.com.    3600    IN    CNAME   nonssl.global.fastly.net.
```

In the above example, the domain set up on Fastly is `www.example.com.`, with a time-to-live (TTL) of `3600` seconds (1 hour), the Record Type is `CNAME`, and the Fastly hostname is `nonssl.global.fastly.net.` because TLS support isn't required and traffic will be routed through Fastly's entire global network.

Best practices when updating a DNS CNAME record

- Be sure you've added all domains you want served by Fastly to the appropriate service. If you don't and you point your domain to Fastly, an `unknown domain` error will occur.
- Make sure your service is properly configured. You can test a Fastly service on your local machine by using curl and testing setup before changing domains.
- If you have multiple hostnames on the same domain (e.g., `api.example.com`, `www.example.com`, `app.example.com`), you can use a DNS wildcard record (`*.example.com`) at your DNS provider so only a single CNAME record is created and maintained. You should also add either a matching `*.example.com` domain or the individual domains to your Fastly service.
- Before changing a CNAME to point to a Fastly hostname, change your service configuration to lower the CNAME's TTL to a small number (we suggest 60 seconds) and wait for the old TTL to expire. Creating a DNS CNAME record for your domain after the TTL expiration ensures you have an easy way to roll back changes if you encounter an issue. Once you confirm everything is working properly using Fastly, you can increase the TTL to its original value.

Checking your CNAME record

To check your CNAME record, run the following command in a terminal window:

```
$ dig www.example.com +short
```

Your output should appear similar to the following:

```
1  nonssl.global.fastly.net.
2  151.101.117.57
```

In most cases, the hostname displayed first will be your current Fastly hostname (in this case, `nonssl.global.fastly.net.`). If you don't see a Fastly hostname in the output or if you see an incorrect Fastly hostname, then either your CNAME isn't properly set at your DNS provider or an older CNAME record is still cached by your local DNS resolver.

You can use various online DNS query tools like OpenDNS Cache Check or whatsmydns.net to test the current DNS responses from the different DNS resolvers worldwide.

Removing CNAME records

If you deactivate a service, delete a service, or cancel your account, we strongly recommend modifying or deleting any CNAME records pointing to Fastly hostnames. Follow the instructions on your DNS provider's website. Doing so will minimize the risk of unauthorized use of your domains.

Content and its delivery

Last updated: 2018-04-24

https://docs.fastly.com/en/guides/content-and-its-delivery

Content types delivered by Fastly

The underlying protocol used by the World Wide Web to define how content is formatted and transmitted is called the Hypertext Transfer Protocol (HTTP). Fastly's CDN Service delivers all HTTP-based file content (e.g., HTML, GIF, JPEG, PNG, JavaScript, CSS) including the following:

- Static content
- Dynamic content
- Video content

Each content type is described below.

Static content

Static content includes content that remains relatively unchanged. Fastly can control static content in two ways:

- using the time to live (TTL) method, where Fastly's cache re-validates the content after expiration of the TTL, or
- using Fastly's Instant Purge functionality, in which content remains valid until the cache receives a purge request that invalidates the content.

Examples of static content include images, css, and javascript files.

Dynamic content

Dynamic content includes content that changes at unpredictable intervals, but can still be cached for a fraction of time. We serve this dynamic content by taking advantage of Fastly's Instant Purge functionality. Using this functionality, dynamic content remains valid only until a Fastly cache receives a purge request that invalidates the content. Fastly understands that the rate of those purge requests cannot be predicted. Dynamic content may change frequently as a source application issues purge requests in rapid succession to keep the content up to date. Dynamic content can, however, remain valid for months if there are no changes requested.

Examples of dynamic content include sports scores, weather forecasts, breaking news, user-generated content, and current store item inventory.

Video content

Video content includes:

- Live video streams
- Video on Demand (VOD) content libraries

Video content can be served using standard HTTP requests. Specifically, Fastly supports HTTP Streaming standards, including HTTP Live Streaming (HLS), HTTP Dynamic Streaming (HDS), HTTP Smooth Streaming (HSS), and MPEG-DASH. For Fastly's CDN Service to deliver video, the video must be packaged.

Content sources supported by Fastly

Fastly caches deliver various types of content from many different sources. Supported sources include:

- Websites
- Internet APIs
- Internet Applications
- Live and Live Linear Video
- Video on Demand (VOD) Libraries

Regardless of the content source, the content's source server must communicate using HTTP. HTTP defines specific types of *methods* that indicate the desired action to be performed on content. The manner in which those HTTP methods are used (the standard, primary methods being GET, POST, PUT, and DELETE) can be labeled as being RESTful or not. Fastly supports RESTful HTTP by default, but also can support the use of non-RESTful HTTP as long as the method used is mapped to its appropriate cache function. Each of the content sources supported by Fastly are described in more detail below.

Websites

Websites are servers that provide content to browser applications (e.g., Google's Chrome, Apple's Safari, Microsoft's Internet Explorer, Opera Software's Opera) when end users request that content. The content contains both the requested data and the formatting or display information the browser needs to present the data visually to the end user.

With no CDN services involved, browsers request data by sending HTTP GET requests that identify the data with a uniform resource locator (URL) address to the origin server that has access to the requested data. The server retrieves the data, then constructs and sends an HTTP response to the requestor. When a CDN Service is used, however, the HTTP requests go to the CDN rather than the origin server because the customer configures it to redirect all requests for data to the CDN instead. Customers do this by adding a CNAME or alias for their origin server that points to Fastly instead.

Internet APIs

Application program interfaces (APIs) serve as a language and message format that defines exactly how a program will interact with the rest of the world. APIs reside on HTTP servers. Unlike the responses from a website, content from APIs contain only requested data and identification information for that data; no formatting or display information is included. Typically the content serves as input to another computing process. If it must be displayed visually to an end user, a device application (such as, an iPad, Android device, or iPhone Weather application) does data display instead.

Legacy internet applications

Legacy internet applications refer to applications not originally developed for access over the internet. These applications may use HTTP in a non-RESTful manner. They can be incrementally accelerated without caching, benefiting only from the TCP Stack optimization done between edge Fastly POPs and the Shield POP, and the Shield POP to the origin. Then caching can be enabled incrementally, starting with the exchanges with the greatest user-experienced delay.

Live and live linear video streams & video on demand libraries

Live and live linear video content (for example, broadcast television) is generally delivered as a *stream* of information to users, which they either choose to watch or not during a specific broadcast time. Video on demand (VOD), on the other hand, allows end users to select and watch video content when they choose to, rather than having to watch at a specific broadcast time.

Regardless of which type of video content an end user experiences, a video player can begin playing before its entire contents have been completely transmitted. End users access the video content from a customer's servers via HTTP requests from a video player application that can be embedded as a part of a web browser. Unlike other types of website content, this content does not contain formatting or display information. The video player handles the formatting and display instead.

When the video content is requested, the customer's server sends the content as a series of pre-packaged file chunks along with a manifest file required by the player to properly present the video to the end user. The manifest lists the names of each file chunk. The video player application needs to receive the manifest file first in order to know the names of the video content chunks to

request.

Pre-packaging in this context refers to the process of receiving the video contents, converting or *transcoding* the stream into segments (chunks) for presentation at a specific dimension and transmission rate, and then packaging it so a video player can identify and request the segments of the live video a user wants to view.

To request video delivery on your account, contact your Fastly Account Representative at sales@fastly.com.

How caching and CDNs work

Last updated: 2021-09-21

https://docs.fastly.com/en/guides/how-caching-and-cdns-work

Fastly is a content delivery network, or CDN. CDNs work on the principle that once a piece of content has been generated it doesn't need to be generated again for a while so a copy can be kept around in a cache. Cache machines are optimized to serve small resources very quickly. CDNs typically have caches placed in data centers all around the world. When a user requests information from a customer's site they're actually redirected to the set of cache machines closest to them instead of the customer's actual servers. This means that a European user going to an American site gets their content anywhere from 200-500ms faster. CDNs also minimize the effects of a cache miss. A cache miss occurs when a user requests a bit of content and it is not in the cache at that moment (because it's expired, because no-one has asked for it before, or because the cache got too full and old content was thrown out).

What can be cached?

CDNs are good at managing a cache of small, static resources (for example, static images, CSS files, Javascripts, and animated GIFs). CDNs are also popular for offloading expensive-to-serve files like video and audio media.

At Fastly, our architecture (known as a *reverse proxy*) is designed to enable customers to go a step further and cache entire web pages for even more efficient handling of your traffic.

> **TIP**
>
> Static files + media objects + web pages = your whole site. With the right service configuration (which we can assist you in setting up) Fastly can reduce your backend traffic by orders of magnitude with no loss in control over the content your users see.

Managing the cache

Caching serves as a powerful weapon in your make-the-site-faster arsenal. However, most objects in your cache aren't going to stay there permanently. They'll need to expire so that fresh content can be served. How long that content should stay in the cache might be mere seconds or a number of minutes or even a year or more.

How can you manage which of your content is cached, where, and for how long? By setting policies that control the cached data. Most caching policies are implemented as a set of HTTP headers sent with your content by the web server (as specified in the configuration or the application). These headers were designed with the client (browser) in mind but CDNs like Fastly will also use those headers as a guide on caching policy.

Expires

The `Expires` header is the original cache-related HTTP header and tells the cache (typically a browser cache) how long to hang onto a piece of content. Thereafter, the browser will re-request the content from its source. The downside is that it's a static date and if you don't update it later, the date will pass and the browser will start requesting that resource from the source every time it sees it.

Fastly will respect the `Expires` header value only if the `Surrogate-Control` or `Cache-Control` headers are not found in the request.

Cache-Control

The `Cache-Control` headers (introduced in the HTTP/1.1 specification) cover browser caches and, in most cases intermediate caches as well, as defined by section 5.2 of RFC 7234:

- `Cache-Control: public` - Any cache can store a copy of the content.
- `Cache-Control: private` - Don't store, this is for a single user.
- `Cache-Control: no-cache` - Re-validate before serving this content.
- `Cache-Control: no-store` - Don't ever store this content.
- `Cache-Control: public, max-age=[seconds]` - Caches can store this content for *n* seconds.
- `Cache-Control: s-maxage=[seconds]` - Same as max-age but applies specifically to proxy caches.

Only the `max-age`, `s-maxage`, and `private` Cache-Control headers will influence Fastly's caching. All other Cache-Control headers will not, but will be passed through to the browser. For more in-depth information about how Fastly responds to these Cache-Control headers and how these headers interact with Expires and Surrogate-Control, check out our documentation on cache freshness.

> **NOTE**
>
> For more information on the rest of the Cache-Control headers, see the relevant section in Mark Nottingham's caching tutorial.

Surrogate Headers

`Surrogate` headers are a relatively new addition to the cache management vocabulary (described in this W3C tech note). These headers provide a specific cache policy for proxy caches in the processing path. `Surrogate-Control` accepts many of the same values as `Cache-Control`, plus some other more esoteric ones (read the tech note for all the options).

One use of this technique is to provide conservative cache interactions to the browser (for example, `Cache-Control: no-cache`). This causes the browser to re-validate with the source on every request for the content. This makes sure that the user is getting the freshest possible content. Simultaneously, a `Surrogate-Control` header can be sent with a longer `max-age` that lets a proxy cache in front of the source handle most of the browser traffic, only passing requests to the source when the proxy's cache expires.

With Fastly, one of the most useful `Surrogate` headers is `Surrogate-Key`. When Fastly processes a request and sees a `Surrogate-Key` header, it uses the space-separated value as a list of tags to associate with the request URL in the cache. Combined with Fastly's Purge API an entire collection of URLs can be expired from the cache in one API call (and typically happens in around 1ms). `Surrogate-Control` is the most specific.

Fastly and Cache Control Headers

Fastly looks for caching information in each of these headers as described in our documentation on cache freshness. In order of preference:

- `Surrogate-Control:`
- `Cache-Control: s-maxage`
- `Cache-Control: max-age`
- `Expires:`

HTTP status codes cached by default

Fastly caches the following response status codes by default. In addition to these statuses, you can force an object to cache under other states using conditions and responses.

Code	Message
`200`	OK
`203`	Non-Authoritative Information
`300`	Multiple Choices
`301`	Moved Permanently

Code	Message
`302`	Moved Temporarily
`404`	Not Found
`410`	Gone

> **TIP**
>
> You can override caching defaults based on a backend response. For example, if you don't want 404 error responses to be cached for the full caching period of a day, you could add a cache object and then create conditions for it.

To cache status codes other than the ones listed above, set `beresp.cacheable = true;` in `vcl_fetch`. This tells Varnish to obey backend HTTP caching headers and any other custom `ttl` logic. A common pattern is to allow all 2XX responses to be cacheable:

```
sub vcl_fetch {
  # ...
  if (beresp.status >= 200 && beresp.status < 300) {
    set beresp.cacheable = true;
  }
  # ...
}
```

Shielding

When an object or collection of objects in the cache expires, the next time any of those objects are requested, the request is going to get passed through to your application. Generally, with a good caching strategy, this won't break things. However, when a popular object or collection of objects expires from the cache, your backend can be hit with a large influx of traffic as the cache nodes refetch the objects from the source.

In most cases, the object being fetched is not going to differ between requests, so why should every cache node have to get its own copy from the backend? With shield nodes, they don't have to. Shielding configured through the Fastly web interface allows you to select a specific data center (most efficiently, one geographically close to your application) to act as a shield node. When objects in the cache expire, the shield node is the only node to get the content from your source application. All other cache nodes will fetch from the shield node, reducing source traffic dramatically.

Resources

- Wikipedia: Reverse Proxy
- Fastly's cache freshness documentation
- Mark Nottingham's caching tutorial
- Surrogate header W3C tech note

How Fastly's CDN Service works

Last updated: 2021-09-21

https://docs.fastly.com/en/guides/how-fastlys-cdn-service-works

Fastly is a content delivery network (CDN). We serve as an internet intermediary and offer the Fastly CDN Service to make transmission of your content to your end users more efficient.

You can make content available through your websites and internet-accessible (hosted) application programming interfaces (APIs). You can create content (customer-generated content), as can your end users (user-generated content). Fastly's CDN Service then makes the transmission of that content (which we sometimes refer to as *content objects*) more efficient by automatically storing copies at intermediate locations on a temporary basis. The process of storing these copies is known as *caching* and the server locations in which they are stored are referred to as *caches*.

Fastly's delivers its CDN service from key access points to the internet called *points of presence* (POPs). Fastly places POPs where their connectivity to the internet reduces network transit time when delivering content to end-users. Each POP has a cluster of Fastly cache servers. When end users request your content objects, Fastly delivers them from whichever of the cache locations are closest to each end user.

Fastly's caches only receive and process your end user requests for content objects. You decide which objects will be cached, for how long, who can access them, whether they are to be encrypted when transmitted over the internet, and when the objects will be deleted from the caching service. You make these decisions by specifically configuring Fastly's CDN Service with these requirements. We refer to this configuration process as *provisioning.*

To provision Fastly's CDN service, you must identify which of your application servers will provide the original content objects for each of your various domains (e.g., company.com, myco.com). Your application servers can be physical servers in a data center or hosting facility, or applications running on cloud services like Amazon, or any combination. Fastly refers to these source servers as *origin* and *backend* servers interchangeably.

The first time each Fastly cache receives a request for a content object, it fetches the object from the appropriate origin server. If multiple origin servers are specified, the cache will distribute the processing load for the fetches across all of them (based on the configuration criteria set by you). After the content object is fetched, the cache stores a copy of it and forwards its response to the end user.

Each time after the first time an end user requests that same content object, the Fastly cache fulfills requests by retrieving the cached copy from storage (or memory) and immediately delivering it to the end user – the fetch step to the original copy is not repeated until the content object either expires or becomes invalidated.

Can Fastly host my content?

We accelerate your site by caching both static assets and dynamic content by acting as a reverse proxy to your origin server (also known as *Origin Pull*), but we do not provide services for uploading your content to our servers.

In addition to using your own servers as the source, we also support various *cloud storage* services as your origin, such as Amazon Simple Storage Service (S3), Google Cloud Storage (GCS), and Google Compute Engine (GCE) as your file origin. Our partnership with Google in particular enables us to have direct connectivity to their cloud infrastructure.

Fastly POP locations

Our points of presence (POPs) on the internet are strategically placed at the center of the highest density Internet Exchange Points around the world. Fastly's Network Map shows a detailed view of current and planned locations for all Fastly POPs. In addition, our data centers API endpoint provides a list of all Fastly POPs, including their latitude and longitude locations.

Once you're signed up for Fastly service (either through a test account or a paid plan) you can see a live, real-time visual representation of the general regions of the world in which Fastly's POPs receive requests for your service.

How Fastly chooses POP locations

Geographic distribution is just one of the factors Fastly considers when building its global infrastructure. Other factors include connectivity, provider diversity, and our ability to build a scalable, performant modern network centered around internet infrastructure hubs to best support our customers' markets. Fastly's focus on automation, operational redundancy, and global delivery when building our infrastructure means our POPs often combine multiple physical sites to better serve densely populated markets.

Will Fastly ever adjust POP locations or service regions? How will I be notified?

Fastly continues to grow its network footprint, adding and combining new service POPs in the process. At times, expansion may result in the addition of new billable regions to our network. We'll announce new POP locations and new billable regions in advance through our network status page at status.fastly.com. Contact sales@fastly.com with specific contract or billing questions.

Self-provisioned Fastly services

You can configure or *provision* Fastly caching and video services personally, independent of Fastly staff, via the Fastly web interface. Fastly calls this *self-provisioning*. Self-provisioning tasks include things like:

- creating and activating services
- adding domains and origin servers

- configuring load balancing
- modifying how services handle HTTP headers
- submitting purge commands

Once provisioned, Fastly services can be activated immediately. If self-provisioned tasks fail to operate in an appropriate or expected manner, you can find answers to a variety of frequently asked questions in Fastly's guides and tutorials. You can also receive personalized assistance by submitting requests directly to Fastly's Customer Support staff.

Always-on DDoS mitigation

Fastly's globally distributed network was built to absorb DDoS attacks. As part of Fastly's standard CDN services, all customers receive:

- **Access to origin shielding.** Fastly allows you to designate a specific point of presence (POP) to host cached content from your origin servers. This POP acts as a *shield* that protects those servers from every cache miss or pass through the Fastly network, reducing the load that directly reaches them.
- **Automatic resistance to availability attacks.** Before they're even processed by our caching infrastructure, we filter out Layer 3 and 4 attacks (e.g., Ping floods, ICMP floods, UDP abuse) as well as distributed reflection and amplification (DRDoS) attacks that rely on anonymity to abuse internet protocols (e.g., DNS and NTP).
- **Access to Fastly cache IP space.** Fastly provides an API endpoint to any customer who would like to know which IP addresses our caches will use to send traffic from our CDN to your origin servers. We make this data available so you can update firewalls at your origin to ensure only our cache traffic can access your resources.
- **Custom DDoS filter creation abilities.** Using custom VCL, we allow you to craft your own DDoS protection rules to protect your network from complex Layer 7 attacks. Once you identify signs of a potential DDoS attack, you can mix and match Fastly VCL with custom VCL to construct filter configurations based on a variety of client and request criteria (e.g., headers, cookies, request path, client IP, geographic location) that block malicious requests before they hit your origin servers.

In addition to these standard DDoS protection services, Fastly offers a DDoS Protection and Mitigation Service. For more information about this or any of our advanced services, including their subscription costs, contact sales-ddos@fastly.com.

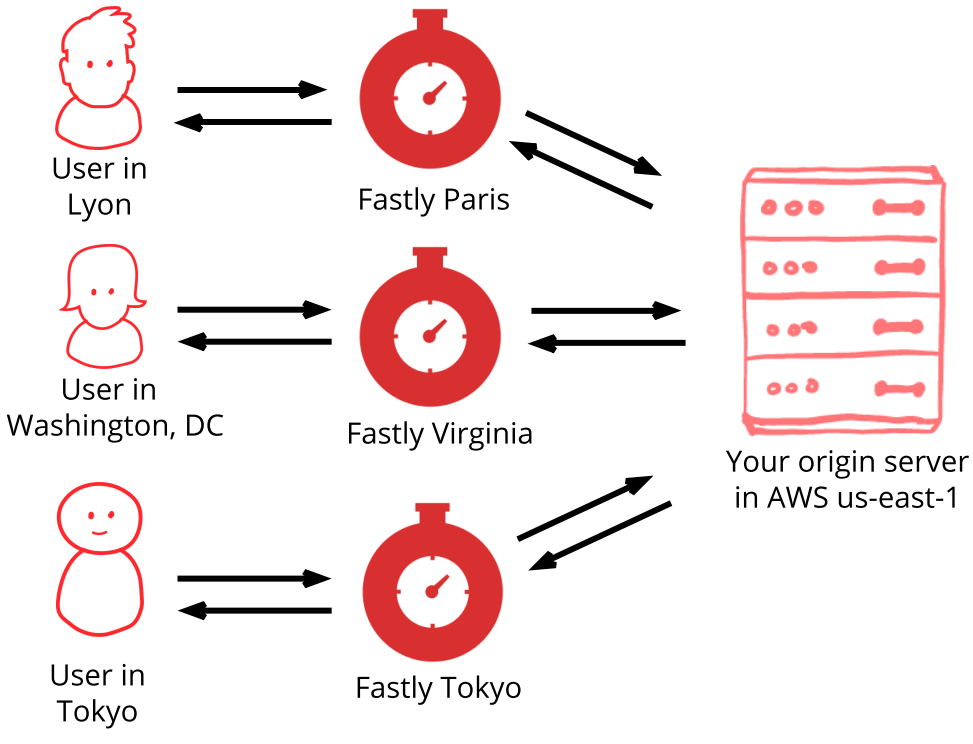
Shielding

Last updated: 2021-04-23

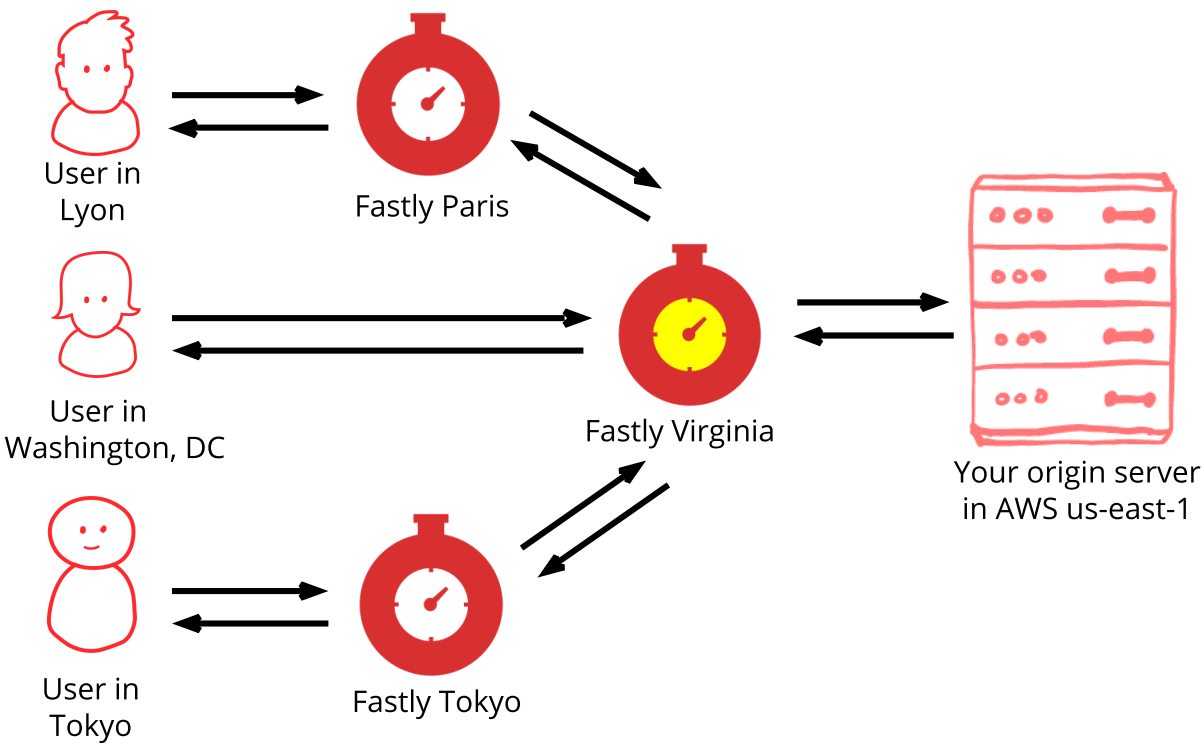
https://docs.fastly.com/en/guides/shielding

As a content delivery network, Fastly works by having any one of our global points of presence (POP) respond to requests that would otherwise be sent directly to your origin server. Because each POP acts independently, any time it doesn't have a cached version of the requested resource, it will make a request directly to your origin, even if another POP may be in the process of making the same request. You can reduce the load on your origin servers if you specify one of Fastly's POPs as an origin "shield." Designating a shield POP ensures requests to your origin will come from a single POP, thereby increasing the chances of an end user request resulting in a cache `HIT`.

For example, consider end users in three regions: Lyon (France), Washington DC (United States), and Tokyo (Japan). Each of these regions has a local Fastly POP available to them that caches information by default. If the user in Lyon requests a resource that is not cached on the Fastly Paris POP, however, that POP will make a request to the origin server, cache the resources in the response, and return the cached resource to the user.



With shielding enabled, however, the POP you designate collects all requests to your origin server instead. For example, the Fastly Virginia POP was designated as the shield for a server located in the AWS `us-east-1` region (as show in the illustration below).



In this scenario, if the user in Lyon requests a resource that isn't cached on the Fastly Paris POP, then that POP will forward the request to the shield POP, in this case in Virginia. If the resource had been previously cached on the shield POP, then it would be returned to the regional POP where it is then cached and returned to the user. Otherwise, the shield POP would make a request to the origin server for the resource, cache the response and return it to the regional POP where it is also cached before being returned to the user.

For more advanced use cases, see Advanced shielding scenarios.

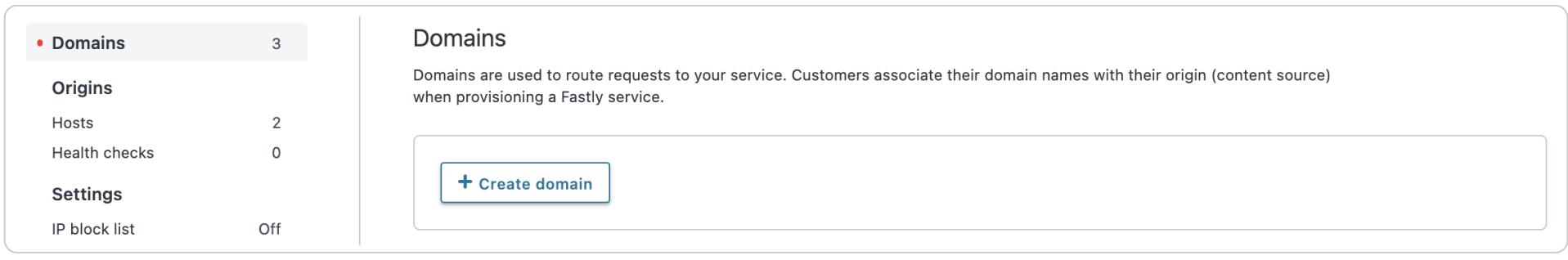
Enabling shielding

> **IMPORTANT**
>
> If you are using Google Cloud Storage as your origin, you need to follow the steps in our GCS setup guide instead of the steps below.

Enable shielding with these steps:

1. Read the caveats of shielding information below for details about the implications of and potential pitfalls involved with enabling shielding for your organization.
2. Log in to the Fastly web interface.
3. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
4. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.

5. Click the **Origins** link. The Origins page appears.
6. Click the name of the Host you want to edit. The Edit this Host page appears.
7. From the **Shielding** menu, select the data center to use as your shield keeping the following in mind:
   - Generally, we recommend selecting a data center close to your backend. Doing this allows faster content delivery because we optimize requests between the shield POP you're selecting (the one close to your server) and the edge POP (the one close to the user making the request). Read our guidance on choosing a shield location for more information.
   - With multiple backends, each backend will have its own shield defined. This allows flexibility if your company has backends selected geographically and different shield POPs are desired.
8. Click **Update** to save your changes.
9. If you have changed the default Host or have added a header to change the Host, add the modified hostname to your list of domains. Do this by clicking the **Domains** link and checking to make sure the Host in question appears on the page. If it isn't included, add it by clicking the **Create domain** button.



   With shielding enabled, queries from other POPs appear as incoming requests to the shield. If the shield doesn't know about the modified hostname, it doesn't know which service to match the request to. Including the origin's hostname in the domain list eliminates this concern.
10. Click the **Activate** button to deploy your configuration changes.

Caveats of shielding

Shielding not only impacts traffic and hit ratios, it affects configuration and performance. When you configure shielding, be aware of the following caveats.

Inbound traffic billing

Inbound traffic to a shield will be billed as regular traffic, including requests to populate remote POPs. Enabling shielding will incur some additional Fastly bandwidth charges, but will be offset by savings of your origin bandwidth (and origin server load). Pass-through requests will not go directly to the origin, they will go through the shield first.

Global HIT ratio calculation

Global HIT ratio calculation may seem lower than the actual numbers. Shielding is not taken into account when calculating the global hit ratio. If an edge node doesn't have an object in its cache, it reports a miss. Local MISS/Shield HIT gets reported as a miss and a hit in the statistics, even though there is no call to the backend. It will also result in one request from the edge node to the shield. Local MISS/Shield MISS will result in two requests, because we will subsequently fetch the resource from your origin. For more information about caching with shielding, see our shielding developer documentation.

Shield failover

If a specified shield POP is inaccessible for a request (e.g., because of intervening network issues), that request will go directly from the edge node to your origin server, bypassing the shield.

Backends manually defined using VCL

You will be unable to manually define backends using VCL. Shielding at this level is completely dependent on backends being defined as actual objects through the web interface or API. Other custom VCL will work just fine.

Automatic load balancing

If you've selected auto load balancing, you can only select one shield total. You must use custom VCL to use multiple shields when auto load balancing is set.

Sticky load balancing

Enabling sticky load balancing and shielding at the same time requires custom VCL. Sticky load balancers use `client.identity` to choose where to send the session. The `client.identity` defaults to the IP request header. That's fine under normal circumstances, but if you enable shielding, the IP will be the original POP's IP, not the client's IP. Thus, to enable shielding and a custom sticky load balancer, you want to use the following:

```
if (req.http.fastly-ff) {
  set client.identity = req.http.Fastly-Client-IP;
}
```

Host header

You'll need to use caution when changing the Host header before it reaches the shield. Fastly matches a request with a Host header. If the Host header doesn't match to a domain within the service, an error of 500 is expected. To ensure consistent behavior of Fastly customer services and origins, we normalize the host header's value to all lowercase in the `vcl_hash` subroutine. This means that no matter how your site's domain name is capitalized in the request, the hash subroutine will behave predictably. This does not apply to any other parts of the URL, which remain case-sensitive.

Purging conflicts can occur if the Host header is changed to a domain that exists in a different service. For example, say Service A has hostname `a.example.com` and Service B has hostname `b.example.com`. If Service B changes the Host header to `a.example.com`, then the edge will think the request is for Service B but the shield will think the request is for Service A. As a precaution, you'll want to purge the object from both Service A and Service B to ensure that Fastly is retrieving the newest version on the next request. If you're changing the Host header before it reaches the shield, the object is split across both services because `a.example.com` and `b.example.com` are treated as separate objects. For information about the caveats to be aware of when you change a Host header, see our article on Specifying an override host.

VCL execution

VCL gets executed twice: once on the edge POP and again on the shield POP. Changes to `beresp` and `resp` can affect the caching of a URL on the shield and edge. Consider the following examples.

Say you want Fastly to cache an object for one hour (3600 seconds) and then ten seconds on the browser. The origin sends `Cache-Control: max-age=3600`. You unset `beresp.http.Cache-Control` and then reset `Cache-Control` to `max-age=10`. With shielding enabled, however, the result will not be what you expect. The object will have `max-age=3600` on the shield and reach the edge with `max-age=10`.

A better option in this instance would be to use `Surrogate-Control` and `Cache-Control` response headers. `Surrogate-Control` overrides `Cache-Control` and is stripped after the edge node. The `max-age` from `Cache-Control` will then communicate with the browser. The origin response headers would look like this:

```
Surrogate-Control: max-age=3600
Cache-Control: max-age=10
```

Another common pitfall involves sending the wrong `Vary` header to an edge POP. For example, there's VCL that takes a specific value from a cookie, puts it in a header, and that header is then added to the `Vary` header. To maximize compatibility with any caches outside of your control (such as with shared proxies as commonly seen in large enterprises), the `Vary` header is updated in `vcl_deliver`, replacing the custom header with `Cookie`. The code might look like this:

```
vcl_recv {
  # Set the custom header
  if (req.http.Cookie ~ "ABtesting=B") {
    set req.http.X-ABtesting = "B";
  } else {
    set req.http.X-ABtesting = "A";
  }
  ...
}

...

sub vcl_fetch {
  # Vary on the custom header
  if (beresp.http.Vary) {
    set beresp.http.Vary = beresp.http.Vary ", X-ABtesting";
  } else {
    set beresp.http.Vary = "X-ABtesting";
  }
  ...
}

...

sub vcl_deliver {
  # Hide the existence of the header from downstream
  if (resp.http.Vary) {
    set resp.http.Vary = regsub(resp.http.Vary, "X-ABtesting", "Cookie");
  }
}
```

When combined with shielding, however, the effect of the above code will be that edge POPs will have `Cookie` in the `Vary` header, and thus will have a terrible hit rate. To work around this, amend the above VCL so that `Vary` is only updated with `Cookie` when the request is not coming from another Fastly cache. The `Fastly-FF` header is a good way to tell. The code would look something like this (including the same `vcl_recv` from the above example):

```
# Same vcl_recv from above code example

sub vcl_fetch {
  # Vary on the custom header, don't add if shield POP already added
  if (beresp.http.Vary !~ "X-ABtesting") {
    if (beresp.http.Vary) {
      set beresp.http.Vary = beresp.http.Vary ", X-ABtesting";
    } else {
      set beresp.http.Vary = "X-ABtesting";
    }
  }
  ...
}

...

sub vcl_deliver {
  # Hide the existence of the header from downstream
  if (resp.http.Vary && !req.http.Fastly-FF) {
    set resp.http.Vary = regsub(resp.http.Vary, "X-ABtesting", "Cookie");
  }
}
```

POP maintenance

As part of our standard maintenance procedures, Fastly may perform maintenance on a POP that you have designated as an origin shielding location. When this happens, the POP will remain in service, but we may bring individual machines within that POP in or out of service as needed. Fastly may also decommission a POP. When this happens, Fastly will move your shielding location to a neighboring data center on your behalf. Before standard maintenance starts, we will post a notification update to status.fastly.com, but if you prefer to be individually alerted before the maintenance happens, you can open a support ticket by emailing support@fastly.com, letting us know your customer ID, and requesting to be notified of upcoming shielding migrations at locations where you shield. We will then contact you via email to announce upcoming migrations. If necessary, we'll ask you to perform certain actions at a time that's convenient for you.

Working with domains

Last updated: 2021-10-05

https://docs.fastly.com/en/guides/working-with-domains

Domains are used to route requests to your service. You associate your domain names with your origin when provisioning a Fastly service so you can properly route requests to your website, and ensure that others cannot serve requests to that domain. For example, you could enter `www.example.com`, `blog.example.com`, or even use wildcards such as `*.example.com`. You can add, edit, or remove domains from your service at any time.

TIP

Due to limitations in the DNS specification, Fastly doesn't recommend using apex or second-level domains. An example of an apex domain is `example.com` rather than `www.example.com`. Our guide on generating redirects at the edge discusses how you can redirect all traffic for apex domains to WWW subdomains.

Before you begin

Be sure you learn about the web interface controls and how to work with services before you start working with your domains.

Creating a domain for the first time

Follow the steps below to add a domain to your service for the first time:

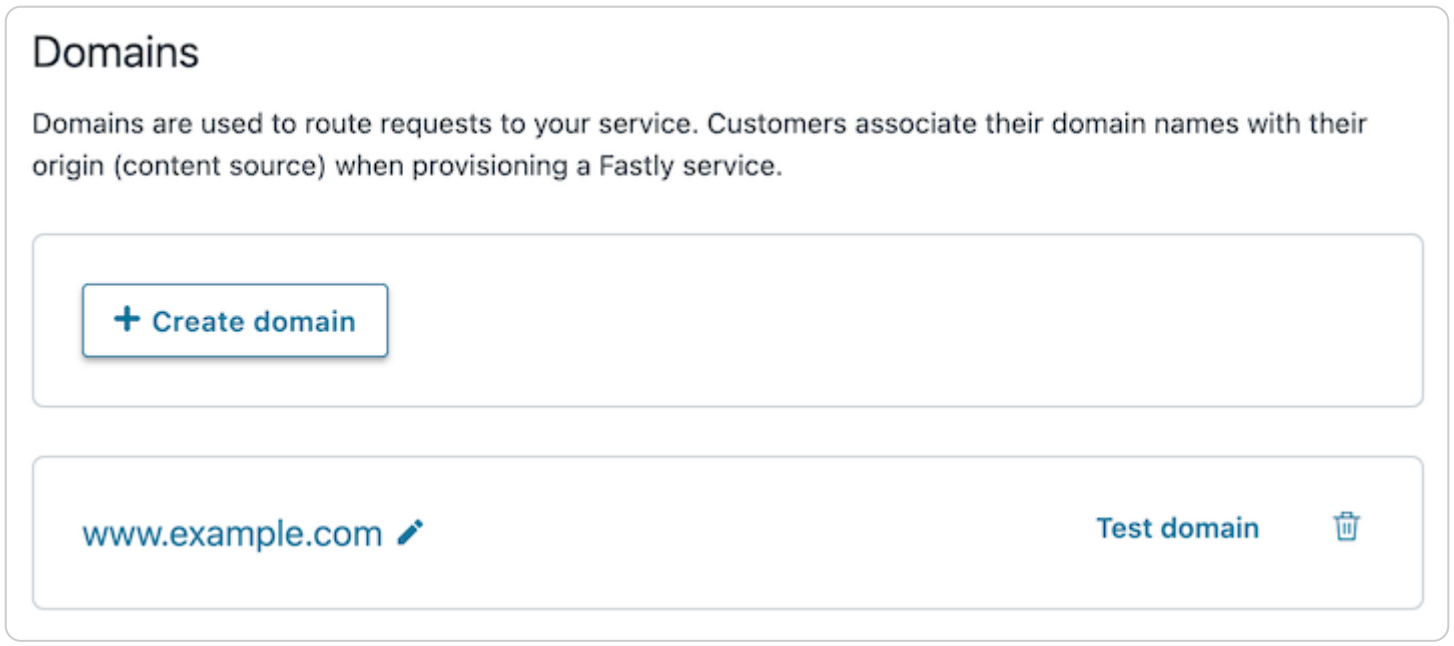
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.

4. Fill out the domain creation fields as follows:
   - In the **Domain Name** field, enter your domain name.
   - In the **Comment** field, optionally enter a comment that describes the domain.
5. Click the **Add** button. Your new domain appears in the list of domains.
6. Click the **Activate** button to deploy your configuration changes.
7. If you haven't already, add a CNAME DNS record for your domain name to begin routing client traffic through Fastly services instead of directly to your origin.

Creating additional domains

Follow the steps below to add additional domains to your service:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.



4. Click the **Create domain** button. The domain creation fields appear.

5. Fill out the domain creation fields as follows:
   - In the **Domain Name** field, enter your domain name.
   - In the **Comment** field, optionally enter a comment that describes the domain.
6. Click the **Add** button. Your new domain appears in the list of domains.
7. Click the **Activate** button to deploy your configuration changes.
8. If you haven't already, add CNAME DNS records for your domain name to begin routing client traffic through Fastly services instead of directly to your origin.

Creating a domain using the API

You can use Fastly's API to programmatically add domains to your service. To add a domain to your service, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/version/<version_id>/domain -d 'name=www.example.com'
```

The response will look like this:

```
{
  "comment": "",
  "name": "www.example.com",
  "service_id": "<service_id>",
  "version": <version_id>
}
```

Domain creation limits

We set a limit on the number of domains you can create per service by default. However, if you email support@fastly.com, we may be able to adjust this number for you by working with you to set up and fine-tune domain handling in your service.

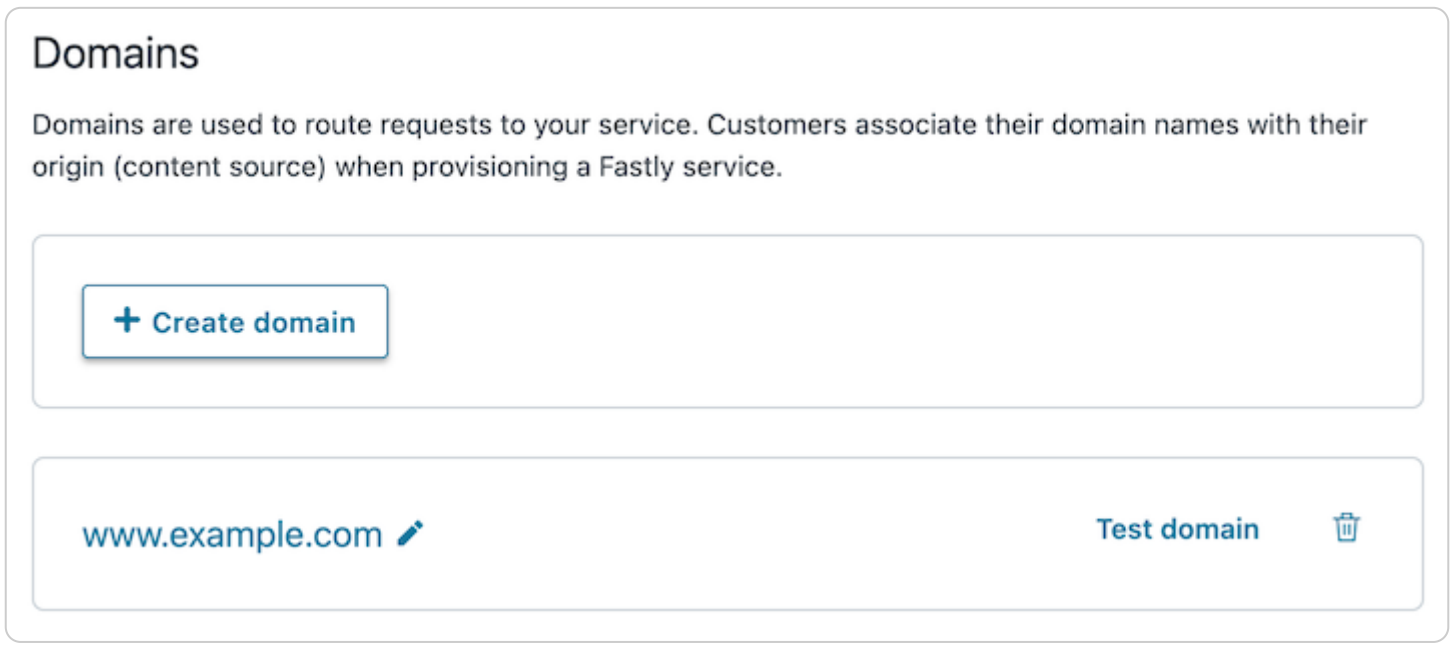
Testing a domain

After you activate your service configuration, but before you change your DNS entries to send your domain to our servers, you can check to see how your service is pulled through our network. Testing your domain can help you identify DNS issues or problems with your Fastly configuration.

Using the web interface

To use the web interface to test your domain on Fastly before you make a final CNAME change, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.



4. Click the **Test domain** link next to the domain you want to test.
5. Verify that your website appears in a new tab in your web browser.

Using command line utilities

To use command line utilities to test your domain on Fastly before you make a final CNAME change, you would:

- find the IP address of a Fastly pop
- add a domain Host entry to your hosts file
- test the domain in a web browser

Determining the IP address of a Fastly POP

Use the nslookup or dig command to determine the IP address of a Fastly POP.

> **TIP**
>
> For non-TLS requests, use `nonssl.global.fastly.net`. For TLS requests, use the custom TLS CNAME record provided by Fastly support. For more information about the Fastly TLS service, see our guide on TLS service options.

For example, running nslookup for `nonssl.global.fastly.net` returns:

```
$ nslookup nonssl.global.fastly.net
Server:         185.121.177.177
Address:        185.121.177.177#53

Non-authoritative answer:
Name:   nonssl.global.fastly.net
Address: 151.101.56.204
```

Find the IP address at the bottom of the nslookup response. In this example, it's `151.101.56.204`.

Alternatively, running dig for `nonssl.global.fastly.net` returns:

```
$ dig nonssl.global.fastly.net

; <<>> DiG 9.8.3-P1 <<>> nonssl.global.fastly.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35146
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;nonssl.global.fastly.net.      IN      A

;; ANSWER SECTION:
nonssl.global.fastly.net. 30    IN      A       151.101.56.204
```

The IP address (A record) is in the `ANSWER SECTION` of the dig results: `151.101.56.204`.

Modifying your hosts file

You can temporarily add a static IP address and domain Host entry to the hosts file on your computer. For example, if the domain you are testing is `www.example.com` and one of the IP addresses returned by nslookup or a dig command is `151.101.56.204`, you would add this entry to the file:

```
151.101.56.204 www.example.com
```

and save the changes.

> **TIP**
>
> On machines running macOS or Linux, your hosts file is `/etc/hosts`. On Windows-based machines, it's `C:\Windows\System32\Drivers\etc\hosts`.

Testing your domain

Test your domain to see how Fastly pulls it through our network by restarting your browser if it's already running, and then typing your domain in the address field. You should now see the updated domain in the address field indicating requests are being sent to the Fastly POP.

Alternatively, you can test the domain using a ping command to verify that your domain is being served by a Fastly POP address. In this case, `ping www.example.com` would display the Fastly POP address `151.101.56.204`.

Be sure to remove the Host entry from your hosts file after you make CNAME changes to point your domain to Fastly.

Deleting a domain

Follow the steps below to delete a domain from your service:

1. On the Domains page, click the trash icon next to the domain you want to delete.
2. Click the **Confirm and delete** button to confirm you want to delete your domain.
3. Click the **Activate** button to deploy your configuration changes.

> **IMPORTANT**
>
> To minimize the risk of unauthorized use of your domains, we strongly recommend modifying or deleting any DNS CNAME records pointing to the Fastly hostname associated with the deleted domain. Follow the instructions on your DNS provider's website.

What's next

Learn more about working with hosts and working with health checks as you continue to refine versions of your service configurations.

Working with health checks

Last updated: 2021-10-05

https://docs.fastly.com/en/guides/working-with-health-checks

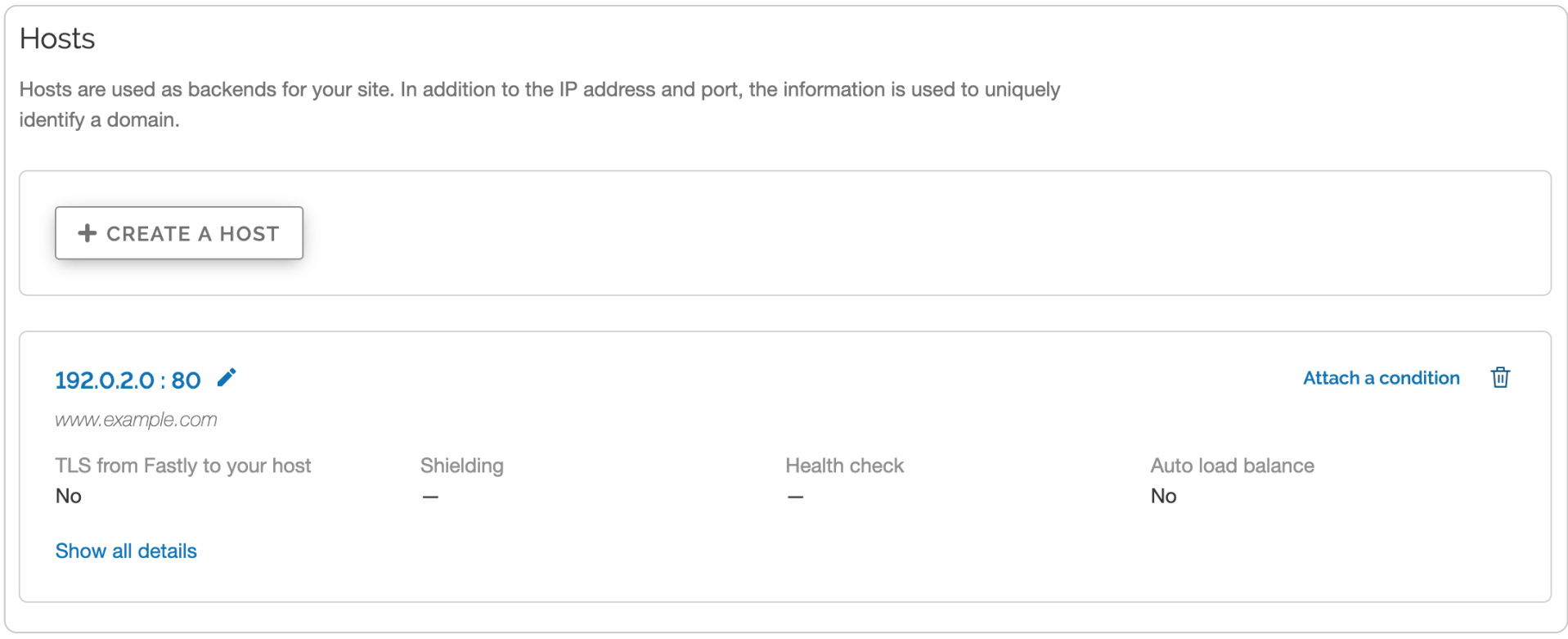
Health checks monitor the status of your hosts. Fastly performs health checks on your origin server based on the Check frequency setting you select in the Create a new health check page. The Check frequency setting you select specifies approximately how many requests per minute Fastly POPs are checked to see if they pass. There is roughly one health check per Fastly POP per period. Any checks that pass will be reported as "healthy."

Before you begin

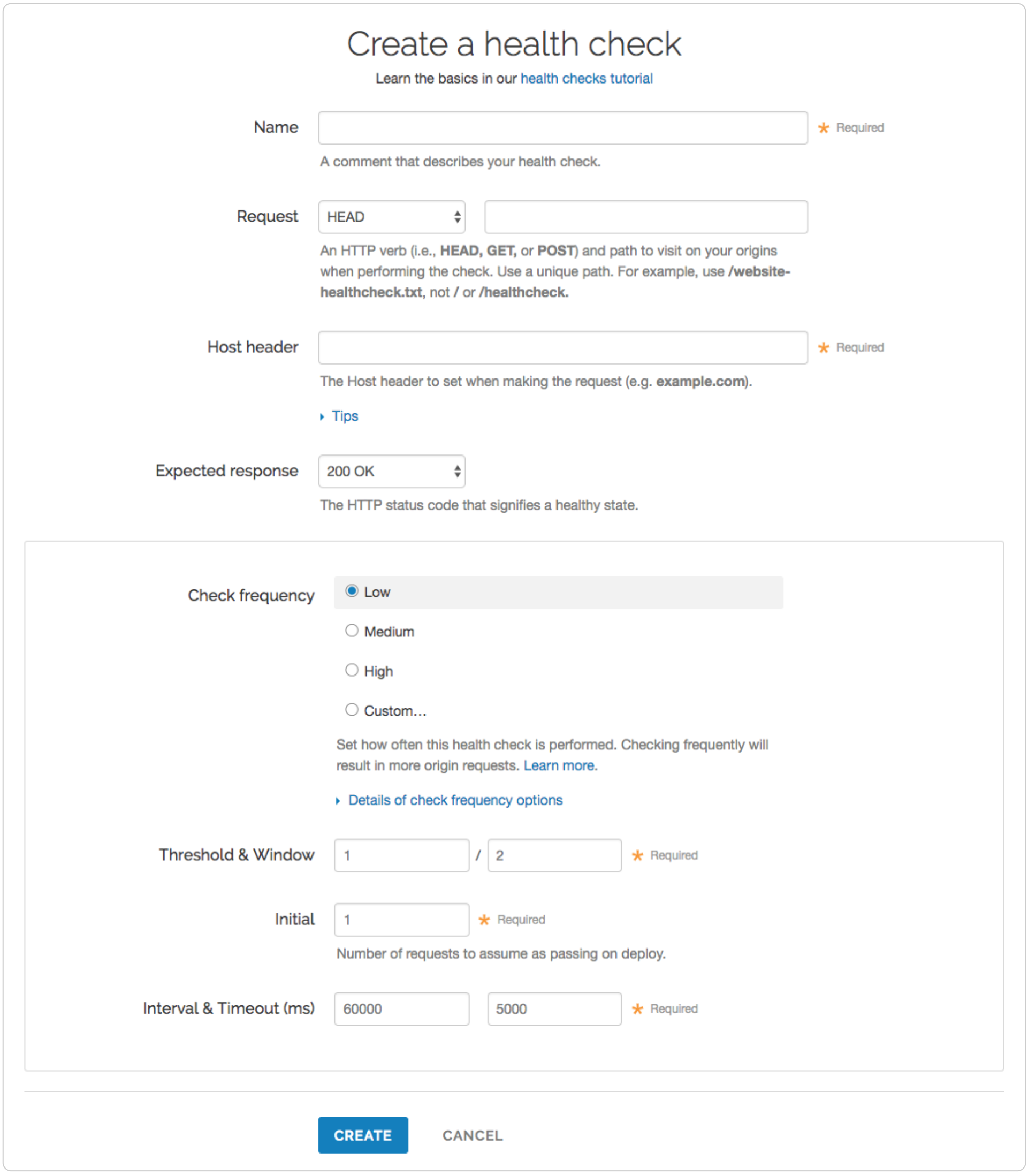
Be sure you learn about the web interface controls and how to work with services before you start setting up health checks.

Creating a health check

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.



5. Click the **Create health check** button. The Create a health check page appears.



6. Fill out the **Create a health check** fields as follows:
   - In the **Name** field, enter a human-readable identifier for the health check (e.g., `West Coast Origin Check`).
   - From the **Request** menu, select an HTTP verb. In the **Request** field, enter the path to visit when performing the check. Use a unique path. For example, use `/website-healthcheck.txt`, not `/` or `/healthcheck`.
   - In the **Host header** field, enter the HTTP Host header to set when making the request (e.g., `example.com`).
   - From the **Expected response** menu, select the HTTP status code the origin servers must respond with for the check to pass (usually `200 OK`).
   - In the **Check frequency** section, select a setting to control how often the health check is performed.
     - **Low:** One request every minute from each data center, where "healthy" means 1 out of 2 must pass.
     - **Medium:** One request every 15 seconds from each data center, where "healthy" means 3 out of 5 must pass.
     - **High:** One request every 2 seconds from each data center, where "healthy" means 7 out of 10 must pass.

- **Custom:** A custom frequency you specify.

- In the **Threshold & Window** fields, enter the number of successes per total number health checks. For example, specifying `3/5` means 3 out of 5 checks must pass to be reported as healthy.
- In the **Initial** field, enter the number of requests to assume as passing on deploy. For example, if the Threshold & Window field is set to `3/5` and the Initial field is set to `1`, a backend would be marked as "unhealthy" until it passed two more health checks to reach the required minimum.
- In the **Interval & Timeout (ms)** fields, enter times. Interval represents the period of time for the requests to run. Timeout represents the wait time until request is considered failed. Both times are specified in milliseconds.

7. Click the **Create** button.

Your new health check now appears in the list of checks.

Assigning a health check

Health checks do nothing on their own, but they can be added as a special parameter to an origin server in your configuration.

1. Edit one of your existing origin servers by clicking the origin server's name. The Edit this host page appears.
2. From the **Health checks** menu, select the health check you just created.
3. Click **Update**.

Fastly will now use the health check to monitor the selected origin server.

Creating and assigning health checks using VCL

Health checks can also be created and assigned using VCL. See our developer documentation on health checks for more information.

Troubleshooting

Fastly will periodically check your origin server based on the options chosen. Pay special attention to the HTTP Host header. A common mistake is setting the wrong host. If the origin server does not receive a host it expects, it may issue a 301 or 302 redirect causing the health check to fail. Also, Varnish requires the origin server receiving the health check requests to close the connection for each request. If the origin server does not close the connection, health checks will time out and fail.

If an origin server is marked unhealthy due to health checks, Fastly will stop attempting to send requests to it. Once all of your origin servers are marked unhealthy, Fastly will return a 503 error (service unavailable) to the client unless you tell it otherwise. You can configure Fastly to attempt to serve stale content instead until your origin servers become available again.

What's next

Learn more about working with domains and working with hosts as you continue to refine versions of your service configurations.

Working with hosts

Last updated: 2022-03-01

https://docs.fastly.com/en/guides/working-with-hosts

To communicate with your origin servers, you can add and edit a host.

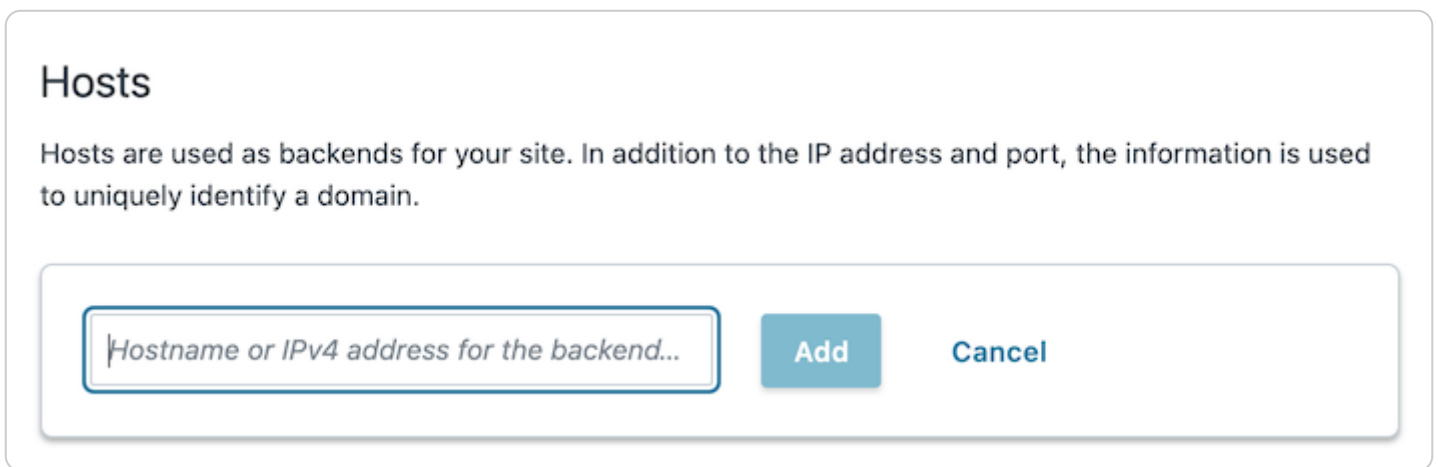
Before you begin

Be sure you learn about the web interface controls and how to work with services before you start working with your hosts.

Adding a host

To add a host, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the **Create a host** button. The Hosts field appears.

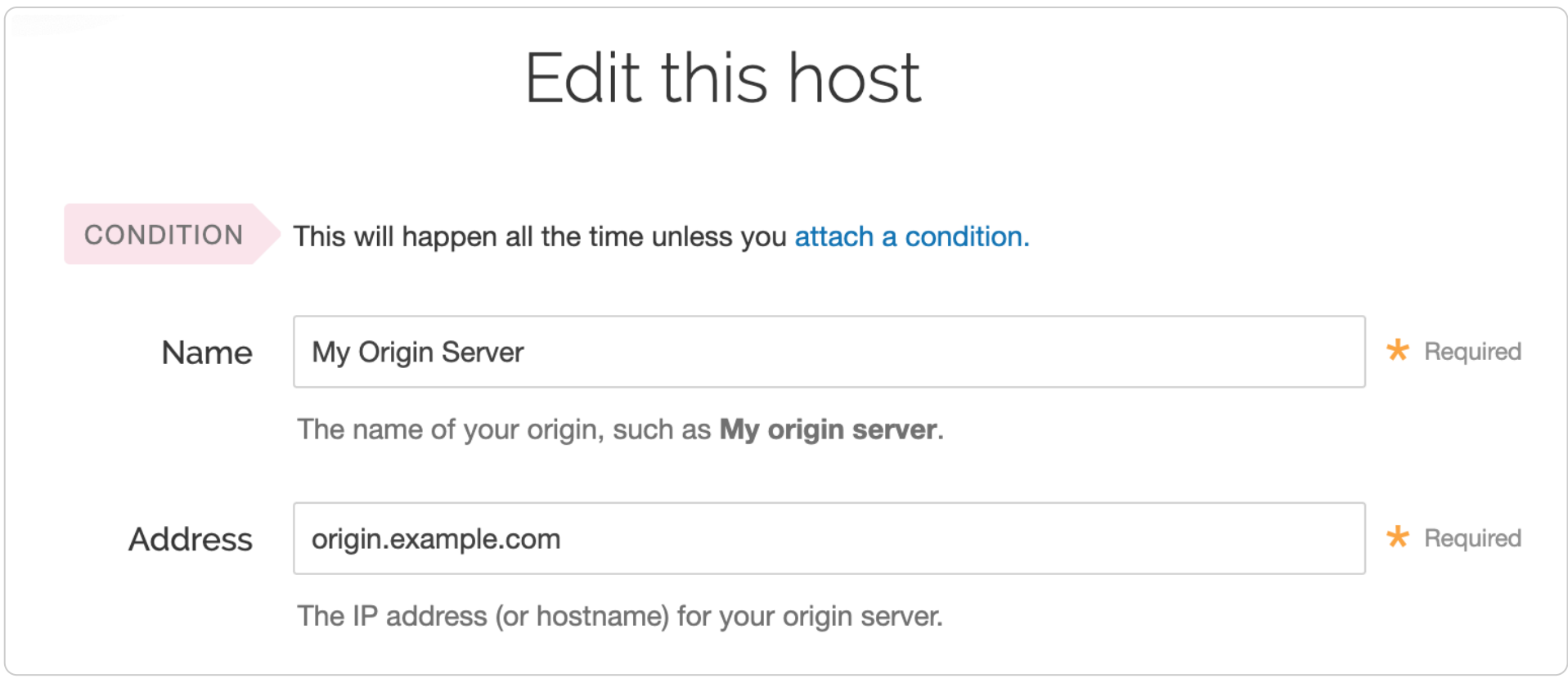


6. Fill out the **Hosts** field by entering the hostname or IP address of your origin server. Entering a hostname automatically enables Transport Layer Security (TLS) and assigns port 443. Entering an IP address disables TLS and assigns port 80.
7. Click **Add** to add your host.

Editing a host

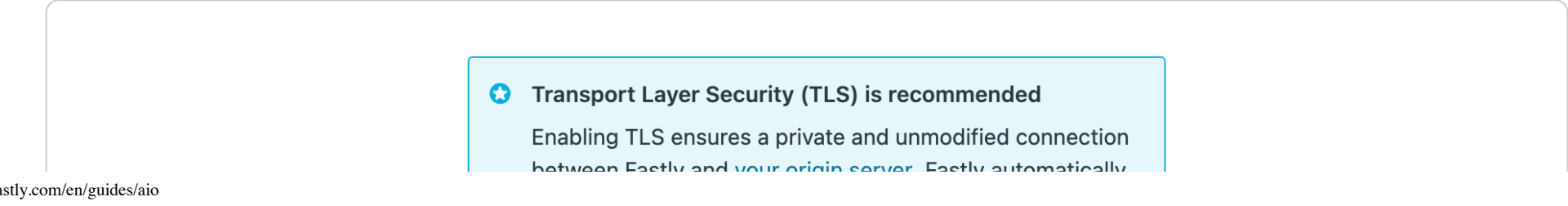
After you've created your host, you can edit the host's details by following the steps below:

1. In the **Hosts** area, click the pencil icon next to the Host you want to edit. The Edit this host page appears.



2. Fill out the **Edit this host** fields as follows:
   - In the **Name** field, enter the name of your server (for example, `My Origin Server`). This name is displayed in the Fastly web interface.
   - In the **Address** field, enter the IP address (or hostname) of your origin server.

     See Understanding the difference between certificate hostname and SNI hostname values for more information about hostnames.



between Fastly and your origin server. Fastly automatically enables TLS for hosts on port 443.

Additionally, enabling TLS ensures your traffic can be routed around internet performance issues using Fastly's Precision Path capability.

Enable TLS?

- (•) **Yes,** enable TLS and connect securely using port [443]
- ( ) **No, do not enable TLS.** Instead connect using port [80]

Verify certificate?

- (•) **Yes,** verify the authenticity of the TLS certificate
- ( ) **No,** do not verify my TLS certificate

Certificate hostname [www.example.com] * Required

This value is matched against the certificate Common Name (CN) or a Subject Alternative Name (SAN).

▸ Need help looking up your Certificate hostname?

▸ What happened to SSL hostname?

SNI hostname [ ]

[ ] Match the SNI hostname to the Certificate hostname. This value identifies which certificate should be used for the request to origin.

▸ When is SNI hostname required?

TLS CA certificate [ ]

CA certificate used to verify the certificate from origin. Must be in PEM format.

▸ When should I specify TLS CA certificate?

› Advanced TLS options Minimum TLS version, Maximum TLS version, Ciphersuites, TLS client certificate, TLS client key

Shielding [(none)]

The shield POP designated to reduce inbound load on your origins by serving cached data. Learn more about POP request handling, the caveats of shielding, and how to select the right shield location for your origin.

Health check [(none)]

+ Create a health check

A small request assigned to test the state of your origin.

Auto load balance [No]

3. Configure the **Transport Layer Security (TLS)** controls as follows:

   - From the **Enable TLS?** options, leave the default set to **Yes** if you want to enable TLS to secure the connection between Fastly and your origin. To enable TLS, a valid SSL certificate must be installed on your origin server and port 443 (or the specified port) must be open in the firewall. You can select **No** if you do not want to use TLS.
   - From the **Verify certificate?** options, leave default set to **Yes** if you want to verify the authenticity of the TLS certificate. Selecting **No** means the certificate will not be verified.

     > **WARNING**
     >
     > Not verifying the certificate has serious security implications, including vulnerability to man-in-the-middle attacks. Consider uploading a CA certificate instead of disabling certificate validation.

   - In the **Certificate hostname** field, enter the hostname associated with your TLS certificate. This value is matched against the certificate common name (CN) or a subject alternate name (SAN) depending on the certificate you were issued.
   - In the **SNI hostname** field, optionally enter the hostname of a different certificate to be used for the request to origin if you are using Server Name Indication (SNI) to put multiple certificates on your origin. Alternatively, select the checkbox to match the SNI hostname to the Certificate hostname. This information also gets sent to the server in the TLS handshake. See our guidance on understanding the difference between certificate hostname and SNI hostname values for additional details.
   - In the **TLS CA certificate** field, optionally provide a certificate in PEM format if you're using a certificate that is either self-signed or signed by a certification authority (CA) not commonly recognized by major browsers. The PEM format looks like this:

```
-----BEGIN CERTIFICATE-----
MIIDrzCCApegAwIBAgIQCDvgVpBCRrGhdWrJWZHHSjANBgkqhkiG9w0BAQUFADBh
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMSAwHgYDVQQDExdEaWdpQ2VydCBHbG9iYWwgUm9vdCBD
QTAeFw0wNjExMTAwMDAwMDBaFw0zMTExMTAwMDAwMDBaMGExCzAJBgNVBAYTAlVT
MRUwEwYDVQQKEwxEaWdpQ2VydCBJbmMxGTAXBgNVBAsTEHd3dy5kaWdpY2VydC5j
b20xIDAeBgNVBAMTF0RpZ2lDZXJ0IEdsb2JhbCBSb290IENBMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEA4jvhEXLeqKTTo1eqUKKPC3eQyaKl7hLOllsB
CSDMAZOnTjC3U/dDxGkAV53ijSLdhwZAAIEJzs4bg7/fzTtxRuLWZscFs3YnFo97
nh6Vfe63SKMI2tavegw5BmV/Sl0fvBf4q77uKNd0f3p4mVmFaG5cIzJLv07A6Fpt
43C/dxC//AH2hdmoRBBYMql1GNXRor5H4idq9Joz+EkIYIvUX7Q6hL+hqkpMfT7P
T19sdl6gSzeRntwi5m3OFBqOasv+zbMUZBfHWymeMr/y7vrTC0LUq7dBMtoM1O/4
gdW7jVg/tRvoSSiicNoxBN33shbyTApOB6jtSj1etX+jkMOvJwIDAQABo2MwYTAO
BgNVHQ8BAf8EBAMCAYYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUA95QNVbR
TLtm8KPiGxvDl7I90VUwHwYDVR0jBBgwFoAUA95QNVbRTLtm8KPiGxvDl7I90VUw
DQYJKoZIhvcNAQEFBQADggEBAMucN6pIExIK+t1EnE9SsPTfrgT1eXkIoyQY/Esr
hMAtudXH/vTBH1jLuG2cenTnmCmrEbXjcKChzUyImZOMkXDiqw8cvpOp/2PV5Adg
06O/nVsJ8dWO41P0jmP6P6fbtGbfYmbW0W5BjfIttep3Sp+dWOIrWcBAI+0tKIJF
PnlUkiaY4IBIqDfv8NZ5YBberOgOzW6sRBc4L0na4UU+Krk2U886UAb3LujEV0ls
YSEY1QSteDwsOoBrp+uvFRTp2InBuThs4pFsiv9kuXclVzDAGySj4dzp30d8tbQk
CAUw7C29C79Fv1C5qfPrmAESrciIxpg0X40KPMbp1ZWVbd4=
-----END CERTIFICATE-----
```

4. To apply advanced TLS options, click the **Advanced TLS options** link and decide which of the optional fields to specify, if any. For more information about these controls, see Advanced TLS options.

5. Configure the remaining **Create a host** controls as follows:

   - From the **Shielding** menu, optionally select a POP to enable the shielding feature. For more information, see our guide on shielding.
   - From the **Health check** menu, optionally select a health check for this origin server. For more information, see our guide on working with health checks.
   - From the **Auto load balance** menu, optionally select **Yes** to enable load balancing for this origin server. For more information, see our guide on load balancing.
   - If you enabled load balancing, enter a weight in the **Weight** field.
   - In the **Override host** field, optionally enter the hostname of your override Host header based on the origin you're using. The value in this field will take precedence over anything you've set using the global override host configuration. To see example Host headers for third-party services, refer to our developer documentation on overriding the Host header.

     > **NOTE**

> To override the incoming Host header on your origin using your Fastly service, refer to the Specifying the override host guide.

6. Click the **Advanced options** link and decide which of the optional fields to change, if any:
   - In the **Maximum connections** field, optionally enter the maximum number of connections for your backend. The default limit is 200 connections per cache node to protect your origins from being overloaded.
   - In the **Error threshold** field, optionally enter the number of errors allowed before an origin is considered down.
   - In the **Connection timeout** field, optionally enter how long, in milliseconds, to wait for a connection timeout. The default is 1000 milliseconds.
   - In the **First byte timeout** field, optionally enter how long, in milliseconds, to wait for a first byte timeout. The default is 15000 milliseconds.
   - In the **Between bytes timeout** field, optionally enter how long, in milliseconds, to wait between bytes. The default is 10000 milliseconds.
7. Click the **Update** button. The new override host appears under the **Show all details** field of the **Override host** section and a code block is added to the origin definition in your VCL that will look similar to the following:

```
Backend F_Host_1 {
   .host = "..."; # IP or hostname
   .host_header = "example.com";
   .always_use_host_header = true;
    ...
}
```

8. Click the **Activate** button to deploy your configuration changes.

Advanced TLS options

If you enabled TLS on your host, there are some optional configurations you can apply by clicking the **Advanced TLS options** link.

Specifying acceptable TLS protocol versions

If your origin server is configured with support for modern TLS protocol versions, you can customize the TLS protocols Fastly will use to connect to it by setting a **Minimum TLS Version** and **Maximum TLS Version**. We recommend setting both to the most up-to-date TLS protocol, currently 1.3, if your origin can support it.

Use the `openssl` command to verify your origin supports a given TLS protocol version. For example:

```
$ openssl s_client -connect origin.example.com:443 -tls1_3
```

Replace `-tls1_3` with `tls1_2`, `tls1_1` and `tls1_0` to test other protocol versions. Fastly does not support SSLv2 or SSLv3.

> **IMPORTANT**
>
> In line with security best practices, Fastly recommends enabling servers with version 1.3 of the TLS protocol by default. For backend connections from our edge nodes to customer origins, Fastly supports TLS 1.3, 1.2, 1.1, and 1.0 depending on the versions of the protocol in use on the origin server. Fastly will continue to support TLS 1.0 based on the ServerHello message as described in RFC 5246 if the server selects TLS 1.0 as the highest supported version.

Specifying acceptable TLS cipher suites

Fastly supports configuring the OpenSSL cipher suites used when connecting to your origin server. This allows you to turn specific cipher suites on or off based on security properties and origin server support. The **Ciphersuites** setting accepts an OpenSSL formatted cipher list. We recommend using the strongest cipher suite your origin will support as detailed by the Mozilla SSL Configuration Generator.

Use the `openssl` command to verify your origin supports a given cipher suite. For example:

```
$ openssl s_client -connect origin.example.com:443 -tls1_2 -cipher ECDHE-RSA-AES128-GCM-SHA256
```

Replace `-cipher ECDHE-RSA-AES128-GCM-SHA256` with the cipher suite to test.

Specifying a TLS client certificate and key

To ensure TLS connections to your origin come from Fastly and aren't random, anonymous requests, set your origin to verify the client using a client certificate. Simply paste the certificate and private key in PEM form into the **TLS client certificate** text box.

> **IMPORTANT**
>
> The private key must not be encrypted with a passphrase.

Then configure your backend to require client certificates and verify them against the CA cert they were signed with. Here are some ways of doing that:

- Apache
- Nginx
- IIS

Fastly also supports securing your connection to origin with mutual TLS (mTLS).

Understanding the difference between certificate hostname and SNI hostname values

The following explains the difference between a certificate and SNI hostname value:

The certificate hostname (ssl_cert_hostname). This hostname validates the certificate at origin. It is always required. This value should match the certificate common name (CN) or an available subject alternate name (SAN). It displays as `ssl_cert_hostname` in VCL. This doesn't affect the SNI certification. You can set this value in **Certificate hostname** field.

The SNI hostname (ssl_sni_hostname). This hostname determines which certificate should be used for the TLS handshake. SNI is generally only required when your origin is using shared hosting, such as Amazon S3, or when you use multiple certificates at your origin. SNI allows the origin server to know which certificate to use for the connection. This value displays as `ssl_sni_hostname` in VCL. This doesn't affect the certificate validation.

The table below shows you what happens when you set the Certificate and SNI hostname values in the TLS settings:

If Certificate hostname contains...	and SNI hostname contains...	then the Certificate Validation value will be...	and the SNI value will be...
www.example.com	nothing	www.example.com	nothing
www.example.com	www.example.org	www.example.com	www.example.org

About the ssl_hostname value (deprecated). The `ssl_hostname` value has been deprecated and replaced with `ssl_cert_hostname` and `ssl_sni_hostname`. Use these two values instead.

> **IMPORTANT**
>
> If you use an IP address for your `.host` value (i.e., by not entering a value in your certificate hostname), this will generate an error where the certificate hostname specified in your service's origin TLS settings doesn't match either the Common Name (CN) or available Subject Alternate Names (SANs).

Using a wildcard certificate

If you're using a wildcard certificate, you can use any SNI hostname that matches the wildcard certificate. The SNI hostname must be a fully-qualified domain name (FQDN), per RFC 6066.

The table below shows a variety of possible combinations of certificate and SNI hostnames that could be used with a wildcard certificate for `*.example.com`:

Certificate hostname	SNI hostname
www.example.com	www.example.com

Certificate hostname	SNI hostname
live.example.com	live.example.com
*.example.com	www.example.com

If you set the certificate hostname to `*.example.com`, Fastly will treat it as a literal.

What's next

Learn more about working with domains and working with health checks as you continue to refine versions of your service configurations.

Working with services

Last updated: 2021-11-02

https://docs.fastly.com/en/guides/working-with-services

A service is a user-defined set of caching rules and behavior for a website or application. You might create new services to do things like:

- add a new website under your control to your list of web properties
- add a new domain to your growing list of existing domains already served by Fastly
- isolate traffic metrics for specific digital assets, like a site's images

Once created, you can edit and activate new versions of your services that include refinements and updates to your configuration settings. The web interface also allows you to do other things with existing services, like compare them to each other, deactivate or reactivate them, and delete them.

Before you begin

Before you begin working with services, be sure you understand how caching and CDNs work. You'll also need to understand the Fastly web interface controls before using them to work with your services.

TIP

This guide includes instructions for creating and interacting with both Deliver and Compute services. Our developer portal provides more information on working with services that take advantage of Compute@Edge.

Creating a new service

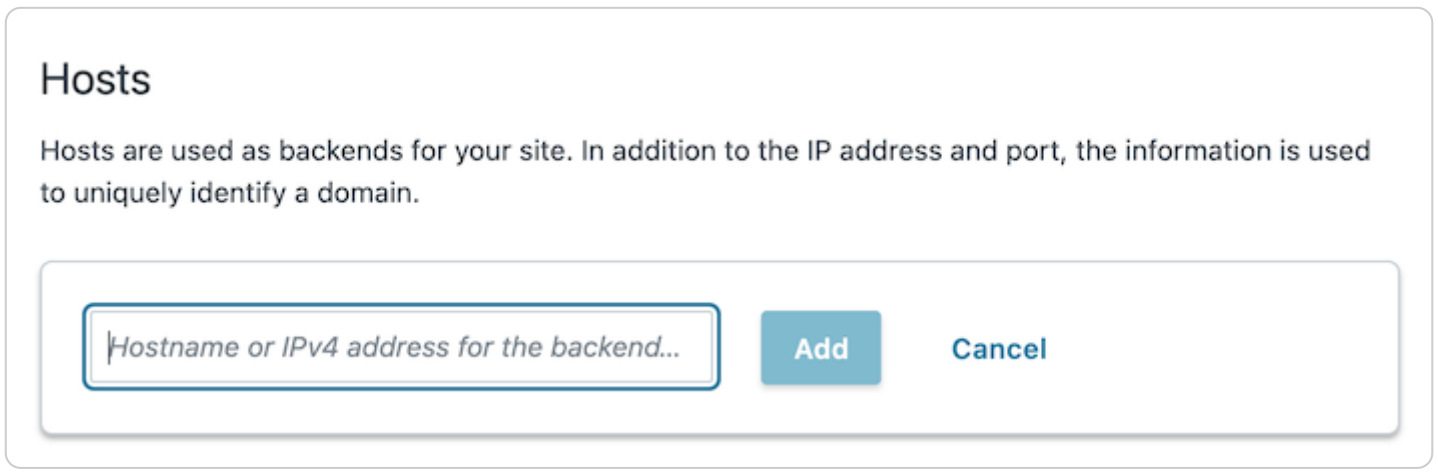
You can create new Deliver and Compute services through the web interface.

Creating a new Deliver service

To create a new Deliver service, follow the steps below:

1. Log in to the Fastly web interface.
2. Click the **Create service** button. The Create service menu appears with the types of services you can create.
3. Select **Deliver** to create a new VCL-based service. A new, unnamed VCL service's configuration page appears.
4. Rename the service as necessary.
5. Optionally, add a comment to help you identify what you're working on.
6. Use the **Domains** fields to add a domain to the service.

7. Use the **Hosts** field to add a host to the service.



> **TIP**
>
> Many customers setting up services specify an override host at the same time. See our guidance on overriding a host at the origin level if you're interested in exploring this functionality.

8. Test your service configurations by opening **http://www.example.com.global.prod.fastly.net** in a new browser window, replacing `www.example.com` with your own website's domain name. Your website should appear, though it may take up to 60 seconds for new configuration settings to take effect.

> **TIP**
>
> You can continue to explore various configuration settings for as long as you like before starting to serve traffic.

9. Click the **Activate** button at the top right of the screen. A confirmation window appears.
10. Click **Confirm and Activate** to confirm you want to activate your new service. The Deliver page appears with details about the configuration settings you've applied.
11. Once you're ready, complete your service setup and start serving traffic through Fastly by setting your domain's CNAME DNS record to point to Fastly.

Creating a new Compute service

To create a new Compute service, follow the steps below:

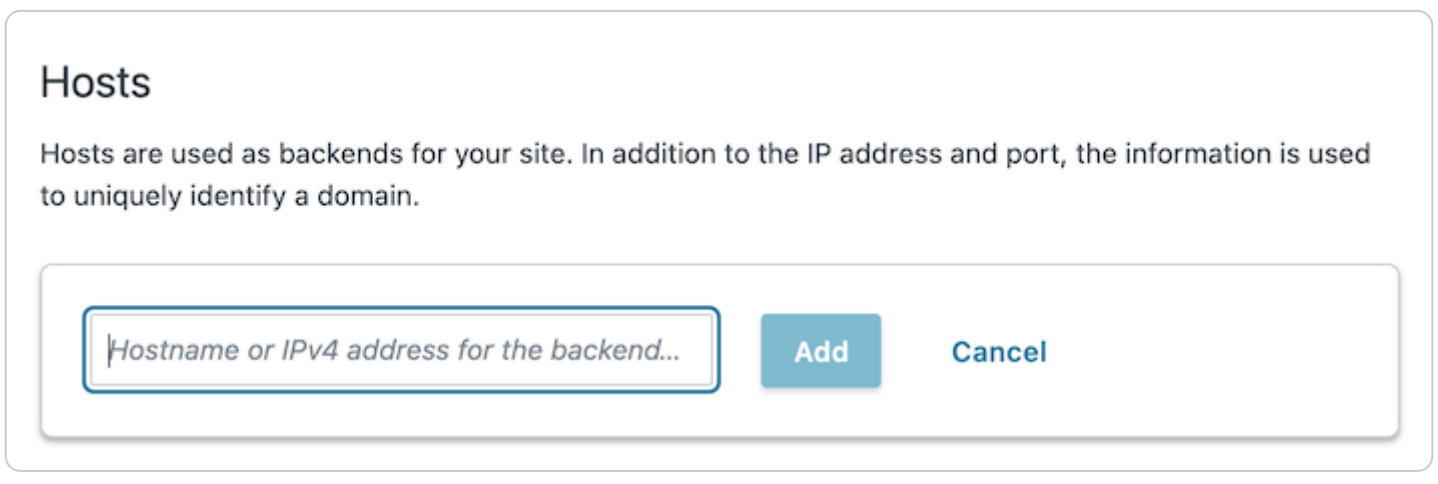
> **NOTE**
>
> Some steps require using the Fastly CLI. If you haven't already, follow the steps for creating an API token, making sure it has `global` scope. Then, download and install the Fastly CLI and use that token to authenticate your account before continuing.

1. Log in to the Fastly web interface.
2. Click the **Create service** button. The Create service menu appears with the types of services you can create.
3. Select **Compute** to create a new Wasm-based service. A new, unnamed Wasm service's configuration page appears.
4. Rename the service as necessary.
5. Optionally, add a comment to help you identify what you're working on.

> **TIP**
>
> After creating your Compute service, you can choose to complete the remaining configuration steps via the Fastly CLI.

6. Use the **Domains** fields to add a domain to the service.

7. Use the **Hosts** field to add a host to the service.



> **TIP**
>
> Many customers setting up services specify an override host at the same time. See our guidance on overriding a host at the origin level if you're interested in exploring this functionality.

8. Using the Fastly CLI, create a new Compute@Edge project.
9. Compile the project into a Wasm binary packaged for Fastly use. The package will be named `<name>.tar.gz`.
10. Return to your service in the Fastly web interface.
11. Click the **Package** link. The Package page appears.
12. Click **Browse for Package** to navigate to the package file on your system.
13. Test your service configurations by opening **http://www.example.com.global.prod.fastly.net** in a new browser window, replacing `www.example.com` with your own website's domain name. Your website should appear, though it may take up to 60 seconds for new configuration settings to take effect.

    > **TIP**
    >
    > You can continue to explore various configuration settings for as long as you like before starting to serve traffic.

14. Click the **Activate** button at the top right of the screen. A confirmation window appears.
15. Click **Confirm and Activate** to confirm you want to activate your new service. The Compute page appears with details about the configuration settings you've applied.
16. Once you're ready, complete your service setup and start serving traffic through Fastly by setting your domain's CNAME DNS record to point to Fastly.

Editing your services

You might want to edit a version of an existing service to do things like:

- change the amount of time information is retained in cache memory for a service
- configure a service to temporarily serve stale content should your origin server need to be unavailable for an extended period of time (for example, taken offline for maintenance)
- decrease the amount of time Fastly will wait for your origin server to respond to a request for content

Editing and activating versions of services

Fastly locks versions of services you've already activated to make rollbacks safer and provide version control. You can duplicate (*clone*) any existing service version, active or inactive, and edit that cloned version. You must *activate* new versions of services to deploy their configurations. Configuration changes are never automatically activated.

To make changes to a service and activate a new version, follow the steps below:

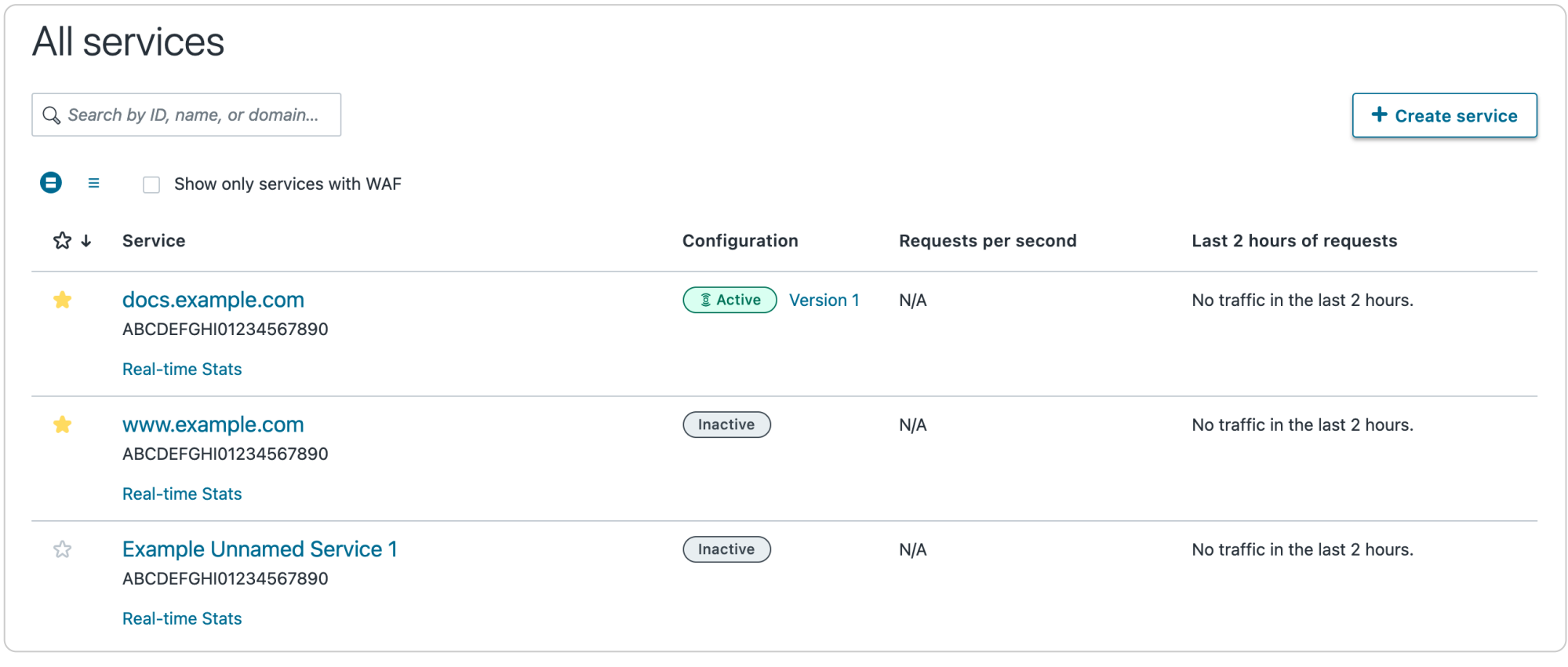
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button. The Edit configuration menu appears.

4. Select the appropriate service configuration action:
   - Select **Clone version *[version number]* (active)** to clone the active version of the service for editing.
   - Select **Edit version *[version number]* (latest draft)** to edit the latest draft of the service.

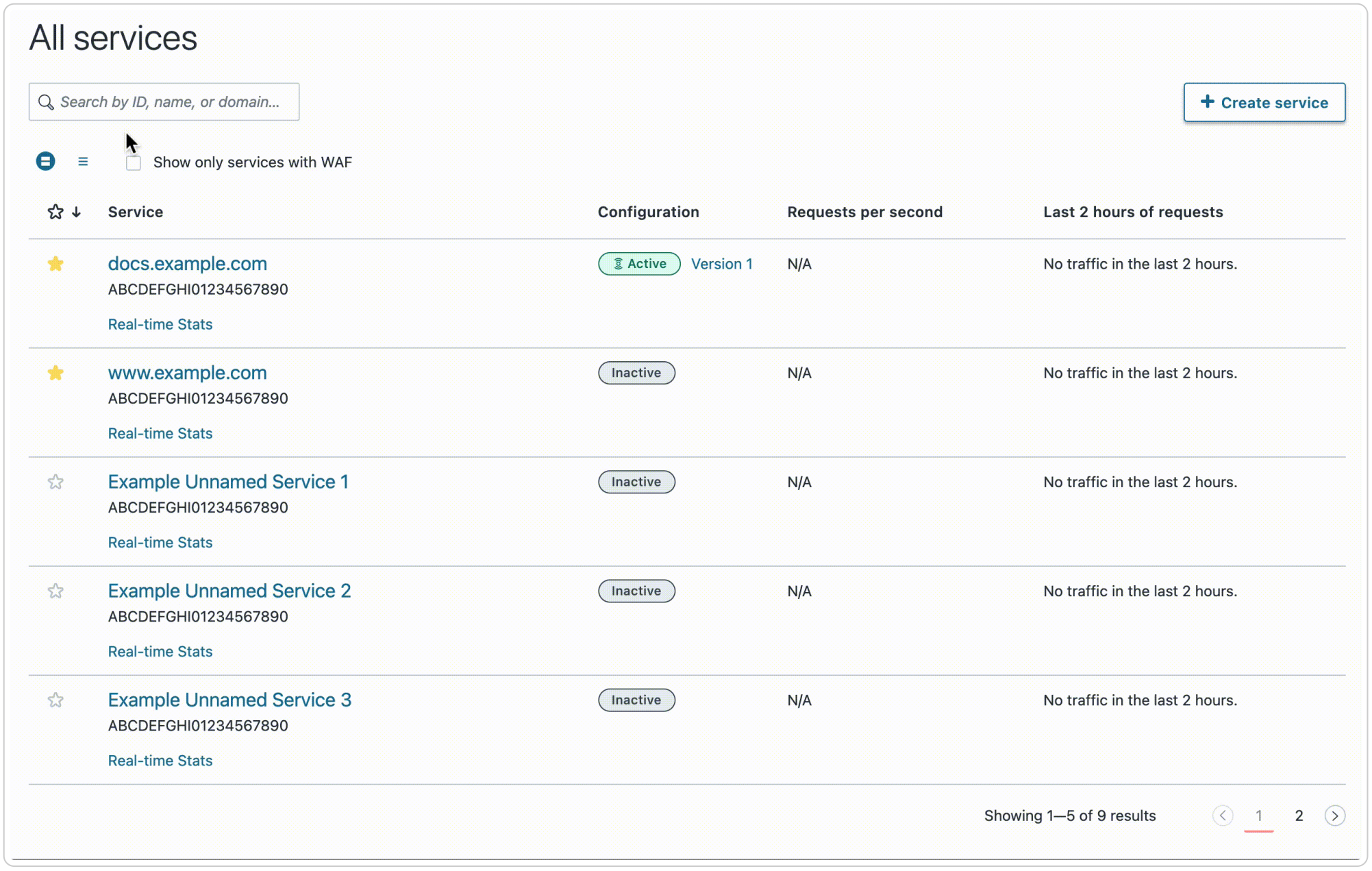
   The service version page appears, listing the version.
5. Click **Activate**. The new version of the service is activated and appears in the event log.

Viewing all services

To view all your services, log in to the Fastly web interface. If you haven't yet created services, Fastly provides several options to help you with ideas for getting started. If you've already created at least one service, however, the Home page appears displaying a summary of all your services, sorted by requests per second, with starred services listed first.

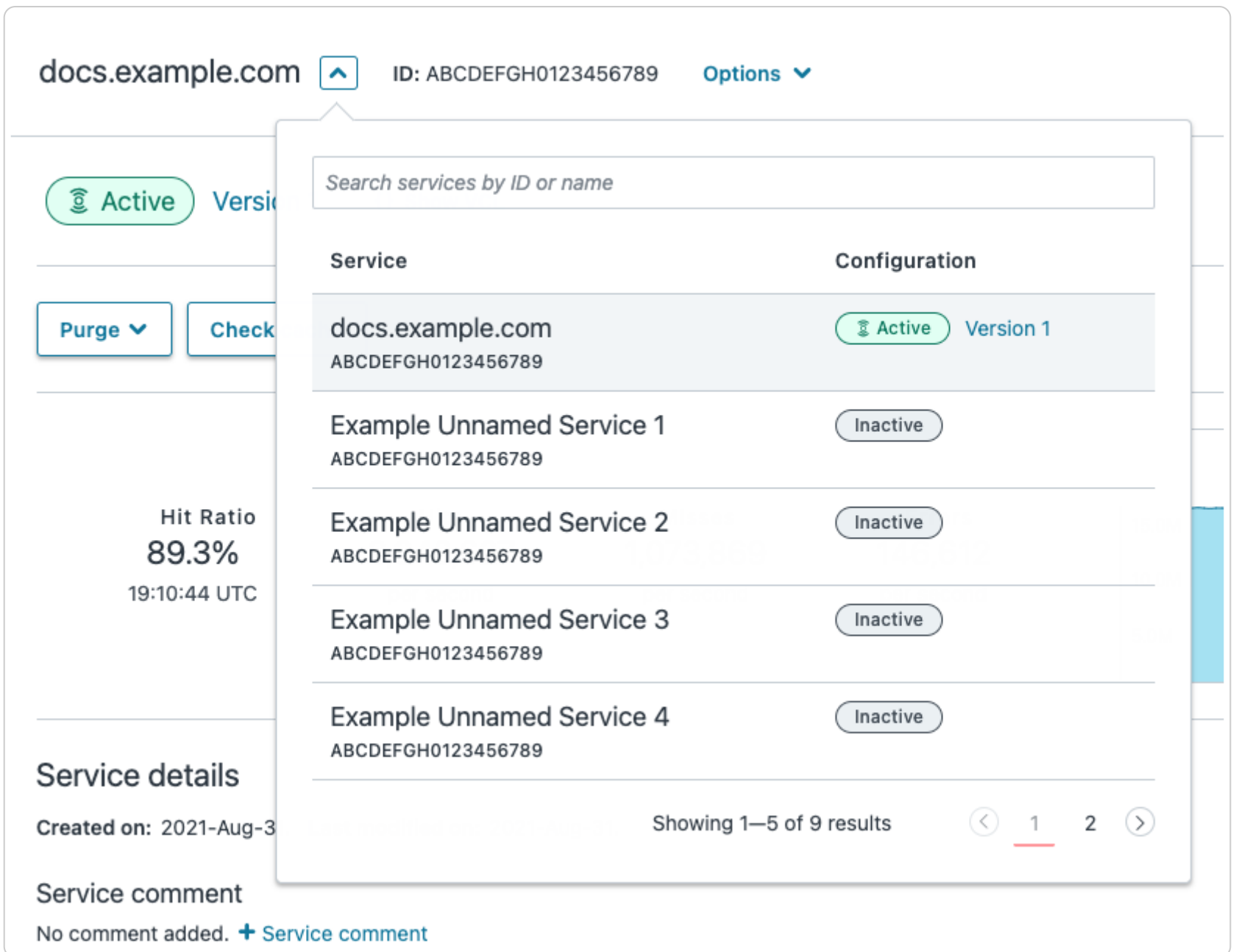


If you have a lot of services, you can view a condensed list by clicking the icon with three lines above the list of services.

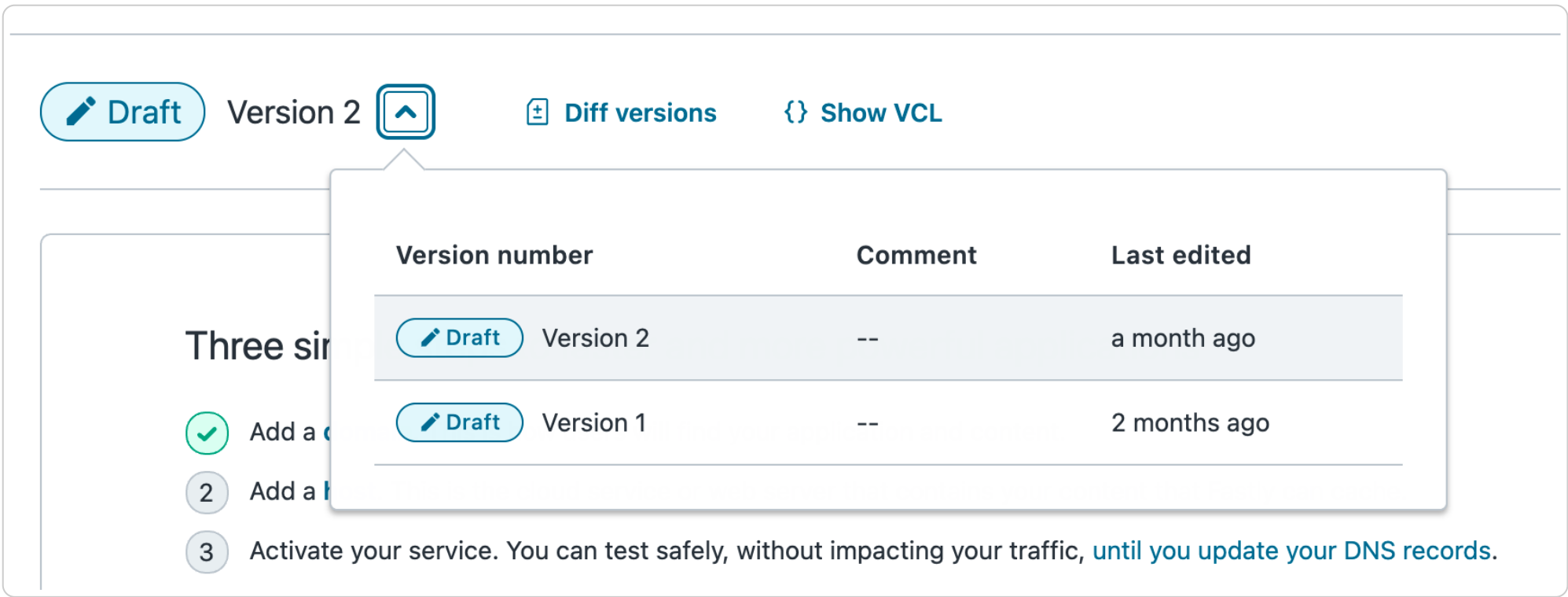


Switching between services and service versions

To switch between services associated with your account, click the switcher icon to the right of the account name and ID and select the appropriate service from the list that appears.



To switch between versions of a specific service, click the switcher icon to the right of the version number and select the appropriate version from the list that appears.

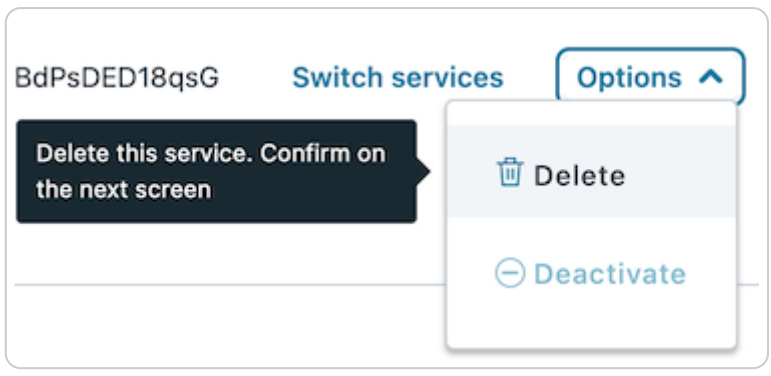


Deleting a service

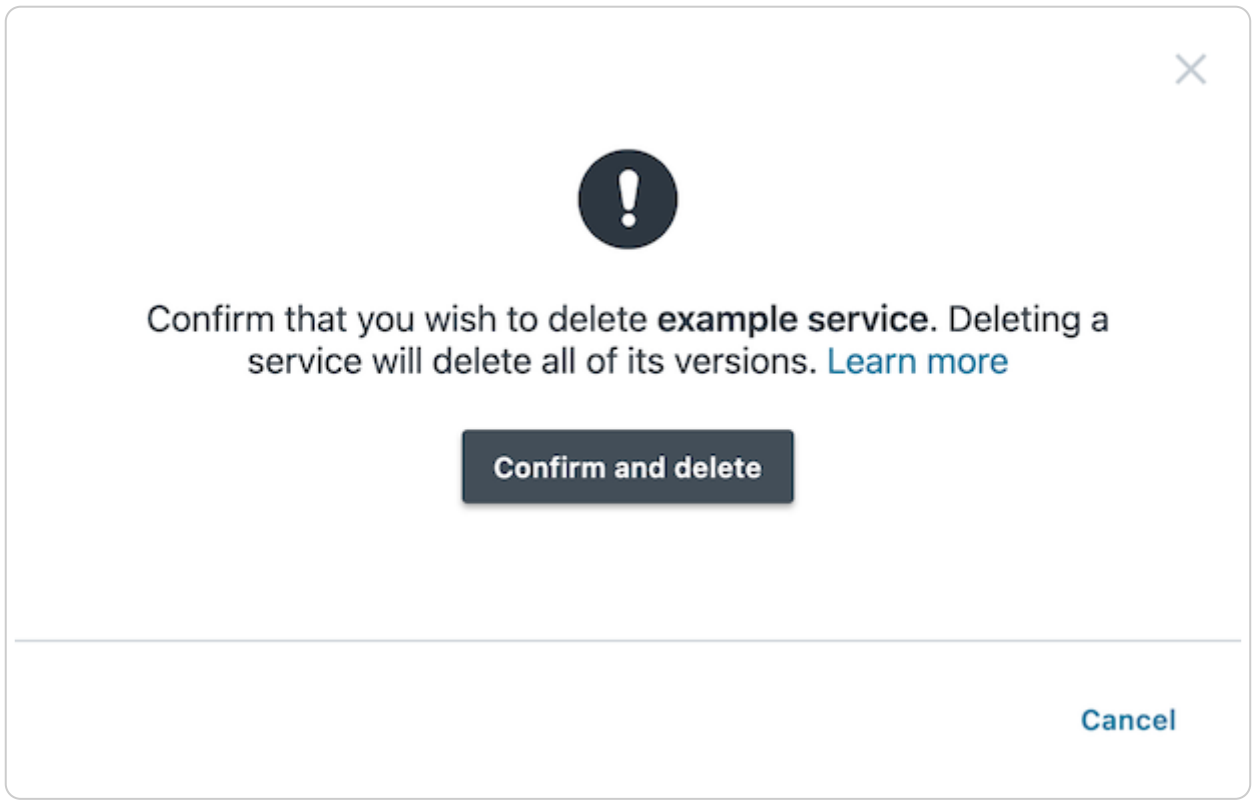
Fastly allows you to delete any service you create, along with all of its versions. Fastly does not offer a way to delete specific versions of a service, however. Service versions are meant to be an historic log of the changes that were made to a service. To undo changes introduced by a particular service version, you can always go back to a previous version and reactivate it or clone a new service version based on any old version.

To delete any service along with all of its versions, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Options** link and select **Deactivate**. The deactivate service warning appears.
4. Click the **Confirm and deactivate** button to confirm you want to deactivate your service and acknowledge that you no longer want to serve traffic with it.
5. Click the **Options** link again and select **Delete**.



The confirm delete window appears.



6. Click the **Confirm and delete** button to confirm that you want to delete the service.

IMPORTANT

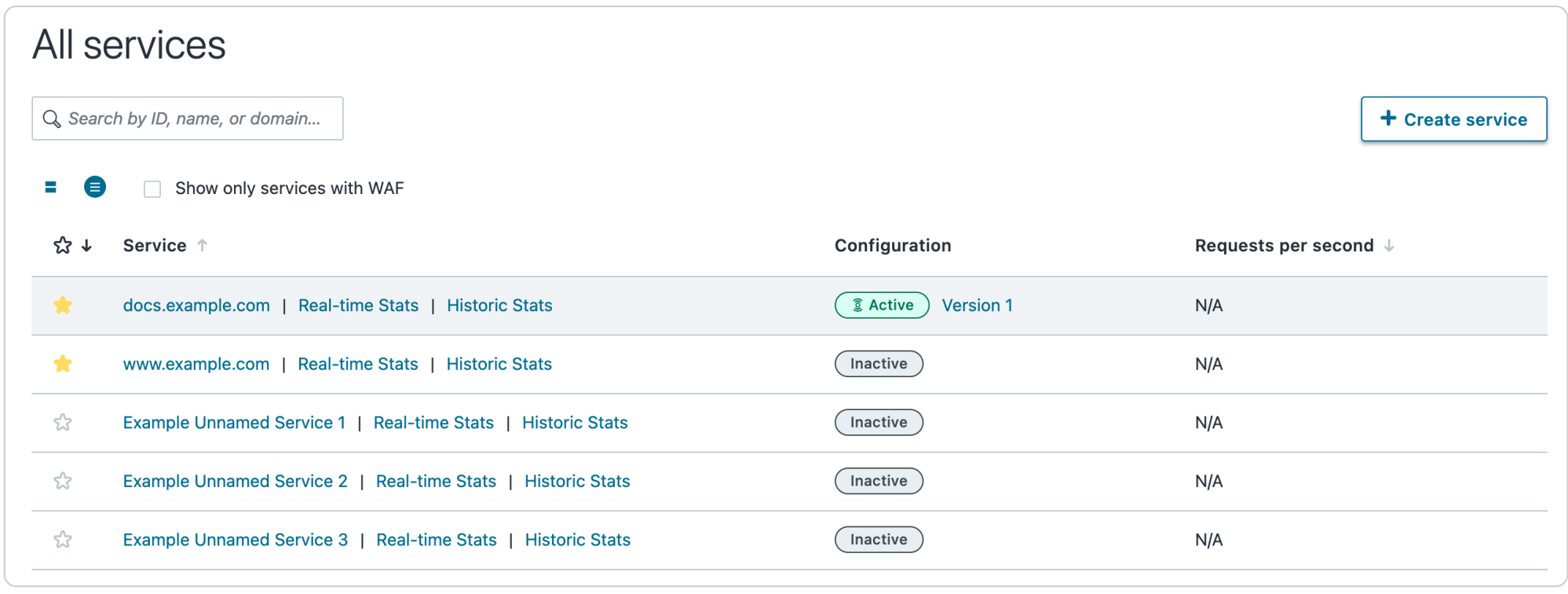
To minimize the risk of unauthorized use of your domains, we strongly recommend modifying or deleting any DNS CNAME records pointing to the Fastly hostname associated with the deleted service. Follow the instructions on your DNS provider's website.

Other things you can do

In addition to creating or editing services, you can view all your services, star them to pin them to the top of the All services page, rename them, compare versions of them, deactivate or reactivate specific versions of them, and delete them.

Starring services

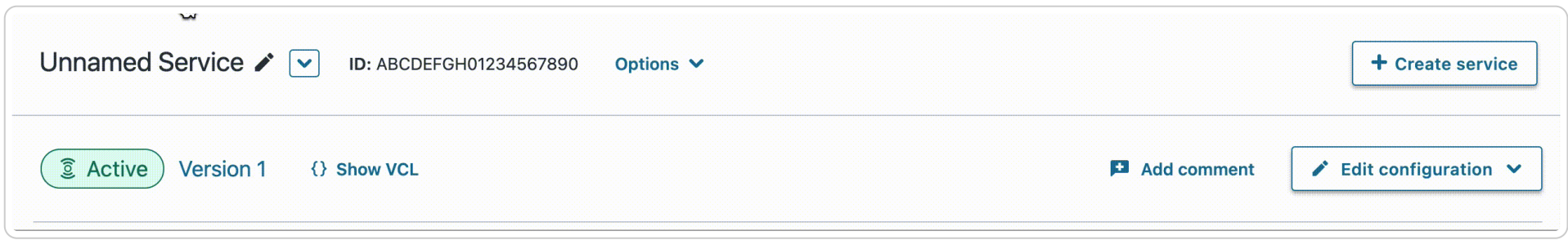
If you have a lot of services, you can star the services you use most often to mark them as important and pin them to the top of the Home page. Click the star next to a service to pin it to the top of the page.



Renaming services

To rename your service, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Select the service name text box and enter a new service name.

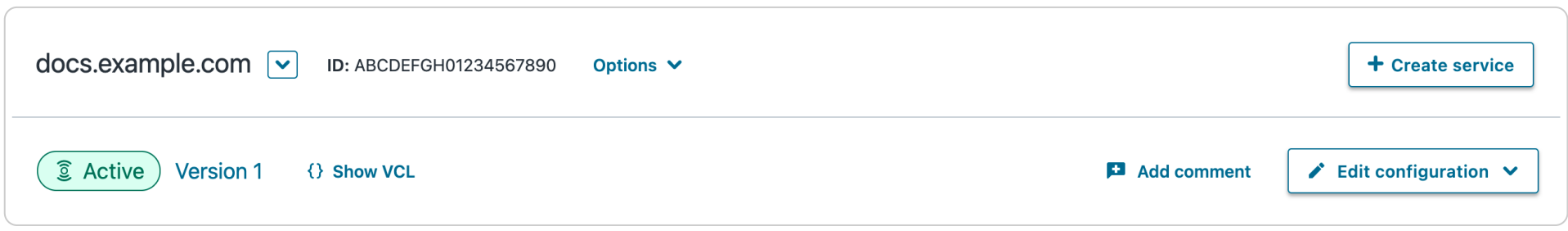


4. Press **enter**. The newly renamed service name appears.

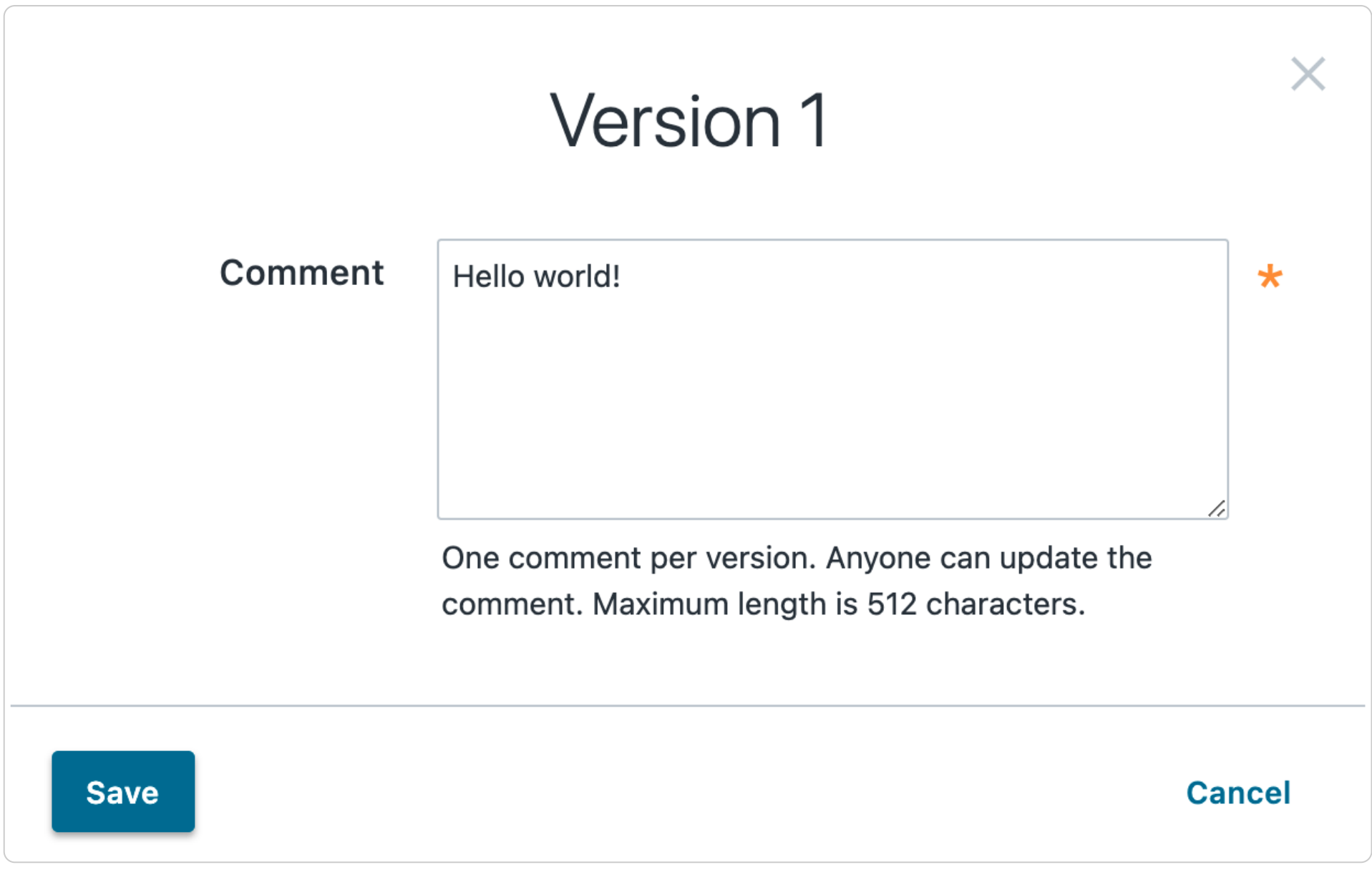
Adding comments to service versions

Service versions can include comments to label them (e.g., to identify reasons for changes in that version). You can add and update version comments on both locked and activated service versions.

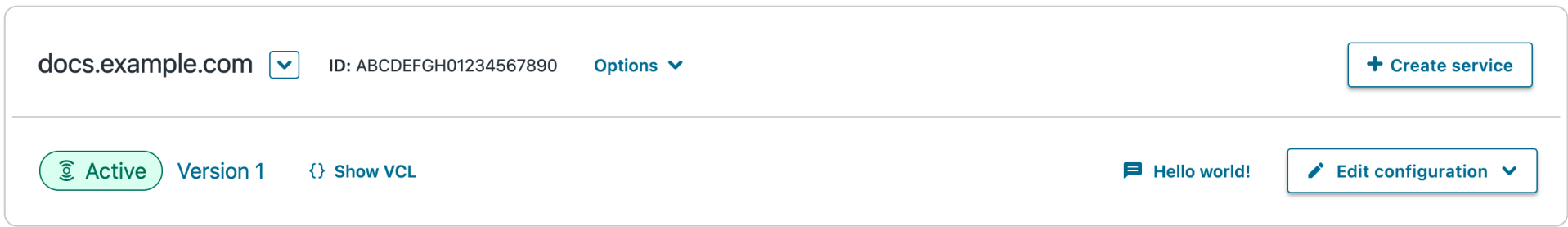
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Add comment** link in the upper right corner of the web interface.



The comment window appears.



5. In the **Comment** field, enter a meaningful comment for the version.
6. Click **Save**. The truncated version of the comment appears where the Add comment link used to be.



TIP

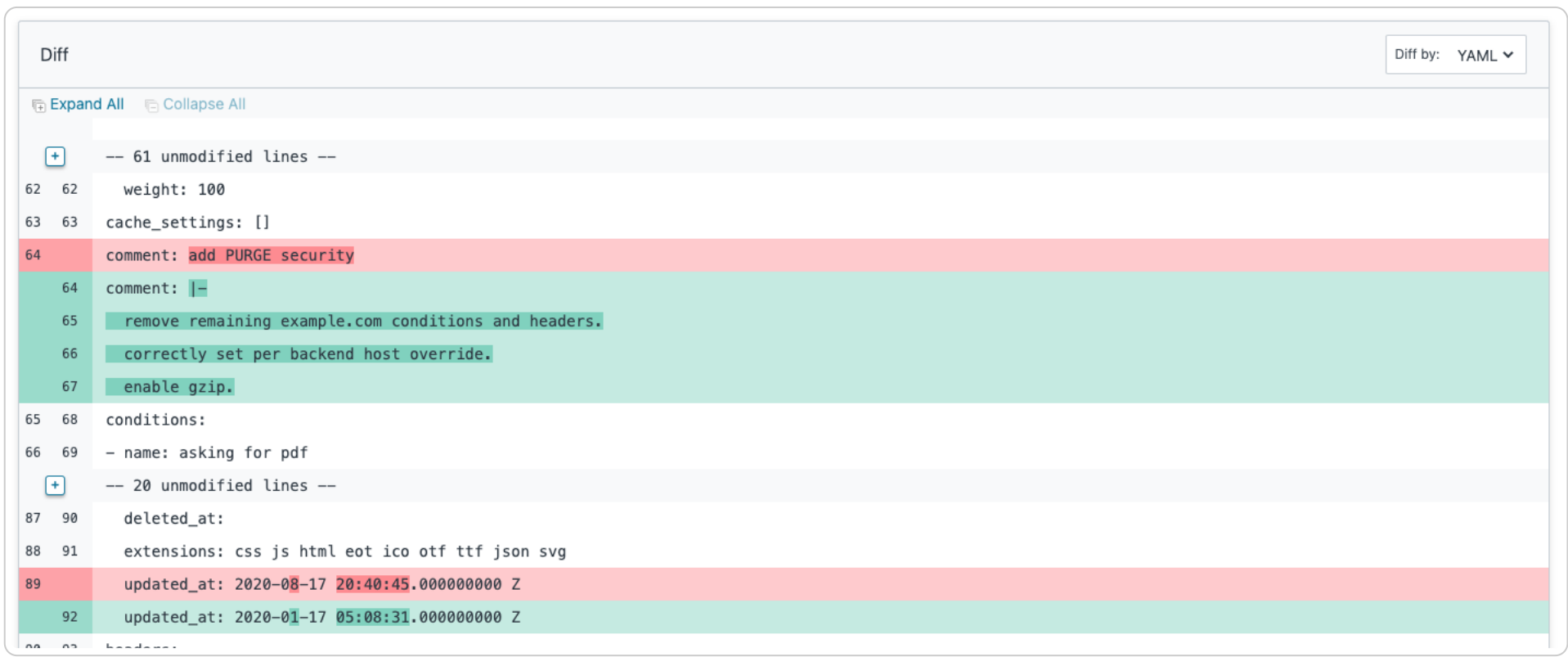
You can view service version comments at any time by clicking the service version number to display the version selection menu or by clicking the version comment icon to display the version comment in a separate window. Version comments also appear in the event log to help with account activity monitoring.

Comparing different service versions

To compare two versions of a service, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Diff versions** link. The Diff versions page appears.

   Removals are highlighted in red and the additions and changes are highlighted in green. Any large blocks of unaffected configuration lines can be expanded and viewed or collapsed and hidden by clicking on the plus (+) sign to the left of the actual changes, next to the line numbers.



TIP

You can change the compared service versions by clicking **Switch versions** and selecting a different version number in the menu that appears.

Deactivating a service

To deactivate a service, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Options** link and select **Deactivate**.

The Deactivate service window appears.

4. In the **Enter service name** field, enter the exact service name to deactivate.
5. Click the **Confirm and deactivate** button to confirm you want to deactivate your service and acknowledge that you no longer want to serve traffic with it.

IMPORTANT

To minimize the risk of unauthorized use of your domains, we strongly recommend modifying or deleting any DNS CNAME records pointing to the Fastly hostname associated with the deactivated service. Follow the instructions on your DNS provider's website.

You can also activate or deactivate a service via the API. Did you accidentally delete a service? We can help.

Reactivating a service

To reactivate a service, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click **Activate**. The service is reactivated.
5. If you removed the DNS CNAME records for the service's domains when you deactivated the service, you should add new DNS CNAME records now.

Getting help with accidental service deletions

Services can be deactivated or deleted. Deactivated services can be reactivated at any time, but once they've been deleted you must contact Customer Support to have them restored. When sending your request, remember to include:

- your customer ID
- your company name
- your service ID (the name of the service you want restored)

Customer Support will notify you when your service has been restored.

What's next

Learn more about working with domains, hosts, and health checks as you continue to refine versions of your service configurations.

§

These articles describe key features of the Fastly web interface controls.

https://docs.fastly.com/en/guides/getting-started#_web-interface

About the account menu

Last updated: 2021-11-30

https://docs.fastly.com/en/guides/about-the-account-menu

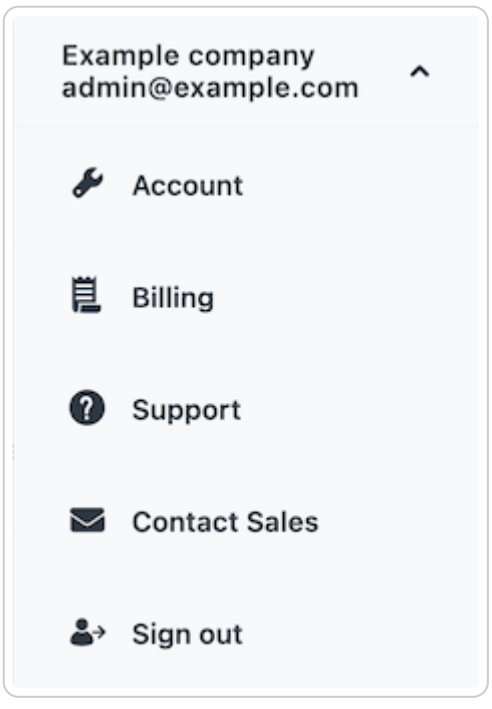
The account menu provides you with access to your account's specific settings and its billing information as determined by the roles and permissions you have been assigned. From here you can access your personal profile information and, if you have access to multiple customer accounts, you'll see an option to switch accounts. The account menu also provides a way for you to contact Support, contact Sales (sales@fastly.com), and log out of the web interface.

Before you begin

Be sure you know how to access the web interface controls before learning about each of the pages you'll encounter there.

About the Account menu

The account menu appears at the far right of the default control group:



About the Account controls

Selecting **Account** from the account menu displays Company Profile and Personal Profile details tied to your account login. The Company Profile details include:

- **Company settings** where you'll find details about your company (e.g., its name and the phone number, the account owner, and various company contacts used to streamline communication with Fastly), as well as the location to enable a login IP allowlist, enable account-wide two-factor authentication, and cancel your account
- **User management** controls where you can control user invitations and configure their roles
- **Account API tokens** created by users within your account to control or restrict access to various services
- **Audit log** keeps track of events related to your account, users, and services
- **Single sign-on** lets you manage user authentication by enabling single sign-on (SSO)
- **Billing** controls where you can review the charges to your account, change your credit card information, and update your company's tax address

The Personal Profile details include:

- **Your profile** including your name and your email address
- **Change password** controls that allow you to update your current login password
- **Two-factor authentication** information where you can manage the multi-factor authentication controls for your personal login
- **Personal API tokens** where you can create and delete your personal API tokens you need to control access to various services and resources within your Fastly account
- **Appearance** controls that allow you to change the appearance of the web interface, such as enabling dark mode.

About the Billing controls

Selecting **Billing** from the account menu displays billing-related account details for your login, including:

- **Invoice history** with a complete history of the monthly bills for your Fastly account and their payment statuses
- **Upgrade account** where you can view your current account type and upgrade it to a paid account if you're currently using a free developer trial
- **Credit card** where you can view and edit your credit card information
- **Tax address** where you can update your tax or billing address for your account

What's next

Dig deeper into details about all areas of the web interface controls before you move on to using them to work with services.

About the Compute page

Last updated: 2021-10-25

https://docs.fastly.com/en/guides/about-the-compute-page

The Compute page serves a web-based alternative to using the Fastly CLI when working with your Compute services using the Compute@Edge platform. The page allows you to define exactly how each instance of your Compute services should behave and interact with its data sources. The Compute page appears automatically for logged in users with the appropriate access permissions.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

Before you begin

Be sure you know how to access the web interface controls before learning about each of the pages you'll encounter there.

About the Compute page

From the Compute page, you can access the **Service summary** tab, where you can review general service details including the event log, duplicate (clone) service versions so you can edit them, and purge content.

The page also provides access to the **Service configuration** tab, with controls for managing service configurations. Specifically you can:

- manage the domains used to route requests to a Compute service
- manage the hosts used as backends for a site and how they should be accessed
- manage the health checks that monitor backend hosts
- specify how logging should be performed and where server logs should be sent
- upload your Compute@Edge compatible Wasm packages once you've built them with the Fastly CLI

Finally, the Compute page also provides a **Learn** tab with specific resources for Fastly's edge serverless compute platform, including:

- details that help you select and learn about languages that can be used on Fastly's edge serverless computing platform
- steps for using the Fastly CLI and a link to the Fastly CLI reference documentation
- information about tools like Terraform to help you automate your workflows
- a way to experiment with code in the Fastly Fiddle so you can create ephemeral Fastly services without logging into a Fastly account and learn how your requests and responses are handled by Fastly

What's next

Dig deeper into details about all areas of the web interface controls before you move on to using them to work with services.

About the Deliver page

Last updated: 2021-10-25

https://docs.fastly.com/en/guides/about-the-deliver-page

The Deliver page allows you to define exactly how each instance of your content delivery network (CDN) cache should behave and control content from data sources. The Deliver page appears automatically for logged in users with the appropriate access permissions.

Before you begin

Be sure you know how to access the web interface controls before learning about each of the pages you'll encounter there.

About the Deliver page

The Deliver page is the key area of the web interface where you work with your CDN services and their configuration settings. From here you can access the **Service summary** tab, where you can review general service details including the event log, duplicate (clone) service versions so you can edit them, and purge content.

The Deliver page also provides access to the **Service configuration** tab, with the main controls for editing your service configurations. Specifically you can edit:

- the domains used to route requests to a CDN service
- the hosts used as backends for a site and how they should be accessed
- the health checks that monitor backend hosts
- various request and cache settings, headers, and responses that control how Fastly caches and serves content for a CDN service
- how logging should be performed and where server logs should be sent
- custom Varnish configuration language (VCL) files
- how conditions are mapped and used for a service at various times (e.g., during request processing, when Fastly receives a backend response, or just before an object is potentially cached)

With the appropriate permissions, you can activate configuration changes immediately and roll back those changes just as quickly should they not have the intended effect. The Deliver page also allows you to compare differences between two configuration versions.

What's next

Dig deeper into details about all areas of the web interface controls before you move on to using them to work with services.

About the Home page

Last updated: 2021-10-25

https://docs.fastly.com/en/guides/about-the-home-page

Before you begin

Be sure you know how to access the web interface controls before learning about each of the pages you'll encounter there.

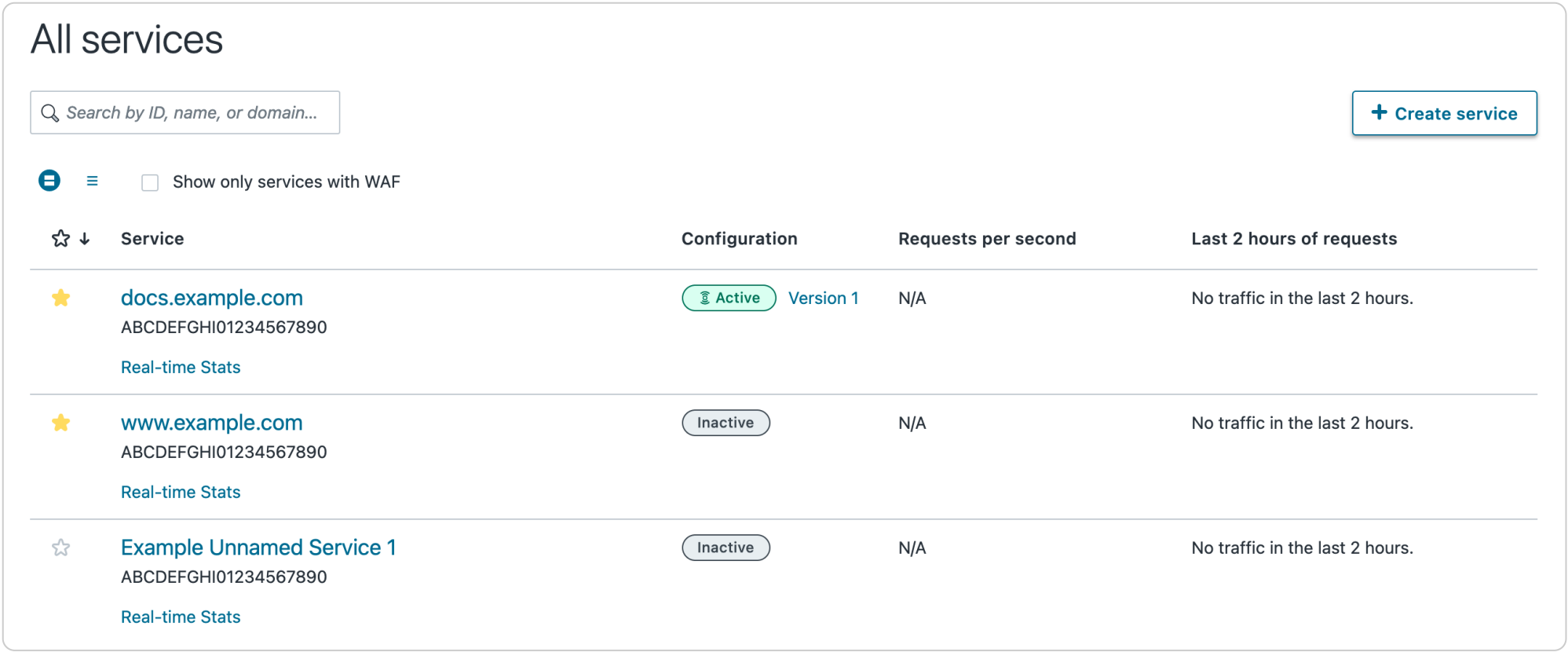
About the Home page

The Home page displays a summary of all your Deliver and Compute services, sorted by requests per second. It appears automatically when users with the appropriate access permissions log in to the Fastly web interface. You can access it by clicking **Home** next to the stopwatch icon once you've logged into your account.

The summary table of all services allows you to:

- access the real-time and historic stats information for a particular service by clicking the related stats link next to the service name
- understand the type of each service you control (Deliver and Compute)
- understand whether or not a particular service is active or inactive (and, if active, which version is active)
- open the current configuration settings for a service by clicking the active version number in the Configuration column
- view the number of requests received per second for a service in the Requests per second column

- view small graphs of the total number of requests received for a service over a two hour period in the Last 2 hours of requests column



You can also search for a specific service associated with a domain by typing the domain name in the **Search by domain** field. The domain name you enter must be an exact match to find the desired service.

What's next

Dig deeper into details about all areas of the web interface controls before you move on to using them to work with services.

About the Origin Inspector stats

Last updated: 2022-03-16

https://docs.fastly.com/en/guides/about-the-origin-inspector-stats

Fastly's Origin Inspector builds on our existing Stats interface by providing real-time and historic visibility into detailed response data for traffic from your origin servers to Fastly.

IMPORTANT

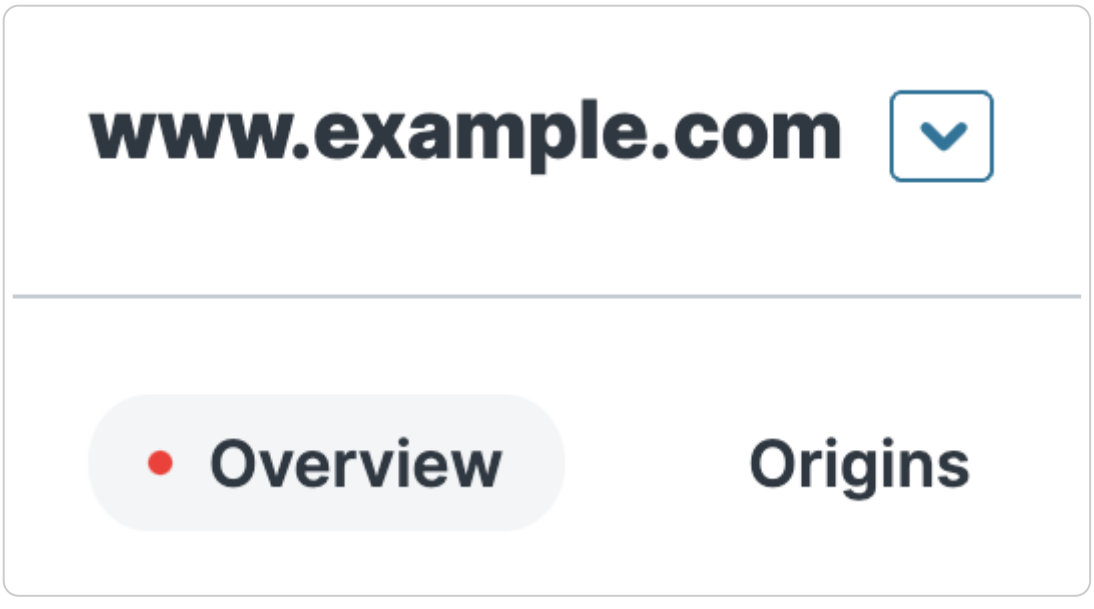
This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

Before you begin

Be sure you know how to access the web interface controls and pay specific attention to basic information about the Stats interface before learning about the Origin Inspector specifics you'll encounter here.

About the Origin Inspector Stats page

Once Origin Inspector is enabled for your account, the Stats page gains additional navigation controls that appear below the service name and ID details.



About the Overview tab

For both real-time and historical stats, an **Overview** tab appears that provides stats details about your entire service as described in our guide about the Stats page.

About the Origins tab summary dashboards

For each origin associated with your service, the Origins tab for both real-time and historic stats displays a small summary dashboard providing immediate visibility into origin egress metrics.

Responses	Bandwidth	Status 1XX	Status 2XX	Status 3XX	Status 4XX	Status 5XX
1	507.7 Kbps	0	1	0	0	0

Specifically, the summary dashboard describes:

- the number of responses received by Fastly from each of your origin servers.
- the bandwidth received from origin. The origin bandwidth calculation formula is `response header bytes + response body bytes`.
- a count of each of the 1XX, 2XX, 3XX, 4XX, and 5XX HTTP status codes returned from your origin, grouped by type.

About the Origins tab graphs

Below the summary dashboard data, for both real-time and historic stats, two graphs appear for each origin associated with your service. These graphs provide a visual representation of dashboard information over time. Specifically:

- a **Response Status Codes** graph detailing the number of responses from your origin.
- an **Origin Bandwidth** graph detailing the amount of bandwidth from your origin.

Viewing more origin details graphs

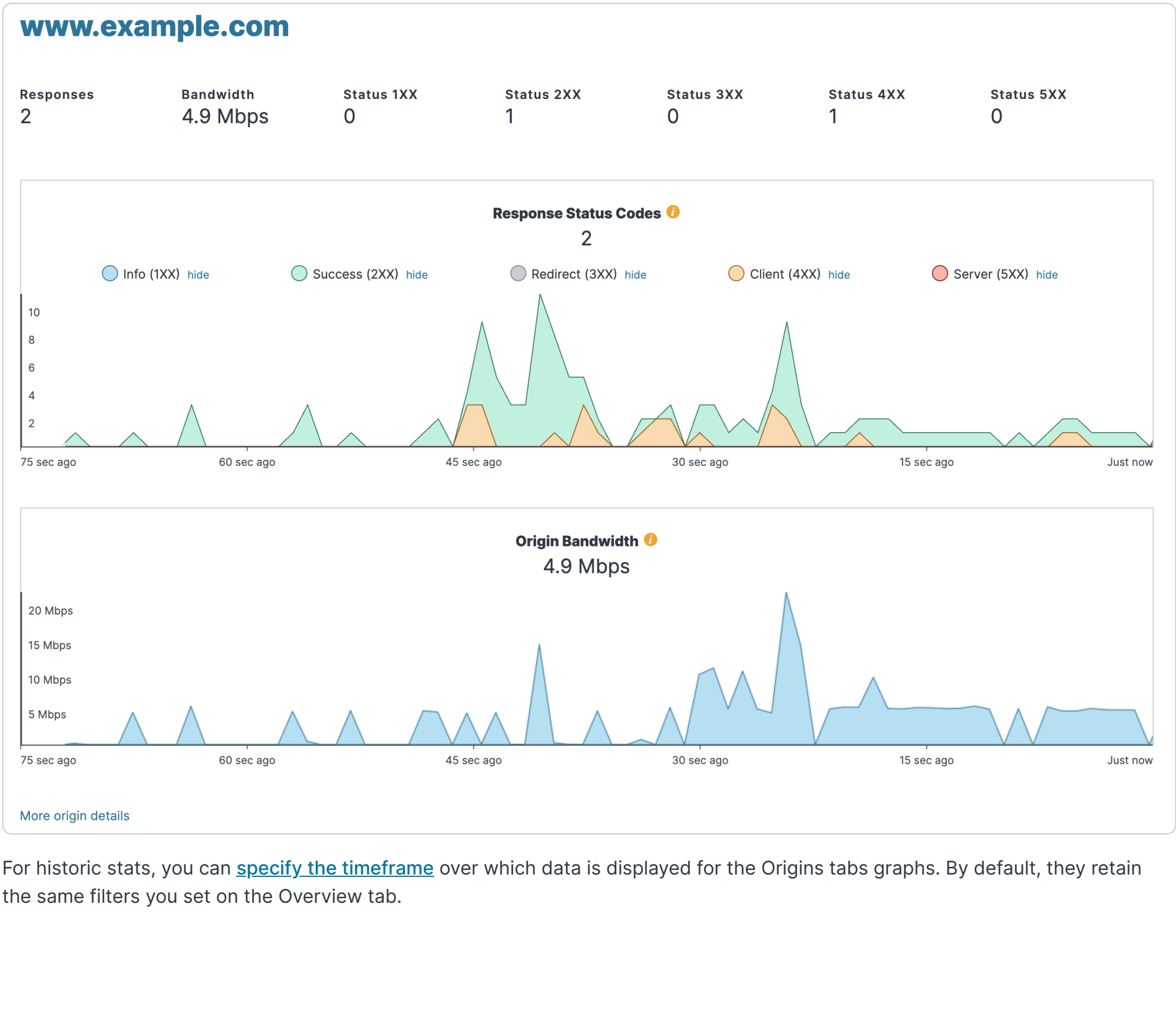
Below the Origins tab graphs, you'll notice a link to **More origin details**. Clicking this link displays the following additional graphs about each origin in between the **Response Status Codes** and **Origin Bandwidth** graphs:

- **Origin Latency** metrics show the distribution of origin latency times, indicating how quickly your origin processes requests when responding to Fastly.
- **Status 5XX Details** metrics show the breakdown between the number of HTTP Status 500 (Internal Server Error), 501 (Not Implemented), 502 (Bad Gateway), 503 (Service Unavailable), 504 (Gateway Timeout), and 505 (HTTP Version Not Supported) requests.
- **Status 4XX Details** metrics show the breakdown between the number of HTTP Status 400 (Bad Request), 401 (Unauthorized), 403 (Forbidden), 404 (Not Found), 416 (Range Not Satisfiable), and 429 (Too Many Requests) requests.
- **Status 3XX Details** metrics show the breakdown between the number of HTTP Status 301 (Moved Permanently), 302 (Found), and 304 (Not Modified) requests.

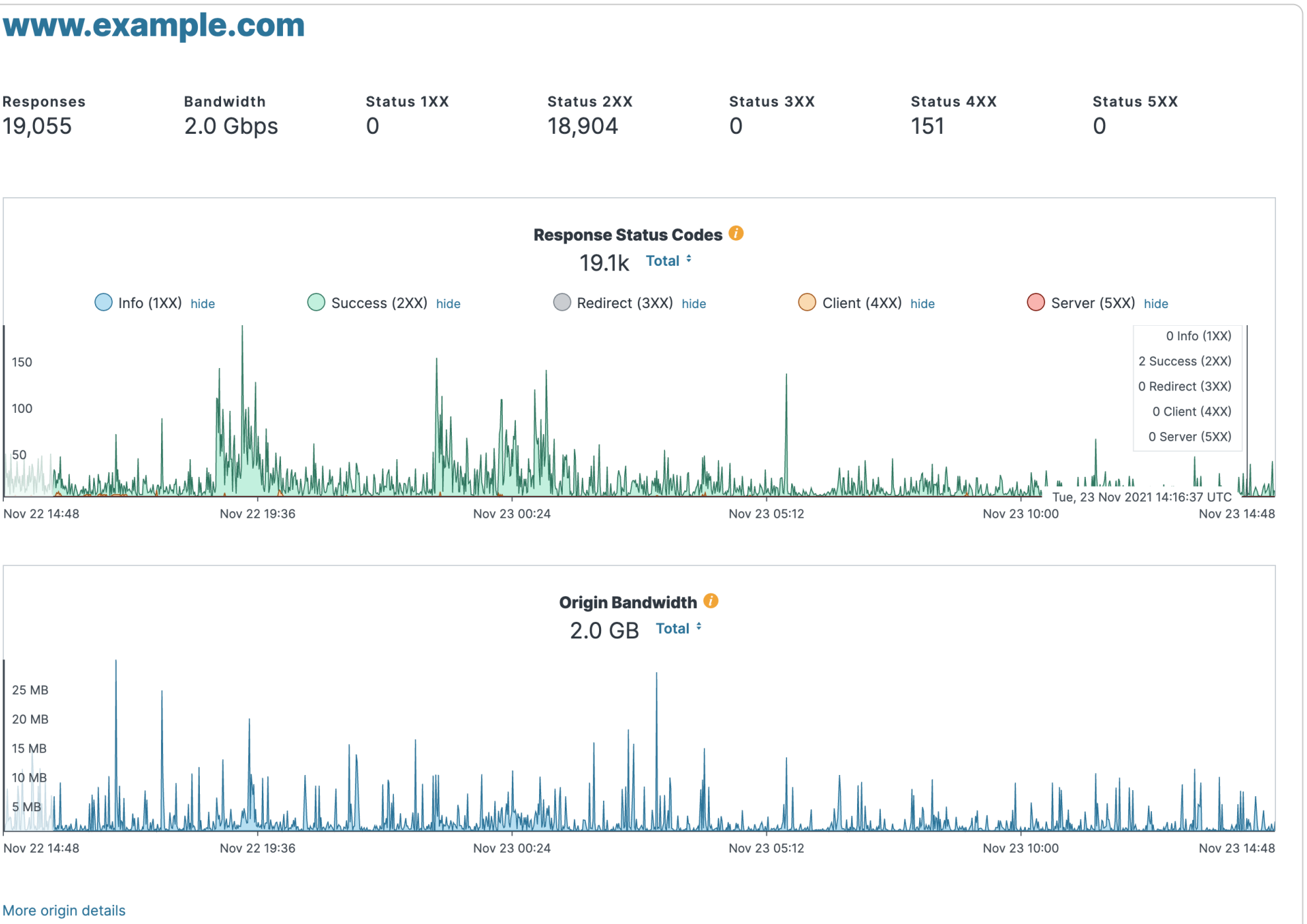
- **Status 2XX Details** metrics show the breakdown between the number of HTTP Status 200 (Success), 204 (No Content), and 206 (Partial Content) requests.

How often graph data is updated

For real-time stats, the Origins tab graphs display data for the last 75 seconds and update continuously as more data is received by each origin listed.



For historic stats, you can specify the timeframe over which data is displayed for the Origins tabs graphs. By default, they retain the same filters you set on the Overview tab.



What's next

Be sure to familiarize yourself with the fundamentals of working with stats graphs before continuing. Dig deeper into details about all areas of the web interface controls before you move on to using them to work with services.

About the Secure page

Last updated: 2022-02-02

https://docs.fastly.com/en/guides/about-the-secure-page

The Secure page gives you access to the different security products Fastly has to offer. Note that you must contact sales@fastly.com to have access to this page and these offerings.

Before you begin

Be sure you know how to access the web interface controls before learning about each of the pages you'll encounter there.

About the Secure page

From the Secure page, you can access the:

- **TLS Management** tab, where you can add HTTPS to your domains for additional privacy and data security for services. You can either upload your own TLS certificates or have Fastly manage this for you.
- **Edge Rate Limiting** tab, where you can implement a rate limiting policy to control the rate of requests sent to your origin servers.

> **IMPORTANT**

This feature is part of a limited availability release. For more information, see our product and feature lifecycle descriptions.

- **DDoS mitigation** tab, where you can contact us for help with a current threat or to protect a specific event. Fastly's high-bandwidth globally distributed network was built with "always-on" DDoS mitigation designed to absorb DDoS attacks. If you currently have a sustained DDoS threat risk or a short term or seasonal event to protect, however, you can also purchase a DDoS Protection and Mitigation Service that provides additional Fastly resources to assist you with mitigating the service and financial impacts of DDoS and related attacks.
- **Next-Gen WAF** tab, where you can contact us for information on the Fastly Next-Gen WAF (powered by Signal Sciences). You can use a WAF to protect your applications from malicious attacks designed to compromise web servers.
- **Fastly WAF** tab, which lists the services where a version of the web application firewall (WAF) is enabled. You can use a WAF to protect your applications from malicious attacks designed to compromise web servers. For more information about our WAF offerings, see our guide on Fastly Next-Gen WAF (powered by Signal Sciences).

Security products note

No security product, such as a WAF or DDoS mitigation product, including those security services offered by Fastly, will detect or prevent all possible attacks or threats. As a subscriber, you should maintain appropriate security controls on all web applications and origins. The use of Fastly's security products do not relieve you of this obligation. As a subscriber, you should test and validate the effectiveness of Fastly's security services to the extent possible prior to deploying these services in production, continuously monitor their performance, and adjust these services as appropriate to address changes in your web applications, origin services, and configurations of the other aspects of your Fastly services.

What's next

Dig deeper into details about all areas of the web interface controls before you move on to using them to work with services.

About the stats for Compute services

Last updated: 2021-12-01

https://docs.fastly.com/en/guides/about-the-stats-for-compute-services

The Stats page for Compute services allows you to monitor your real-time analytics and view your historical statistics for your services on the web interface.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

Before you begin

Be sure you know how to access the web interface controls and pay specific attention to basic information about the Stats interface before learning about the Compute service specifics you'll encounter here.

Viewing the Real-time stats

The Real-time stats graphs allow you to separately monitor metrics for your services in real time, as they operate on a second-by-second basis. In addition to a menu allowing you to select the specific data center from which to view data (it defaults to data from all data centers), the top of the dashboard includes the following real-time cache activity:

- **Requests:** the number of requests Fastly receives for your site per second.
- **Errors:** the number of errors from your Wasm service, including exceeding limit errors.
- **Bits transferred (client):** the total number of header and body bits sent and received by Fastly for your Wasm service.
- **Bits transferred (backend):** the total number of header and body bits sent and received by Fastly to and from your Wasm service's backend.

- **Bandwidth:** the bandwidth served from Fastly's servers to your website's visitors.
- **Memory Resource Limits Exceeded:** the number of times that guests have exceeded their stack, heap, or global limits.
- **RAM Usage:** the amount of RAM used for your site by Fastly, including a maximum indicating the highest usage within the timeframe and a static cap for usage.
- **Total Wall Clock Time:** the total, actual amount of time used to process your requests, including active CPU time.
- **CPU Time:** the amount of active CPU time used to process your requests.
- **Logs:** the number of logs sent to your endpoints from Fastly.
- **Log Bandwidth:** the total bandwidth size of the logs sent to your endpoints from Fastly.

If you've just created your service, you might see a message saying there's nothing to see yet. This might be because:

- **Not enough data is going to your site.** If this is the case, visit the site yourself to trigger some traffic.
- **You've made a CNAME change.** If this is the case, it could take from a few minutes to hours for the change to propagate your DNS servers. See how to edit your DNS record to point to Fastly for more information.

Once you start seeing real-time cache activity, you also can interact with your stats graphs.

Viewing the Historic stats

The Historic stats graphs provide a visual interface to our stats API for a selected Fastly service. You can also display historical metrics aggregated across all your Fastly services by clicking **All services**. The graphs display metrics derived from your site's statistical information. If you've just created your service, you might see a message saying there's nothing to see yet.

The displayed performance metrics help you optimize your Wasm service and analyze your service's traffic over time. These metrics include the following:

- **Requests** metrics show you the total number of requests over time that were received for your site by Fastly.
- **Response Status Codes** metrics show you the number of HTTP Info (1xx), Success (2xx), Redirect (3xx), Client Errors (4xx), and Server Errors (5xx) statuses served for your Wasm service using Fastly.
- **Bytes transferred to client** metrics show you the relative values of bytes transferred by Fastly for your Wasm service when serving the body and header portion of the Wasm requests.
- **Bytes transferred from client** metrics show you the total relative values of bytes transferred to Fastly for your Wasm service when receiving the body and header portion of the Wasm requests.
- **Bytes transferred to backend** metrics show you the relative values of bytes transferred to Fastly from your service's backend when serving the body and header portion of the Wasm requests.
- **Bytes transferred from backend** metrics show you the relative values of bytes transferred to Fastly from your service's backend when serving the body and header portion of the Wasm requests.
- **Bandwidth** metrics show you the amount of bandwidth served for your site by Fastly.
- **Memory Resource Limits Exceeded** metrics show you the number of times that guests have exceeded their stack, heap, or global limits.
- **RAM Usage** metrics show you the amount of RAM used for your site by Fastly, including a maximum indicating the highest usage within this timeframe and a static cap for usage.
- **Total Wall Clock Time** metrics show you the total, actual amount of time used to process your requests, including active CPU time.
- **CPU Time** metrics show you amount of active CPU time used to process your requests.
- **Logs** metrics show the number of logs sent to your endpoints from Fastly.
- **Log Bandwidth** metrics show the total bandwidth size of the logs sent to your endpoints from Fastly.

What's next

Once you start to see your performance metrics, you also can interact with your stats graphs.

About the stats for Deliver services

Last updated: 2022-03-29

https://docs.fastly.com/en/guides/about-the-stats-for-deliver-services

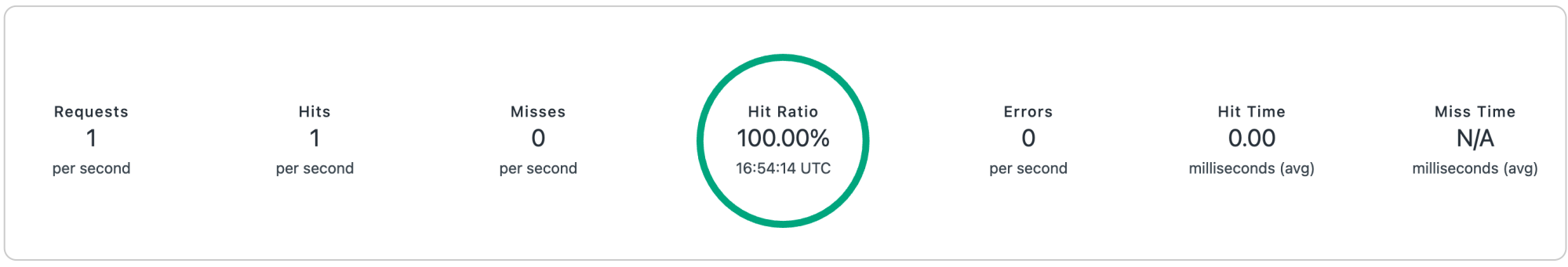
The Stats page for Deliver services allows you to monitor your real-time analytics and view your historical caching statistics for your services on the web interface.

Before you begin

Be sure you know how to access the web interface controls and pay specific attention to basic information about the Stats interface before learning about the Deliver service specifics you'll encounter here.

Viewing the Real-time stats

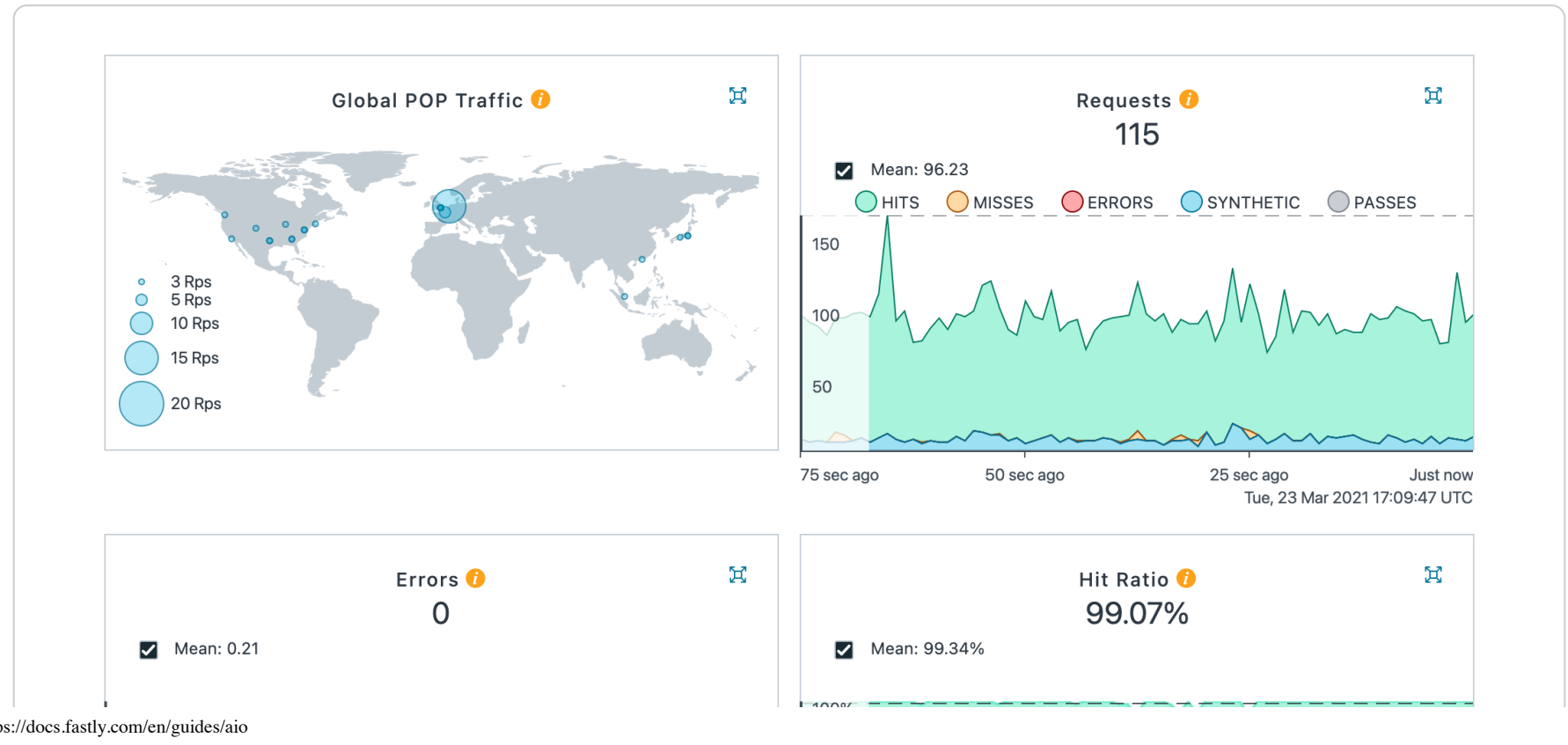
The Real-time stats graphs allow you to separately monitor caching for your services in real time, as they operate on a second-by-second basis.

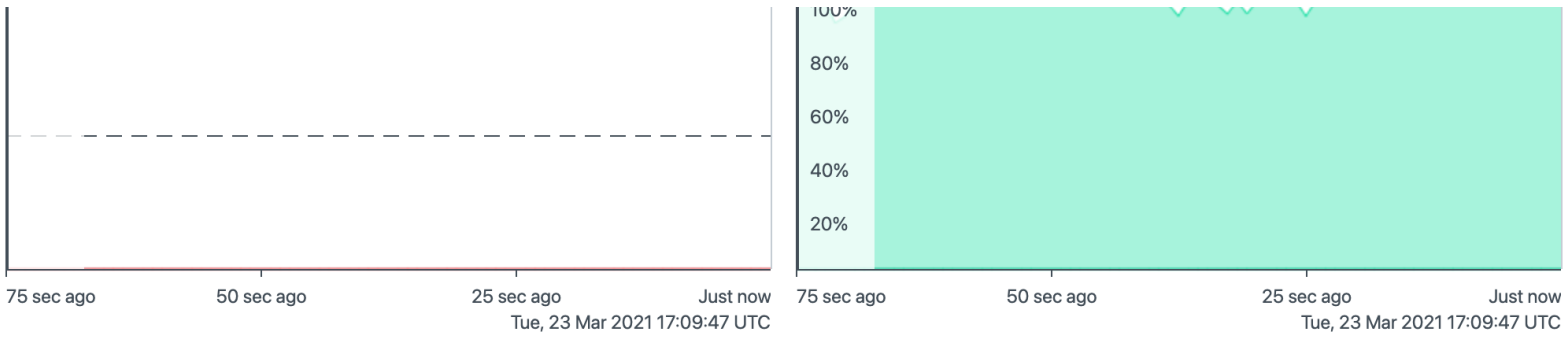


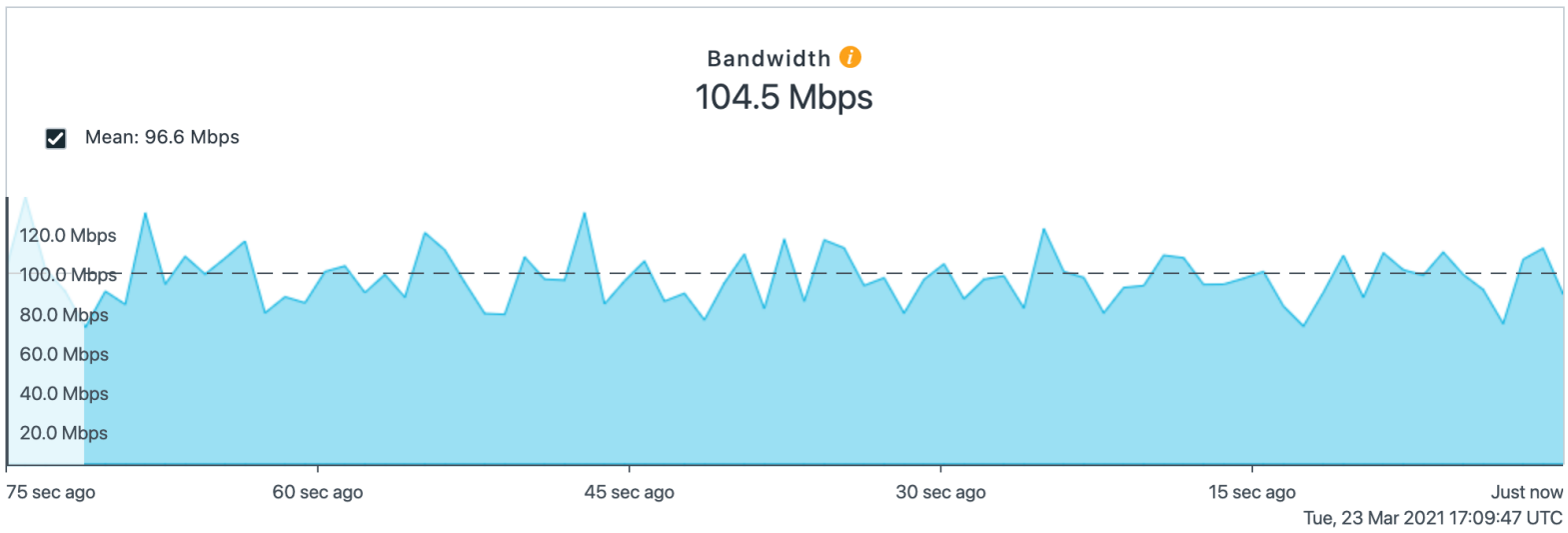
In addition to a menu allowing you to select the specific data center from which to view data (it defaults to data from all data centers), the top of the dashboard includes the following real-time cache activity:

- **Requests:** the number of requests Fastly receives for your site per second.
- **Hits:** the number of times requested data is found in cache and does not require making a fetch your origin server.
- **Misses:** the number of times requested data is not found in cache and has to be requested from your origin server.
- **Hit Ratio:** the percentage of content being accessed that is currently cached by Fastly. Also known as Fastly's definition of cache hit ratio. While there are many ways to calculate cache hit ratio, at Fastly this is defined as the proportion of cache hits (hits) to all cacheable content (hits + misses). If shielding is enabled, it can cause the hit ratio to be inaccurate.
- **Errors:** the number of error responses per second that occur as Fastly receives requests for your site.
- **Hit Time:** the average amount of time (in milliseconds) spent processing cache hits.
- **Miss Time:** the average amount of time (in milliseconds) spent processing cache misses.

Below the real-time cache activity summary data, several graphs appear:




100%
80%
60%
40%
20%
75 sec ago
50 sec ago
25 sec ago
Just now
Tue, 23 Mar 2021 17:09:47 UTC


Bandwidth
104.5 Mbps
Mean: 96.6 Mbps
120.0 Mbps
100.0 Mbps
80.0 Mbps
60.0 Mbps
40.0 Mbps
20.0 Mbps
75 sec ago
60 sec ago
45 sec ago
30 sec ago
15 sec ago
Just now
Tue, 23 Mar 2021 17:09:47 UTC

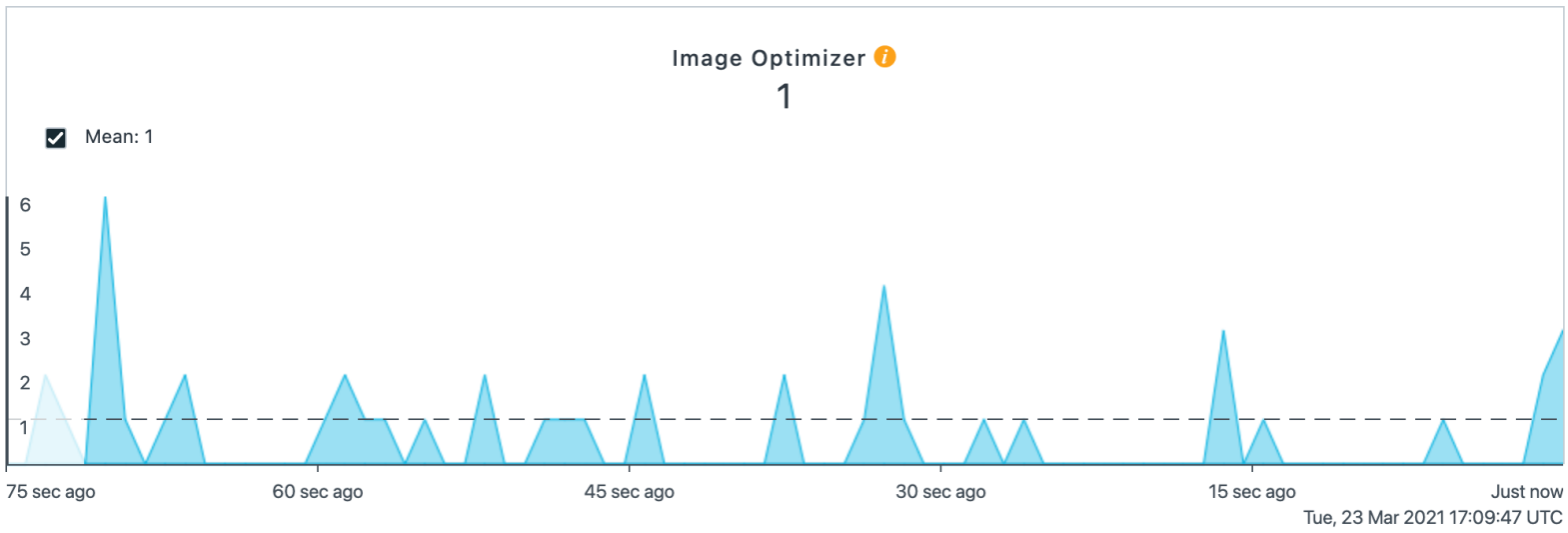


Image Optimizer
1
Mean: 1
6
5
4
3
2
1
75 sec ago
60 sec ago
45 sec ago
30 sec ago
15 sec ago
Just now
Tue, 23 Mar 2021 17:09:47 UTC

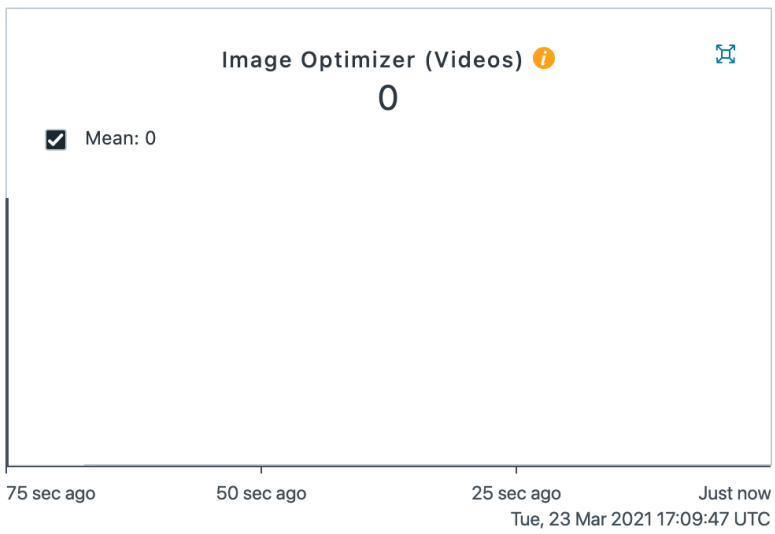


Image Optimizer (Videos)
0
Mean: 0
75 sec ago
50 sec ago
25 sec ago
Just now
Tue, 23 Mar 2021 17:09:47 UTC

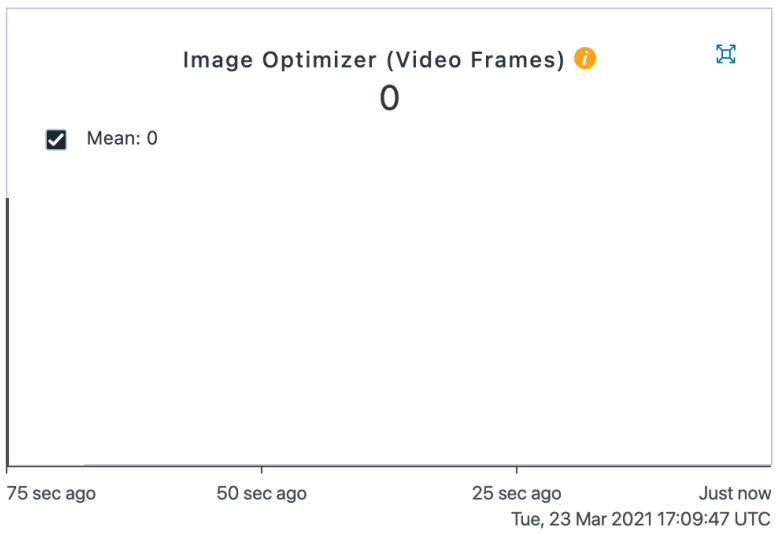
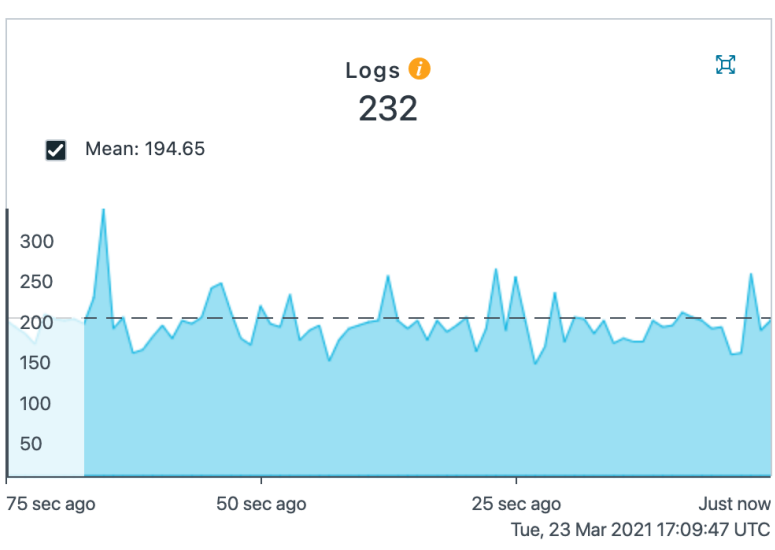
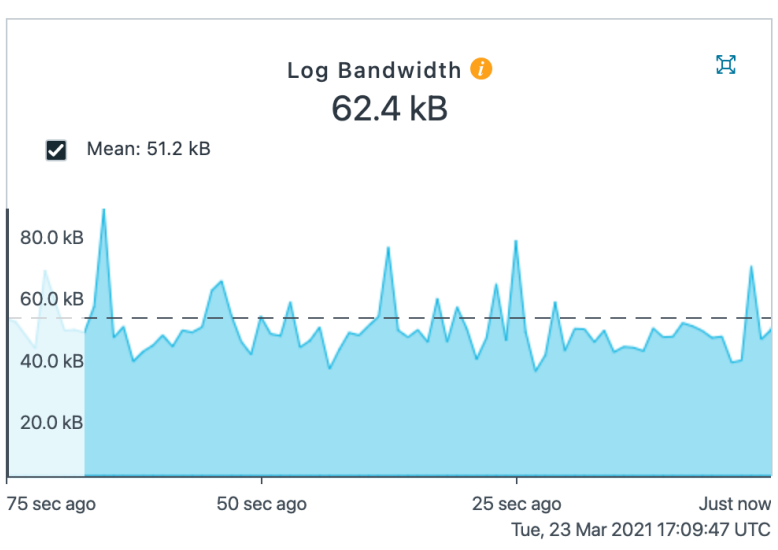
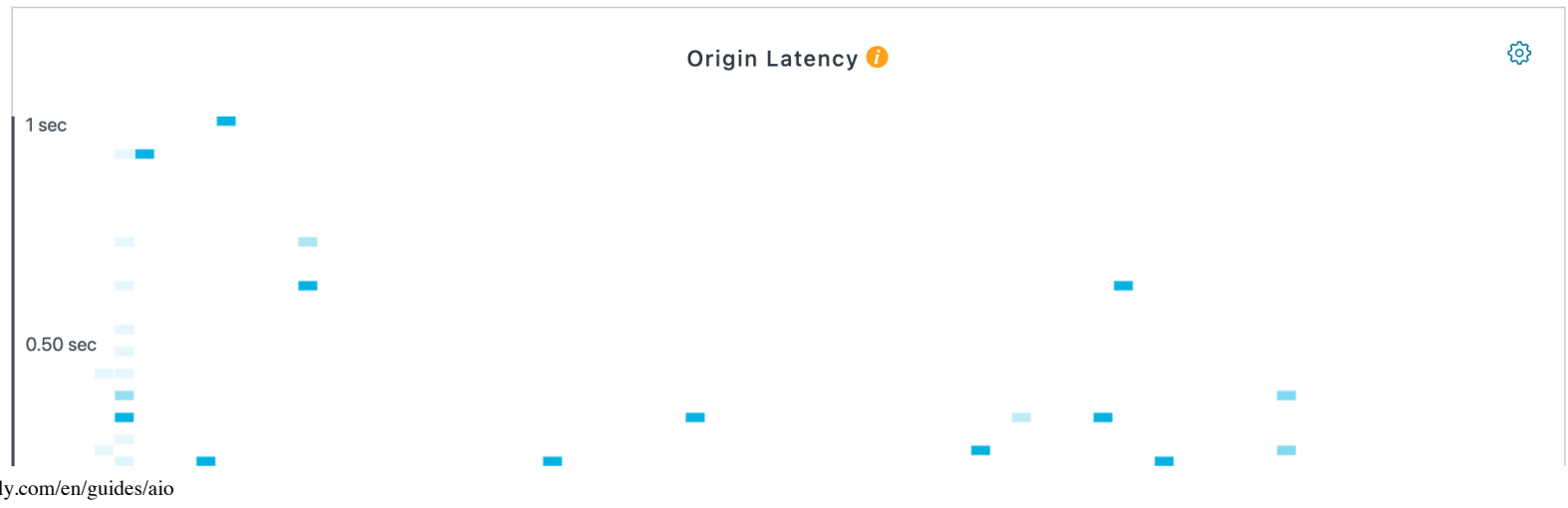


Image Optimizer (Video Frames)
0
Mean: 0
75 sec ago
50 sec ago
25 sec ago
Just now
Tue, 23 Mar 2021 17:09:47 UTC


Logs
232
Mean: 194.65
300
250
200
150
100
50
75 sec ago
50 sec ago
25 sec ago
Just now
Tue, 23 Mar 2021 17:09:47 UTC


Log Bandwidth
62.4 kB
Mean: 51.2 kB
80.0 kB
60.0 kB
40.0 kB
20.0 kB
75 sec ago
50 sec ago
25 sec ago
Just now
Tue, 23 Mar 2021 17:09:47 UTC


Origin Latency
1 sec
0.50 sec

The graphed cache activity includes:

- **Global POP Traffic:** a heat map displaying global POP traffic through all POPs for your service.
- **Requests:** a graph displaying the total number of requests received for your site by Fastly over time.
- **Errors:** a graph displaying the number of error requests that occurred over time.
- **Hit Ratio:** a graph displaying the percentage of content being accessed that is currently cached by Fastly over time.
- **Bandwidth:** a graph displaying the bandwidth served from Fastly's servers to your website's visitors.
- **Image Optimizer:** when enabled, a graph displaying the number of responses that came from the Fastly Image Optimizer service over time.
- **Image Optimizer (Videos):** when enabled, a graph displaying the number of videos responses that came from the Fastly Image Optimizer.
- **Image Optimizer (Video Frames):** when enabled, a graph displaying the number of video frames that came from the Fastly Image Optimizer. A video frame is an individual image in a sequence of images that make up a video.
- **Logs:** a graph displaying the number of logs sent to your endpoints from Fastly.
- **Log Bandwidth:** a graph displaying the total bandwidth size of the logs sent to your endpoints from Fastly.
- **Origin Latency:** a histogram displaying the average amount of time to first byte (measured in milliseconds) on a cache miss or pass. High origin latency means that your backends are taking longer to process requests.

If you've just created your service, you might see a message saying there's nothing to see yet. This might be because:

- **Not enough data is going to your site.** If this is the case, visit the site yourself to trigger some traffic.
- **You've made a CNAME change.** If this is the case, it could take from a few minutes to hours for the change to propagate your DNS servers. See how to edit your DNS record to point to Fastly for more information.

Once you start seeing real-time cache activity, you also can interact with your stats graphs.

Viewing the Historic stats

The Historic stats graphs provide a visual interface to our stats API for a selected Fastly service. You can also display historical metrics aggregated across all your Fastly services by clicking **All services**. The graphs display metrics derived from your site's statistical information. If you've just created your service, you might see a message saying there's nothing to see yet.

The displayed caching and performance metrics help you optimize your website's speed. These metrics include the following:

- **Hit Ratio** metrics tell you how well you are caching content using Fastly. This metric represents the proportion of cache hits (hits) to all cacheable content (hits + misses). Increasing your hit ratio improves the overall performance benefit of using Fastly.
- **Cache Coverage** metrics show how much of your site you are caching with Fastly. This metric represents the ratio of cacheable requests (i.e., non "pass" requests) to total requests. Improving your cache coverage by reducing passes can improve site performance and reduce load on your origin servers.
- **Caching Overview** metrics compare Cache Hits, Cache Misses, Synthetic Responses (in VCL edge responses), and Passes (or requests that cannot be cached according to your configuration).

The traffic metrics analyze your website's traffic as it evolves over time. These metrics include the following:

- **Requests** metrics show you the total number of requests that were received for your site by Fastly.
- **Bytes Transferred** metrics show you the total number of bytes transferred by Fastly for your service.
- **Header & Body Bytes Transferred** metrics show you the relative values of bytes transferred when serving the body portion of HTTP requests and the header portion of the requests.
- **Miss Latency** metrics show the distribution of only the miss latency times for your origin.

- **Error Ratio** metrics show you the ratio of error responses (4xx and 5XX status code errors) compared to the total number of requests for your site. This metric allows you to filter types of error responses and quickly identify error spikes at given times.
- **HTTP Info, Success, & Redirects** metrics show the number of HTTP Info (1XX), Success (2XX), and Redirect (3XX) statuses served for your site using Fastly.
- **Status 3XX Details** metrics show the breakdown between the number of HTTP Status 301s, 302s, 304s, and other 3XX requests.
- **HTTP Client and Server Errors** metrics show the number of HTTP Client Errors (4XX), and Server Errors (5XX) served for your site by Fastly.
- **HTTP versions** metrics show the number of requests using HTTP/1.1, HTTP/2, and HTTP/3 (QUIC) protocols.
- **TLS versions** metrics show the number of requests using TLS 1.0, TLS 1.1, TLS 1.2, and TLS 1.3 protocols.
- **Logs** metrics show the number of logs sent to your endpoints from Fastly.
- **Log Bandwidth** metrics show the total bandwidth size of the logs sent to your endpoints from Fastly.
- When enabled, **Image Optimization Requests** metrics show you the number of responses that came from the Fastly Image Optimization service.

What's next

Once you start to see your caching and performance metrics, you also can interact with your stats graphs.

About the Stats page

Last updated: 2022-03-16

https://docs.fastly.com/en/guides/about-the-stats-page

The Stats page allows you to monitor your real-time analytics and view your historical caching statistics for your services on the web interface.

Before you begin

Be sure you know how to access the web interface controls before learning about the details you'll encounter here.

NOTE

This page describes Stats details displayed when you haven't purchased any additional Fastly products. If you've purchased Origin Inspector, be sure to read our guide about the Origin Inspector stats in addition to this one.

About the Stats page

On the Stats page you can access:

- Real-time stats information that allows you to monitor cache activity for your services
- Historic stats information that displays your historical stats derived from your site's statistical information

The exact graphs displayed depend on your access permissions and the type of service you create, as well as other products and features you may have purchased. Different stats graphs and navigation appear for:

- Deliver services
- Compute services
- Origin Inspector

The data on the Stats page may also appear grayed out or blank to some users, with no information displayed in the controls, when a service hasn't yet received enough requests for Fastly to display meaningful information about it.

Working with stats graphs

You can interact with and control your Real-time and Historic stats graphs as follows.

Limiting data viewed to specific data centers

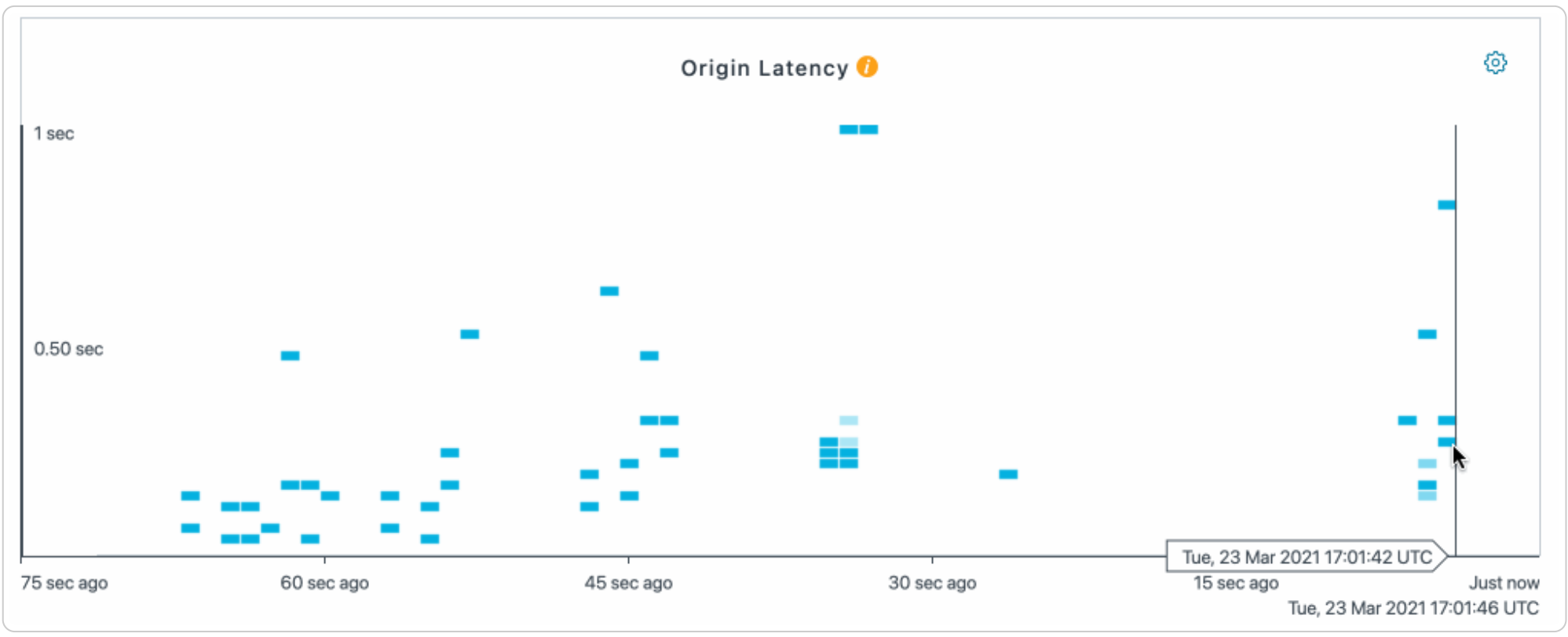
By default, Real-time graphs display data from all data centers. To view data from a single, specific data center, select it from the **All datacenters** menu.

Disabling smooth scrolling

The Real-time graphs update continuously. Leaving the graphs open for long periods of time, however, can occasionally lead to higher CPU utilization. To improve performance, you can deselect the **Smooth scrolling** checkbox. The graphing animations may not be as smooth when this checkbox is deselected.

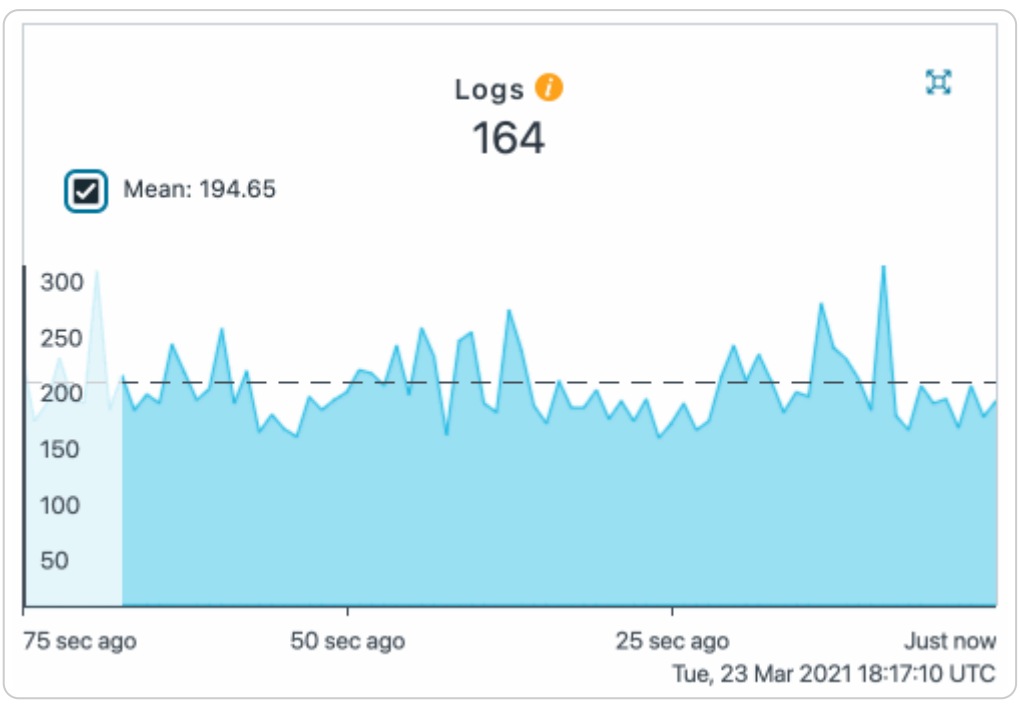
Viewing the real-time stats timestamp indicator

Hovering the cursor over any part of a graph displays a timestamp indicator that updates itself as you move the mouse.



Hiding and displaying the mean value

A dashed line indicating the mean value of the graph's data appears on some graphs. To hide the mean line, deselect the **Mean** checkbox.



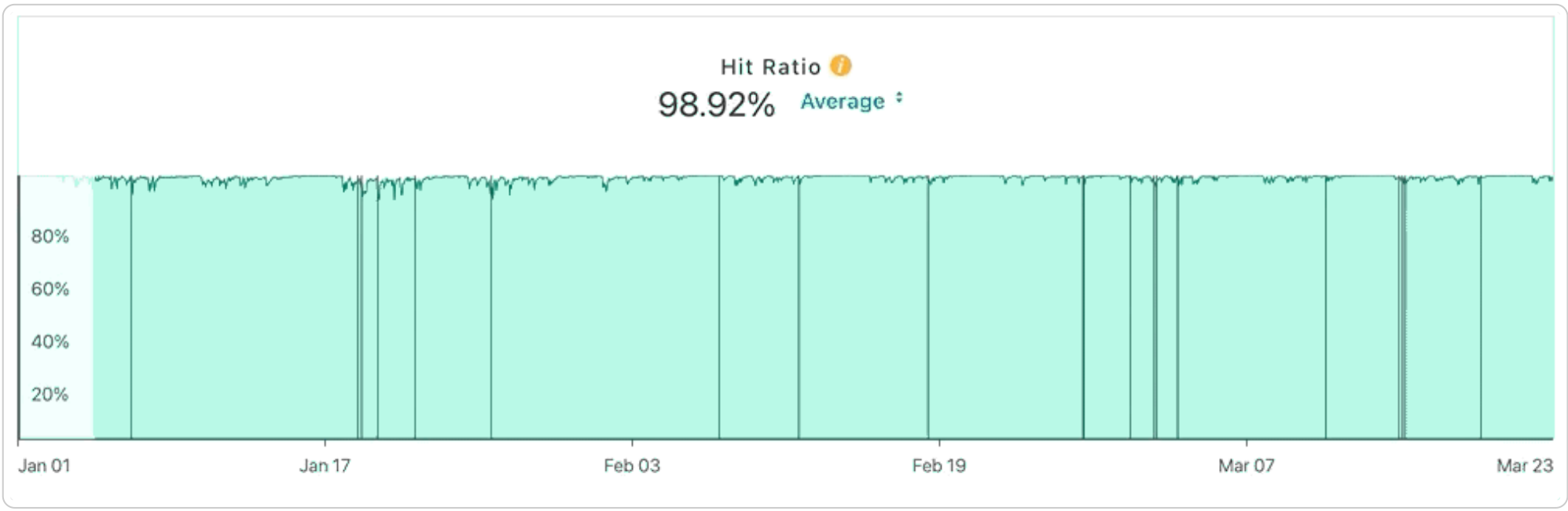
Expanding and minimizing graph views

You can expand and minimize the view of some of the graphs using the quadruple arrow button in the right-hand corner of the graph to display an expanded view of the graph or special features it offers. Specifically:

- the Global POP Traffic heat map displays a larger view of the graph as well as the traffic in each POP region, with continuously updating data on the POP's current requests per second, the request error ratio, and the bandwidth going through that POP.
- the Requests, Errors, and Hit Ratio graphs expand to larger versions of themselves along with the already expanded versions of the Bandwidth and Origin Latency graphs.
- the Origin Latency graph specifically includes a small gear icon in the upper right corner that allows you to change the interval limit displayed by the graph from the default 15 second interval to a different time frame.

Viewing service version activation

Service version activations appear as vertical lines on the Historic graphs. Hovering your cursor over any line displays the version's number and its activation timestamp.



IMPORTANT

You cannot retrieve minutely historical statistics data older than 35 days from the current date. Contact support@fastly.com to discuss your minutely data needs.

Controlling the historic stats date displayed

You can control how you view the historic stats date ranges. For all displayed graphs, you can choose:

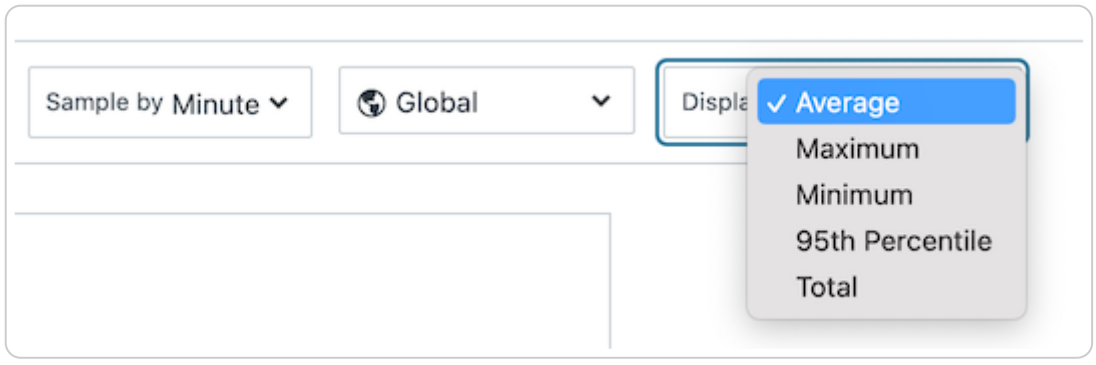
- the exact local date and time range of the graphed data
- how often to sample the data displayed
- whether to view global data for the graphs or only data from a specific region
- how to display the statistical values

Keep in mind, however, that data won't appear yet for time periods that haven't ended. Data aggregated per day is collected based on UTC days and each day's data becomes available around 2am the following day. Data aggregated by hour becomes available approximately 15 minutes after the end of each hour. Data aggregated by minute usually becomes available two minutes after the end of the minute, but can take up to 15 minutes. If your use case requires data closer to real-time, consider using the real-time stats instead.

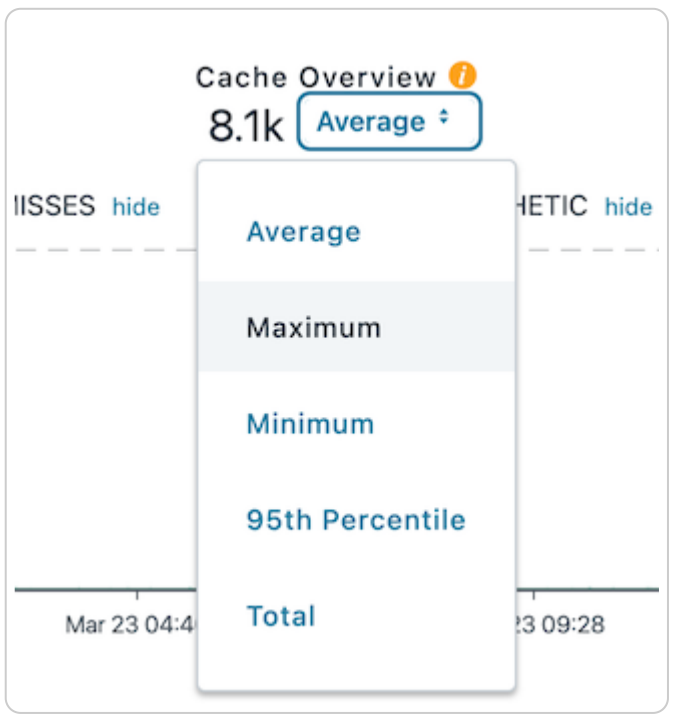
Changing the stats displayed in historical graphs

You can change the statistics displayed in any historical graph to display an average, a 95th percentile, a minimum, a maximum, or a total. When set to average, the graph displays the average (mean) as a dashed line.

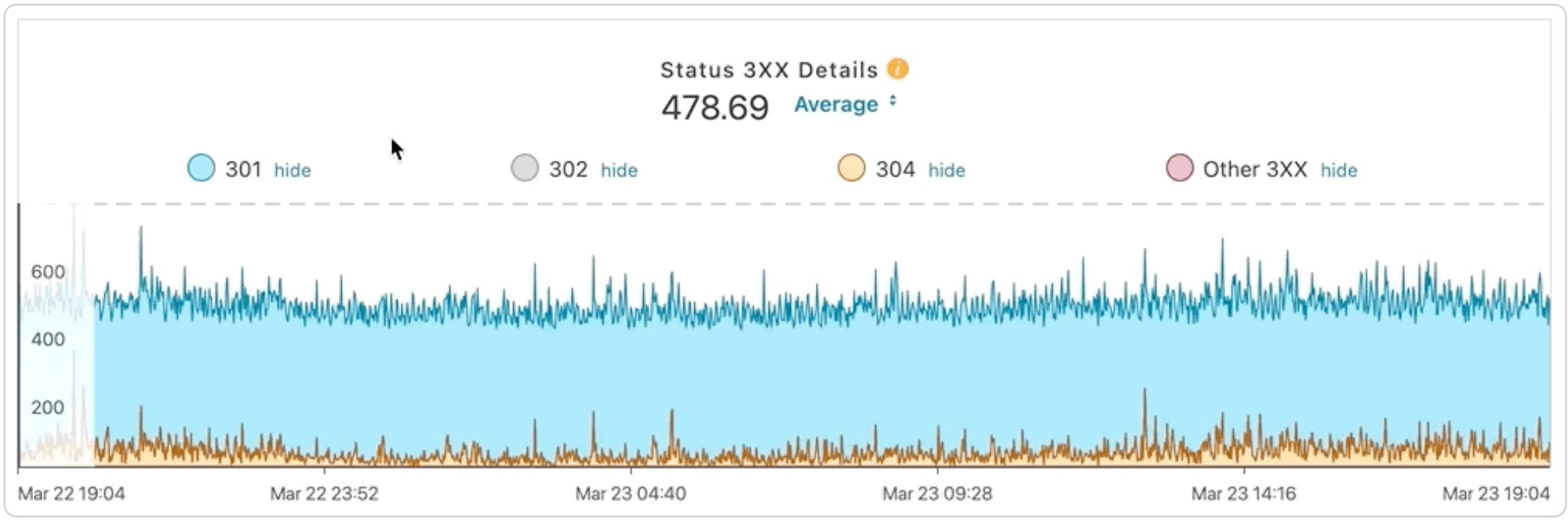
To change the statistics displayed by all historical graphs, use the Display menu in the upper right corner, as shown below.



To change the statistics displayed for a individual graph, click the menu below the graph's name, as shown below.



You can also exclude certain data entirely from some graphs. For example, in the Status 3XX Details graph, you can click **Show** or **Hide** or the corresponding color ship next to the specific 3XX errors (301, 302, etc.) to show or hide those error types, respectively.



What's next

Dig deeper into details about all areas of the web interface controls before you move on to using them to work with services.

About the web interface controls

Last updated: 2022-03-24

https://docs.fastly.com/en/guides/about-the-web-interface-controls

Fastly provides web interface access to all of its features and functions, which are also accessible using Fastly's application programming interface (API).

Before you begin

Before you begin learning about the Fastly web interface controls, it's useful to know the basics of content delivery and how caching and CDNs work.

Access to Fastly's web interface controls

Access to Fastly's web interface controls requires you to sign up for a Fastly account. Creating an account is free. Once you've created an account, you can navigate to the controls via the Fastly login page at https://manage.fastly.com, either directly using any standard web browser or by clicking a link at the top right of almost all pages at the Fastly website.

If it's your first time logging in, the controls may appear a bit empty. As you get started and create your first service, however, you'll begin to see more, including:

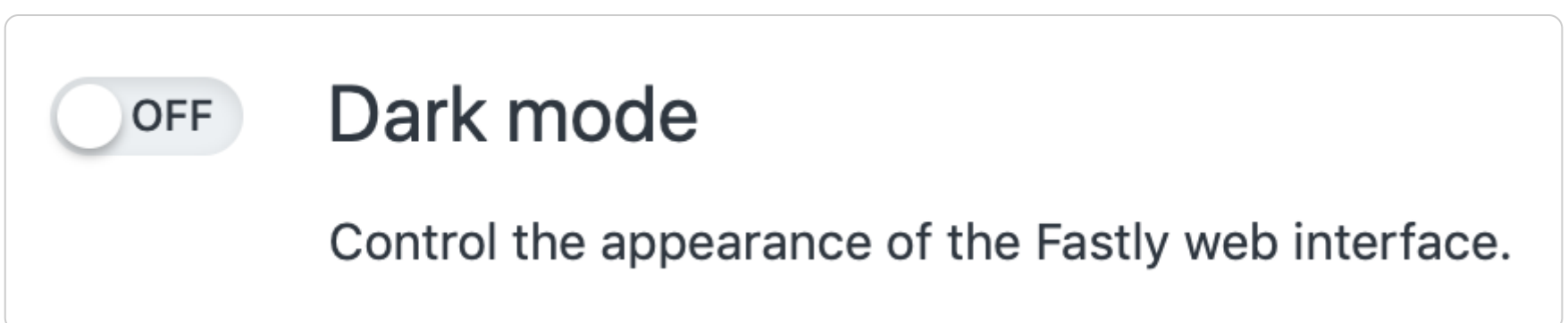
- the Home page where you view summarized details about all your services
- the Stats page where you monitor real-time and historic caching-related details about a specific service
- the Deliver page where you define the settings that specifically control how cached content should behave and be delivered
- the Compute page where you can define how each instance of your Wasm services should behave and interact with its data sources
- the Secure page where you access the different security products Fastly has to offer
- the account menu where you access account-specific settings, personal profile information, and billing-related details

Keep in mind that the controls that appear will be based on the roles and permissions assigned to you by whoever manages your company's account access. Also, not all Fastly service features are enabled by default. The appearance of the web interface controls may change from the defaults displayed once these services are enabled for your account.

Changing the appearance of the web interface

You can change the theme of the Fastly web interface. Follow these instructions to change the appearance for your personal profile:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Appearance** link.
3. To enable dark mode, click the **Dark mode** switch.



You can click the switch again to change the theme back to light mode.

Browser recommendations

We support a minimum display width of 768 pixels on the latest version of the following browsers:

- Google Chrome
- Firefox
- Safari

If you aren't using one of these browsers, then some visual styling may not be correct when using the Fastly web interface.

We strongly recommend updating your browser before beginning any debugging of Fastly services and before reporting problems to Fastly Customer Support. You can find the latest, downloadable versions of all major browsers online. The list at Browse Happy may help you.

What's next

Dig deeper into details about all areas of the web interface controls before you move on to using them to work with services.

Configuration

These articles provide basic instructions for configuring Fastly services after getting started.

https://docs.fastly.com/en/guides/configuration

§

These articles describe configuration settings and changes you can make to your cache settings when setting up Fastly services.

https://docs.fastly.com/en/guides/configuration#_caching

Caching configuration best practices

Last updated: 2021-04-23

https://docs.fastly.com/en/guides/caching-best-practices

To ensure optimum origin performance during times of increased demand or during scheduled downtime for your servers, consider the following best practices for your service's caching configurations.

Integrate Fastly with your application platform

You can optimize caching with Fastly by customizing your application platform settings. For instructions, see our documentation on integrating third-party services and configuring web server software. We also provide a variety of plugins to help you directly integrate Fastly with your content management system.

Check your cache hit ratio

The number of requests delivered by a cache server, divided by the number of cacheable requests (hits + misses), is called the *cache hit ratio*. A high cache hit ratio means you've kept request traffic from hitting your origin unnecessarily. Requests come from cache instead. In general, you want your cache hit ratio as high as possible, usually in excess of 90%. You can check your hit ratio by viewing the Stats page for your service.

Set a fallback TTL

The amount of time information can be retained in cache memory is considered its *time to live* or TTL. TTL is set based on the cache related headers information returned from your origin server. You can specifically set a fallback TTL (sometimes called a *default TTL*).

WARNING

If you're using custom VCL, the fallback TTL will be specified in the VCL boilerplate and the fallback TTL configured via the web interface or the API will not be applied. See our documentation on cache freshness and TTLs for more information.

TIP

If there's no other source of freshness in the response, setting the fallback TTL to 0 seconds in the web interface will set `return(pass)` in `vcl_fetch`.

We set a default fallback TTL that you can update at any time as follows:

1. Log in to the Fastly web interface.

2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. In the **Fallback TTL** area, click the pencil icon next to the TTL setting.

6. In the **Fallback TTL (sec)** field, enter the new TTL in seconds.
7. Click **Save** to save your changes.
8. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> See our Google Cloud Storage instructions if you're changing the default TTL for a GCS bucket.

Configure Fastly to temporarily serve stale content

If your origin becomes unavailable for an extended period of time (for example, being taken offline for maintenance purposes), temporarily serving stale content may help you. Serving stale content can also benefit you if your site's static content is updated or published quite frequently.

You can instruct Fastly to serve stale content by adding a `stale-while-revalidate` or `stale-if-error` statement on your `Cache-Control` or `Surrogate-Control` headers. Our guide to serving stale content describes this in more detail.

Decrease your first byte timeout time

After you have configured Fastly to temporarily serve stale, decreasing your first byte timeout time will cause stale content to be served to the requestor faster while fetching fresh content from the origin. Decreasing your first byte timeout time as well as serving stale will reduce unnecessary 503 first byte timeout errors. Decrease the first byte timeout time to your origin as follows:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. In the **Hosts** area, find your origin server and click the pencil icon to edit the host. The Edit this host page appears.
6. Click the **Advanced options** link at the bottom of the page. The Advanced options controls appear.
7. In the **First byte timeout** field, enter the new first byte timeout in milliseconds. Approximately 15000 milliseconds is a good default to start with.
8. Click **Update** to save your changes.
9. Click the **Activate** button to deploy your configuration changes.

Increase Cache-Control header times

During times of increased demand, you can instruct Fastly to keep objects in cache as long as possible by increasing the times you set on your Cache-Control headers. Consider changing the `max-age` on your `Cache-Control` or `Surrogate-Control` headers. Our documentation on cache freshness describes this in more detail.

Configure caching actions for specific workflows

When you create a cache setting, the **Action** setting determines how the request will be handled. The action settings and the most common workflow for each are described below.

- **Do nothing now** - Selecting this option will apply the request setting options, but won't force a lookup or a pass action.
- **Pass (do not cache)** - Selecting this option will make the request and subsequent response bypass the cache and go straight to origin. Use this option if you need to conditionally prevent pages from caching or when using conditions
- **Restart processing (not common)** - Selecting this option will restart processing of the request. Use this option if you need to check multiple backends for a single request
- **Deliver (not common)** - Selecting this option will deliver the object to the client. Use this option if you need to create an override condition.

Consider custom error handling

When downtime can't be avoided, standard error messages might not ensure the best user experience. Consider creating custom error messages that include information specific to the request being made and pertinent to the user. Our guide to creating error pages with custom responses provides more detail.

Inform Fastly Customer Support

We like to be sure we're readily available for assistance during customer events. When you know in advance that an event is forthcoming, contact support with details. Be sure to include details about:

- the date and time of the event
- the type of event happening
- how long you expect it to last (if it's planned)
- the Fastly services that might be affected

If the event you're planning is designed to validate the security of your service behind Fastly, be sure to read our guide to penetration testing first.

Controlling caching

Last updated: 2021-12-16

https://docs.fastly.com/en/guides/controlling-caching

When we store your content in cache, we calculate a Time to Live (TTL). The TTL is the maximum amount of time we will use the content to answer requests without consulting your origin server. After the TTL expires, we may keep the content in storage, but we won't use it to answer requests unless we're able to revalidate it with your origin.

There's no guarantee that the content will stay cached for the length of time specified in the TTL. Depending on the size of the object and how popular it is relative to other objects, we may delete it before its TTL expires.

For more information about controlling how long Fastly caches your resources, start with our documentation on cache freshness. In general, we will honor any Cache-Control headers you send to us from your origin.

Determining the TTL of an object

You can determine the TTL of an individual object as follows:

- If you've set the `Surrogate-Control: max-age`, `Cache-Control: max-age`, or `Expires` headers, the TTL is whatever you specified in those headers

- If you've specified the TTL in the web interface or custom VCL, the TTL is whatever you specified
- If you've specified the TTL in the web interface or custom VCL and you've set the `Surrogate-Control: max-age`, `Cache-Control: max-age`, or `Expires` headers, the TTL specified in the web interface might override the TTL specified in the origin response
- If you haven't specified the TTL in the web interface or custom VCL and you haven't set the `Surrogate-Control: max-age`, `Cache-Control: max-age`, or `Expires` headers, the TTL is 3600 seconds

You can change this limit on the Deliver page.

Setting different TTLs for Fastly cache and web browsers

Purging objects from the Fastly cache is easy. Clearing the caches of users' web browsers is much harder. For that reason, it can make sense to set different TTLs for content in the Fastly cache versus users' web browsers. You can set different TTLs for the Fastly cache and web browsers through Surrogate-Control headers defined by the W3C. For example, if you wanted Fastly to cache something for a year but you also wanted to set a maximum age of a single day for users viewing that object in a web browser, then you could return the following HTTP headers:

```
Surrogate-Control: max-age=31557600
Cache-Control: max-age=86400
```

The Surrogate-Control header in this example tells Fastly to cache the object for a maximum of 31557600 seconds (one year). The Cache-Control header in this example tells the browser to cache the object for a maximum of 86400 seconds (1 day).

> **TIP**
>
> CDNs can't invalidate an end user's web browser cache. If your content is going to change frequently or if you want new content immediately available to end users, the best strategy is to cache that content on Fastly and specifically instruct browsers not to cache it. Then, when you purge the Fastly cache, an end user's browser will display the new version the next time the content is requested.
>
> One way to do this is to return the following headers:
>
> ```
> Surrogate-Control: max-age=31557600
> Cache-Control: no-store, max-age=0
> ```
>
> This Surrogate-Control header tells Fastly to cache the object for a maximum of 31557600 seconds (one year). The Cache-Control header tells the browser not to cache the object.

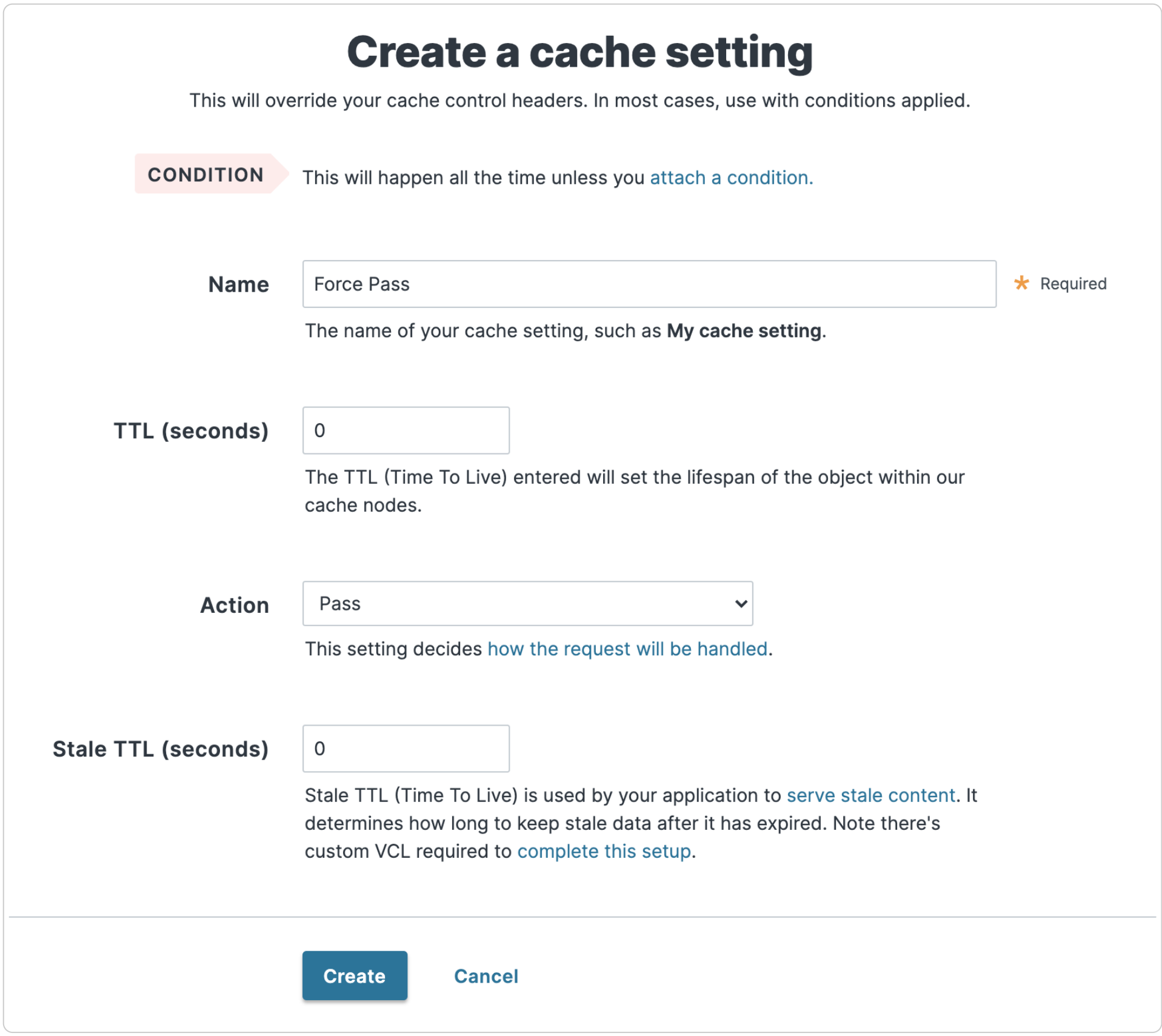
For Surrogate-Control, Fastly supports the `max-age`, `stale-if-error`, and `stale-while-revalidate` parameters.

For more information about controlling caching, see our documentation on cache freshness.

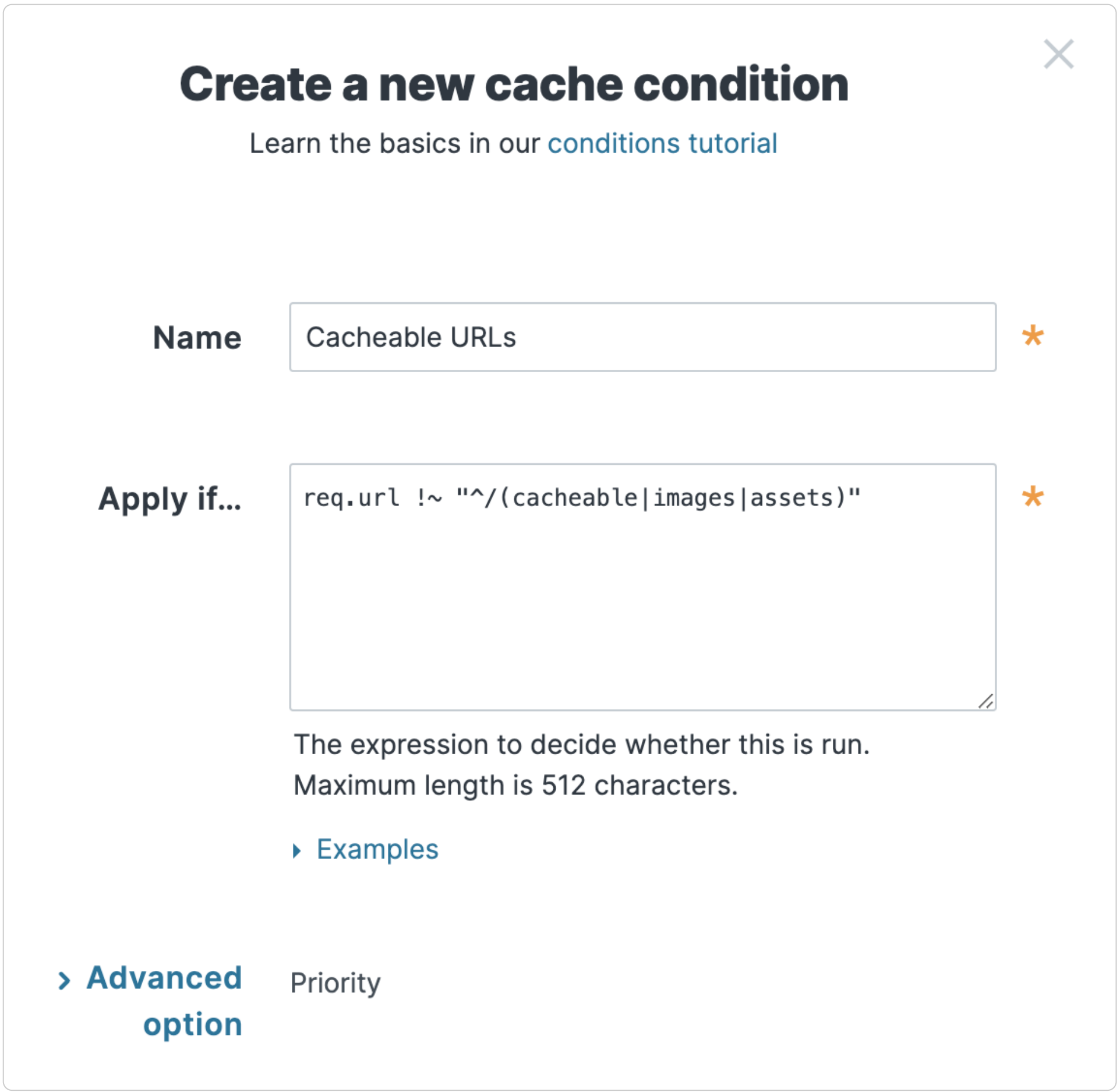
Conditionally preventing pages from caching

To conditionally prevent pages from caching, follow the steps below.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **Create cache setting** button to create a new cache setting. The Create a cache setting page appears.



6. Fill out the **Create a cache setting** fields as follows:
   - In the **Name** field, enter a descriptive name for the new cache setting (e.g., `Force Pass`).
   - In the **TTL (seconds)** field, enter `0` to set the lifespan of the object in our cache nodes to zero.
   - From the **Action** menu, select **Pass (do not cache)** to pass the request and avoid caching it.
   - In the **Stale TTL (seconds)** field, enter `0` to set the amount of time to serve stale content, in seconds, to zero.
7. Click the **Create** button. The new cache setting appears on the Settings page.
8. Click the **Attach a condition** link to the right of the newly created cache setting. The Create a new cache condition window appears.



9. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, enter a descriptive name for the condition (e.g., `Cacheable URLs`).
   - In the **Apply if** field, create a condition that matches the URLs to not cache. For example, you could enter `req.url !~ "^/(cacheable|images|assets)"` to set the condition to look for URLs that do not start with `/cacheable`, `/images`, or `/assets`. If the condition finds them, the URLs should be cached. If the condition doesn't find them, the cache setting that is set to `Pass` ensures the URLs are explicitly not cached.
10. Click the **Save and apply to** button.
11. Click the **Activate** button to deploy your configuration changes.

> **TIP**
>
> You can use these steps to override default caching based on a backend response.

Enabling API caching

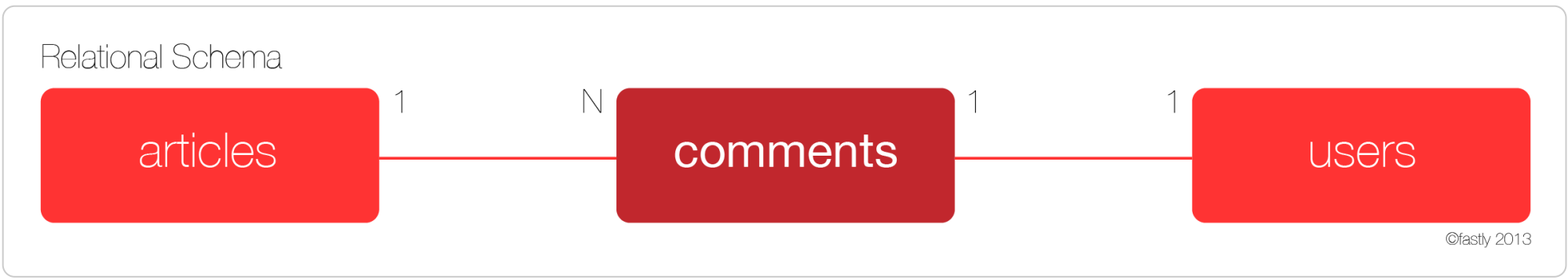
Last updated: 2021-10-25

https://docs.fastly.com/en/guides/enabling-api-caching

Application Programming Interfaces (APIs) allow you to retrieve data from a variety of web services. Fastly makes it possible for you to cache your API so you can accelerate the performance of your service-oriented architecture. It optimizes your API's performance by efficiently handling traffic bursts and reducing latency.

An example

Let's look at an example to learn how API caching works. Imagine we're an online magazine with articles on which users can make comments. Each article can have many comments, and each comment is created by exactly one user.



We'll design a RESTful API specification and use it to manipulate and retrieve comments:

- `GET /comment` - Returns a list of all comments
- `GET /comment/:id` - Returns a comment with the given ID
- `POST /comment` - Creates a new comment
- `PUT /comment/:id` - Updates a comment with the given ID
- `DELETE /comment/:id` - Deletes a comment with the given ID

The create, read, update, and delete (CRUD) methods ensure the API can perform its basic operations, but they don't expose the relational aspect of the data. To do so, you would add a couple of relational endpoints:

- `GET /articles/:article_id/comments` - Get a list of comments for a given article
- `GET /user/:user_id/comments` - Get all comments for a given user

Endpoints like these allow programmers to get the information they need to do things like render the HTML page for an article, or display comments on a user's profile page. While there are many other possible endpoints we could construct, this set should suffice for the purposes of this guide. Let's assume that the API has been programmed to use an Object-Relational Mapper (ORM), such as ActiveRecord, when interacting with the database.

Determining which API endpoints to cache

Start by identifying the URLs you want to cache. We recommend splitting the specification endpoints into two groups.

The first group, called *accessors*, retrieves or accesses the comment data. These are the endpoints you want to cache using Fastly. Using the example, four endpoints match this description:

- `GET /comment`
- `GET /comment/:id`
- `GET /article/:article_id/comments`
- `GET /user/:user_id/comments`

The second group, called *mutators*, changes or mutates the comment data. These endpoints are always dynamic, and are therefore uncacheable. Using the example, three endpoints match this description:

- `POST /comment`
- `PUT /comment/:id`
- `DELETE /comment/:id`

You should see a pattern emerging. Because the example API is RESTful, we can use a simple rule to identify the accessor and mutator endpoints: GET endpoints can be cached, but PUT, POST, and DELETE endpoints cannot.

Once you've gathered this information, you're ready to program the API to configure PURGE requests.

Configuring PURGE requests

Don't be tempted to point at the PUT, POST, and DELETE endpoints as the place where data is modified. In most modern APIs, these endpoints represent an interface to the actual model code responsible for handling the database modifications.

In the example, we assumed that we'd be using an ORM to perform the actual database work. Most ORMs allow programmers to set special *callbacks* on models that will fire when certain actions have been performed (e.g., before or after validation, or after creating a new record).

For purging, we are interested in whether a model has saved information to the database — whether it's a new record, an update to an existing record, or the deleting of a record. At this point, we'd add a callback that tells the API to send a PURGE request to Fastly for each of the cacheable endpoints.

For an ActiveRecord comments model, you could do something like this:

```
require 'fastly'

class Comment < ActiveRecord::Base
  fastly = Fastly.new(api_key: 'FASTLY_API_TOKEN')

  after_save do
    fastly.purge "/comment"
    fastly.purge "/comment/#{self.id}"
    fastly.purge "/article/#{self.article_id}/comments"
    fastly.purge "/user/#{self.user_id}/comments"
  end
end
```

Keep two things in mind when creating the callback:

- The purge code should be triggered *after* the information has been saved to the database, otherwise a race condition could be created where Fastly fetches the data from the origin server before the data has been saved to the database. This would cache the old data instead of the new data.
- These URLs are being purged because they have content that changes when a comment is changed.

With the model code in place, the API is now ready to be cached.

Setting up Fastly

The final step to enabling API caching involves setting up Fastly. You'll need to:

- Create a new service
- Add the domain for the API
- Add the origin server that powers the API

In addition, you can optionally create rules that tell Fastly how to work with the specific elements that are exclusive to your API.

> **NOTE**
>
> By default, Fastly will not cache PUT, POST, and DELETE requests. For more information, see our guide on default caching behavior of HTTP methods.

Creating a new service

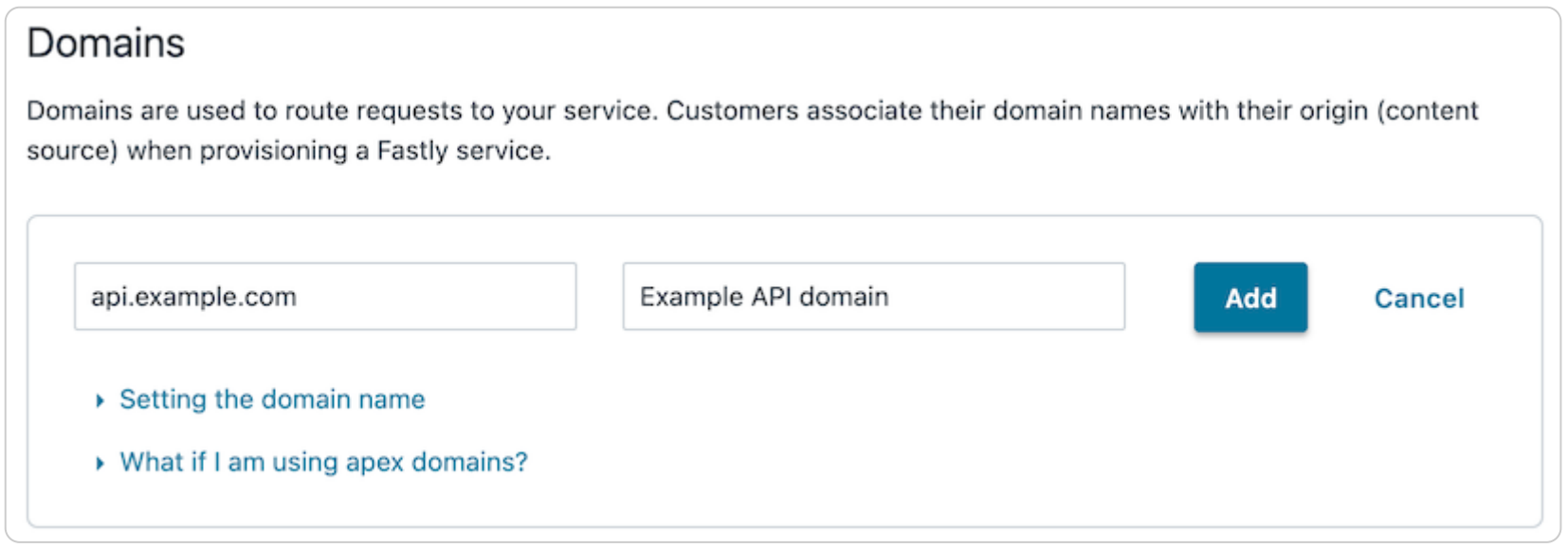
Follow the instructions for creating a new service. You'll add specific details about your API server when you fill out the **Create a new service** fields:

- In the **Name** field, enter a name for this service that helps you identify it's related to caching your API information (e.g., `My API Service`).
- In the **Domain** field, enter the domain name associated with your API (e.g., `api.example.com`).
- In the **Address** field, enter the IP address or hostname of your API server.

Adding the domain

Follow these instructions to add the API's domain name to your Fastly service:

1. On the **Deliver** page, click the **Edit configuration** menu and then clone the active version of the service. The Domains page appears.
2. Click the **Create domain** button. The Create a domain page appears.



3. Fill out the **Create a domain** fields as follows:
   - In the **Domain Name** field, enter the domain name for the API.
   - In the **Comment** field, enter an optional comment that describes your domain.
4. Click **Create**. Your API's domain name appears in the list of domains.

Adding the origin server

Follow the instructions for working with hosts. You'll add specific details about your API server when you fill out the **Create a host** fields:

- In the **Name** field, enter a name for the origin server that helps you identify it's related to caching your API information.
- In the **Address** field, enter the IP address (or hostname) of the API server.

Implementing API cache control

Last updated: 2016-12-22

https://docs.fastly.com/en/guides/implementing-api-cache-control

This guide explains how to implement API cache control. Once you've enabled API caching, and ensured purging works properly with your cached data, you can set up specific headers like Cache-Control and Surrogate-Control to change when data is cached.

Understanding Cache-Control headers

In general, we assume that GET requests are cached and PUT, POST, and DELETE requests are not. For an ideal REST API, this rule works well. Unfortunately, most APIs are far from ideal and require additional caching rules for some requests.

For these reasons, it's a good idea to set Cache-Control headers when migrating APIs to Fastly. Cache-Control, as defined by RFC 7234 (the HTTP specification), includes many different options for appropriate handling of cached data. Specifically, Cache-Control headers tell user agents (e.g., web browsers) how to handle the caching of server responses. For example:

- `Cache-Control: private`
- `Cache-Control: max-age=86400`

In the first example, `private` tells the user agent the information is specific to a single user and should not be cached for other users. In the second example, `max-age=86400` tells the user agent the response can be cached, but that it expires in exactly 86,400 seconds (one day).

Fastly respects Cache-Control headers by default, but you can also use another proxy-specific header: Surrogate-Control. Surrogate-Control headers are similar to Cache-Control headers, but provide instructions to reverse proxy caches like Fastly. You can use Cache-Control and Surrogate-Control headers together. For more information about Cache-Control and Surrogate-Control headers, see our documentation on cache freshness.

An updated example

Let's take a look at how the Cache-Control headers could be used in our original example, the comments API. Recall the API endpoint that provided a list of comments for a given article:

```
GET /article/:article_id/comments
```

When a user submits a comment for a given article, the response from this endpoint will be purged from the Fastly cache by the comment model. It's hard to predict when content will change. Therefore, we'd like to ensure the following:

1. If the content doesn't change, it should stay cached in Fastly for a reasonable amount of time.
2. If the content does change, it should not be cached by the client longer than it needs to be.

The goal is to ensure that API responses will reach clients in a timely manner, but we also want to ensure that clients always have the most up-to-date information. The first constraint can be solved by using the Surrogate-Control header, and the second constraint can be solved by using the Cache-Control header:

```
Surrogate-Control: max-age=86400
Cache-Control: max-age=60
```

These headers tell Fastly that it is allowed to cache the content for up to one day. In addition, the headers tell the client that it is allowed to cache the content for 60 seconds, and that it should go back to its source of truth (in this case, the Fastly cache) after 60 seconds.

Implementing cache control

Migrating APIs isn't easy, even for experienced teams. When migrating an API to Fastly, we recommend separating the task into three strategic endpoint migrations to make the process more manageable while still maintaining the validity of the API as a whole.

Preparing the API

To ensure that the API bypasses the cache during the piecewise migration, we must have every API endpoint return a specific control header:

```
Cache-Control: private
```

This header tells Fastly that a request to any endpoint on the API should bypass the cache and be sent directly to the origin. This will allow us to serve the API via Fastly and have it work as expected.

> **NOTE**
>
> Modern web frameworks allow for blanket rules to be overridden by specific endpoints (for example, by the use of middlewares). Depending on how the API has been implemented, this step might be as simple as adding a single line of code.

Serving traffic with Fastly

The next step is configuring a Fastly service to serve the API's traffic. After you save the configuration, there will be an immediate speed improvement. This happens because Fastly's cache servers keep long-lasting connections to the API's origin servers, which reduces the latency overhead of establishing multiple TCP connections.

Migrating endpoints

Now we can implement instant purge caching for each cacheable API endpoint, one at a time. The order in which this is done depends on the API, but by targeting the slowest endpoints first, you can achieve dramatic improvements for endpoints that need them the most. Because each endpoint can be worked on independently, the engineering process is easier to manage.

Excluding endpoints

The last step is deciding which API endpoints you don't want Fastly to cache. To disable caching for endpoints, you'll need to add new conditions for the endpoints. As you learned in Preparing the API, using the `Cache-Control: private` header is another option for disabling caching.

Overriding caching defaults based on a backend response

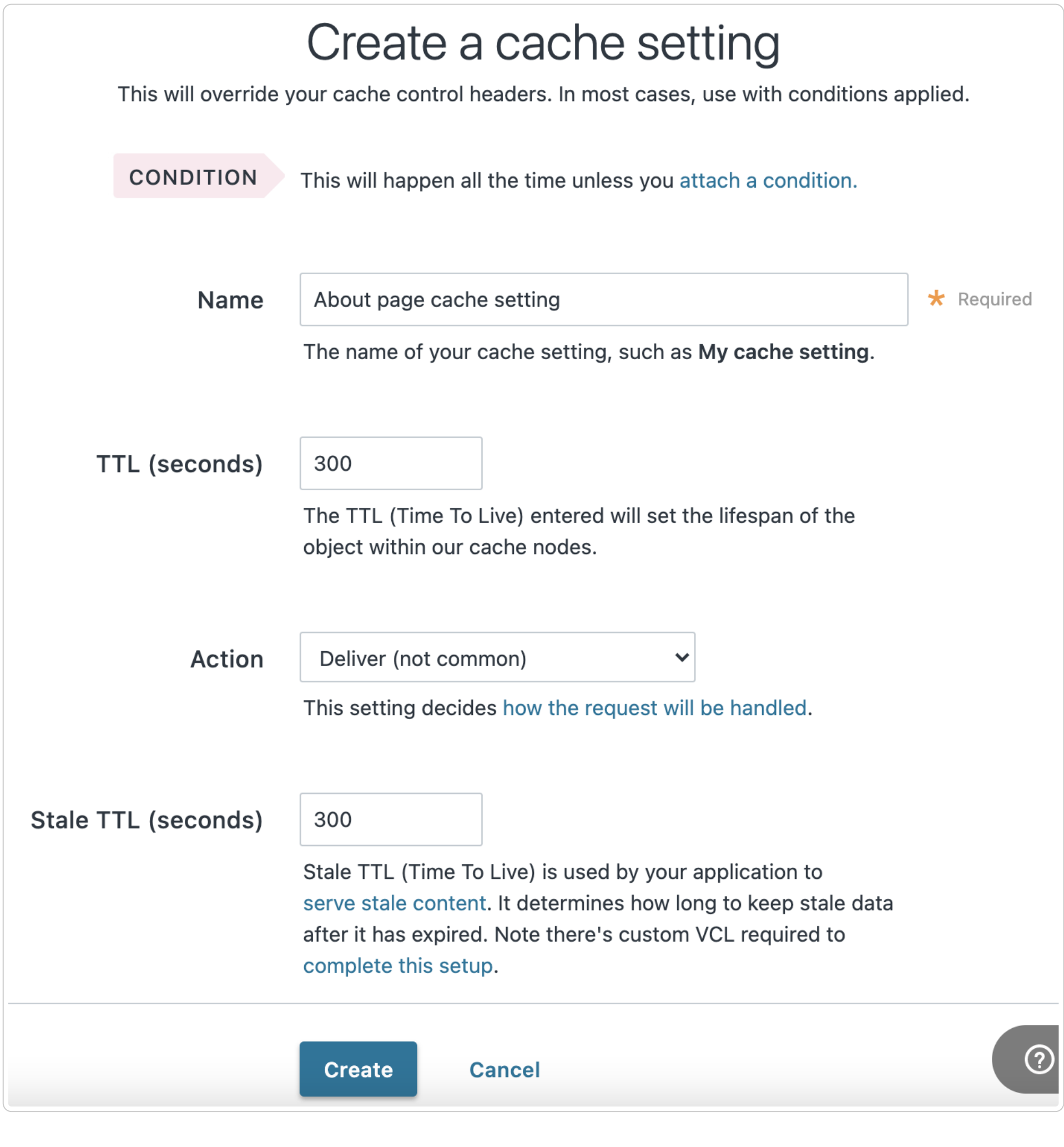
Last updated: 2021-05-11

https://docs.fastly.com/en/guides/overriding-caching-defaults-based-on-a-backend-response

In certain situations you may want to conditionally apply a different caching policy based on a backend response. In this particular case we have backend that on occasion returns 404 errors (e.g., document not found). We don't want those responses to be cached for the full caching period of a day but only for 5 minutes. To override default caching we add a cache object and then create conditions for it.

Creating the new Cache Object

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **Create cache setting** button. The Create a cache setting page appears.

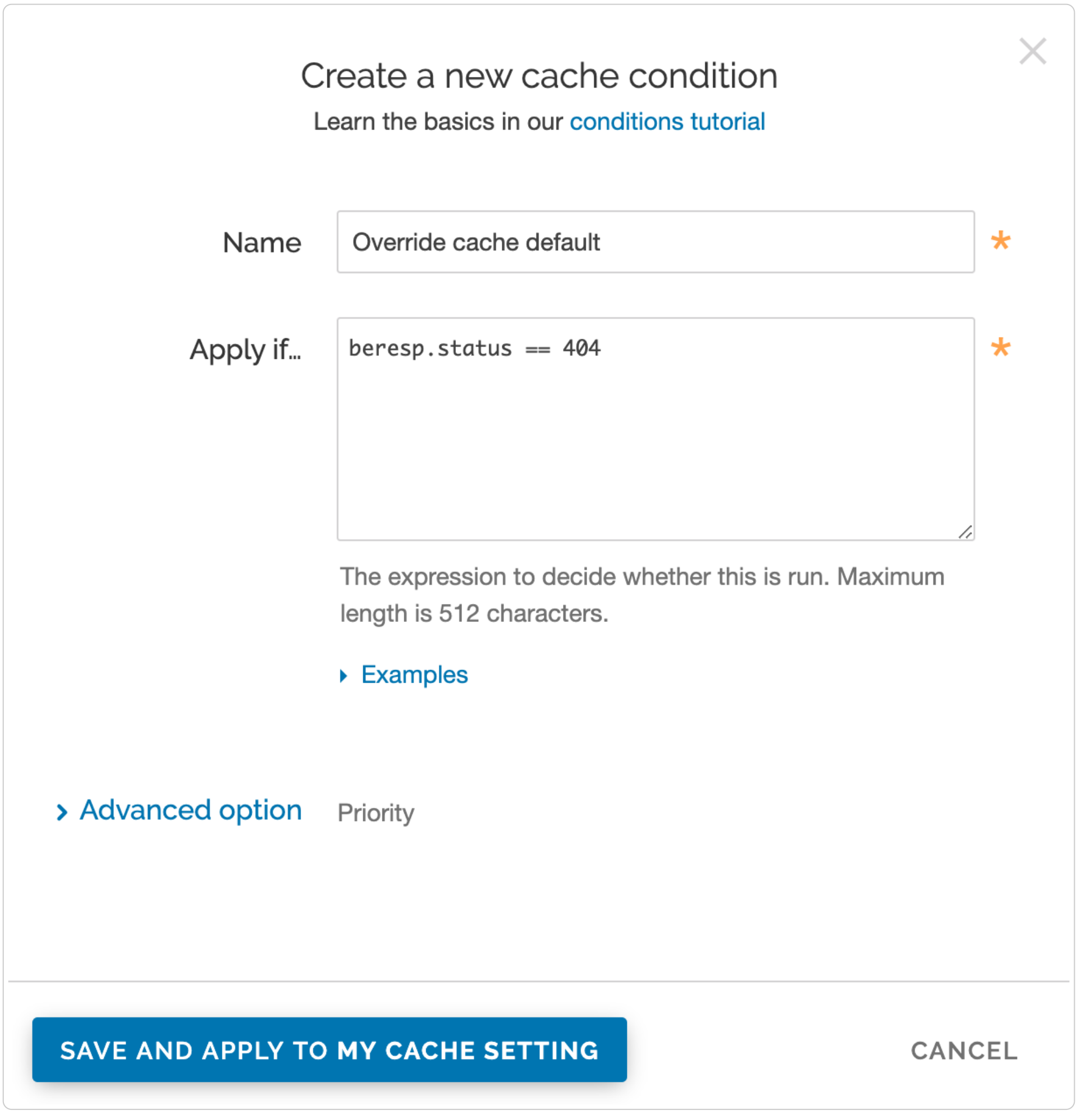


6. Fill out the **Create a cache setting** fields as follows:
   - In the **Name** field, enter a descriptive name for the new cache settings.
   - In the **TTL (seconds)** field, enter the amount of time, in seconds, to cache the objects (e.g., `300`).
   - From the **Action** menu, select **Deliver**.
   - In the **Stale TTL (seconds)** field, enter the amount of time to serve stale or expired responses, in seconds, should the backend become unavailable (e.g., `300`).
7. Click the **Create** button.

Creating an Override Condition for the new Cache Object

Once the object is created, add a condition to it.

1. Click the **Attach a condition** link to the right of the object.
2. Click **Create cache setting** button. The Create a new cache condition window appears.



3. Fill out the **Create a new cache setting** fields as follows:
   - In the **Name** field, enter a descriptive name for the new condition. For example, `Override cache default`.
   - In the **Apply if** field, enter an appropriate backend response header to specify when the condition will be applied. For example, `beresp.status == 404`.
4. Click the **Save and apply to** button.
5. Click the **Activate** button to deploy your configuration changes.

Other notes

You can use any backend response header in the **Apply if** field to make decisions on caching.

For example, `beresp.http.Content-Type ~ "^text/html"` can be used to specify different caching rules for HTML documents.

Preventing cache poisoning via HTTP X-headers

Last updated: 2020-08-14

https://docs.fastly.com/en/guides/preventing-cache-poisoning

Fastly service configurations may be vulnerable to cache poisoning if they do not take into consideration the interaction between HTTP "X-" headers used by backends to select content. This vulnerability can be mitigated using a VCL patch or by modifying backend configurations.

When cache poisoning can occur

If one or more of your backends uses the contents of the X-Forwarded-Host, X-Rewrite-URL, or X-Original-URL HTTP request headers to decide which of your users (or which security domain) it sends an HTTP response for, you could be impacted by vulnerability. If your site's Fastly configuration passes one of these headers to your backend and does not factor the contents of it into the effective edge cache key (for example, explicitly or via the Vary HTTP response header), an attacker could potentially cause the edge to store a response with arbitrary content inserted into a victim's cache.

An attacker might be able to poison a Fastly customer URL by sending an HTTP request to a site that causes the affected backend to respond with an attacker-controlled response. The malicious response object would be stored in the site's cache at a poisoned URL. An attacker could then potentially lure a victim site user into browsing the poisoned URL, where they would be served malicious content.

How to mitigate cache poisoning

If your origin uses special values to select content for users or to otherwise select between security domains, consider reconfiguring your origin server and applying any corresponding security updates as suggested in our original security advisory. In addition, strip or normalize the values of X-Forwarded-Host, X-Rewrite-URL, or X-Original-URL in VCL.

To do this, set the vulnerable headers to a known-safe value or unset the headers completely. For example, the X-Forwarded-Host header can be set to the value of the Host header via the following VCL snippet:

```
set req.http.x-forwarded-host = req.http.host;
```

The X-Original-URL header can be unset via the following VCL snippet:

```
unset req.http.x-original-url;
```

And X-Rewrite-URL can be unset via the following VCL snippet:

```
unset req.http.x-rewrite-url;
```

Alternatively, the values could be included in your cache key or Vary header to prevent caching of content across security domains. See our guide to manipulating the cache key for more information.

Segmented Caching

Last updated: 2021-07-28

https://docs.fastly.com/en/guides/segmented-caching

Fastly's Segmented Caching feature allows you to cache resources of any size. Segmented Caching works by breaking resources into smaller segments in Fastly's cache then recombining or splitting these resources to respond to arbitrary size byte `Range:` requests from clients. Once enabled, Segmented Caching improves performance for `Range:` requests and allows Fastly to efficiently cache resources of any size.

WARNING

Fastly recommends enabling Segmented Caching on services that will be serving large resources. Without Segmented Caching enabled, the resource size limits for your account depend on when you become a Fastly customer:

- If you created your account on or after June 17, 2020 and haven't enabled Segmented Caching, your Fastly services have a maximum object size of 20 MB.
- If you created your account prior to June 17, 2020 and haven't enabled Segmented Caching, your Fastly services have a maximum cacheable object size of 2 GB for requests without Streaming Miss or 5 GB for requests with Streaming Miss.

How Segmented Caching works

When an end user makes a `Range:` request for a resource with Segmented Caching enabled and a cache miss occurs (that is, at least part of the range is not cached), Fastly will make the appropriate `Range:` requests back to origin. Segmented Caching will then ensure only the specific portions of the resource that have been requested by the end user (along with rounding based on object size) will be cached rather than the entire resource. Partial cache hits will result in having the cached portion served from cache and the missing pieces fetched from origin. (Requests for an entire resource would be treated as a byte `Range:` request from 0 to end of resource.)

Once Fastly has all of the objects necessary to respond to an end user's request, the Segmented Caching feature will assemble the response by concatenating or pulling portions of objects. The requests back to origin, also called "inner requests," will have a `true` value for `segmented_caching.is_inner_req` and requests from end users, also called "outer requests," will have a `true` value for `segmented_caching.is_outer_req`. If a request is made for an object without segmented caching enabled, both variables will have a FALSE value.

> **NOTE**
>
> The feature will only go back to origin for missing objects needed to handle the end-client's byte `Range:` request. Cache hits will occur based on having that portion of resource in cache even if the end user's range exact request is unique.

Limitations and considerations

This feature has the following limitations and considerations you should take into account:

- **Segmented Caching is not compatible with** object compression **and** ESI**.** To use either of these features, you must ensure Segmented Caching is disabled.
- **Segmented Caching is not compatible with Fastly's** Image Optimizer (IO)**.** If IO is enabled, Segmented Caching is disabled automatically.
- **HTTP chunked transfer encoding between Fastly and origin isn't supported.** Your origin server must frame responses to `Range:` requests with the `Content-Length` header.
- **URL purges must be authenticated.** Segmented caching allows you to purge all range objects for the resource by URL purge, but authentication for URL purge needs to be enabled due to its underlying implementation. Make sure you've provided an authorization token for URL purges as described in our purging documentation.
- **Segmented Caching cannot be enabled based on resource size.** The VCL code to enable Segmented caching must run *before* the resource is requested from cache, so it is not possible to know how large it will be. However, it is possible to make segmented caching conditional upon the URL (e.g. `/video/`) or file extension (e.g. *.m4v), and this is a common use case for resources which will benefit from it. Our instructions below contain an example of how to enable the Segmented Caching feature based on extension.
- **Resources cached prior to enabling Segmented Caching are not used.** If you are enabling Segmented Caching on an existing service, whole resources already in cache are ignored and Varnish will go back to origin to build range request objects. You can choose to purge these or ignore them to be aged out of cache.
- **Your cache hit ratio (CHR) will appear lower than it actually is** because only outer requests are used in the calculation. For example, if there is a request for the first 100 MB of a resource but Fastly only has 99 MB of 100 MB in cache, the entire request will be counted as a miss in the CHR stat. In practice, only 1 MB had to be fetched from origin and you experienced 99% origin offload.

Enabling Segmented Caching

Use the following steps to enable Segmented Caching.

1. Determine which resources should use Segmented Caching.

   > **TIP**
   >
   > We recommend focusing on a set of file extensions or a well-defined URL structure that distinguishes the resources.

2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.

4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the **Create your first VCL snippet** button. The Create a VCL snippet page appears.
6. In the **Name** field, enter an appropriate name (e.g., `Enable segmented caching`).
7. From the **Type (placement of the snippets)** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **recv (vcl_recv)**.
9. In the **VCL** field, add a VCL snippet that sets the `req.enable_segmented_caching` VCL variable to `true` in `vcl_recv`. For example, to ensure proper caching of the large resources you've identified that contain MPEG-2-compressed video data, you could add this VCL snippet in vcl_recv:

```
# my custom enabled Segmented Caching code
if (req.url.ext == "ts") {
   set req.enable_segmented_caching = true;
}
```

   This snippet tells Fastly to look for requests for files with the `ts` extension and then enable Segmented Caching for those files.

10. Click **Create** to create the snippet.
11. Click the **Activate** button to deploy your configuration changes.

---

§

These articles describe conditions and how to use them in VCL and the Fastly web interface.

https://docs.fastly.com/en/guides/configuration#_conditions

About conditions

Last updated: 2018-05-11

https://docs.fastly.com/en/guides/about-conditions

Conditions control how requests are processed. You can use them to add logic to any basic configuration object in a service and have them control if and when that object is applied. Conditions require minimal programming. They allow you to wrap configuration objects attached to your service in a VCL IF statement.

Before you start using conditions

Be sure you understand the construction of basic logical expressions before you start using conditions. Specifically, you should understand basic C-style logical expression syntax (e.g., basic logic, operators such as && and precedence) when working with conditions. A basic programming guide that deals with IF style expressions in either the C or Perl language (the Tizag Perl tutorial is a good one to start with). Even though they aren't directly applicable to our condition examples, the syntax of these languages is similar to VCL.

A simple condition example

The simplest way to explain how Fastly handles conditions is this IF statement:

```
IF
   this condition happens
THEN
   respond this way
```

A practical example can demonstrate this. The vast majority of the time, your site processes requests for information as usual, but every so often customers mistype a search term or simply can't find what they're looking for and you're forced to display a 404 Not Found error. You've realized that when that happens, the standard 404 Not Found errors on your website aren't as helpful as they

could be. To fix this, any time your server can't find what a customer is looking for (a condition), you want to display a customized 404 message instructing customers to contact your support team for help (a response).

In plain English, the IF statement might look like this:

```
IF
   404 Not Found is what we have to tell the customer
THEN
   respond with the special Contact Support page
```

The IF line in the example above is the *condition* you've set. The THEN line describes what will happen if that condition is met.

If you were to replace the English in the example above with VCL variables and a little bit of HTML, it might look like this instead:

```
IF
  beresp.status == 404
THEN
  respond with <html><body><h1>Can't find it?</h1><p>Contact support@example.com for help.</p></body></html>
```

Interested in doing this? We have step-by-step instructions for creating error pages with custom responses.

Ideas for using conditions

Need some more ideas for when you could use conditions? Explore these:

Condition	Response	Learn how
A web robot wants to crawl a particular area of your website	Provide a customized robots.txt file defining which areas of your website should not be processed or scanned	Creating and customizing a robots.txt file
Your server needs to return a 404 Not Found response	Change the default caching time for only 404 responses from 3600 seconds (60 minutes) to 120 seconds (2 minutes)	Overriding caching defaults based on a backend response
Users request a popular page on your site but it's been moved to a different area	Have Fastly redirect the page requests at the edge, without having to go back to your origin server for it	Generating HTTP redirects at the edge

Types of conditions and when you can use them

We group conditions into three types:

- request conditions
- response conditions
- cache conditions

A condition's type dictates which configuration objects it can be applied to during a specific stage of the caching process. In addition, each stage of caching works with a different set of VCL variables that can be used to create conditions.

Condition type	Applied when Fastly ...	Works with which VCL variables
Request	processes a request	`client.*` `server.*` `req.*`
Response	processes a response to a request	`client.*` `server.*` `req.*` `resp.*`

Condition type	Applied when Fastly ...	Works with which VCL variables
Cache	receives a response from your origin, just before that response is (potentially) cached	`client.*` `server.*` `req.*` `beresp.*`

Where to go for more information

The Varnish Cache documentation provides a complete list of variables you can use to craft conditions. Keep in mind, however, some of the variables Varnish allows may not be available or may have no meaning in the context of Fastly services. For more information, see our Guide to VCL.

Troubleshooting conditions

Last updated: 2018-09-21

https://docs.fastly.com/en/guides/troubleshooting-conditions

If you are having problems using conditions, here are some common things to look for.

Check the Apply if field for if statements

Most problems with conditions occur in the Apply if parameter because it uses logical expressions to represent actual VCL variables that specify when a condition should be applied to a configuration object. If you are having problems using conditions, start by checking to see if you've put an `if ()` statement in the wrong place. A condition's if statement is implied and doesn't need to be placed in the Apply if field of the condition window. You only need to enter an evaluated expression (e.g., `req.url ~ "^/special/"`).

Check the construction of inverse regex matches

Consider if inverse regular expression (regex) matching might be the issue, especially if you're using it to exclude a particular URL in your condition. When using the `!~` (inverse regex match) to build expressions that exclude particular URLs, be thoughtful when also using the `||` or `&&` operators and multiple patterns.

For example, if you want to apply something to all URLs except those that start with `/admin`, the condition you'd enter into the Apply if field would be `req.url !~ "^/admin"`. If you also wanted to exclude URLs starting with `/internal`, that expression would be `!(req.url ~ "^/admin" || req.url ~ "^/internal")`.

TIP

Keep in mind De Morgan's laws if you're using multiple conditions and negation.

Check for general regex formatting mistakes

Consider the following general regex issues that may have caused trouble:

- **Is case sensitivity the problem?** Varnish regex is case sensitive by default. To use a case insensitive check, you must use the `(?i)` flag.
- **Have you escaped forward slashes?** Forward slashes don't need to be escaped in Varnish regex.

Our cheatsheet provides additional examples of using VCL with regular expressions.

Using conditions

Last updated: 2018-10-02

https://docs.fastly.com/en/guides/using-conditions

Conditions use the Varnish Configuration Language (VCL) to define when a configuration object should be applied while processing requests to a cache server. Once you understand some basics about conditions, use this guide to learn about how to create conditions using the Fastly web interface and when to use them.

Where to find conditions

Conditions appear in two areas of Fastly's web interface:

- The Manage conditions page lists all conditions available to your configuration settings.
- Each configuration object displays conditions specifically attached to them.

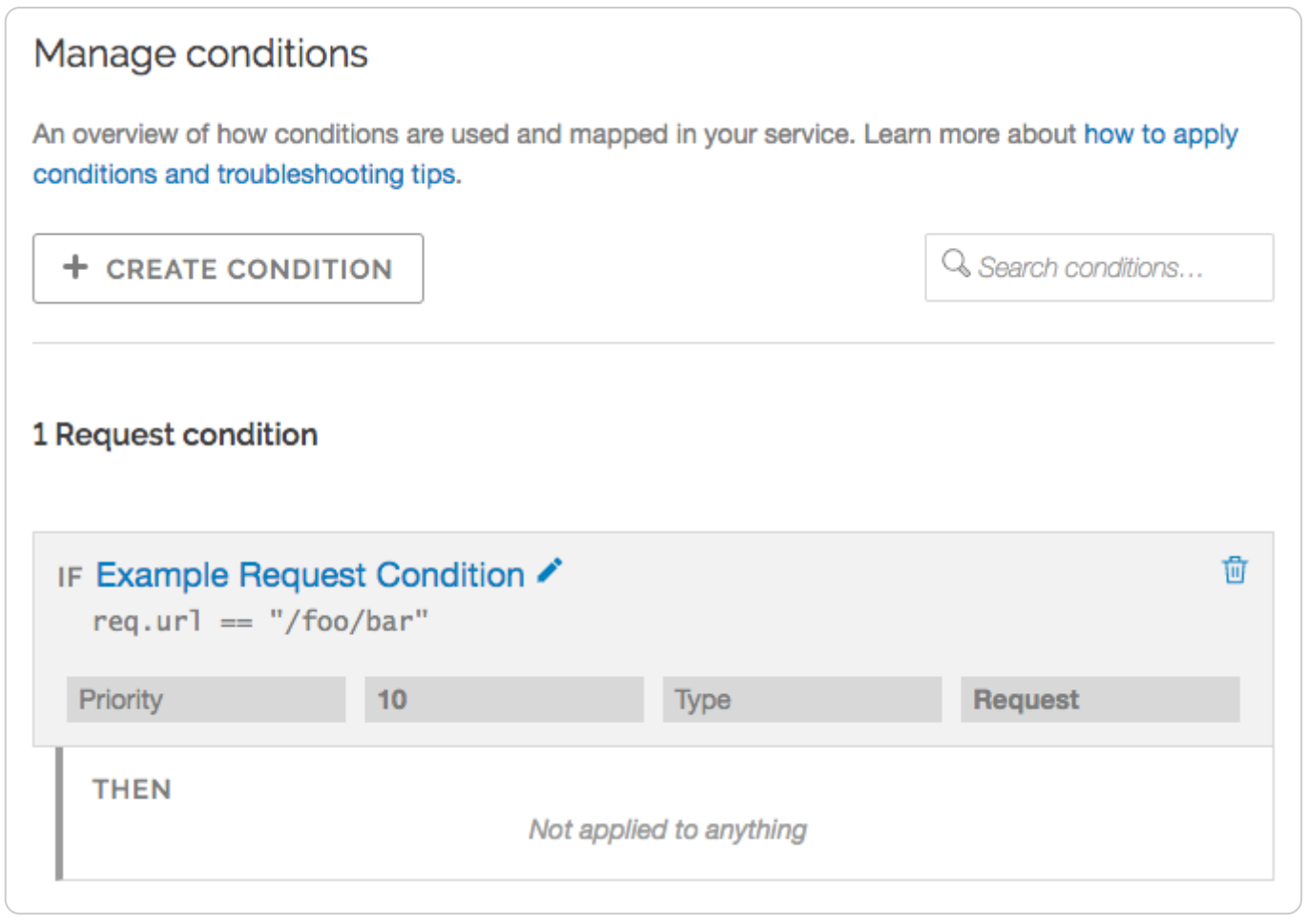
Conditions on the Manage conditions page

The Manage conditions page provides an overview of all the conditions currently available to your service. You can see at a glance which conditions are mapped to configuration objects. It allows you to create new conditions and search for existing ones.

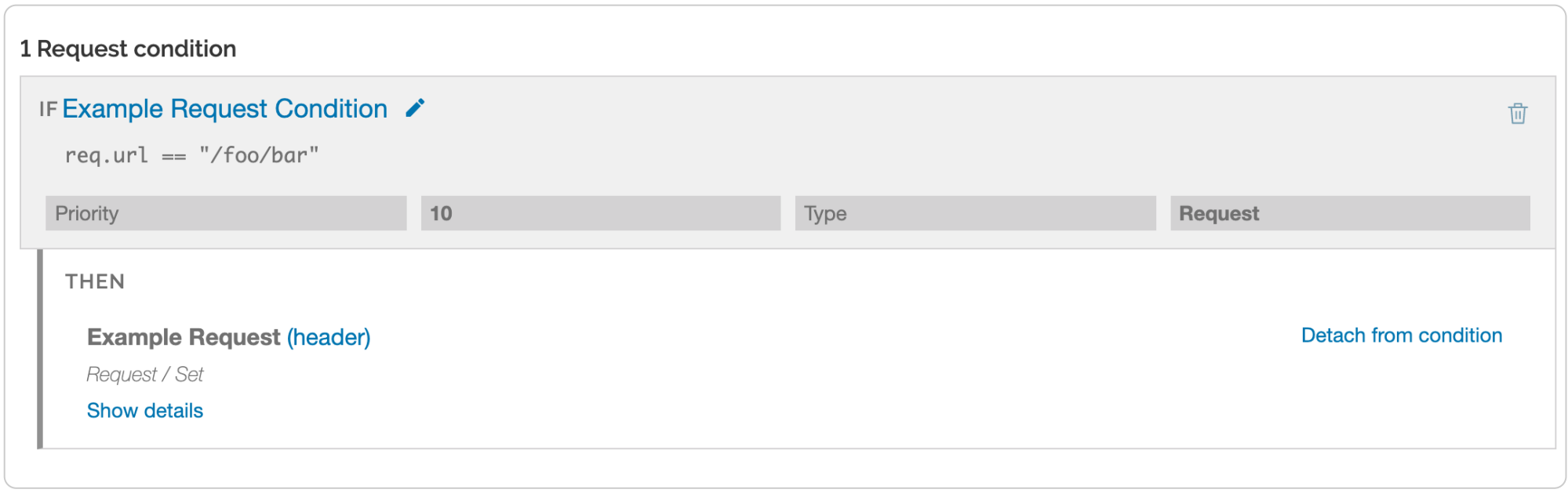
To view conditions on the Manage conditions page:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Conditions** link. A list of all conditions for your service appears.

For example, this service has one request condition available:

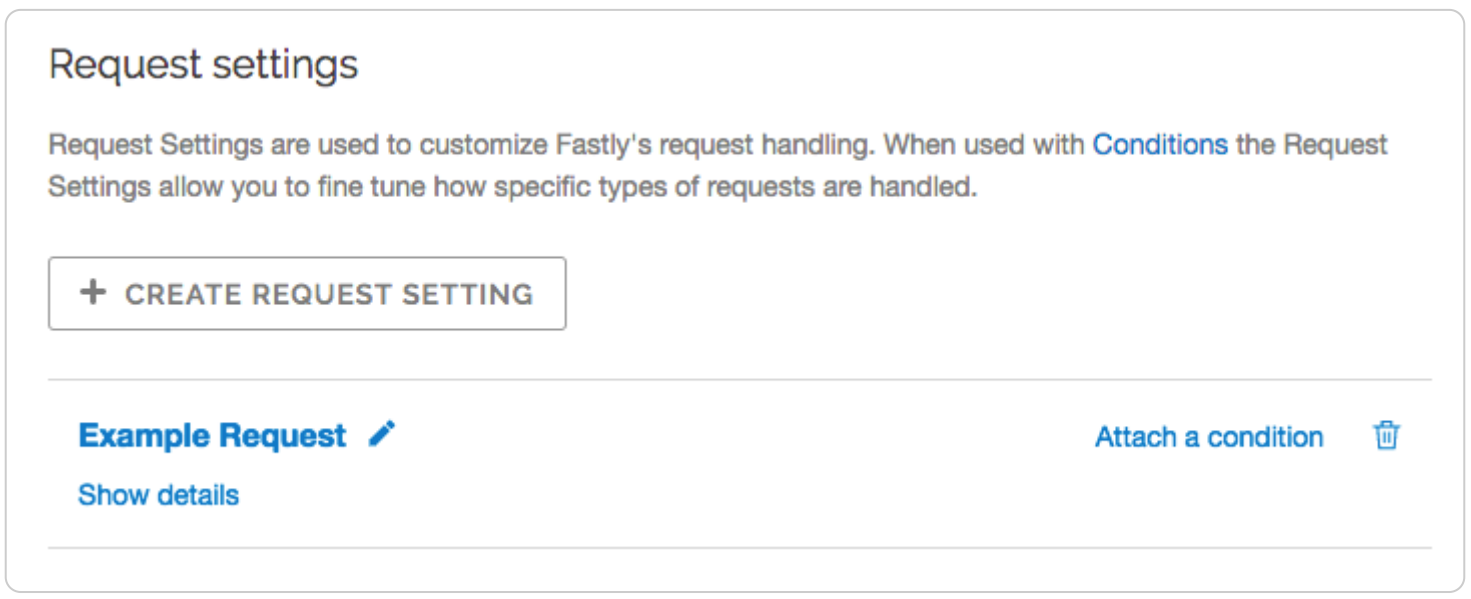


The Example Request Condition shown above currently isn't applied to a configuration object (as indicated by "Not applied to anything"). If it was, it would instead appear similar to this:

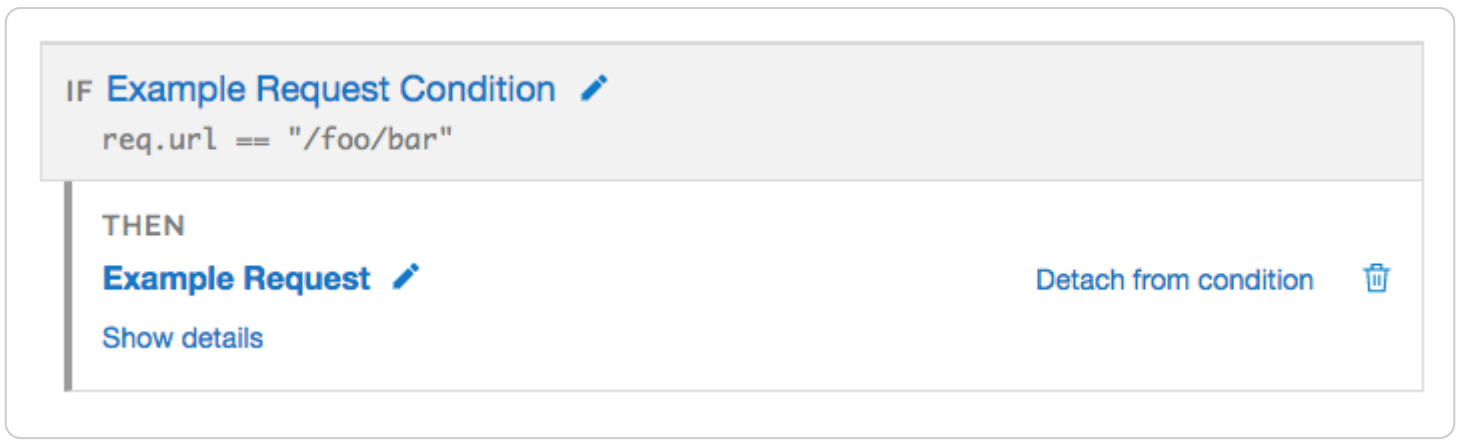


Conditions attached directly to a configuration object

Configuration objects appear differently in the web interface when conditions are attached to them. For example, this request setting has no condition attached to it:



Once you click the Attach a condition link to create a new condition or attach an existing condition, however, the web interface changes how the configuration object appears:



By default, configuration objects hide the majority of details for any attached conditions. You can unhide those details by clicking the Show details link. When expanded, the details vary depending on the type of condition.

Parts of a properly configured condition

Conditions require only a few parameters, making them appear deceptively simple. Specifically, they require:

- a **Type** parameter that classifies the condition being added. If added via the Manage conditions page, the type can always be manually selected. If added via the Attach a condition link on a configuration object, the type is automatically applied whenever possible.
- a **Name** parameter that serves human-readable identifier of the condition.
- an **Apply if** statement containing the logical expression to execute in VCL to determine if condition resolves as True or False.

Most problems with conditions occur in the Apply if parameter.

Performing matches on basic logical expressions

Properly configured conditions can perform matches on complicated logical expressions specified in the Apply if parameter. For example:

This logical expression ...	Matches when ...
`client.ip == "127.0.0.1"`	The client requesting a resource on your service has the IP `127.0.0.1`.
`req.http.host == "example.com"`	The Host header of the incoming request is `example.com`.
`req.method == "POST" && req.url ~ "^/api/articles/"`	The request is a POST and the URL begins with `/api/articles/`.

The `client.ip`, `req.http.host`, `req.method`, and `req.url` conditions shown above all represent configuration variables in VCL.

Using operators to perform matches on complex logical expressions

You could also get creative and create a more complex condition used by Fastly that might have an Apply if parameter that looked like this:

```
req.http.host == "www.example.com" && (req.url !~ "/foo" && req.url !~ "/bar" && req.url !~ "^/baz")
```

This condition tells the cache server that the Host should equal `www.example.com` and the URL cannot match `/foo` or `/bar` or `/baz`. You might use this type of condition when you have multiple variables or options and want to fine-tune your results. In this example, you are indicating that you don't want URLs that contain `foo`, `bar`, or `baz` by using the following operators:

This operator ...	Does this ...
`( )`	groups expressions and restricts alteration to part of the regex
`&&`	ensures each equation is true
`!~`	excludes any URLs that include the specified variables

An example of adding conditions

The scenario: You want to add a new origin server that handles a specific portion of your API requests. Some requests to this API must be cached differently than other requests to your API, so you want to set special headers for specific types of requests. Specifically, you don't want your new origin server to cache PUT, POST, or DELETE requests because they're special for this particular API and send back extra, time dependent, meta-information in each response. And finally, you want to track the effectiveness of doing this. To accomplish all of this using conditions via the Fastly web interface, you would:

1. Create a new origin server to handle the special API traffic.
2. Create a new condition that tells the cache how to route some of the API requests to that origin server.
3. Create a new cache setting object to ensure the origin server caches only the correct responses.
4. Create a new condition that specifies when the cache settings object should be applied.
5. Create a new header to track the specific type of API requests.
6. Create a new response condition to make sure that the header is only set on specific type of request.
7. Check your work.

Create a new origin server

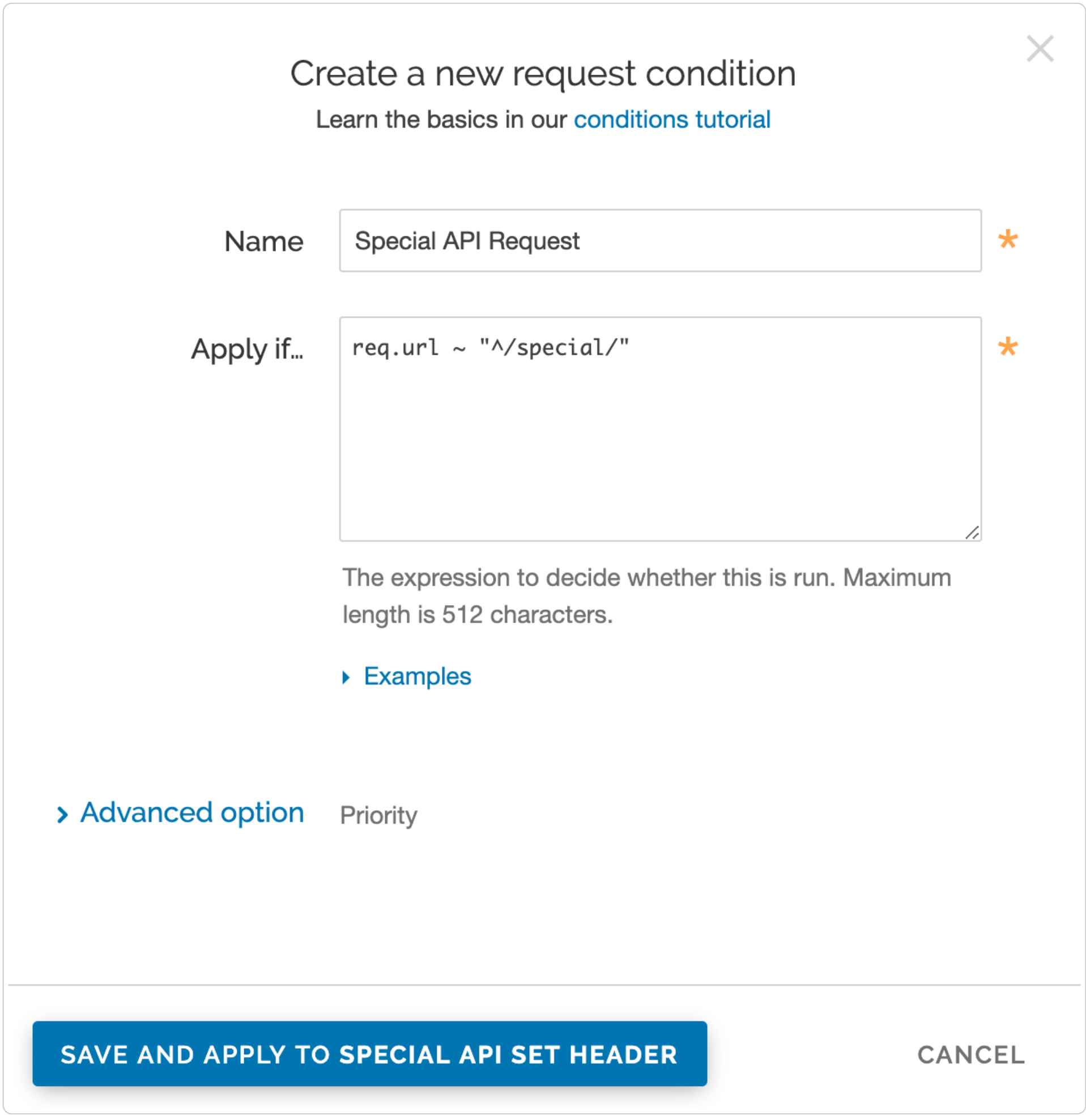
To create a new origin server that will handle the special API traffic, follow the instructions for working with hosts. You'll add specific details about your API server when you fill out the **Create a host** fields:

- In the **Name** field, enter a name for your API server (for example, `Special API Server`).
- In the **Address** field, enter the IP address (or hostname) of the API server.

Create a request condition

Once you've created a new origin server to handle the special API traffic, tell the cache how to route requests to this origin server by creating a request condition.

1. In the **Hosts** area, click the **Attach a condition** link next to the name of the origin server you just created. The Add a condition to window appears.
2. You can either select an available condition or you can click the **Create a new request condition** button. The Create a new request condition window appears.

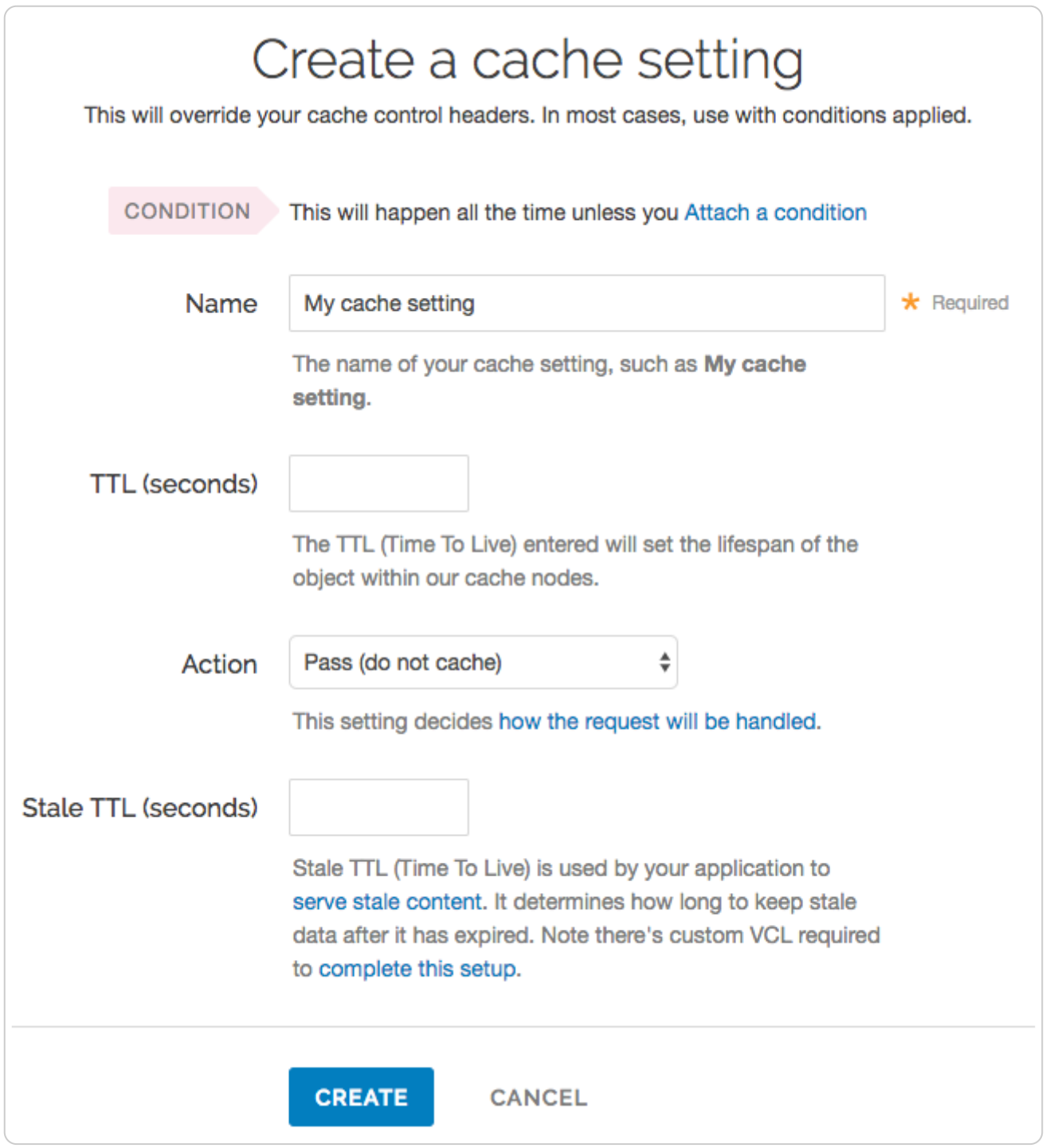


3. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter a descriptive name for the new condition (for example `Special API Request`).
   - In the **Apply if** field, enter the appropriate request condition that will be applied (for example, `req.url ~ "^/special/"` could address all requests related to the special API server).
4. Click the **Save and apply to** button to create the new condition for the host.

Create a cache settings object

Requests are now are being properly routed to the new origin server. Next, create a cache settings object to ensure Fastly doesn't cache any responses from PUT, POST, or DELETE requests. They're special for this particular API and send back extra, time dependent, meta-information in each response.

1. Click the **Settings** link. The Settings page appears.
2. In the **Cache Settings** area, click the **Create cache setting** button. The Create a cache setting page appears.

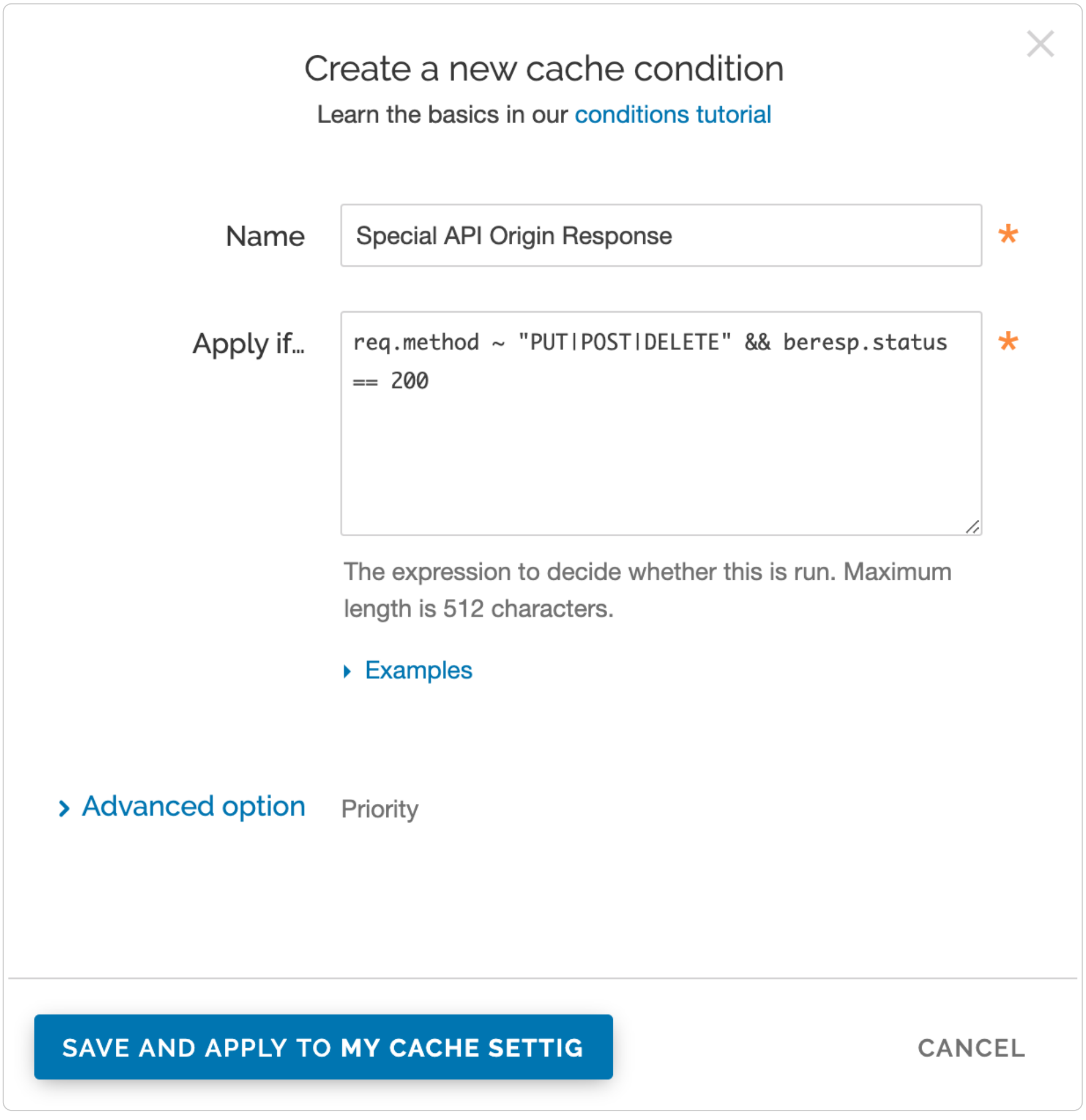


3. Fill out the **Create a cache setting** fields as follow:
   - In the **Name** field, enter a descriptive name for the new cache settings.
   - Leave the **TTL (seconds)** field set to its default value.
   - From the **Action** menu, select **Pass (do not cache)**.
   - Leave the **Stale TTL (seconds)** field set to its default value.
4. Click the **Create** button.

Create and apply a condition to the cache settings object

Create a new condition that specifies when the cache settings object should be applied.

1. In the **Cache Settings** area, click the **Attach a condition** link next to the name of the cache setting you just created. The Add a condition to window appears.
2. Click **Create a new cache condition** button. The Create a new cache condition window appears.

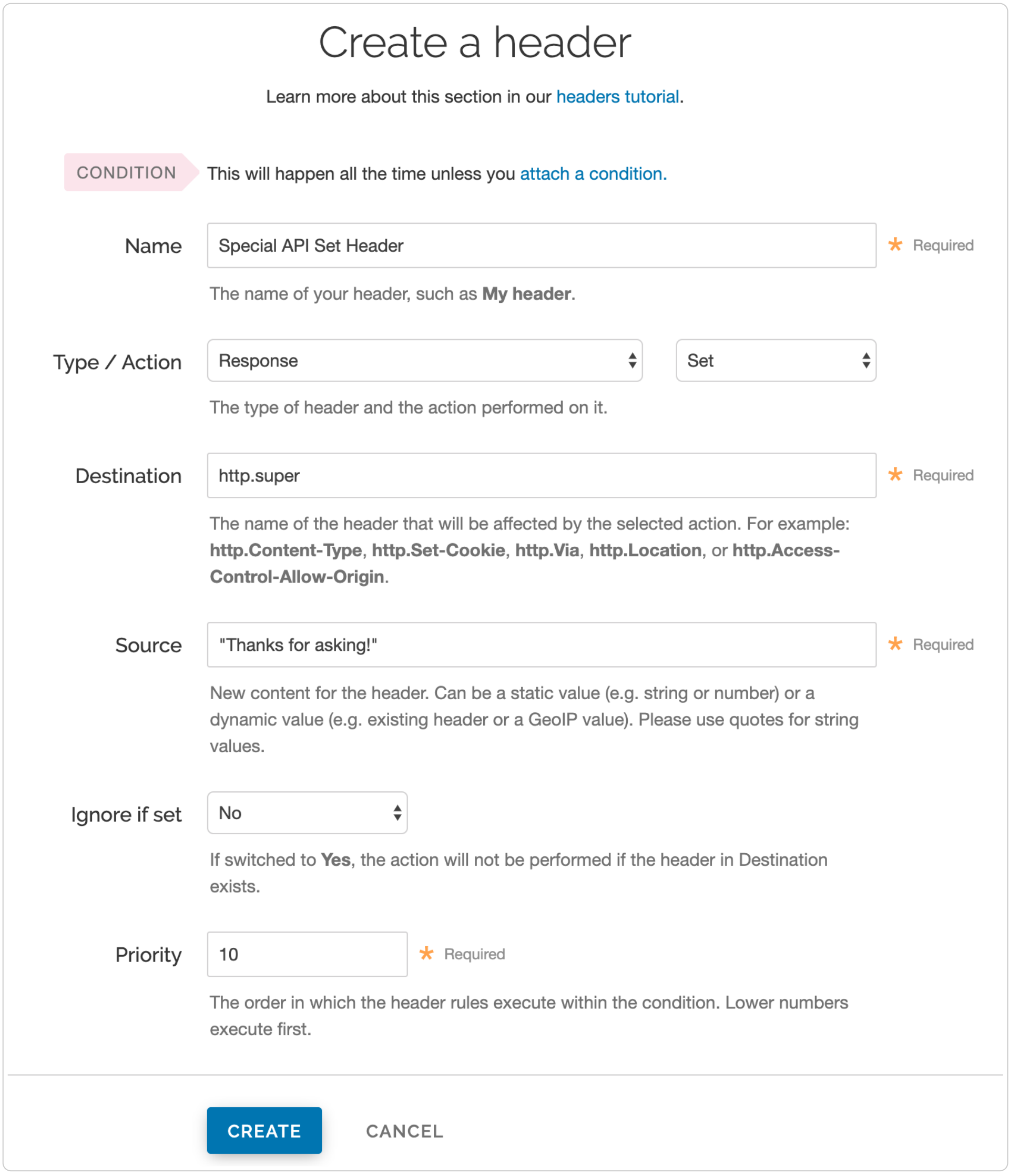


3. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, enter a descriptive name for the new condition (for example, `Special API Origin Response`).
   - In the **Apply if** field, enter the appropriate request condition that will be applied (for example, `req.method ~ "PUT|POST|DELETE" && beresp.status == 200`).
4. Click the **Save and apply to** button to create the new condition for the cache setting.

Create a new header

To make sure you can track the effectiveness the new API, create a new header so you can use it to gather information about the special API requests as they happen.

1. Click the **Content** link. The Content page appears.
2. In the **Headers** area, click the **Create header** button to create a new header. The Create a header page appears.

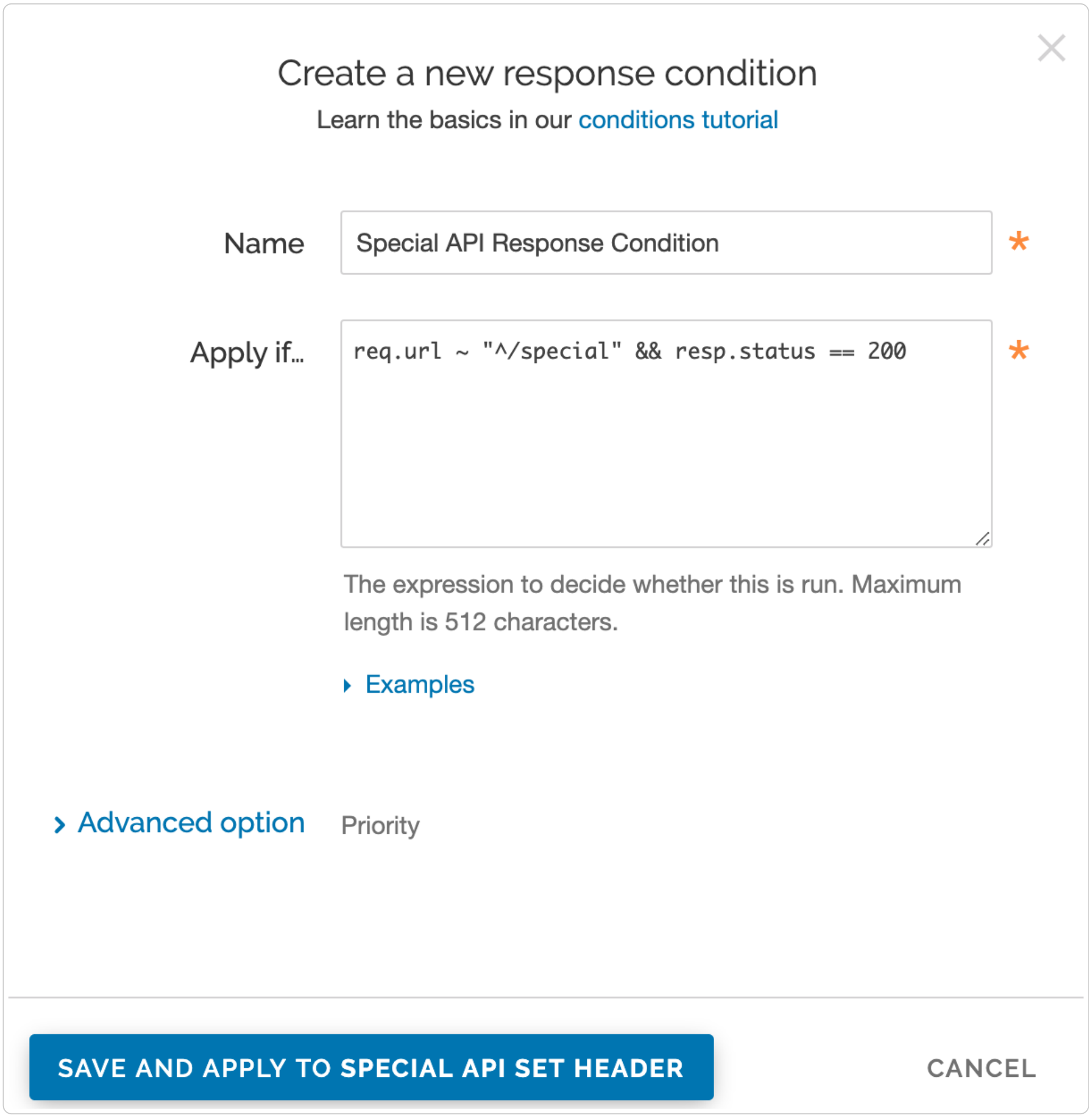


3. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a descriptive name for the new header (for example, `Special API Set Header`).
   - From the **Type** menu, select **Response** and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter the name of the header that will be affected by the action (for example, `http.super`).
   - In the **Source** field, enter a description of the source where the content for this header comes from (for example, `"Thanks for asking!"`).
   - Leave the **Ignore if set** and **Priority** fields set to their default settings.
4. Click **Create**.

Create a response condition for the new header

Once the header is created, create an associated condition to ensure this header is only set on that special type of request.

1. In the **Headers** area, click the **Attach a condition** link next to the name of the new header you just created. The Create a new response condition window appears.



2. Fill out the **Create a new response condition** fields as follows:
   - In the **Name** field, enter a descriptive name for the new condition (for example, `Special API Response Condition`).
   - In the **Apply if** field, enter the appropriate request condition that will be applied (for example, `req.url ~ "^/special" && resp.status == 200`).
3. Click the **Save and apply to** button to create the new condition for the header.

Check your work

Before activating the configuration, review the generated VCL to see how Fastly converted the objects and conditions into actual VCL. For the example shown above, the VCL for the request condition appears as:

```
# Condition: Special API Request Prio: 10
if (req.url ~ "^/special/") {
  set req.backend = F_Special_API_Server;
}
#end condition
```

The cache settings and condition VCL appears as:

```
if (req.method ~ "PUT|POST|DELETE" && beresp.status == 200) {
  set beresp.ttl = 0s;
  set beresp.grace = 0s;
  return(pass);
}
```

And the new header response condition VCL appears as:

```
# Condition Special API Response Condition Prio: 10
if (req.url ~ "^/special" && resp.status == 200) {

  # Header rewrite Special API Set Header: 10
  set resp.http.super = "Thanks for asking!";
}
```

As you become more familiar with the VCL syntax and programming, look at the generated VCL to see if the configuration is doing what you think it is doing (most VCL is pretty simple once you know what the variables are referring to).

---

§

These articles describe how to create your own VCL files with specialized configurations.

https://docs.fastly.com/en/guides/configuration#_custom-vcl

About VCL Snippets

Last updated: 2019-05-13

https://docs.fastly.com/en/guides/about-vcl-snippets

VCL Snippets are short blocks of VCL logic that can be included directly in your service configurations. They're ideal for adding small sections of code when you don't need more complex, specialized configurations that sometimes require custom VCL. Fastly supports two types of VCL Snippets:

- Regular VCL Snippets get created as you create versions of your Fastly configurations. They belong to a specific service and any modifications you make to the snippet are locked and deployed when you deploy a new version of that service. You can treat regular snippets like any other Fastly objects because we continue to clone them and deploy them with a service until you specifically delete them. You can create regular snippets using either the web interface or via the API.
- Dynamic VCL Snippets can be modified and deployed any time they're changed. Because they are versionless objects (much like Edge Dictionaries or ACLs at the edge), dynamic snippets can be modified independently from service changes. This means you can modify snippet code rapidly without deploying a service version that may not be ready for production. You can only create dynamic snippets via the API.

Limitations of VCL Snippets

- Snippets are limited to 1MB in size by default. If you need to store snippets larger than the limit, contact support@fastly.com.
- Snippets don’t currently support conditions created through the web interface. You can, however, use `if` statements in snippet code.
- Snippets cannot currently be shared between services.

Accept-Language header VCL features

Last updated: 2018-05-11

https://docs.fastly.com/en/guides/accept-language-header-vcl-features

Fastly provides a number of extensions to VCL, including functions to parse and normalize the `Accept-Language` header.

Language lookup

We've implemented Lookup functionality as defined by RFC 4647, section 3.4.

Syntax

`accept.language_lookup(<available languages>, <default>, <priority list>)`

Argument	Explanation
`available languages`	A colon-separated list of languages to choose from. Typically the languages that the origin can provide. For example: `en:de:fr:pt:es:zh-CN`
`default`	The default language to return if none from the `priority list` match. For example: `en`
`priority list`	The `Accept-Language` header. A comma-separated list of languages, optionally accompanied by weights (q-values). For example: `pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4`

Return values

The best matching language (as per the RFC) is returned. If none are found, the default language is returned, unless a weight of zero (`q=0`) was indicated by the priority list, in which case `NULL` is returned.

Examples

```
set req.http.Normalized-Language =
  accept.language_lookup("en:de:fr:pt:es:zh-CN", "en", req.http.Accept-Language);
```

The above would result in `Normalized-Language: pt` given an `Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4` header.

`accept.language_lookup("en", "nl", "en-GB")` results in `en`, as each subtag is removed and the match retried when a tag does not match.

`accept.language_lookup("en:nl", "nl", "en-GB,nl;q=0.5")` results in `en` still, even if `nl` is a more exact match, because the q-value of nl is lower, and that has precedence. Exactness just does not come into the equation.

`accept.language_lookup("en-US:nl", "nl", "en-GB,nl;q=0.5")` results in `nl`, because subtags are not removed from the available languages, only from language tags on the priority list.

`accept.language_lookup("en-US:nl", "nl", "en-us,nl;q=0.5")` results in `en-US`, as the lookup is case insensitive.

`accept.language_lookup("en-US:nl", "nl", "en-GB,nl;q=0")` results in `NULL` (the value, not a string) since `en-GB` and `en` do not match, and `nl` (the default) is listed as unacceptable.

If `q=0` for the default language is to be ignored, the following VCL can be used:

```
set req.http.Normalized-Language =
  accept.language_lookup("en-US:nl", "nl", req.http.Accept-Language);
if (!req.http.Normalized-Language) {
  # User will get Dutch even if he doesn't want it!
  # (Because none of our languages were acceptable)
  set req.http.Normalized-Language = "nl";
}
```

Language filter (Basic)

We've implemented Basic Filtering functionality as defined by RFC 4647, section 3.3.1. The implementation is not exact when the wildcard tag (`*`) is used. If a wildcard is encountered and no matches have been found yet, the default is returned. If there are matches, those are returned and the remainder of the priority list is ignored. There is no implementation of Extended Filtering, but if

you are in need you could always file a feature request with Support.

Syntax

`accept.language_filter_basic(<available languages>, <default>, <priority list>, <limit>)`

Argument	Explanation
`available languages`	A colon-separated list of languages choose from. Typically the languages that the origin can provide. For example: `en:de:fr:pt:es:zh-CN`
`default`	The default language to return if none from the `priority list` match. For example: `en`
`priority list`	The `Accept-Language` header. A comma-separated list of languages, optionally accompanied by weights (q-values). For example: `pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4`
`limit`	The maximum amount of languages returned.

Return values

The best matching language (as per the RFC) is returned. If none are found, the default language is returned, unless a weight of zero (`q=0`) was indicated by the priority list, in which case `NULL` is returned.

Examples

```
set req.http.Filtered-Language =
  accept.language_filter_basic("en:de:fr:pt:es:zh-CN", "en", req.http.Accept-Language, 2);
```

The above would result in `Filtered-Language: pt,en` given an `Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4` header.

`accept.language_filter_basic("en", "nl", "en-GB", 2)` results in `en`, as each subtag is removed and the match retried when a tag does not match.

`accept.language_filter_basic("en:nl", "nl", "en-GB,nl;q=0.5", 2)` results in `en,nl`, even if `nl` is a more exact match, because the q-value of nl is lower, and that has precedence. Exactness just does not come into the equation.

`accept.language_filter_basic("en-US:nl", "nl", "en-GB,nl;q=0.5", 2)` results in `nl`, because subtags are not removed from the available languages during the search.

`accept.language_filter_basic("en-US:nl", "nl", "en-us,nl;q=0.5", 2)` results in `en-US,nl`, as the lookup is case insensitive.

`accept.language_filter_basic("en-US:nl", "nl", "en-GB,nl;q=0", 2)` results in `NULL` (the value, not a string) since `en-GB` and `en` do not match, and `nl` (the default) is listed as unacceptable.

If `q=0` for the default language is to be ignored, the following VCL can be used:

```
set req.http.Filtered-Language =
  accept.language_filter_basic("en-US:nl", "nl", req.http.Accept-Language, 2);
if (!req.http.Filtered-Language) {
  # User will get Dutch even if he doesn't want it!
  # (Because none of our languages were acceptable)
  set req.http.Filtered-Language = "nl";
}
```

`accept.language_filter_basic("en:nl:de:fr", "nl", "en-GB,*;q=0.5", 2)` results in `en` and `accept.language_filter_basic("en:nl:de:fr", "nl", "*", 2)` results in `nl`.

Authenticating before returning a request

Last updated: 2018-10-18

https://docs.fastly.com/en/guides/authenticating-before-returning-a-request

Performing authentication before returning a request is possible if your authentication is completely header-based and you do something like the following using custom VCL:

```
sub vcl_recv {

  /* unset state tracking header to avoid client sending it */
  if (req.restarts == 0) {
    unset req.http.X-Authed;
  }

  if (!req.http.X-Authed) {
    /* stash the original URL and Host for later */
    set req.http.X-Orig-URL = req.url;

    /* set the URL to what the auth backend expects */
    set req.url = "/authenticate";

    /* Auth requests won't be cached, so pass */
    return(pass);
  }

  if (req.http.X-Authed == "true") {
    /* were authed, so proceed with the request */
    /* reset the URL */
    set req.url = req.http.X-Orig-URL;

  } else {
    /* the auth backend refused the request, so 403 the client */
    error 403;
  }

#FASTLY recv

  ...etc...
}

sub vcl_deliver {

  /* if we are in the auth phase */
  if (!req.http.X-Authed) {

    /* if we got a 5XX from the auth backend, we should fail open */
    if (resp.status >= 500 && resp.status < 600) {
      set req.http.X-Authed = "true";
    }

    if (resp.status == 200) {

      /* the auth backend responded with 200, allow the request and restart */
      set req.http.X-Authed = "true";
    } else if (resp.status == 401) {

      return(deliver);

    } else {

      /* the auth backend responded with non-200, deny the request and restart */
      set req.http.X-Authed = "false";
    }

    restart;
  }

#FASTLY deliver

  ...etc...
}
```

NOTE

Be sure to change `/authenticate` to whatever your authentication endpoint is.

WARNING

Caching authentication might result in users receiving responses intended for other authenticated users. For example, if you cache the response from the `/authenticate` endpoint for User A, User B could receive the same response when logging in.

If you feel like you can cache the authentication, then add the appropriate headers to the hash in `vcl_hash` and `return(lookup)` instead of `(pass)`.

Basic authentication

Last updated: 2021-08-16

https://docs.fastly.com/en/guides/basic-authentication

Basic authentication is a simple way of protecting a website at the edge. Users enter a username and password combination to access pages protected by basic authentication. You can use basic authentication to restrict access to low-risk assets like testing and staging environments.

WARNING

Basic authentication shouldn't be used to restrict access to sensitive information. See the security considerations section for more information.

Implementing basic authentication

Follow our HTTP basic auth example to implement basic authentication using custom VCL or Compute@Edge.

Using basic authentication with GCS

To use basic authentication with Google Cloud Storage (GCS) as a origin server, add a request header to delete the `http.Authorization` header and prevent it from being sent to GCS. That header causes GCS to respond with a "Not Authorized" message instead of your request.

Security considerations

There are several security considerations you should take into account before using basic authentication:

- Basic authentication can't protect high-risk information. Don't use it to restrict access to sensitive information.
- If you're not using TLS, the password will be transmitted over the wire in Base64 encoding. The encoded string could easily be captured using an application like Wireshark and converted to plaintext.
- The password is cached by the user's web browser, and it can be permanently saved by the user's web browser.

Using access control lists

As an alternative to basic authentication, you can use access control lists (ACLs) to restrict access to your assets by allowlisting a set of IP addresses. To allowlist IP addresses with an ACL, add custom VCL to Fastly's boilerplate VCL.

```
# Who is allowed access ...
acl local {
    "localhost";
    "192.168.1.0"/24; /* and everyone on the local network */
    ! "192.168.1.23"; /* except for the dial-in router */
}
```

See our ACL guides for more information.

Custom responses that don't hit origin servers

Last updated: 2019-09-11

https://docs.fastly.com/en/guides/custom-responses-that-dont-hit-origin-servers

Fastly can send custom responses for certain requests that you don't want to hit your origin servers.

Creating a quick response

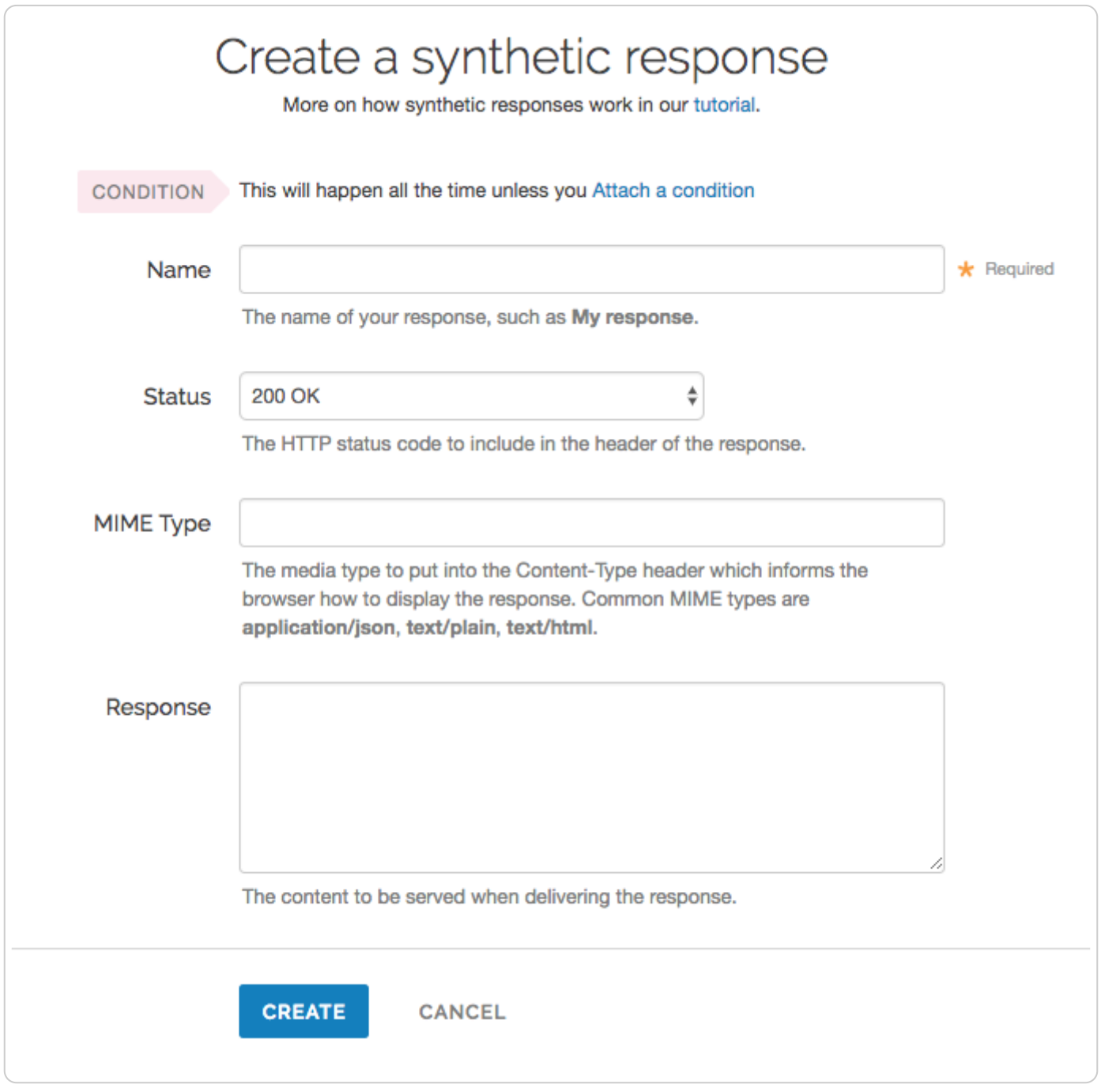
Fastly provides features that allow you to quickly enable and configure responses for a robots.txt file and 404 and 503 errors. For more information, see our guides on creating and customizing a robots.txt file and creating error pages with custom responses.

Creating an advanced response

You can create an advanced response to specify the HTTP status code, MIME type, and content of the response. For example, if you wanted to restrict caching to a URL subtree that contains images and scripts, you could configure Fastly to return an `HTTP 404 Not Found` response to requests for anything other than `/Content/*` or `/Scripts/*`. To illustrate how to implement this example, we'll show you how to create a response and corresponding request condition.

Follow these instructions to create an advance response:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Set up advanced response** button. The Create a synthetic response page appears.

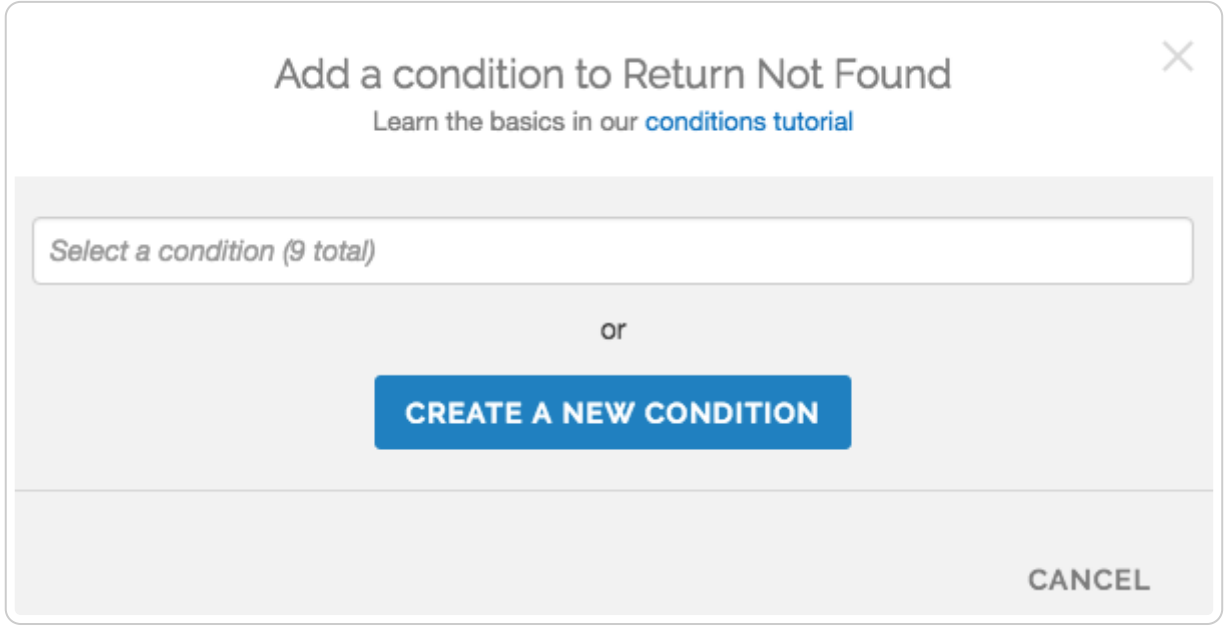


6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, enter a human-readable name for the response. For example `Return Not Found`.
   - From the **Status** menu, select an HTTP code to return to the client. For example, `404 Not Found`.
   - In the **MIME Type** field, enter the MIME type of the response. For example, `text/html`.
   - In the **Response** field, enter the plaintext or HTML content to return to the client. For example `Page not found`.
7. Click the **Create** button to create the response. The new response appears in the Responses area of the Content page.

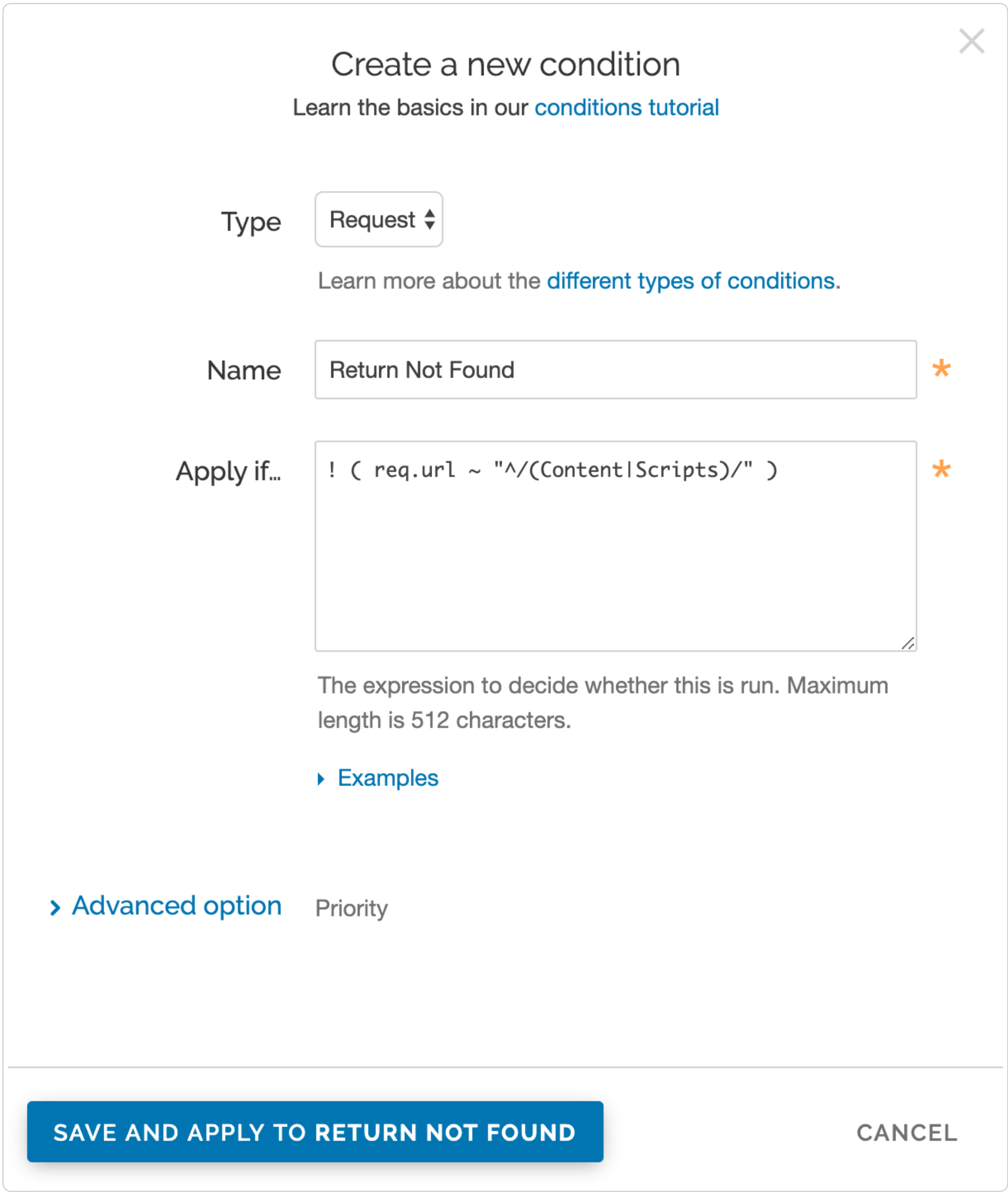
Creating the request condition

Follow these instructions to attach a request condition to the response you just created:

1. Click the **Attach a condition** link next to the response that you just created. The Add a condition window appears.



2. Click the **Create a new condition** button. The Create a new condition window appears.



3. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select the type of condition you want to create.
   - In the **Name** field, enter a human-readable name for the condition. For example, `Return Not Found`.
   - In the **Apply if** field, enter the request condition you want inserted into a VCL if statement. For example, `! ( req.url ~ "^/(Content|Scripts)/" )`. See below for more examples of request conditions.
4. Click the **Save and apply to** button. The Responses area now displays the condition that must be met in order for your response to begin being used.

5. Click the **Activate** button to deploy your configuration changes.

Example request conditions

Respond only if URLs don't match a certain mask, in this case `/Content/*` or `/Scripts/*`:

```
! (req.url ~ "^/(Content|Scripts)/")
```

Respond only if URLs match `/secret/*` or are Microsoft Word or Excel documents (`*.doc` and `*.xls` file extensions):

```
! (req.url ~ "^/secret/" || req.url ~ "\.(xls|doc)$")
```

Ignore POST and PUT HTTP requests:

```
req.method == "POST" || req.method == "PUT"
```

Deny a spider or crawler using `user-agent` `"annoying_robot"`:

```
req.http.user-agent ~ "annoying_robot"
```

Prevent a specific IP from connecting, in this case the IP `225.0.0.1`:

```
client.ip == "225.0.0.1"
```

Use geographic variables to block traffic from a specific location (e.g., China):

```
client.geo.country_code == "CN"
```

Match the `client.ip` against a CIDR range, such as `240.24.0.0/16` (this requires first creating an ACL object in VCL):

```
client.ip ~ ipRangeObject
```

Enabling URL token validation

Last updated: 2021-06-30

https://docs.fastly.com/en/guides/enabling-url-token-validation

Token validation allows you to create URLs that expire. Tokens are generated within your web application and appended to URLs in a query string. Requests are authenticated at Fastly's edge instead of your origin server. When Fastly receives a request for the URL, the token is validated before serving the content. After a configurable period of time, the token expires.

Adding custom VCL

To enable token validation, you'll need to create a Varnish configuration named `vcl_recv` and add the following example code to it.

> **TIP**
>
> You can also use the custom VCL and Compute@Edge examples in our time-limited URL tokens documentation.

```
# only do this once per request
if (fastly.ff.visits_this_service == 0 && req.restarts == 0) {
  declare local var.token STRING;
  declare local var.token_expiration STRING;
  declare local var.token_signature STRING;
  declare local var.to_sign STRING;

  # extract and remove the token
  set var.token = querystring.get(req.url, "token");
  set req.url = querystring.filter(req.url, "token");

  # make sure there is a token
  if (var.token == "") {
    error 403;
  }

  # make sure there is a valid expiration and signature
  if (var.token !~ "^(\d{10,11})_([a-f0-9]{40})$") {
    error 403;
  }

  # extract token expiration and signature
  set var.token_expiration = re.group.1;
  set var.token_signature = re.group.2;

  # calculate string to sign
  set var.to_sign = req.url + var.token_expiration;

  # make sure the signature is valid
  if (!digest.secure_is_equal(var.token_signature, regsub(digest.hmac_sha1(digest.base64_decode("YOUR%SECRET%KEY%IN%BASE64%HERE"), var.to_sign), "^0x", ""))) {
    error 403;
  }

  # make sure the expiration time has not elapsed
  if (time.is_after(now, std.integer2time(std.atoi(var.token_expiration)))) {
    error 410;
  }
}
```

> **IMPORTANT**

Be sure to generate your own key for use with this VCL (the example key shown here will intentionally cause an error). Due to limitations in VCL, the binary form of the key should not contain NUL (0×00) bytes. In Linux, use the command:

```
$ while (b=$(openssl rand -base64 32) ; echo $b; echo $b | base64 -d | hd | grep " 00 " > /dev/null); do :; done | t
ail -1
```

In macOS, use the command:

```
$ while (b=$(openssl rand -base64 32) ; echo $b; echo $b | base64 -D | hexdump | grep " 00 " > /dev/null); do :; don
e | tail -1
```

The custom VCL code above checks for two things:

- It verifies the signature supplied matches the signature of the token
- It ensures the current time is less than the expiration time specified in the token

If the signature is invalid, Varnish returns a 403 response. If the signature is valid but the expiration time has elapsed, Varnish returns a 410 response. The different response codes are helpful for debugging.

The token information

A token is expected in the `?token=` GET parameter. Tokens take the format `[expiration]_[signature]` and look like this:

```
1441307151_4492f25946a2e8e1414a8bb53dab8a6ba1cf4615
```

The full request URL with the token looks like this:

```
http://www.example.com/foo/bar.html?token=1441307151_4492f25946a2e8e1414a8bb53dab8a6ba1cf4615
```

The signature validation

The key found in `digest.hmac_sha1` can be any string. The one in this example was generated with the following command:

```
$ openssl rand -base64 32
```

The example key `YOUR%SECRET%KEY%IN%BASE64%HERE` will intentionally cause an error if you use it. You must replace it with your own randomly generated secret key.

WARNING

Anyone who learns your key can bypass your token validation, so it's critical that you keep it secret.

Configuring your application

You'll need to write custom code in your application to generate tokens and authenticate with Varnish. We provide examples in our token functions repository on GitHub. Review the examples in the repository to learn how to generate custom tokens within your application.

Testing

To test your configuration, append a token generated by your application to a URL in a query string. For example:

`http://www.example.com/foo/bar.html?token=1441307151_4492f25946a2e8e1414a8bb53dab8a6ba1cf4615`

If the token is valid, you will receive a normal response. If it is invalid, you will receive a 403 response.

Troubleshooting NUL bytes

You should verify that your secret key is devoid of NUL bytes. If the Base64-decoded string contains a NUL byte (0×00), then that byte and any bytes following it will not be included in the response. See Base64 decoding for more information.

Guide to VCL

Last updated: 2021-04-20

https://docs.fastly.com/en/guides/guide-to-vcl

Fastly's Edge Cloud services use the Fastly Varnish Configuration Language (VCL), a scripting language used to configure and add logic to Varnish caches. Fastly VCL allows you to generate and compile changes to your Fastly services. These changes can be distributed to all Fastly caches worldwide and then loaded and activated without requiring maintenance windows or service downtime. Fastly VCL is generated automatically per your service configurations specified via the web interface.

VCL and what you can do with it

We allow you to create your own VCL files with specialized configurations. Your custom VCL files can be uploaded into Fastly caches and activated.

You can also mix and match custom VCL and Fastly VCL, using them together at the same time. You will never lose the options on the Fastly web interface when you use custom VCL, but keep in mind that custom VCL always takes precedence over any VCL generated by the web interface. Be mindful of where your custom VCL sits in the default VCL.

IMPORTANT

Personal data should not be incorporated into VCL. Our Compliance and Law FAQ describes in detail how Fastly handles personal data privacy.

Embedding inline C code in VCL

Currently, we don't provide embedded C access to our users. Fastly is a shared infrastructure. By allowing the use of inline C code, we could potentially give a single user the power to read, write to, or write from everything. As a result, our varnish process (i.e., files on disk, memory of the varnish user's processes) would become unprotected because inline C code opens the potential for users to do things like crash servers, steal data, or run a botnet.

We appreciate feedback from our customers. If you are interested in a feature that requires C code, contact support@fastly.com. Our engineering team looks forward to these kinds of challenges.

Where to learn more about VCL and Varnish

Fastly's Developer Hub provides a reference of Fastly VCL for programming custom edge logic on VCL services. You can also discover more about learning to build on the Fastly platform using VCL and the current best practices involved.

The official Varnish documentation is a good place to start when looking for online information. In addition, Varnish Software, who provides commercial support for Varnish, has written a free online book.

Roberto Moutinho's book *Instant Varnish Cache* also provides information.

Isolating header values without regular expressions

Last updated: 2020-09-24

https://docs.fastly.com/en/guides/isolating-header-values-without-regular-expressions

Fastly supports the ability to extract header subfield values without regular expressions in a human-readable way. *Headers subfields* are headers with a body syntax style similar to `value1=123value123; testValue=asdf_true; staff_user=true;` or `max-age=0, surrogate-control=3600` These headers include Cookie, Set-Cookie, Cache-Control, or a custom header. Fastly allows you to isolate these key values with the following syntax:

```
req.http.Header-Name:key-name
```

For example, if a `Cookie` request from a client was `value1=123value123; testValue=asdf_true; staff_user=true;`, you could isolate the `staff_user` value using this logic:

```
set req.http.Staff-User = req.http.Cookie:staff_user;
```

The same can be accomplished by using the subfield function:

```
set req.http.Staff-User = subfield(req.http.Cookie, "staff_user", ";");
```

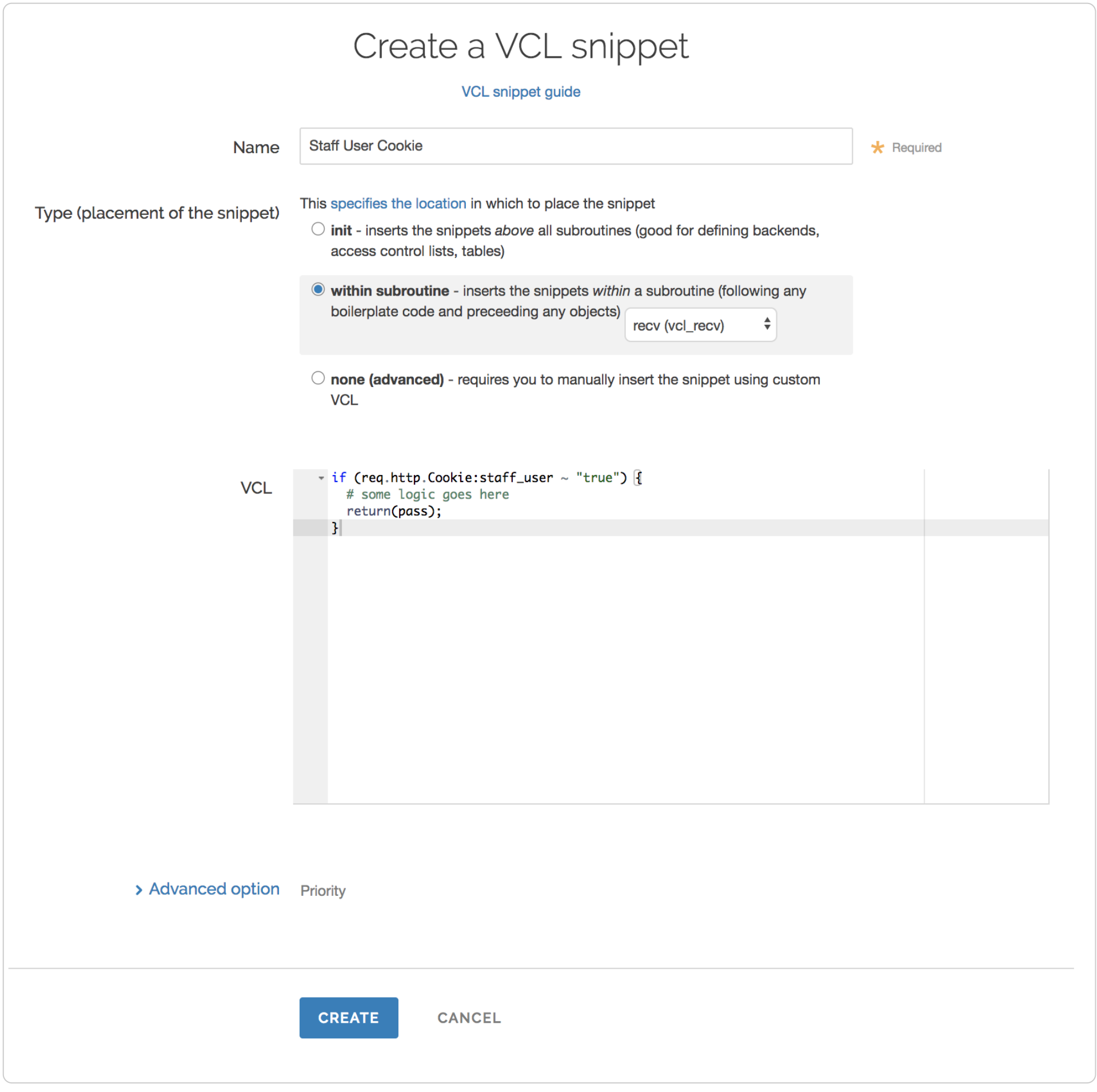
You can add this logic using VCL Snippets or using a custom header.

> **WARNING**
>
> The subfield function as well as the `:` accessor cannot be used for `Set-Cookie` headers.

Using VCL Snippets

To execute this logic based on the value of `staff_user` within `req.http.Cookie` using a VCL Snippet, you would:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click **Create Snippet**. The Create a VCL snippet page appears.



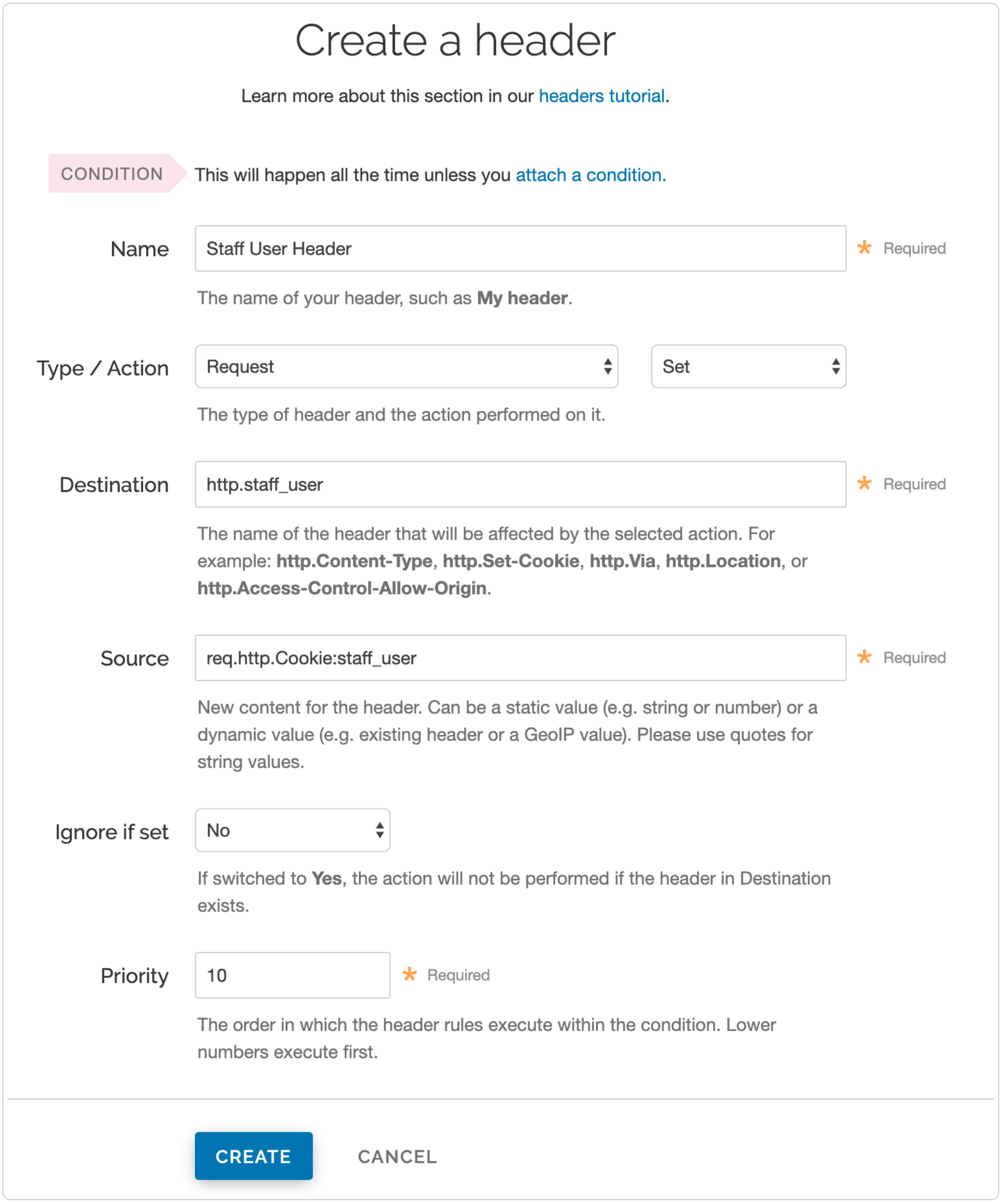
6. In the **Name** field, enter an appropriate name (e.g., `Staff User Cookie`).
7. From the **Type** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **recv (vcl_recv)**.
9. In the **VCL** field, add the following condition:

```
# in vcl_recv
if (req.http.Cookie:staff_user ~ "true") {
  # some logic goes here
  return(pass);
}
```

10. Click **Create** to create the snippet.
11. Click the **Activate** button to deploy your configuration changes.

Using a custom header

You can isolate the value of `staff_user` from `Cookie` to the header `req.http.staff_user` by creating a custom header with the following settings:



Fill out the **Create a header** fields as follows:

- In the **Name** field, enter `Staff User Header`.
- From the **Type** menu, select **Request**, and from the **Action** menu select **Set**.
- In the **Destination** field, enter `http.staff_user`.
- In the **Source** field, enter `req.http.Cookie:staff_user`.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, enter `10`.

This will send the `staff_user` header in every inbound request.

> **NOTE**
>
> You can use the `Attach a condition` link to only create this header when it's needed. See our Using Conditions docs for more information.

Manipulating the cache key

Last updated: 2018-09-04

https://docs.fastly.com/en/guides/manipulating-the-cache-key

Before you begin

If your origin uses special values (e.g., request headers) to select content for users or to otherwise direct requests to appropriate security domains, consider including those values in your cache key or Vary header. Doing so will prevent you from accidentally caching content across security domains and could prevent malicious attackers from poisoning your cache.

Redefining the cache key

> **WARNING**
>
> By default, Fastly uses the URL and the Host of a request (plus a special, internal Fastly variable for purging purposes) to create unique HTTP objects. Although Fastly allows you to explicitly set the cache key to define this more precisely, changing the default behavior risks the following:
>
> 1. If you add too much information to the cache key, you can significantly reduce your hit ratio.
> 2. If you make a mistake when explicitly setting the cache key, you can cause all requests to get the same object.
> 3. If you add anything to the hash, you will need to send a purge for each combination of the URL and value you add in order to purge that specific information from the cache.
>
> To avoid these dangers, consider using the Vary header instead of following the instructions below.

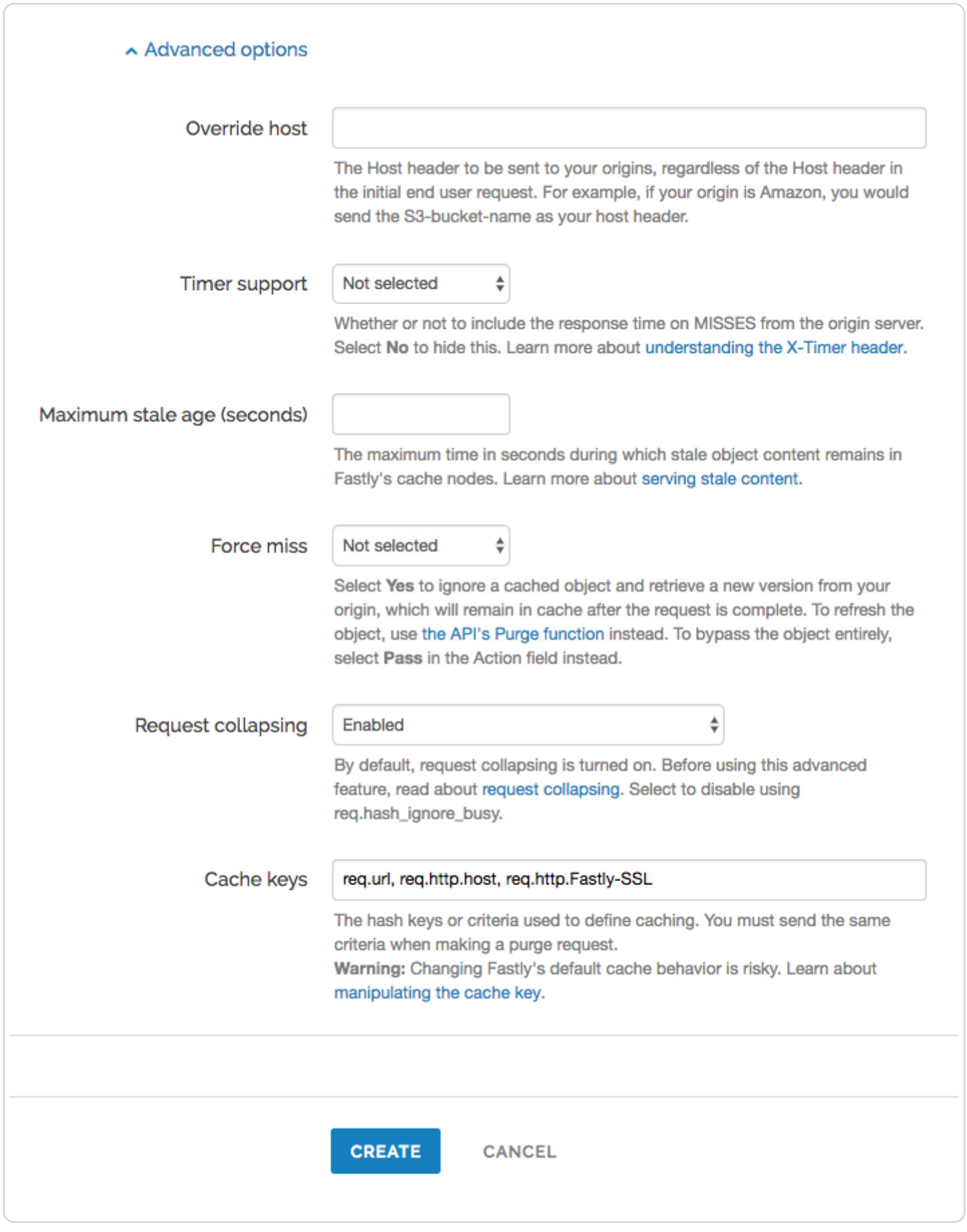
Explicitly setting the cache key

You can set the cache key explicitly (including attaching conditions) by adding a request setting via the Settings page in the configuration controls and including a comma-separated list of cache keys. The values of the cache keys listed are combined to make a single hash, and each unique hash is considered a unique object.

For example, if you don't want the query string to be part of the cache key, but you don't want to change `req.url` so that the query string still ends up in your logs, you could use the following text for the hash keys:

```
req.url.path, req.http.host
```

In the web interface, the text would appear in the Cache keys field:



As a general rule, you should always have `req.url` as one of your cache keys or as part of one.

Purging adjustments when making additions to cache keys

Because purging works on individual hashes, additions to cache keys can complicate purging URLs. However, it can also be simplified.

For example, if you were to change your cache key to just `req.url` and not the default `req.url, req.http.host`, then purging `http://foo.example.com/file.html` would also purge `http://bar.example.com/file.html`. Keep in mind this is because they're actually the same object in the cache!

On the other hand, if you were to change your cache key `req.url, req.http.host, req.http.Fastly-SSL`, you would have to purge `http://example.com/` and `https://example.com/` individually.

In the latter case, if you were to use the Vary header instead of changing the cache key, you could still have different content on the two URLs, yet purge them with a single purge. In this case you would add a new Cache Header, use `http.Vary` as the Destination, and use the following as the Source:

```
if(beresp.http.Vary, beresp.http.Vary ",", "") "Fastly-SSL"
```

Using a cookie as a cache key

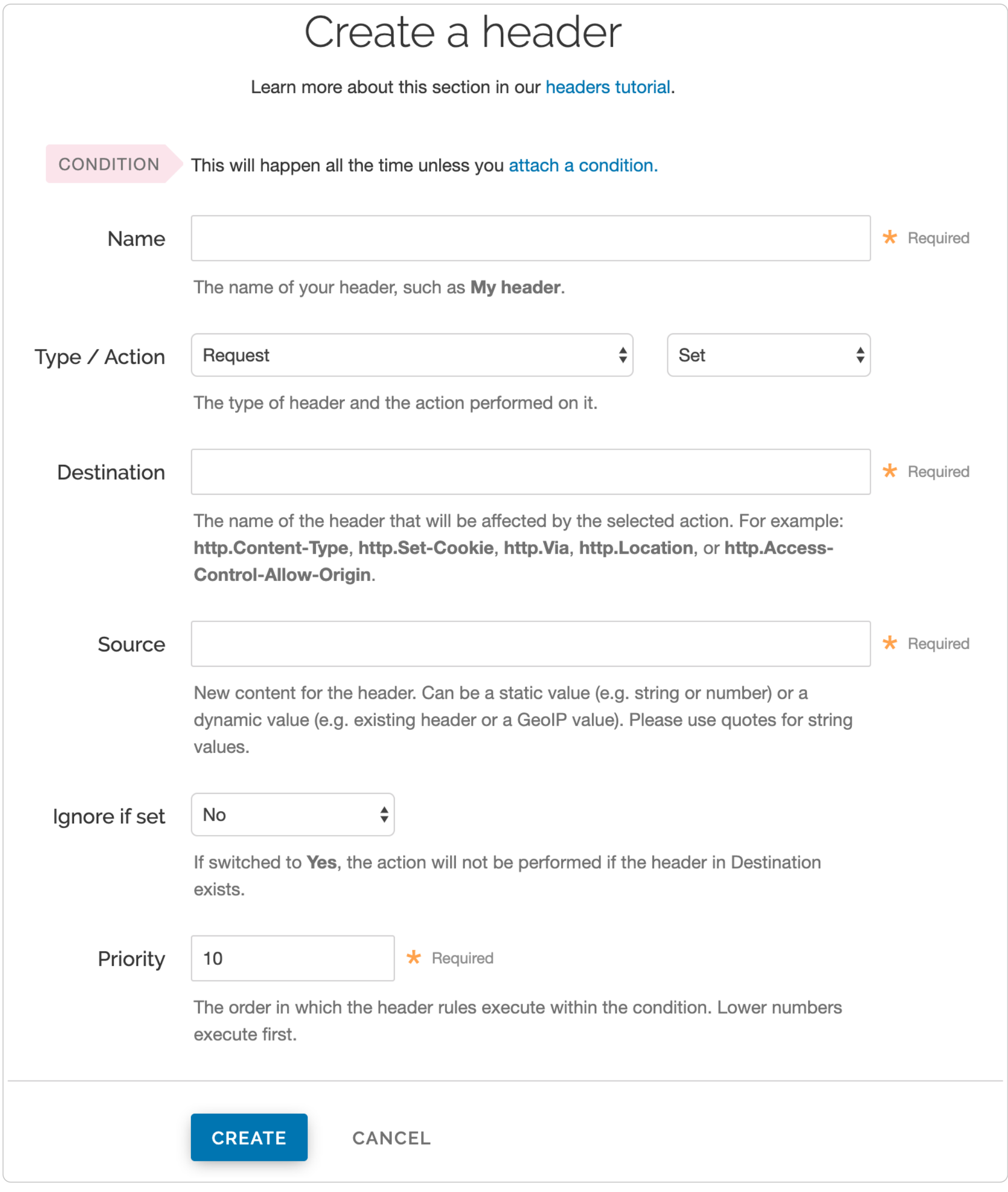
You can use a cookie as a cache key or just check for the presence of a cookie set to a specific value by controlling its request conditions. Both methods are simple and shown in the steps below.

To use a cookie as a cache key

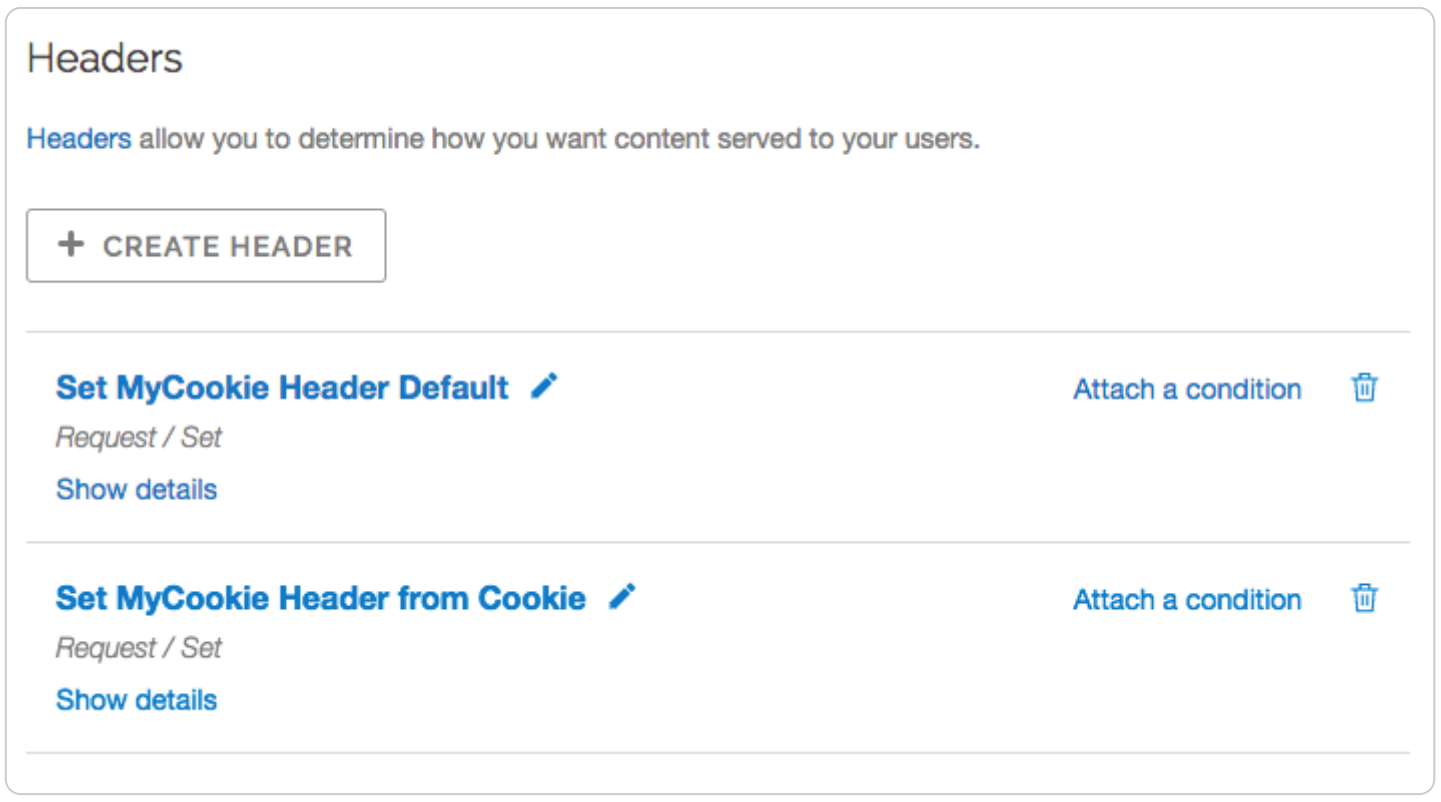
Using a cookie as a cache key looks complicated but it's actually quite simple. Let's say your cookie is called "MyCookie" and it looks like `mycookie=`.

Create new headers

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.

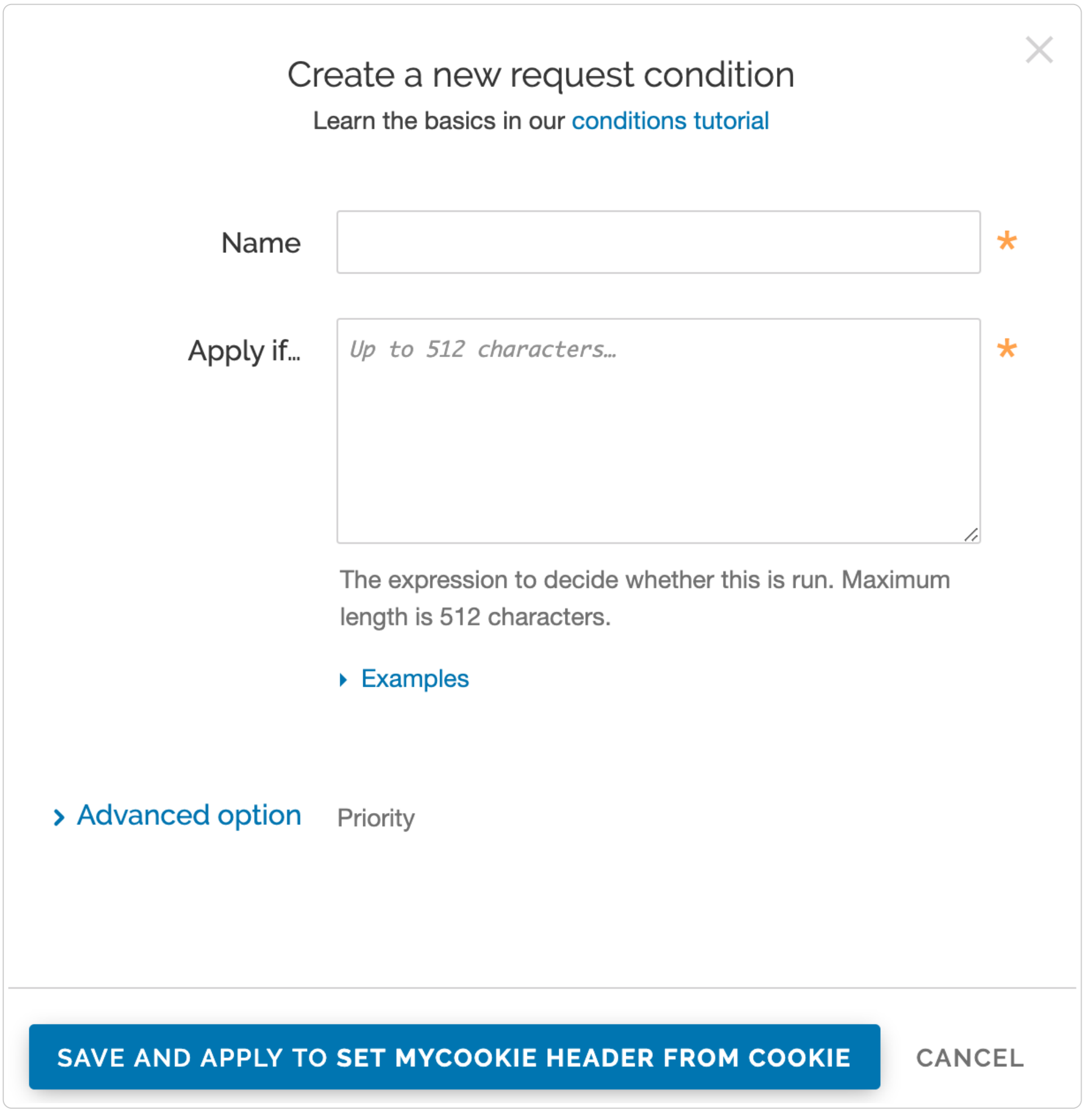


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Set MyCookie Header Default`.
   - From the **Type** menu, select **Request**, and from the **Action** menu select **Set**.
   - In the **Destination** field, enter `http.X-MyCookie`.
   - In the **Source** field, enter `"0"` (with quotes).
   - Leave the **Ignore if set** menu set to the default, **No**.
   - In the **Priority** field, enter a number representing the order in which the header rule should execute. The default is set to `10` for new headers.
7. Click the **Create** button. The new header appears in the Headers area of the Content page.
8. Click the **Create header** button again and create a second new header by filling out the fields as follows:
   - In the **Name** field, enter `Set MyCookie Header from Cookie`.
   - From the **Type** menu, select **Request**, and from the **Action** menu select **Set**.
   - In the **Destination** field, enter `http.X-MyCookie`.
   - In the **Source** field, enter `req.http.cookie:mycookie`.
   - Leave the **Ignore if set** menu set to the default, **No**.
   - In the **Priority** field, enter a larger number than the priority of previous header you just created. For example, if you left the default priority set to `10`, enter `20`.
9. Click the **Create** button. The second header appears in the Headers area of the Content page.

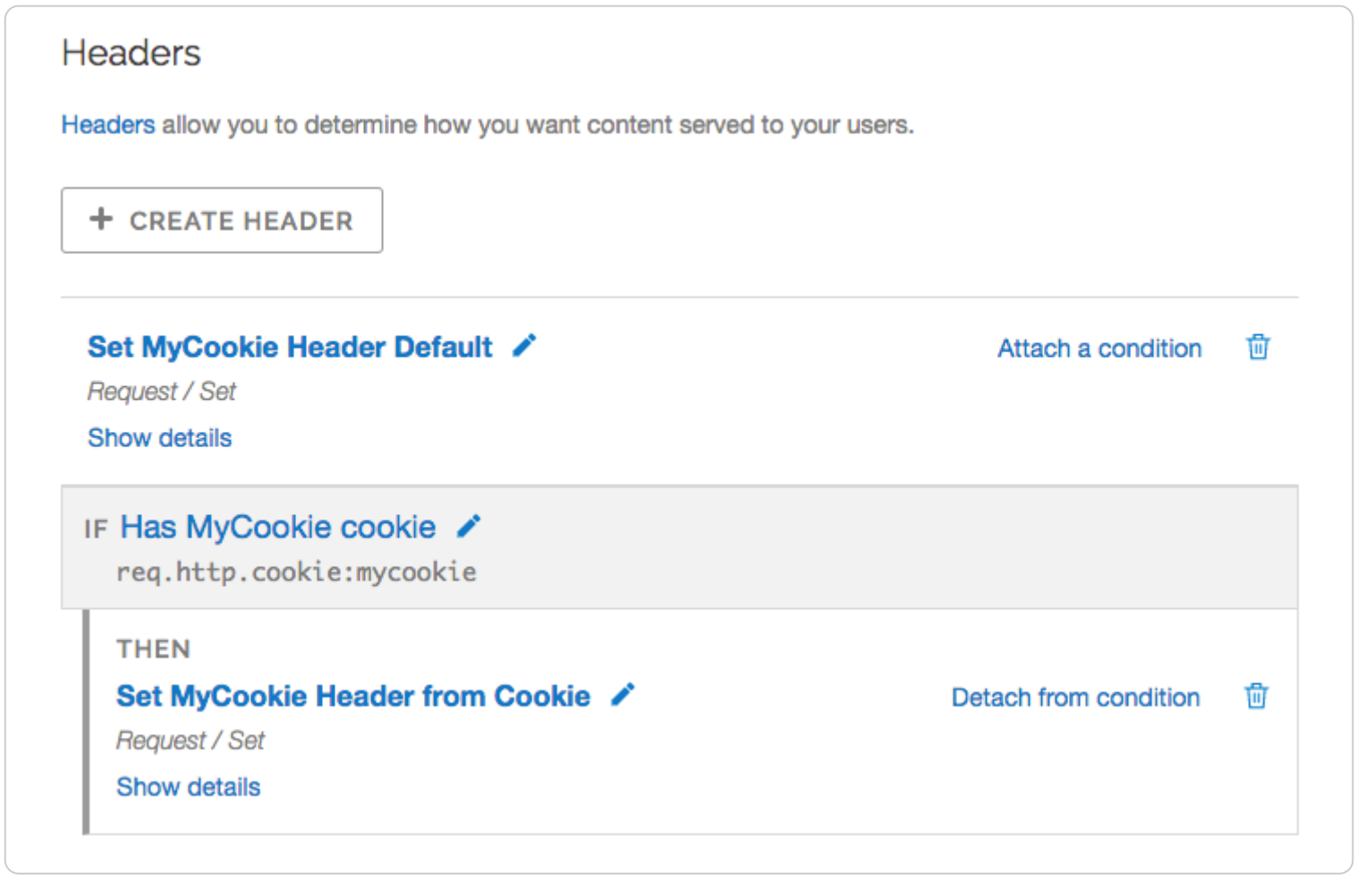


Attach conditions to the new headers

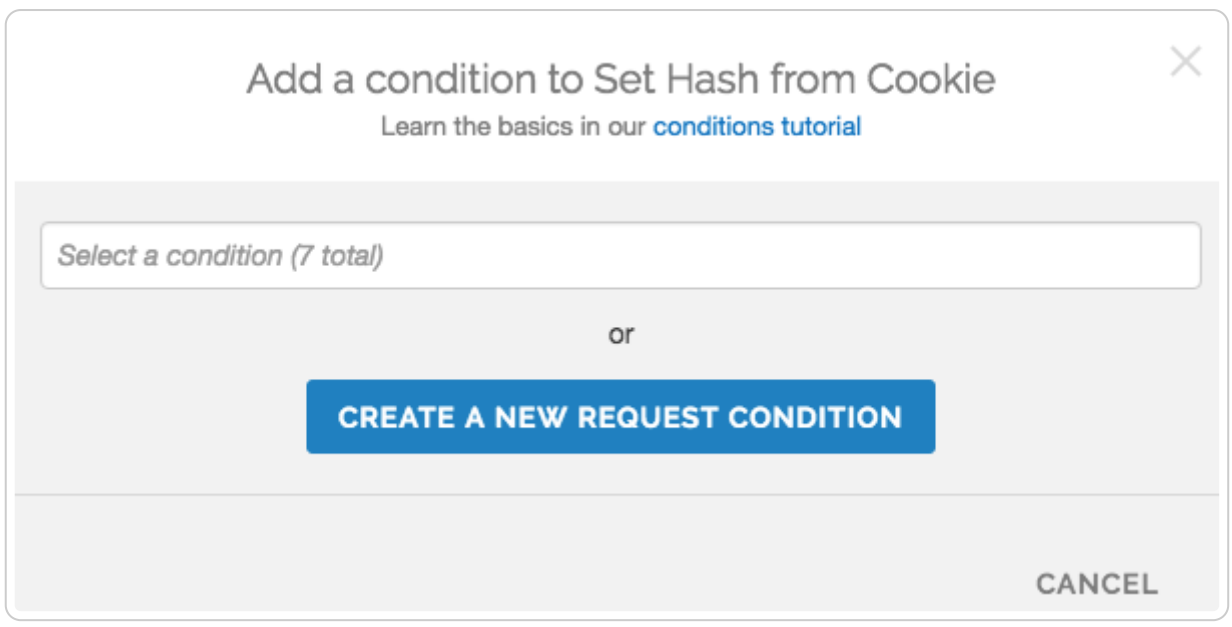
1. Click the **Attach a condition** link next to the `Set MyCookie Header from Cookie` header. The add a condition window appears.
2. Click the **Create a new request condition** button. The Create a new request condition window appears.



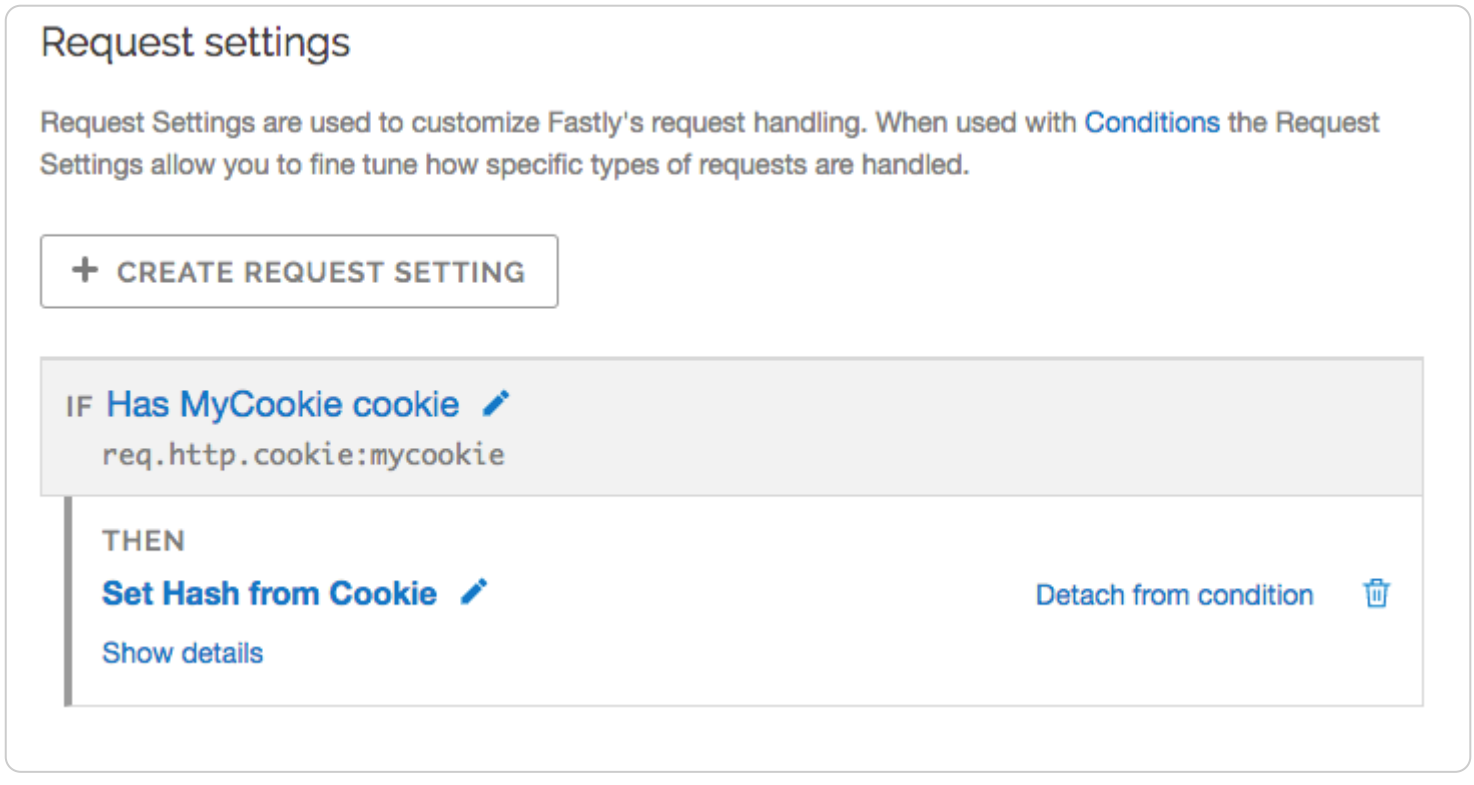
3. Fill out the fields of the **Create a new request condition** page as follows:
   - In the **Name** field, enter `Has MyCookie cookie`.
   - In the **Apply if** field, enter `req.http.cookie:mycookie`.
4. Click the **Save and apply to** button. The Headers area now displays the condition that must be met in order for your header to begin being used.



5. Click the **Settings** link. The Settings page appears.
6. Click the **Create request setting** button. The Create a request setting page appears.
7. In the **Name** field, enter `Set Hash from Cookie`.
8. Click the **Advanced options** link. The Advanced options appear.
9. In the **Cache keys** field, enter `req.url, req.http.host, req.http.X-MyCookie`
10. Click the **Create** button. The new request appears in the Request settings area.
11. Click the **Attach a condition** link next to the new request. The Add a condition window appears.



12. From the **Select a condition** menu, select `Has MyCookie cookie`. The Request settings area now displays the condition that must be met in order for your request to begin being used.



13. Click the **Activate** button to deploy your configuration changes.

To check for the presence of a cookie set to a specific value

An alternative way if you're just checking for the presence of the cookie set to some specific value (e.g., 1):

1. Add a new Request setting where the Cache key field is set to `req.url, req.http.host, "Has mycookie"`.
2. Add a condition to that Request setting where the Apply if field contains `req.http.cookie:mycookie`.

Response Cookie handling

Last updated: 2021-05-10

https://docs.fastly.com/en/guides/response-cookie-handling

The traditional way to read response cookies in VCL is to inspect either the `beresp.http.Set-Cookie` or the `resp.http.Set-Cookie` variables and then extract values using regular expressions. However this is not ideal since attempting to parse potentially complicated or quoted strings with regular expressions is brittle and prone to being tripped up by edge cases. It also doesn't allow for reading multiple headers with the same name such as when an origin sends multiple `Set-Cookie` headers. Because of these two reasons Fastly supports a method for extracting a named value out of `Set-Cookie` headers no matter how many there are.

To access a named value simply use the function with either `beresp` or `resp` depending on what part of the request you're in - so either

```
setcookie.get_value_by_name(beresp, "name")
```

or

```
setcookie.get_value_by_name(resp, "name")
```

as appropriate, replacing `"name"` with whatever the name of the value is. So for example, given this HTTP response from an origin

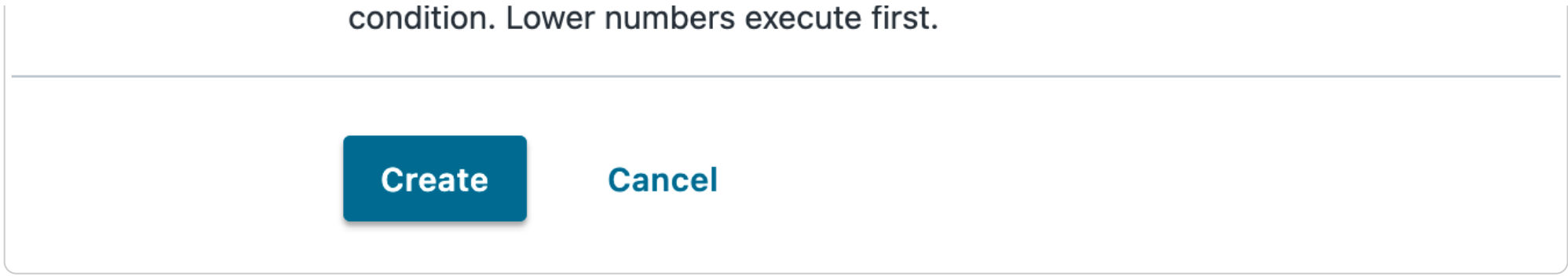
```
HTTP/1.1 200 OK
Cache-Control: max-age=60
Content-Type: text/html; charset=utf-8
Content-Length: 80806
Accept-Ranges: bytes
Date: Tue, 11 Aug 2015 19:00:04 GMT
Age: 123
Connection: keep-alive
Set-Cookie: one=a; httponly; secure
Set-Cookie: two=b or not to b; httponly
```

then using the function like this

```
set resp.http.X-One = setcookie.get_value_by_name(resp, "one");
set resp.http.X-Two = setcookie.get_value_by_name(resp, "two");
```

will set `resp.http.X-One` to be "a" and `resp.http.X-Two` to "b or not to b".

This logic can be used in uploaded custom VCL, as well as throughout the web interface. For example:



Support for the Edge-Control header

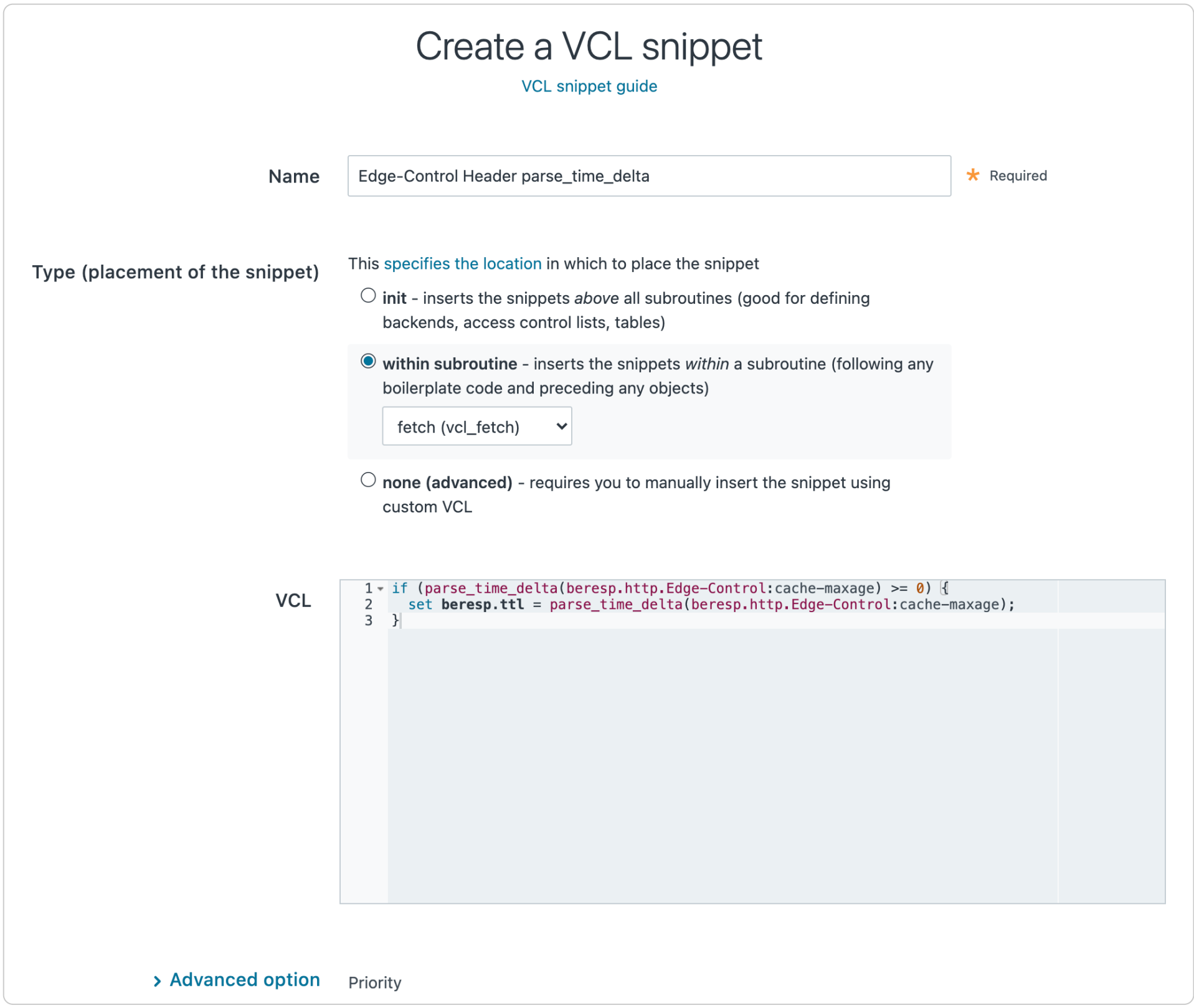
Last updated: 2021-08-19

https://docs.fastly.com/en/guides/support-for-the-edge-control-header

VCL provides the building blocks to access information inside the Edge-Control response header field from the origin. We support this by honoring `cache-maxage` from Edge-Control as the time to live (TTL) of the object on the Fastly edge, and honoring `downstream-ttl` from Edge-Control as the TTL to be sent down from the Fastly edge to the end user's browser.

In order to incorporate this Edge-Control header support, use VCL Snippets to update your `vcl_fetch`:

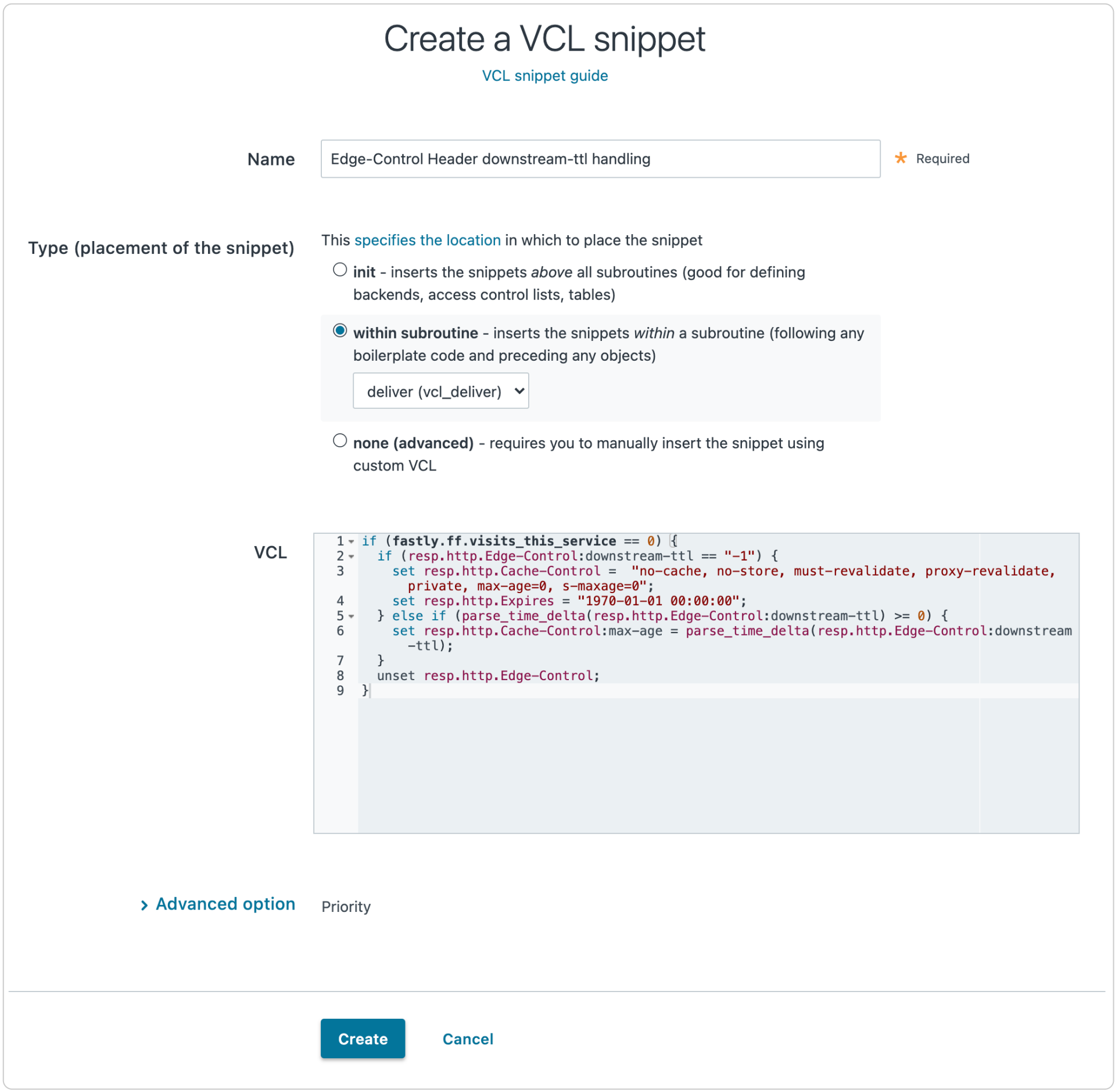
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click **Create Snippet**. The Create a VCL snippet page appears.



6. In the **Name** field, enter an appropriate name (e.g., `Edge-Control Header parse_time_delta`).
7. From the **Type** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **fetch (vcl_fetch)**.
9. In the **VCL** field, add the following conditions:

```
if (parse_time_delta(beresp.http.Edge-Control:cache-maxage) >= 0) {
  set beresp.ttl = parse_time_delta(beresp.http.Edge-Control:cache-maxage);
}
```

10. Click **Create** to create the snippet.
11. Click **Create Snippet**. The Create a VCL snippet page appears.



12. In the **Name** field, enter an appropriate name (e.g., `Edge-Control Header downstream-ttl handling`).
13. From the **Type** controls, select **within subroutine**.
14. From the **Select subroutine** menu, select **deliver (vcl_deliver)**.
15. In the **VCL** field, add the following conditions:

```
if (fastly.ff.visits_this_service == 0) {
  if (resp.http.Edge-Control:downstream-ttl == "-1") {
    set resp.http.Cache-Control =  "no-cache, no-store, must-revalidate, proxy-revalidate, private, max-age=0, s-max
age=0";
    set resp.http.Expires = "1970-01-01 00:00:00";
  } else if (parse_time_delta(resp.http.Edge-Control:downstream-ttl) >= 0) {
    set resp.http.Cache-Control:max-age = parse_time_delta(resp.http.Edge-Control:downstream-ttl);
  }
  unset resp.http.Edge-Control;
}
```

16. Click **Create** to create the snippet.
17. Click the **Activate** button to deploy your configuration changes.

The subfield function parses the Edge-Control field for subfields, and the `parse_time_delta` function converts time values like "7m" into a number of seconds. You can then use that number of seconds to populate `beresp.ttl` (the TTL of the object on the Fastly edge) or you can use it to construct a Cache-Control header field for downstream. The `parse_time_delta` function will return -1 if the subfield is not well-formed as a time value, or if it is entirely absent. The above snippet honors `cache-maxage` and `downstream-ttl` from Edge-Control if present and usable.

Understanding the different PASS action behaviors

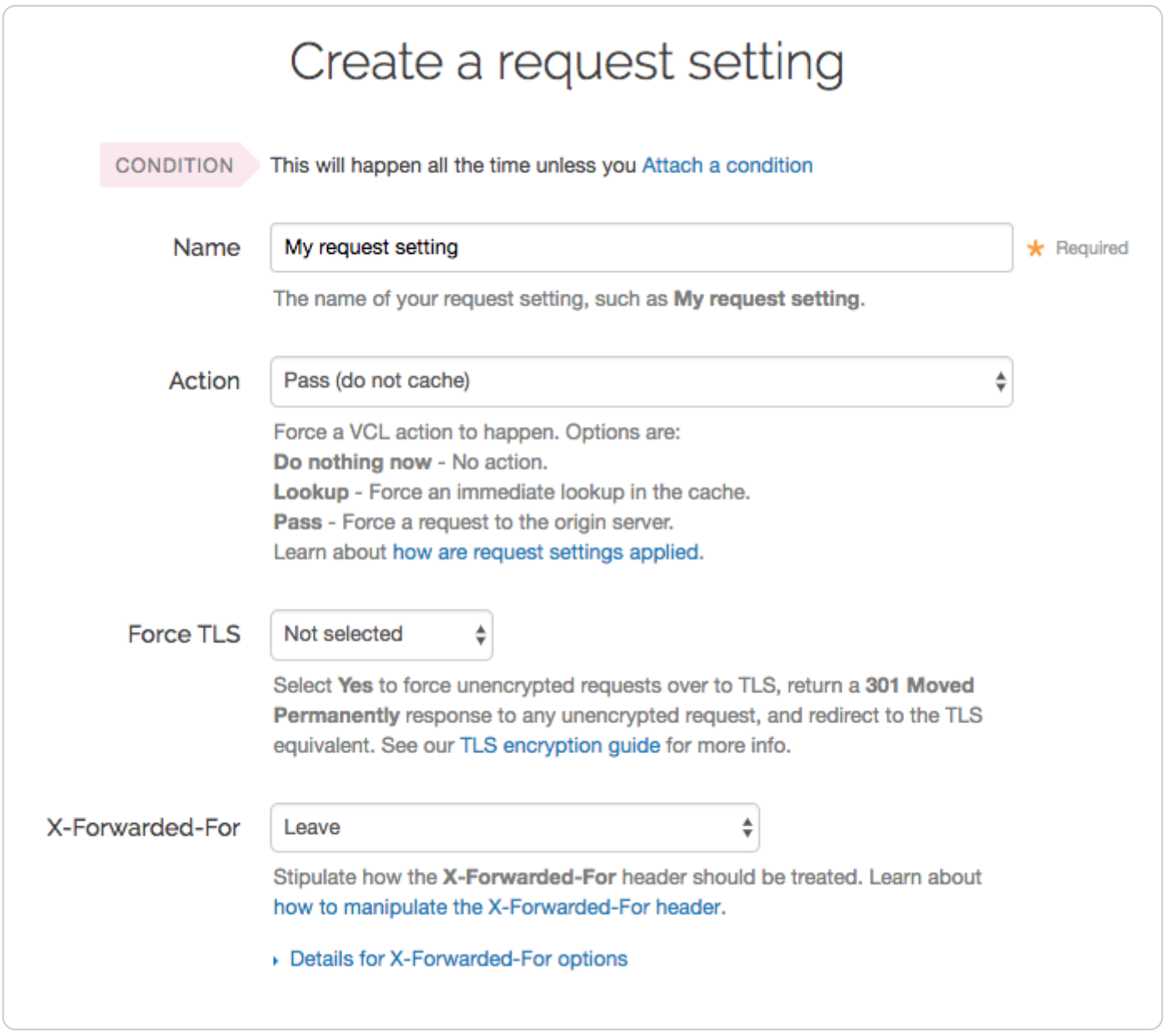
Last updated: 2018-08-01

https://docs.fastly.com/en/guides/understanding-the-different-pass-action-behaviors

Passing with a request setting and with a cache setting triggers very different behavior in Varnish. Within VCL, passing with a request setting is the same as `return(pass)` in `vcl_recv`. Passing with a cache setting is the same as `return(pass)` in `vcl_fetch`. If you are familiar with Varnish 3+, passing with a cache setting is equivalent to `return(hit_for_pass)`.

Using a request setting

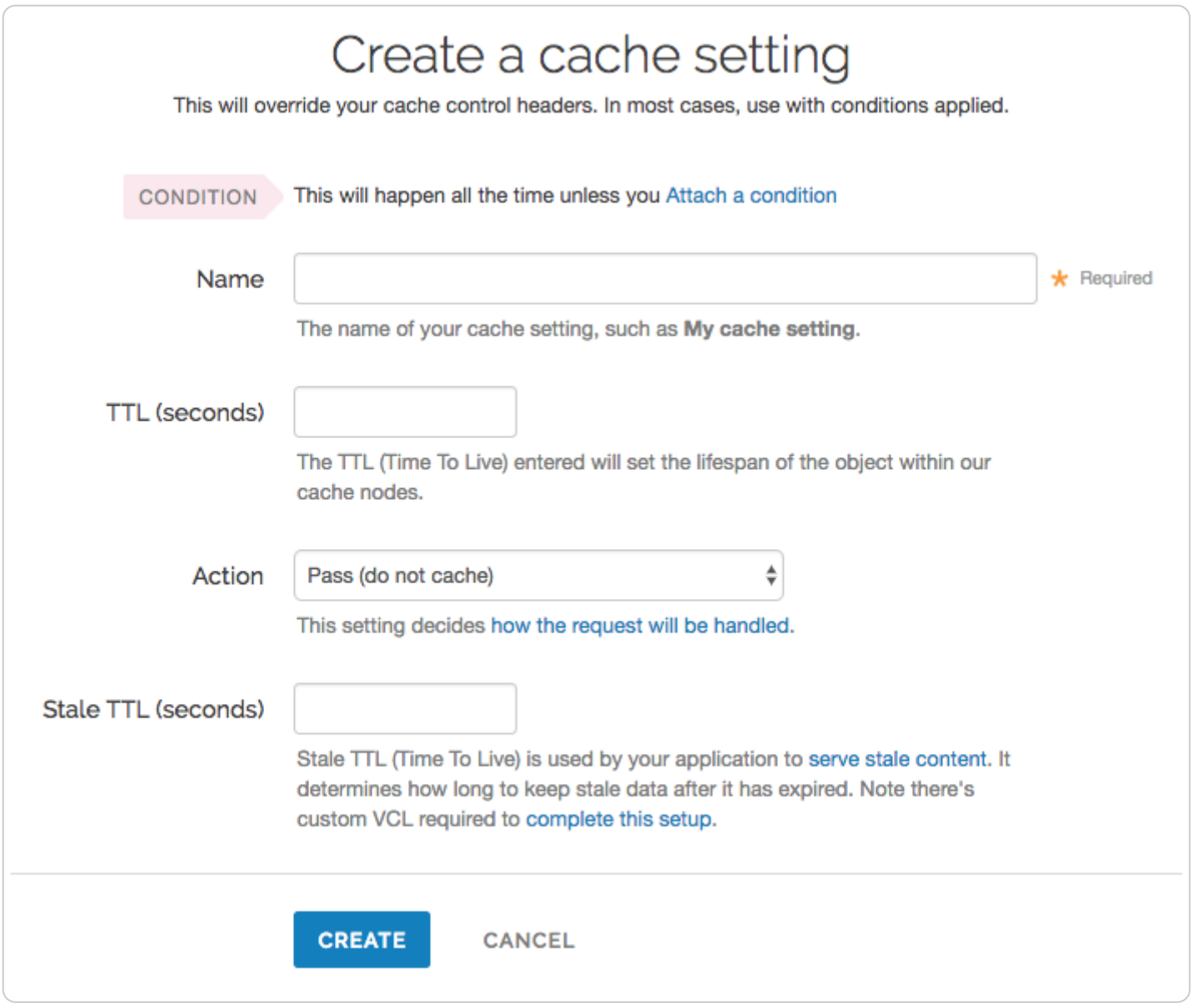
Passing with a request setting translates within your generated VCL to `return(pass)` in `vcl_recv`. Varnish will not perform a lookup to see if an object is in cache and the response from the origin will not be cached.



Passing in this manner disables request collapsing. Normally simultaneous requests for the same object that result in cache misses will be collapsed down to a single request to the origin. While the first request is sent to the origin, the other requests for that object are queued until a response is received. When requests are passed in `vcl_recv`, they will all go to the origin separately without being collapsed.

Using a cache setting

Passing with a cache setting translates within your generated VCL to `return(pass)` in `vcl_fetch`. At this point in the flow of a request, Varnish has performed a lookup and determined that the object is not in cache. A request to the origin has been made; however, in vcl_fetch we have determined that the response is not cacheable. In Fastly's default VCL, this can happen based on the presence of a `Set-Cookie` response header from the origin.



Passing in `vcl_fetch` is often not desirable because request collapsing is *not* disabled. This makes sense since Varnish is not aware in `vcl_recv` that the object is uncacheable. On the first request for an object that will be later passed in `vcl_fetch`, all other simultaneous cache misses will be queued. Once the response from the origin is received and Varnish has realized that the request should be passed, the queued requests are sent to the origin.

This creates a scenario where two users request an object at the same time, and one user must wait for the other before being served. If these requests were passed in `vcl_recv`, neither user would need to wait.

To get around this disadvantage, when a request is passed in `vcl_fetch`, Varnish creates what is called a hit-for-pass object. These objects have their own TTLs and while they exist, Varnish will pass any requests for them as if the pass had been triggered in `vcl_recv`. For this reason, it is important to set a TTL that makes sense for your case when you pass in `vcl_fetch`. All future requests for the object will be passed until the hit-for-pass object expires. Hit-for-pass objects can also be purged like any other object.

Even with this feature, there will be cases where simultaneous requests will be queued and users will wait. Whenever there is not a hit-for-pass object in cache, these requests will be treated as if they are normal cache misses and request collapsing will be enabled. Whenever possible it is best avoid relying on passing in `vcl_fetch`.

Using req.hash_always_miss and req.hash_ignore_busy

Setting `req.hash_always_miss` forces a request to miss whether it is in cache or not. This is different than passing in `vcl_recv` in that the response will be cached and request collapsing will not be disabled. Later on the request can still be passed in `vcl_fetch` if desired.

A second relevant variable is `req.hash_ignore_busy`. Setting this to true disables request collapsing so that each request is sent separately to origin. When `req.hash_ignore_busy` is enabled all responses will be cached and each response received from the origin will overwrite the last. Future requests for the object that are served from cache will receive the copy of the object from the last cache miss to complete. `req.hash_ignore_busy` is used mostly for avoiding deadlocks in complex multi-Varnish setups.

Setting both these variables can be useful to force requests to be sent separately to the origin while still caching the responses.

Uploading custom VCL

Last updated: 2020-08-14

https://docs.fastly.com/en/guides/uploading-custom-vcl

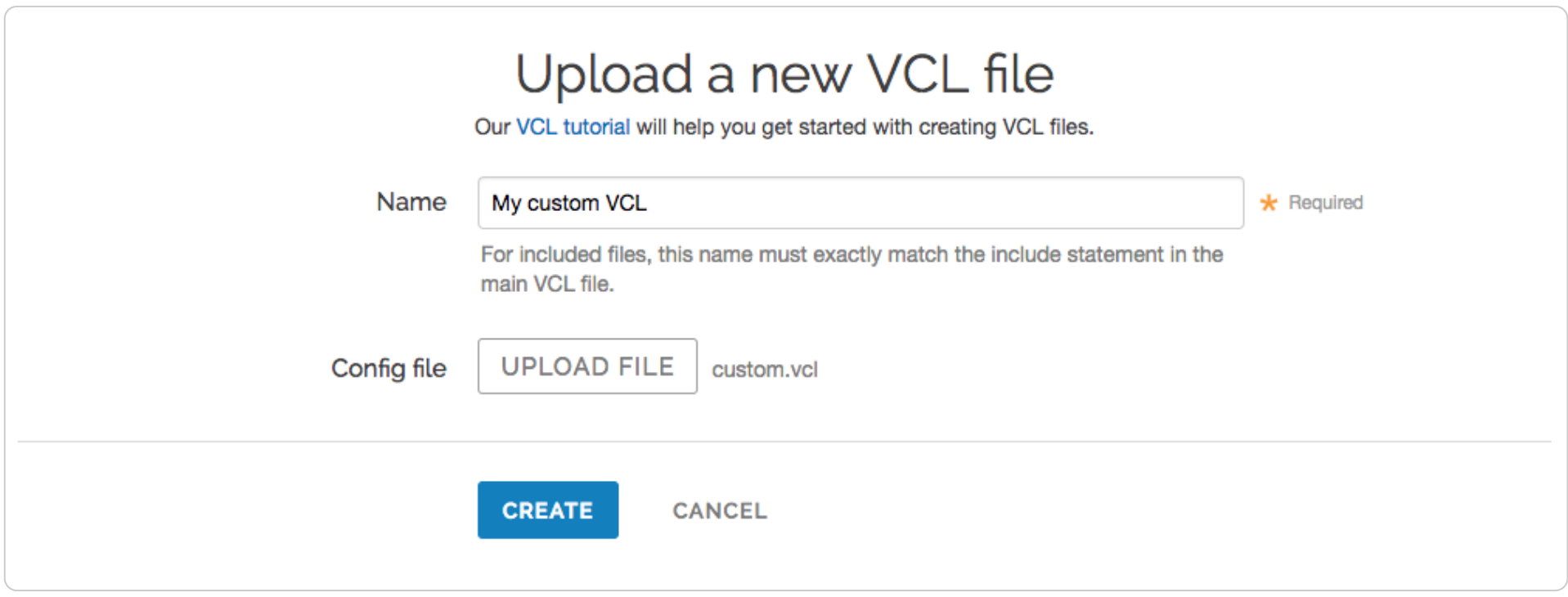
Fastly allows you create your own Varnish Configuration Language (VCL) files with specialized configurations. By uploading custom VCL files, you can use custom VCL and Fastly VCL together at the same time. Any time you upload VCL files, you can preview the VCL prior to activating a new version of your service. Keep in mind that your custom VCL always takes precedence over VCL generated by Fastly.

> **IMPORTANT**
>
> Personal data should not be incorporated into VCL. Our Compliance and Law FAQ describes in detail how Fastly handles personal data privacy.

Uploading a VCL file

Follow these instructions to upload a custom VCL file:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Custom VCL** tab. The Custom VCL page appears.
5. Click the **Upload a new VCL file** button. The Upload a new VCL file page appears.



6. In the **Name** field, enter the name of the VCL file. For included files, this name must match the include statement in the main VCL file. See how to include additional VCL configurations for more information.
7. Click **Upload file** and select a file to upload. The name of the uploaded file appears next to the button.

> **IMPORTANT**
>
> Don't upload generated VCL that you've downloaded from the Fastly web interface. Instead, edit and then upload a copy of Fastly's VCL boilerplate to avoid errors.

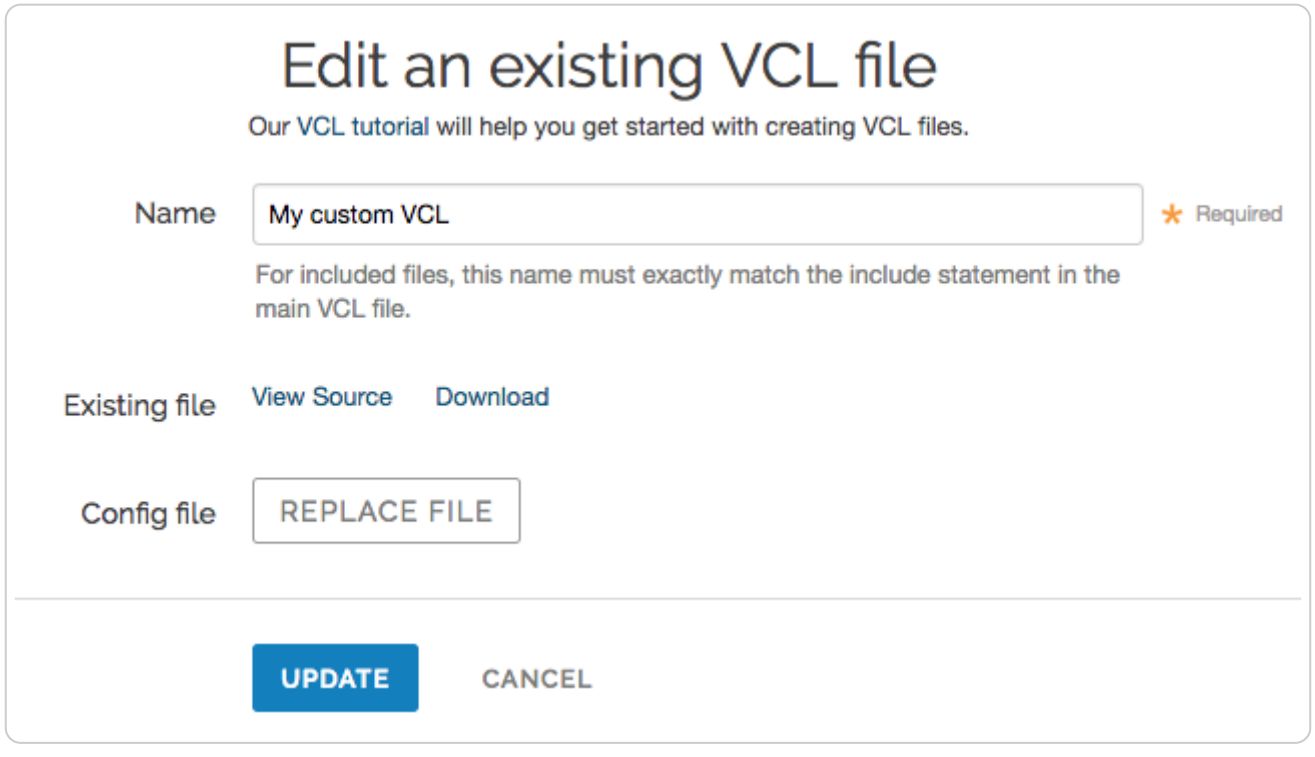
8. Click the **Create** button. The VCL file appears in the Varnish Configurations area.
9. Click the **Activate** button to deploy your configuration changes.

Editing a VCL file

To edit an existing VCL file, follow these instructions:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Custom VCL** tab. The Custom VCL page appears.

5. In the **Varnish Configurations** area, click the VCL file you want to edit. The Edit an existing VCL file page appears.

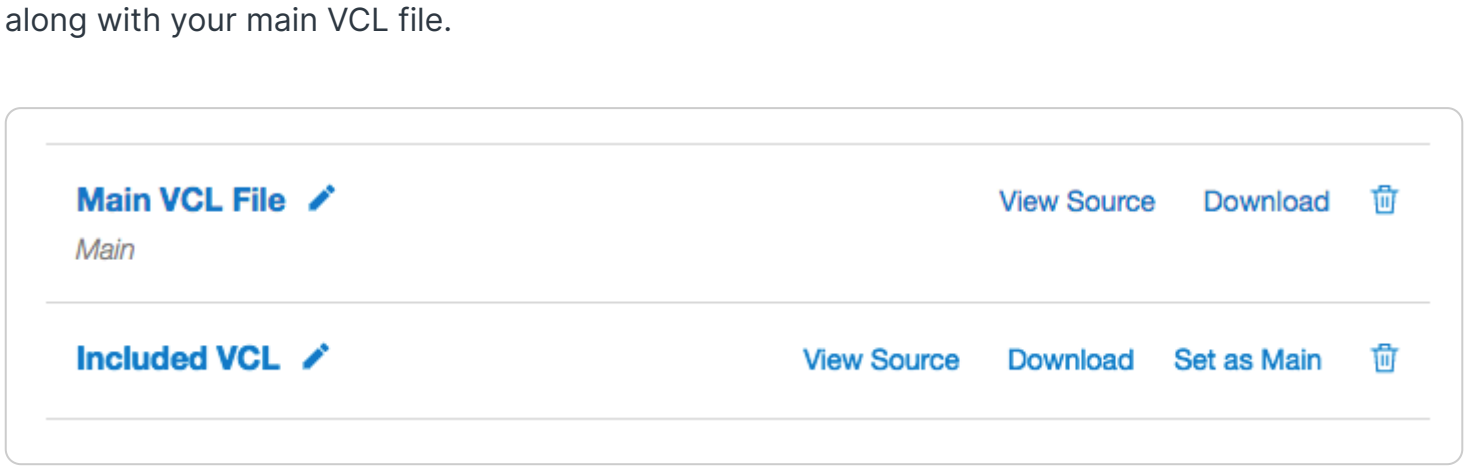


6. In the **Name** field, optionally enter a new name of the VCL file.
7. Click the **Download** link to download the appropriate file.
8. Make the necessary changes to your file and save them.
9. Click the **Replace file** button and select the file you updated. The selected file replaces the current VCL file and the file name appears next to the button.
10. Click the **Update** button to update the VCL file in the Fastly application.
11. Click the **Activate** button to deploy your configuration changes.

Including additional VCL configurations

To make your full VCL configuration easier to maintain, you can split it up into multiple files that are accessed by a main VCL file. This allows you to separate out chunks of logic (for example, logic that has a specific purpose or that might change frequently) into as many separate files as makes sense.

1. Start by isolating a portion of VCL and placing it in a separate file. The name of the file doesn't matter, nor does the file extension. A `foo.vcl` file will work just as well as a `bar.txt` file.
2. Upload the file to include it in your Varnish configurations and give it a unique name when you fill out the **Name** field at the time of upload (for example, you could call it `Included VCL`). The uploaded file will appear in the Varnish Configurations area along with your main VCL file.



3. Enter the name of the included VCL file on a separate line in the main VCL configuration file. For example, your **Included VCL** file would get added to the main VCL file in a single line like this:

```
include "Included VCL";
```

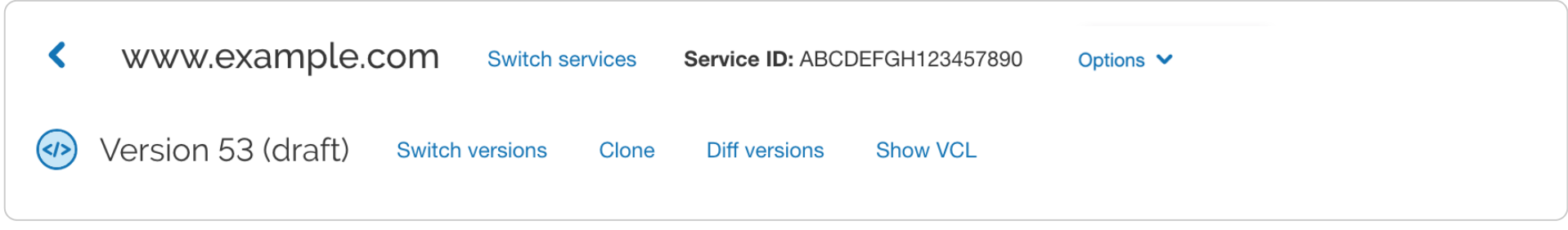
4. Continue uploading VCL files and then including them in your main VCL using the syntax `include "<VCL FILE>";` where `<VCL FILE>` exactly matches the name you entered in the **Name** field.

> **TIP**
>
> Our guide to manually creating access control lists demonstrates a common example of using included VCL.

Previewing VCL before activation

Follow these instructions to preview VCL prior to activating a service version:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Show VCL** link.



The VCL preview page appears.

Using dynamic VCL Snippets

Last updated: 2021-10-25

https://docs.fastly.com/en/guides/using-dynamic-vcl-snippets

Dynamic VCL Snippets are one of two types of snippets that allow you to insert small sections of VCL logic into your service configuration without requiring custom VCL (though you can still include snippets in custom VCL when necessary).

You can only create dynamic snippets via the API. Because they are versionless objects (much like Edge Dictionaries or ACLs at the edge), dynamic snippets can be modified independently from changes to your Fastly service. This means you can modify snippet code rapidly without deploying a service version that may not be ready for production.

Creating and using a dynamic VCL Snippet

Using the curl command line tool, make the following API call in a terminal application:

```
$ curl -X POST -s https://api.fastly.com/service/<Service ID>/version/<Editable Version>/snippet -H "Fastly-Key:FASTLY_API_TOKEN" -H 'Content-Type: application/x-www-form-urlencoded' --data $'name=my_dynamic_snippet_name&type=recv&dynamic=1&content=if ( req.url ) {\n set req.http.my-snippet-test-header = "true";\n}';
```

Fastly returns a JSON response that looks like this:

```
{
 "service_id": "<Service Id>",
 "version": "<Editable Version>",
 "name": "my_dynamic_snippet_name",
 "type": "recv",
 "priority": 100,
 "dynamic": 1,
 "content": null,
 "id": "decafbad12345",
 "created_at": "2016-09-09T20:34:51+00:00",
 "updated_at": "2016-09-09T20:34:51+00:00",
 "deleted_at": null
}
```

NOTE

The returned JSON includes `"content": null`. This happens because the content is stored in a separate, unversioned object.

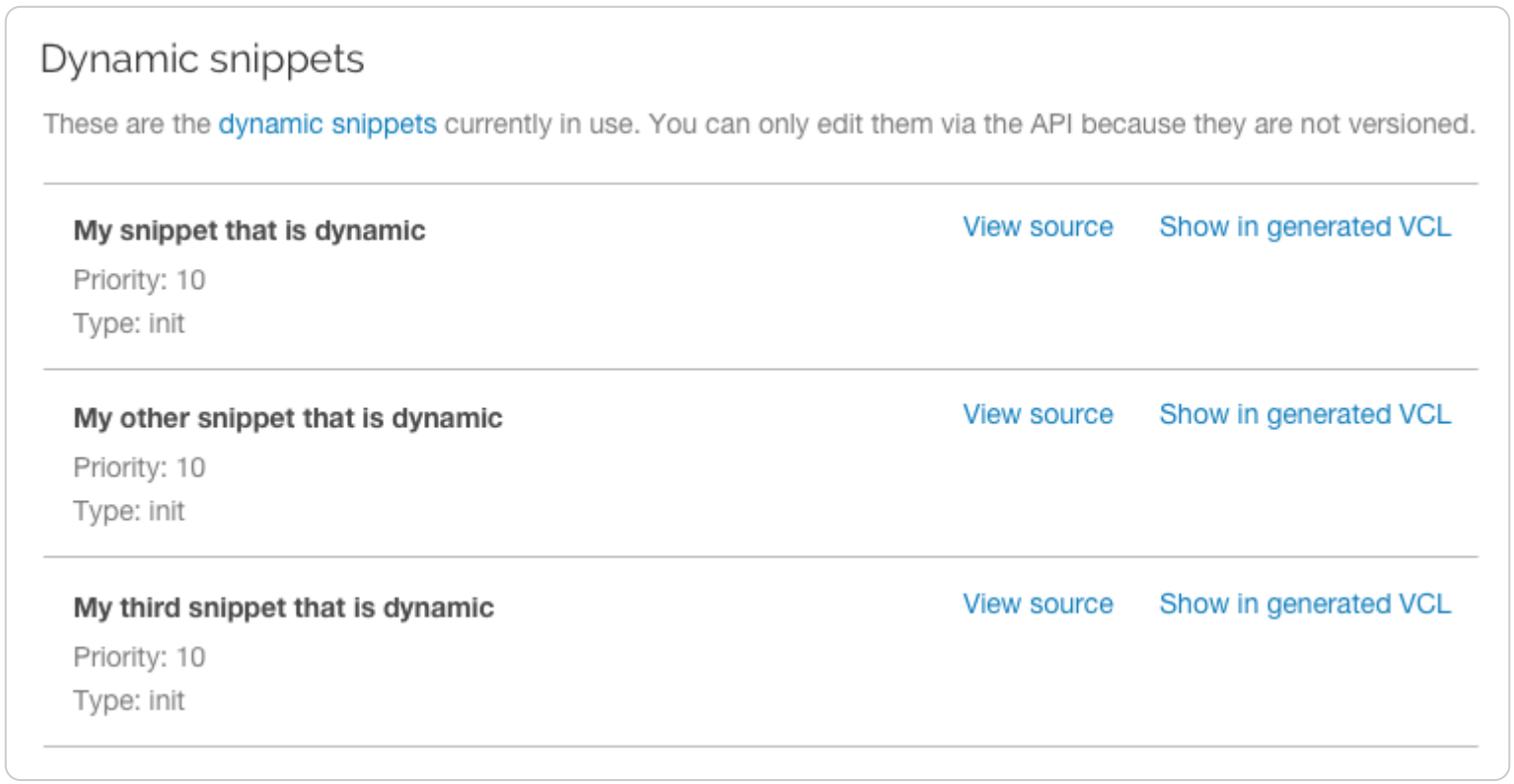
Viewing dynamic VCL Snippets in the web interface

You can view a list of dynamic VCL snippets. You can also view just the source of a specific snippet or a specific snippet's location in generated VCL.

Viewing a list of dynamic VCL Snippets

To view the entire list of a service's dynamic VCL Snippets directly in the web interface:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate VCL service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears listing all dynamic VCL Snippets for your service in the Dynamic snippets area.



Viewing the source of a specific snippet

You can view just the source of a specific snippet:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate VCL service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the **View Source** link to the right of the name of the snippet. A view source window appears.

Viewing the location of a specific snippet in generated VCL

You can view a specific snippet's location in generated VCL:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate VCL service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the **Show in Generated VCL** link to the right of the name of the snippet. The Generated VCL window appears.

Fetching a list of all dynamic VCL Snippets

To list all dynamic VCL Snippets attached to a service, make the following API call in a terminal application:

```
$ curl -X GET -s https://api.fastly.com/service/<Service ID>/version/<Editable Version>/snippet -H "Fastly-Key:FASTLY_API_TOKEN"
```

Fetching an individual dynamic VCL Snippet

To fetch an individual snippet, make the following API call in a terminal application:

```
$ curl -X GET -s https://api.fastly.com/service/<Service ID>/snippet/<my_dynamic_snippet_id> -H "Fastly-Key:FASTLY_API_TOKEN"
```

Unlike fetching regular VCL Snippets, you do not include the version in the URL and you must use the ID returned when the snippet was created, not the name.

Updating an existing dynamic VCL Snippet

To update an individual snippet, make the following API call in a terminal application on an editable version of your service configuration:

```
$ curl -X PUT -s https://api.fastly.com/service/<Service ID>/snippet/<my_dynamic_snippet_id> -H "Fastly-Key:FASTLY_API_TOKEN" -H 'Content-Type: application/x-www-form-urlencoded' --data $'content=if ( req.url ) {\n set req.http.my-snippet-test-header = \"affirmative\";\n}';
```

Deleting an existing dynamic VCL Snippet

To delete an individual snippet, make the following API call in a terminal application on an editable version of your service configuration:

```
$ curl -X DELETE -s https://api.fastly.com/service/<Service ID>/version/<Editable Version>/snippet/<my_dynamic_snippet_name> -H "Fastly-Key:FASTLY_API_TOKEN"
```

Including dynamic snippets in custom VCL

By specifying a location of `none` for the `type` parameter, snippets will not be rendered in VCL. This allows you to include snippets in custom VCL using the following syntax:

```
include "snippet::<snippet name>"
```

The same VCL Snippet can be included in custom VCL in as many places as needed.

Example use: blocking site scrapers

Say you wanted to implement some pattern matching against incoming requests to block someone trying to scrape your site. Say also that you've developed a system that looks at all incoming requests and generates a set of rules that can identify scrapers using a combination of the incoming IP address, the browser, and the URL they're trying to fetch. Finally, say that the system updates the rules every 20 minutes.

If, during system updates, your colleagues are also making changes to the rest of your Fastly configuration, you probably don't want the system to automatically deploy the latest version of the service since it might be untested. Instead you could generate the rules as a Dynamic VCL Snippet. Whenever the snippet is updated, all other logic remains the same as the currently deployed version and only your rules are modified.

Using regular VCL Snippets

Last updated: 2018-08-09

https://docs.fastly.com/en/guides/using-regular-vcl-snippets

Regular VCL Snippets are one of two types of snippets that allow you to insert small sections of VCL logic into your service configuration without requiring custom VCL (though you can still include snippets in custom VCL when necessary).

Unlike dynamic snippets, regular snippets can be created via the web interface or via the API. They are considered *versioned* objects. They belong to a specific service and any modifications you make to the snippet are locked and deployed when you deploy a new version of that service. We continue to clone them and deploy them with a service until you specifically delete them.

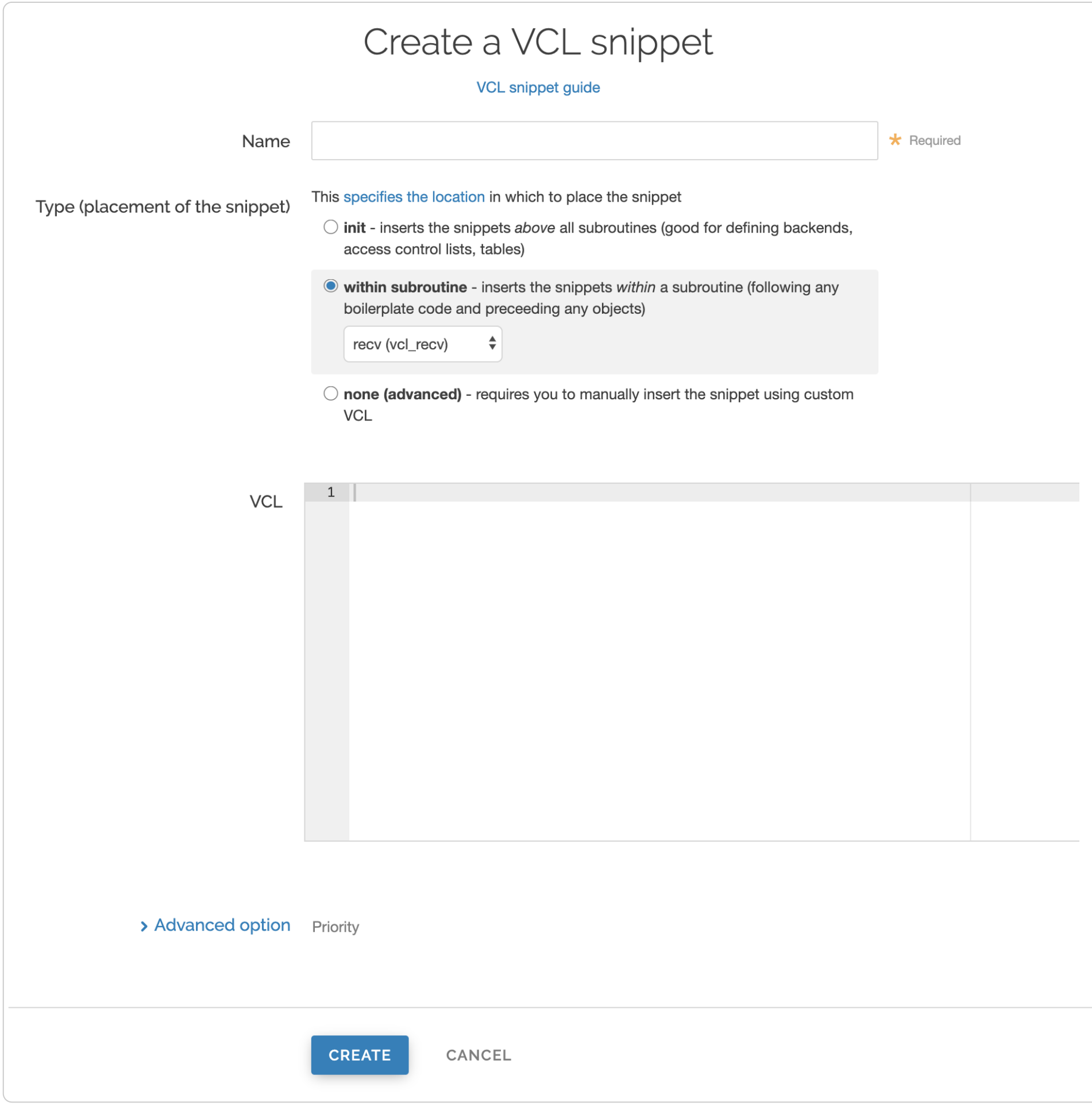
Creating a regular VCL Snippet

You can create regular VCL Snippets via the web interface or via the API.

Via the web interface

To create a regular VCL Snippet via the web interface:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click **Create Snippet**. The Create a VCL snippet page appears.



6. In the **Name** field, enter an appropriate name (for example, `Example Snippet`).
7. Using the **Type** controls, select the location in which the snippet should be placed as follows:
   - Select `init` to insert it above all subroutines in your VCL.
   - Select `within subroutine` to insert it within a specific subroutine and then select the specific subroutine from the **Select subroutine** menu.
   - Select `none (advanced)` to insert it manually. See Including regular snippets in custom VCL for the additional manual insertion requirements if you select this option.
8. In the **VCL** field, enter the snippet of VCL logic to be inserted for your service version.
9. Click **Create** to create the snippet.

Via the API

To create a regular VCL Snippet via the API, make the following API call using the curl command line tool in a terminal application:

```
$ curl -X POST -s https://api.fastly.com/service/<Service ID>/version/<Editable Version>/snippet -H "Fastly-Key:FASTLY_API_TOKEN" -H `fastly-cookie` -H 'Content-Type: application/x-www-form-urlencoded' --data $'name=my_regular_snippet&type=recv&dynamic=0&content=if ( req.url ) {\n set req.http.my-snippet-test-header = "true";\n}';
```

Fastly returns a JSON response that looks like this:

```
{
 "service_id": "<Service Id>",
 "version": "<Editable Version>",
 "name": "my_regular_snippet",
 "type": "recv",
 "content": "if ( req.url ) {\n set req.http.my-snippet-test-header = \"true\";\n}",
 "priority": 100,
 "dynamic": 0,
 "id": "56789exampleid",
 "created_at": "2016-09-09T20:34:51+00:00",
 "updated_at": "2016-09-09T20:34:51+00:00",
 "deleted_at": null
}
```

> **NOTE**
>
> When regular VCL snippets get created, an `id` field will be returned that isn't used. The field only applies to dynamic VCL Snippets. In addition, the returned JSON includes a populated `content` field because the snippet content is stored in a versioned object.

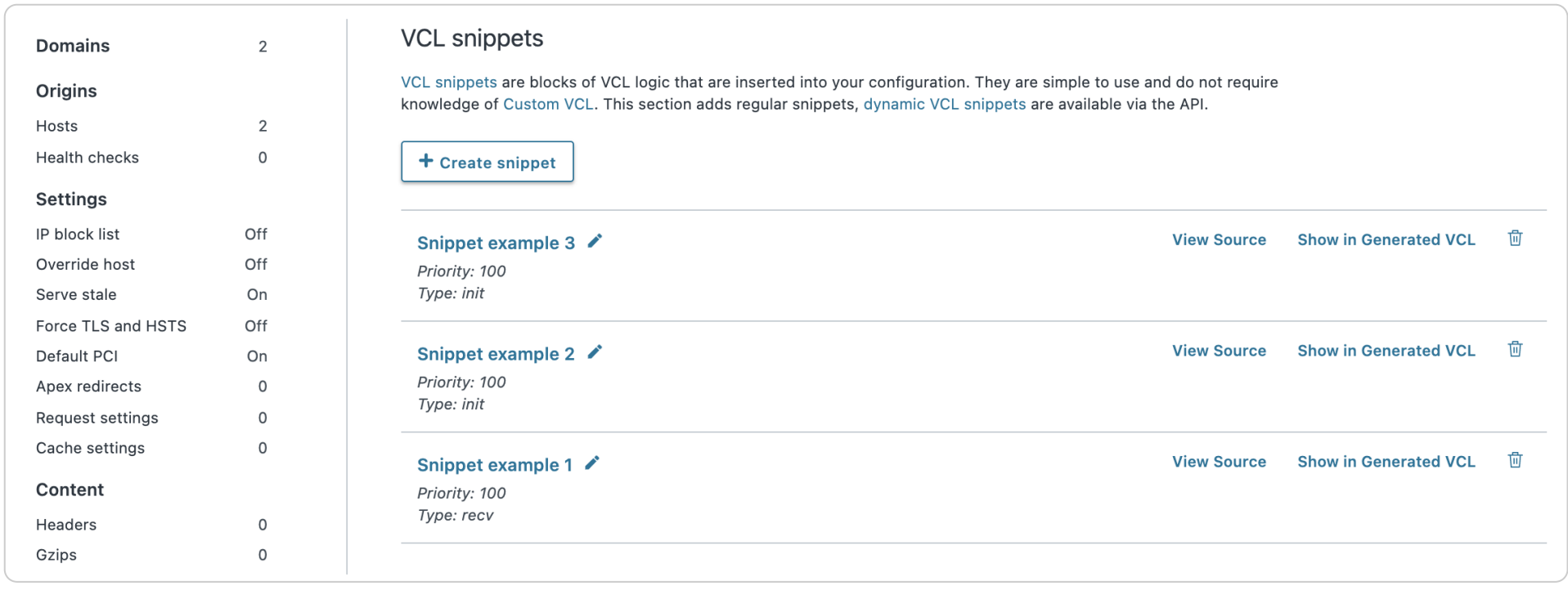
Viewing regular VCL Snippets in the web interface

You can view a list of regular VCL snippets. You can also view just the source of a specific snippet or a specific snippet's location in generated VCL.

Viewing a list of regular VCL Snippets

To view the entire list of a service's regular VCL Snippets directly in the web interface:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears listing all available VCL snippets for your service.



Viewing the source of a specific snippet

You can view just the source of a specific snippet:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the **View Source** link to the right of the name of the snippet. A view source window appears.

Viewing the location of a specific snippet in generated VCL

You can view a specific snippet's location in generated VCL:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the **Show in Generated VCL** link to the right of the name of the snippet. The Generated VCL window appears.

Fetching regular VCL Snippets via the API

You can fetch regular VCL Snippets for a particular service via the API either singly or all at once.

Fetching an individual regular VCL Snippet

To fetch an individual snippet, make the following API call in a terminal application:

```
$ curl -X GET -s https://api.fastly.com/service/<Service ID>/version/<Editable Version>/snippet/<Snippet Name e.g my_regular_snippet> -H "Fastly-Key:FASTLY_API_TOKEN"
```

Unlike fetching dynamic VCL Snippets you include the version in the URL and you must use the name of the snippet, not the ID.

Fetching a list of regular VCL Snippets

To list all regular VCL Snippets attached to a service, make the following API call in a terminal application:

```
$ curl -X GET -s https://api.fastly.com/service/<Service ID>/version/<Editable Version>/snippet/ -H "Fastly-Key:FASTLY_API_TOKEN"
```

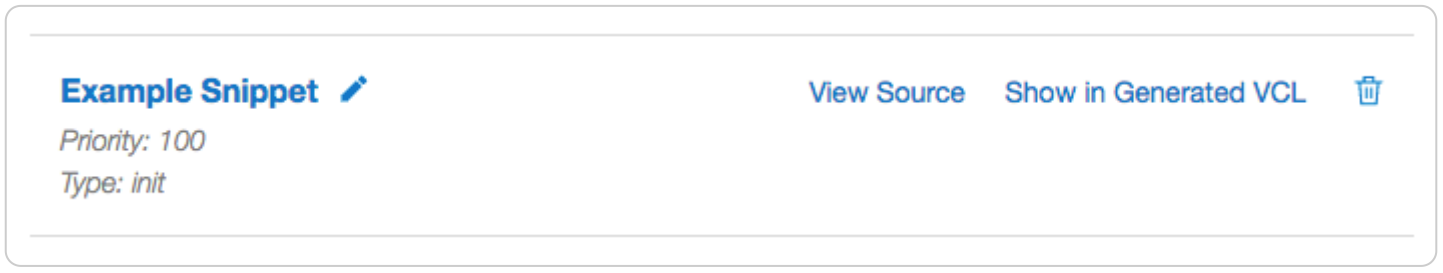
Updating an existing regular VCL Snippet

You can update existing regular VCL Snippets via the web interface or via the API.

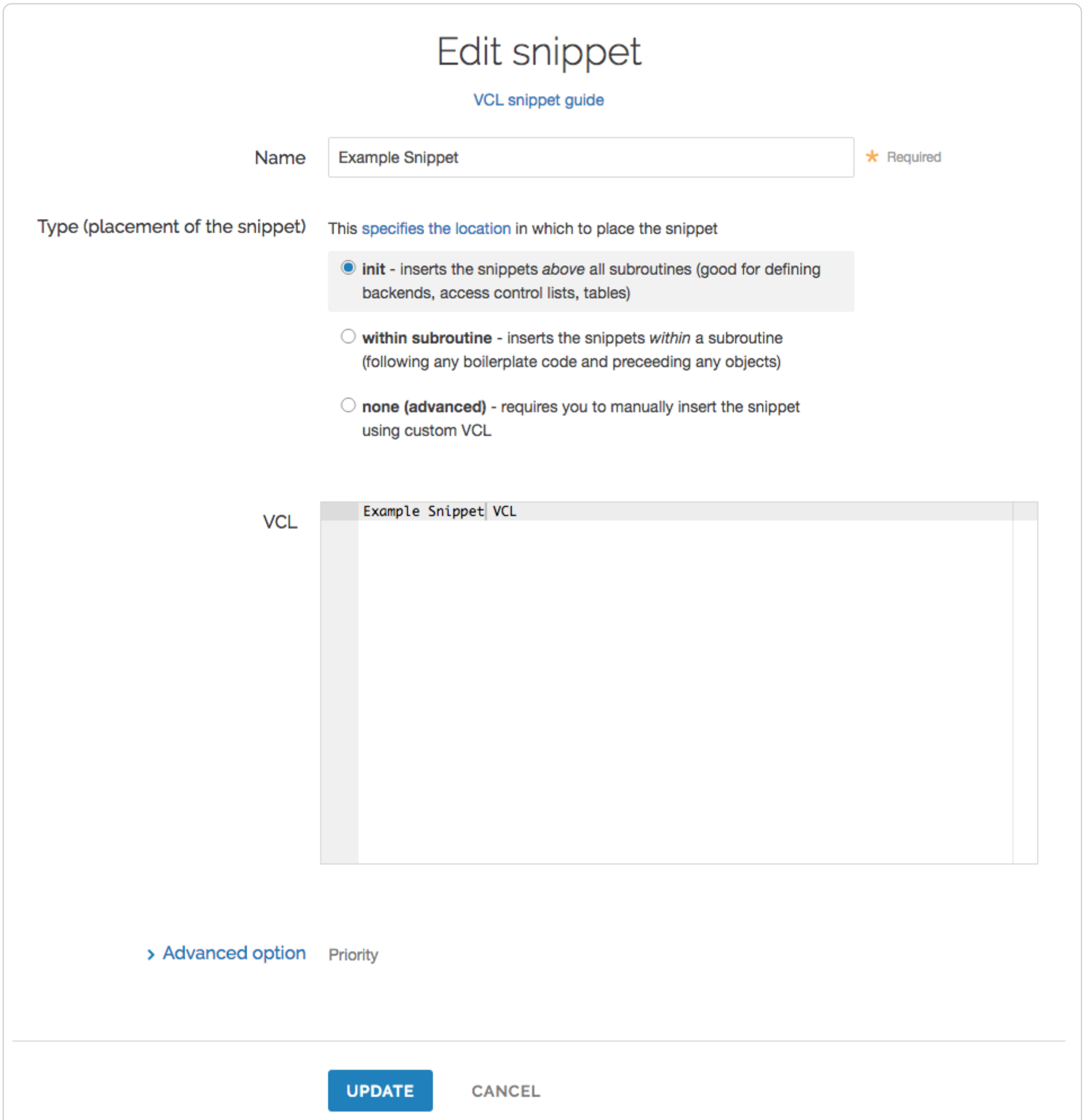
Via the web interface

To update an individual snippet via the web interface:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the pencil icon next to the name of the snippet to be updated.



The Edit snippet page appears.



6. Update the snippet's settings or VCL as appropriate.

7. Click **Update** to save your changes.

Via the API

To update an individual snippet via the API, make the following API call in a terminal application:

```
$ curl -X PUT -s https://api.fastly.com/service/<Service ID>/version/<Editable Version>/snippet/<Snippet Name e.g my_regular_snippet> -H "Fastly-Key:FASTLY_API_TOKEN" -H 'Content-Type: application/x-www-form-urlencoded' --data $'content=if ( req.url ) {\n set req.http.my-snippet-test-header = \"affirmative\";\n}';
```

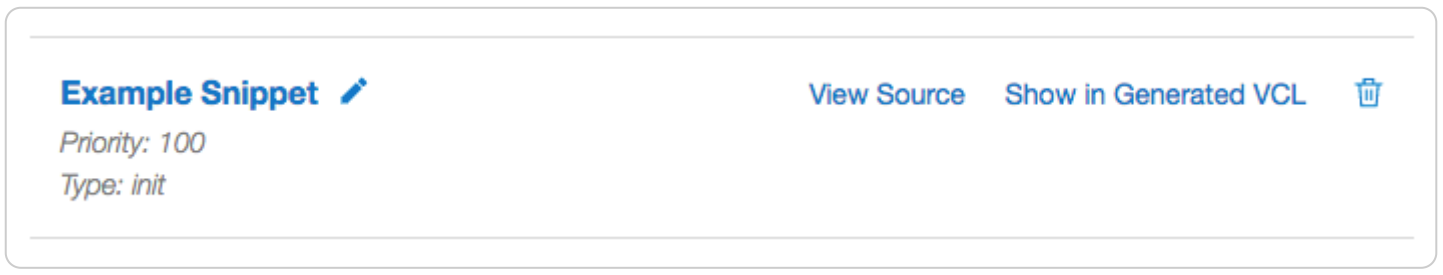
Deleting an existing regular VCL Snippet

You can update existing regular VCL Snippets via the web interface or via the API.

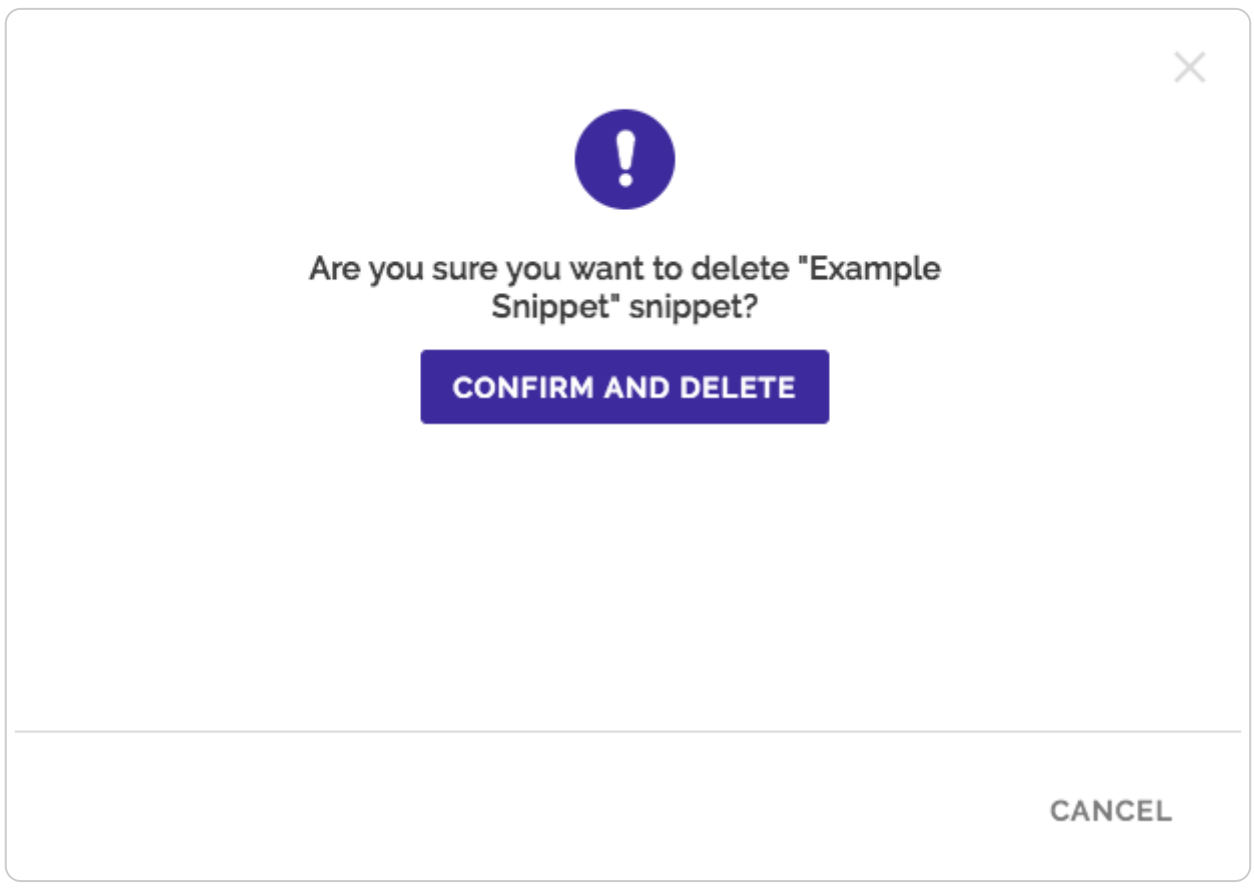
Via the web interface

1. Log in to the Fastly web interface.

2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the trashcan icon to the right of the name of the snippet to be updated.



A confirmation window appears.



6. Click **Confirm and Delete**.

Via the API

To delete an individual snippet via the API, make the following API call in a terminal application:

```
$ curl -X DELETE -s https://api.fastly.com/service/<Service ID>/version/<Editable Version>/snippet/<Snippet Name e.g my_regu
lar_snippet> -H "Fastly-Key:FASTLY_API_TOKEN"
```

Including regular snippets in custom VCL

Snippets will not be rendered in VCL if you select `none (advanced)` for the snippet type in the web interface or specify a location of `none` for the `type` parameter in the API. This allows you to manually include snippets in custom VCL using the following syntax:

```
include "snippet::<snippet name>"
```

The same VCL Snippet can be included in custom VCL in as many places as needed.

Example use: location-based redirection

Say that you work at a large content publisher and you want to redirect users to different editions of your publication depending on which country their request comes from. Say also that you want the ability to override the edition you deliver to them based on a cookie.

Using regular VCL snippets, you could add a new object with the relevant VCL as follows:

```
if (req.http.Cookie:edition == "US" || client.geo.country_code == "US") {
  set req.http.Edition = "US";
  set req.backend = F_US;
} elseif (req.http.Cookie:edition == "Europe" || server.region ~ "^EU-" ) {
  set req.http.Edition = "EU";
  set req.backend = F_European;
} else {
  set req.http.Edition = "INT";
  set req.backend = F_International;
}
```

This would create an Edition header in VCL, but allow you to override it by setting a condition. You would add the Edition header into Vary and then add a false condition (e.g., `!req.url`) to your other backends to ensure the correct edition of your publication gets delivered (Remember: VCL Snippets get added to VCL before backends are set.)

VCL regular expression cheat sheet

Last updated: 2018-08-02

https://docs.fastly.com/en/guides/vcl-regular-expression-cheat-sheet

Fastly VCL uses a subset of Perl Compatible Regular Expression (PCRE) syntax. This is case sensitive and forward slashes don't need to be escaped. To disable case sensitivity, add `(?i)` to the start of your expression.

Basic matching

```
req.url == "/phrase"
```

Matches only if `req.url` is exactly `/phrase`.

```
req.url ~ "phrase"
```

Matches `phrase` anywhere.

Matching at the beginning or end of a string

```
req.http.host ~ "^www"
```

Matches if `req.http.host` starts with `www`.

```
req.url ~ "\.jpg$"
```

Matches if `req.url` ends with `.jpg`.

Multiple matches

```
req.url ~ "\.(png|jpg|css|js)$"
```

Matches if `req.url` ends with either `.png`, `.jpg`, `.css`, or `.js`.

```
req.url ~ "\.php(\?.*)?$"
```

Matches if `req.url` ends with `.php`, `.php?foo=bar` or `.php?`, but not `.phpa`.

NOTE

You can also use `req.url.ext` to find the file extension specified in a URL. For example, in the request `www.example.com/1/hello.gif?foo=bar`, `req.url.ext` will contain `gif`.

```
req.url ~ "\.[abc]server$"
```

Matches if `req.url` ends with `.aserver`, `.bserver` or `.cserver`.

Matching wildcards

```
req.url ~ "jp.g$"
```

Matches if `req.url` ends with `jpeg`, `jpag`, and `jp0g`, but doesn't match if `req.url` ends with `jpg`. It also matches if any other character is between the `jp` and the `g`.

```
req.url ~ "jp.*g$"
```

Matches `jp` followed by 0 or more random characters ending with the letter `g` (`jpeg`, `jpg`, and `jpeeeeg` all match).

Capturing matches

```
set req.http.Foo = "abbbccccc";
if (req.http.Foo ~ "^(a+)(b+)(c+)") {
  set resp.http.match0 = re.group.0; # now equals 'abbbccccc'
  set resp.http.match1 = re.group.1; # now equals 'a'
  set resp.http.match2 = re.group.2; # now equals 'bbb'
  set resp.http.match3 = re.group.3; # now equals 'cccccc'
}
```

The `re.group.[0-9]` objects allow you to capture matches. The `re.group.0` object evaluates to the entire matched string even if no capture groups have been supplied. You can use these objects to replace this example:

```
if (req.url ~ "(?i)\?.*some_query_arg=([^&]*)") {
  set req.http.Thing-I-Want = regsub(req.url, "(?i)\?.*some_query_arg=([^&]*).*", "\1");
}
```

You can use `re.group` to greatly simplify the previous example:

```
if (req.url ~ "(?i)\?.*some_query_arg=([^&]*)") {
  set req.http.Thing-I-Want = re.group.1;
}
```

You could even get really fancy and do something like this:

```
set req.http.Thing-I-Want = if(req.url ~ "(?i)\?.*some_query_arg=([^&]*)", re.group.1, "");
```

Replacing content

```
set req.http.host = regsub(req.http.host, "^www\.","");
```

Removes a leading `www.` from the host, if present.

```
set req.http.api-test = regsub(req.http.host, "^www\.","api.");
```

Sets the `api-test` header to contain the host-header, but replaces a leading `www.` with `api.`:

```
Host: www.example.com ->
Host: www.example.com
api-test: api.example.com
Host: example.com ->
Host: example.com
api-test: example.com
```

```
set req.url = regsuball(req.url, "/+", "/");
```

Changes all occurrences of multiple slashes in the URL to a single slash. For example, `//docs///intro.html` will be transformed to `/docs/intro.html`.

---

§

These articles describe how to move rapid key/value pair decision logic to the edge using dictionaries.

https://docs.fastly.com/en/guides/configuration#_dictionaries

About Edge Dictionaries

Last updated: 2022-03-02

https://docs.fastly.com/en/guides/about-edge-dictionaries

Edge Dictionaries are a type of container that allow you to store data as key-value pairs that can be used in a service without being attached to a single version.

Using Edge Dictionaries, you can turn frequently repeated statements like this:

```
if (something == "value1") {
  set other = "result1";
} else if (something == "value2") {
  set other = "result2";
}
```

into a single function that acts as constant, like this:

```
table <ID> {
  "KEY_STRING": "VALUE_STRING",
  "KEY_STRING2": "VALUE_STRING2",
  ...
}
```

This allows you to easily make changes to these statements without it being tied to a specific version.

When Edge Dictionaries might be useful

- Content sharing and social media outlets updating large referer block lists
- Mobile advertisers validating a key to prevent cache-bust guessing
- Customers authenticating valid user keys at the edge (see private Edge Dictionaries
- Global publishers redirecting users to a specific country site based on geo-location
- Image providers performing token checks for certain objects
- Advertising technology companies blocking bad actors at edge
- Customers deploying web interface versions with simple value change via API

How dictionaries work

Edge Dictionaries are made up of dictionary containers and the dictionary items within them. Once you attach a dictionary container to a version of your service and that service is activated, the data in it becomes *versionless*. This means you can add to and update the data an Edge Dictionary contains at any time after it is created, without ever incrementing a service's version.

For example, say you have a referer block list that changes frequently and you want to associate it with a service. Any time that service's configuration changes, especially if the configuration rolls back to a previous version, you would want the block-listed referer domains to continue to remain with the service configuration instead of being removed. Edge Dictionaries would help you do this.

How to create and use dictionaries

To create an Edge Dictionary and use it within your service, start by creating an empty dictionary container and then add its entries in a working version of a service that's unlocked and not yet activated. You can create dictionaries:

- via the Fastly web interface).
- via VCL snippets.
- via the Fastly API.

> **TIP**
>
> You can create a private Edge Dictionary to store dictionary items that can't be listed or read via the web interface or the API.

Limitations and considerations

When creating Edge Dictionaries, keep the following things in mind:

- **Edge Dictionaries created with custom VCL cannot be manipulated using the API or the web interface.** If you create a dictionary container using custom VCL, that dictionary must always be manipulated via custom VCL. Dictionaries uploaded via custom VCL aren't versionless.
- **Dictionary containers, item keys, and their values have specific limits.** Dictionary containers are limited to 1000 items. Dictionary item keys are limited to 256 characters and their values are limited to 8000 characters. If you find your dictionaries approaching these resource limits, contact us at support@fastly.com. We may be able to help you figure out more efficient ways to do things.
- **Dictionary item keys are case sensitive.** The names of dictionary items are case sensitive. When designing your Edge Dictionaries, be sure to take this into account.
- **The contents of Edge Dictionaries are stored as VCL.** Personal data should not be incorporated into VCL. Our Compliance and Law FAQ describes in detail how Fastly handles personal data privacy.

When making changes to Edge Dictionaries, keep the following things in mind:

- **When you delete a dictionary container, you'll only delete it from the service version you're editing.** Dictionary containers are tied to versions and can be cloned and reverted. When using Edge Dictionaries, we want you to be able to do things like delete a dictionary container from a current version of your service in order to roll back your configuration to a previous version using as few steps as possible.
- **When you delete a dictionary container, we don't delete the dictionary items inside it.** The dictionary items in a dictionary container are versionless. When you change service versions, we want you to still be able to access the data.
- **Dictionary item deletions are permanent.** Because we don't store data, if you delete a dictionary item, the entry is gone forever from all service versions.
- **Event logs don't exist for Edge Dictionary changes.** If you add, update, or remove a dictionary item, there will be no record of it. The only record of a change will exist when you compare service versions to view the point at which the dictionary container was associated with the service version in the first place.

> **IMPORTANT**
>
> Personal information, secrets, or sensitive data should not be included in dictionaries or incorporated into VCL. In addition, we do not maintain version histories of your dictionaries. Our Compliance and Law FAQ describes in detail how Fastly handles personal data privacy.

Working with Edge Dictionaries using the web interface

Last updated: 2022-03-02

https://docs.fastly.com/en/guides/working-with-dictionaries-using-the-web-interface

Edge Dictionaries are a type of container that allow you to store data as key-value pairs that can be used in a service without being attached to a single version. Edge Dictionaries are made up of dictionary containers and dictionary items. You use dictionary items to create and store the key-value pairs, which are then added to a dictionary container. The dictionary container is attached to a service version but can be updated at any time after it is created, without ever incrementing a service's version.

You can work with dictionaries using the web interface, custom VCL snippets, or Fastly API. You can also create private edge dictionaries.

Before you begin

Before working with edge dictionaries, be sure to review our About edge dictionaries guide to familiarize yourself with how edge dictionaries work, common use cases, and limitations to using dictionaries.

Working with dictionaries in the web interface

You can create and interact with edge dictionaries using the web interface.

Viewing dictionaries created using the web interface

To view a dictionary via the web interface, navigate to the dictionary management area of your service:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Dictionaries** link under **Data**. Existing dictionaries, if any, associated with the currently selected service version appear.

> **NOTE**
>
> Remember that dictionary containers are versioned. If you don't see an dictionary attached to your service, check the service version to make sure you're looking at the right one.

Creating a dictionary via the web interface

Creating a dictionary via the web interface requires you to create a dictionary container and then create the items that will exist in it.

Creating a dictionary container

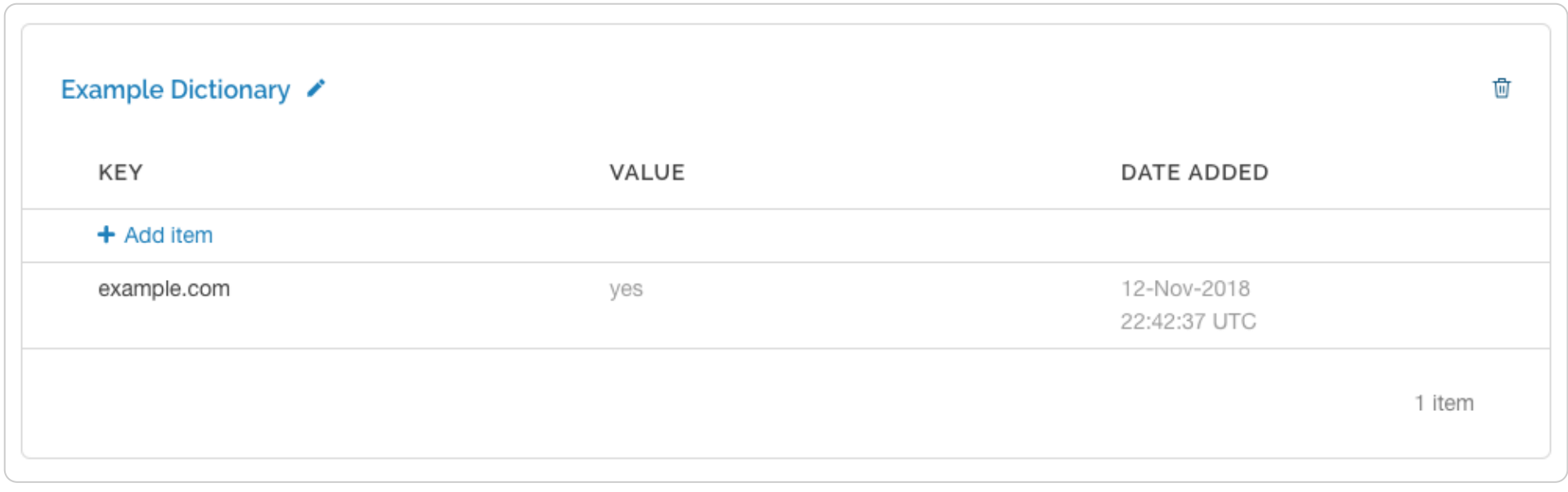
Start by creating a dictionary container using the following steps:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Dictionaries** link under **Data**. The Dictionaries page appears.
5. Click **Create a dictionary**. The dictionary container name field appears.
6. In the **Name of dictionary** field, enter a descriptive name for the dictionary (e.g., `Example Dictionary`).
7. Click the **Add** button. The empty dictionary container you created appears.
8. Click the **Activate** button to deploy your configuration changes to the service version you're editing.

Creating a dictionary item

Once you've created a dictionary container, add items into it:

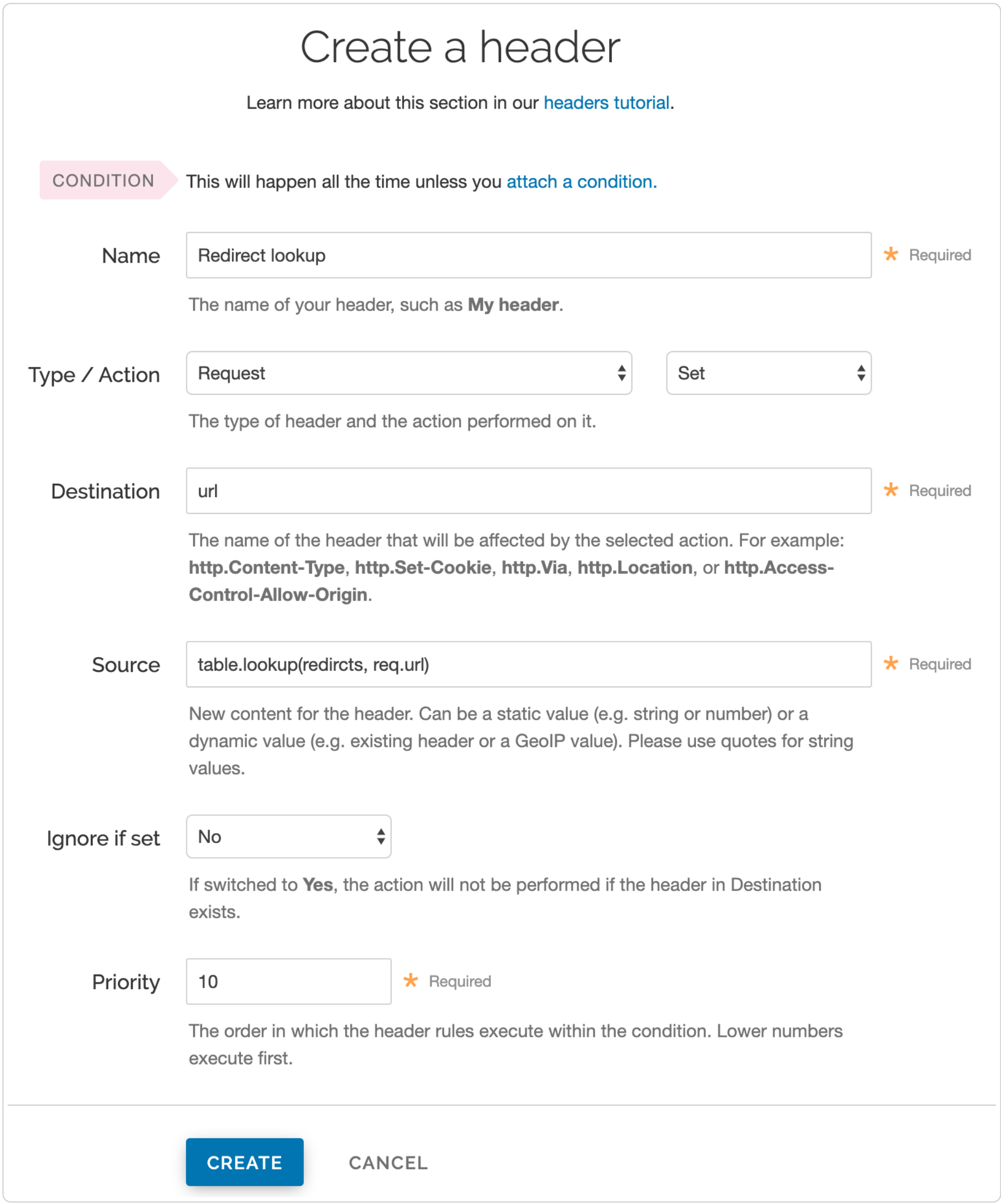
1. Click the **Add item** link. The dictionary item fields appear.
2. In the **Key** field, enter the unique identifier for some item of data (e.g., `example.com`).
3. In the **Value** field, enter the value associated with the unique identifier (e.g., `yes`)
4. Click the **Add** button. The key-value pair appears in the dictionary container. This addition will become effective immediately.



Using a dictionary via the web interface

Once you've created a dictionary, you can start using it. To use a dictionary via the web interface, you'll need to create and add a header. Follow the steps below:

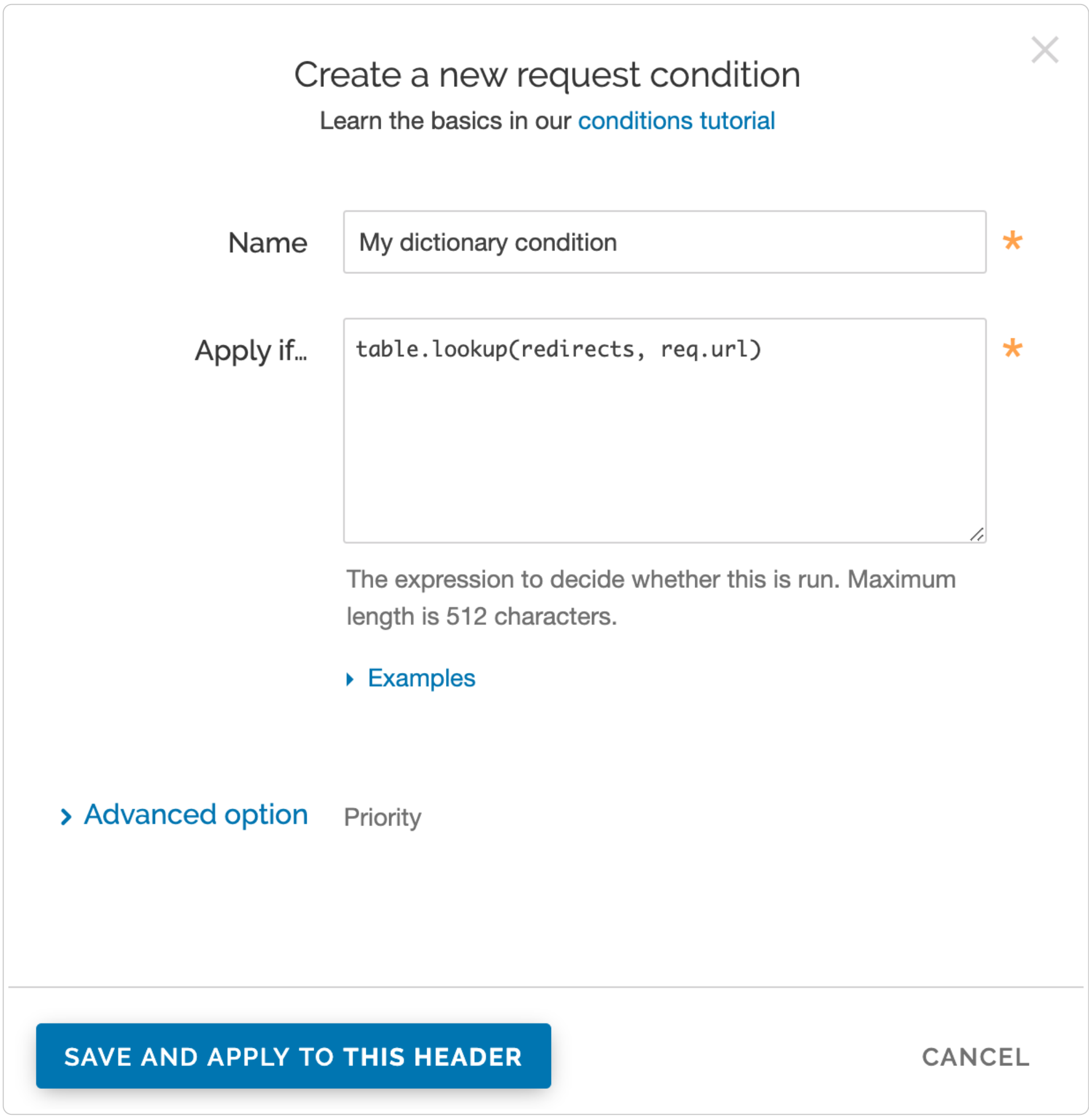
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter the name of your header rule (for example, `Redirect lookup`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter the name of the header affected by the selected action (e.g., `url`).
   - In the **Source** field, enter where the content for the header comes from (e.g., `table.lookup(redirects, req.url)`).
   - From the **Ignore if set** menu, select **No**
   - Leave the **Priority** field as is.
7. Click the **Create** button to create the header. A new header appears on the Content page.

Once you've created a header, you can create a condition to specify when to use the dictionary:

1. Click the **Attach a condition** link next to new header you just created. The Add a condition message appears.
2. Click the **Create a new request condition** button. The Create a new condition page appears.



3. In the **Name** field, enter a name for your condition (e.g., `My dictionary condition`)
4. In the **Apply if** field, enter a condition (e.g., `table.lookup(redirects, req.url)`)
5. Click the **Save and apply to** button.
6. Click the **Activate** button to deploy your configuration changes.

Editing a dictionary container via the web interface

You can edit the name of a dictionary container that was created via the web interface in any unlocked service version:

1. Find a dictionary associated with an unlocked version of your service.
2. Click the pencil icon next to the dictionary container name.
3. Change the name, then click the **Save** button.

Editing a dictionary item via the web interface

You can edit the dictionary items within a container at any time. To edit the key-value pair in a dictionary container that was created via the web interface:

1. Find any dictionary associated with your service in which the key-value pairs appear. Because dictionary items are versionless, the service version you choose doesn't matter. Choose the one that makes the most sense to you.
2. Hover your cursor over a dictionary item, then click the pencil icon that appears.
3. Edit the key or value as necessary.
4. Click the **Save** button. The changes you make will be immediately applied to your configuration. If your dictionary container has already been associated with a deployed service version, those changes will happen live.

Deleting a dictionary via the web interface

Keeping in mind their limitations, dictionary containers and the items within them can be deleted via the web interface.

Deleting a dictionary container

You can delete a dictionary container that was created via the web interface in any unlocked service version:

1. Find a dictionary associated with an unlocked version of your service.
2. Click the trash can icon in the top right corner of the dictionary.
3. Click the **Confirm and delete** button.
4. Click the **Activate** button to deploy your configuration changes to the service version you're editing.

Deleting a dictionary entry

You can delete the dictionary entries within a container at any time. To delete a key-value pair included in a dictionary container that was created via the web interface:

1. Find any dictionary associated with your service in which the key-value pairs appear. Because dictionary items are versionless, the service version you choose doesn't matter. Choose the one that makes the most sense to you.
2. Hover your cursor over a dictionary item, then click the trash can icon that appears.
3. Click the **Confirm and delete** button.

Working with dictionaries using VCL snippets

You can create and interact with edge dictionaries using custom VCL snippets. Edge Dictionaries created with custom VCL cannot be manipulated using the API or the web interface. If you create a dictionary container using custom VCL, that dictionary must always be manipulated via custom VCL. Dictionaries uploaded via custom VCL aren't versionless.

Viewing dictionaries created by VCL snippets

To view a dictionary under the VCL Snippets link, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The snippet titles associated with the currently selected service version appear.
5. To view the contents of the dictionary, click the **View source** button.

Creating a dictionary using VCL snippets

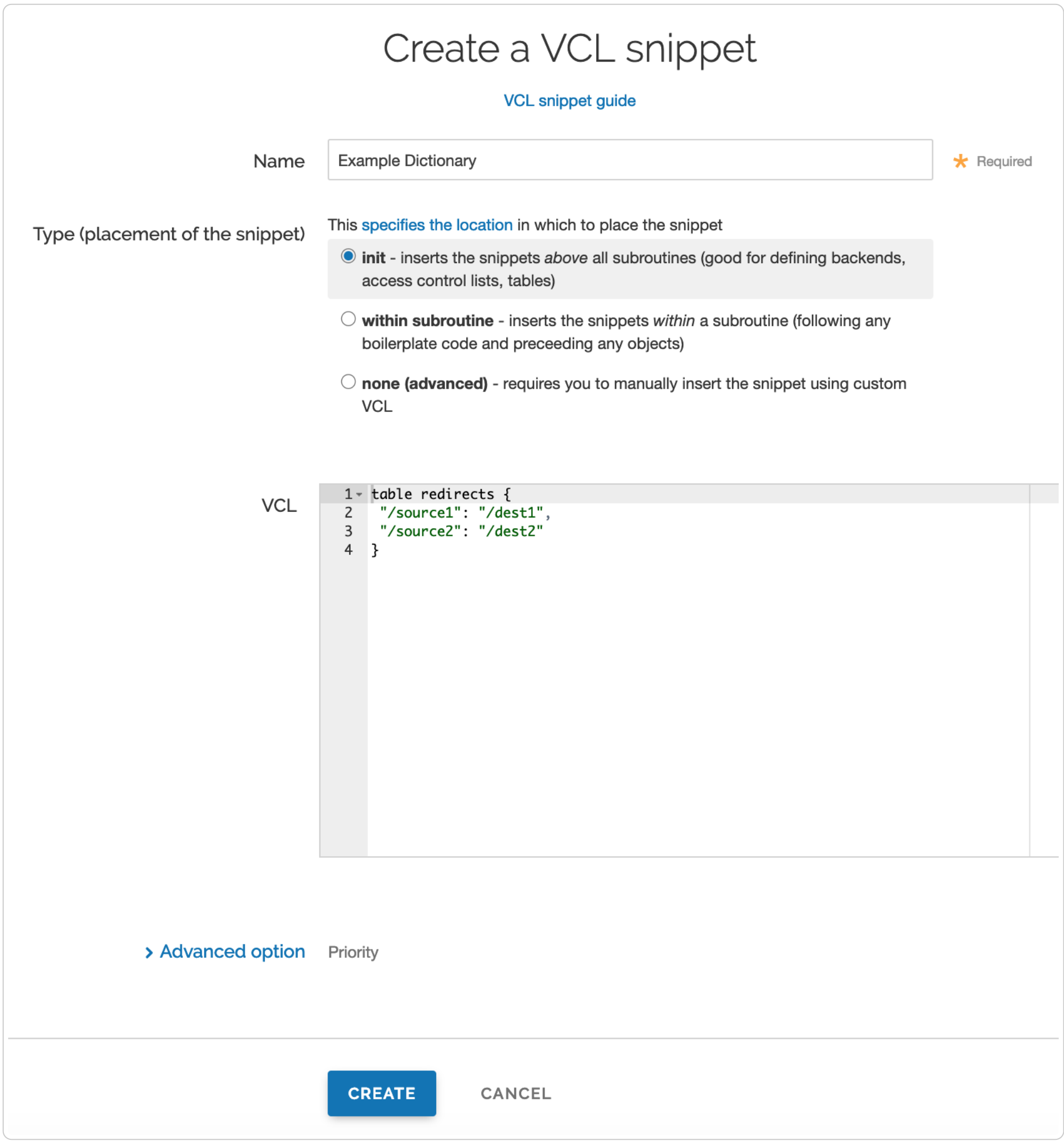
To create a dictionary using VCL snippets, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.

3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.

4. Click the **VCL snippets** link. The VCL Snippets page appears.

5. Click **Create snippet**. The Create a VCL snippet page appears.



6. In the **Name** field, enter an appropriate name (e.g., `Example Dictionary`).

7. From the **Type (placement of the snippet)**, select **init**.

8. In the **VCL** field, create a table and add key-value pairs. For example, if you want to create a table that redirects a URL to another path:

```
table redirects {
  "/source1": "/dest1",
  "/source2": "/dest2"
}
```

where the table is a set of key-value pairs that you can reference in your code. You can replace the contents of this table with different key-value pairs.

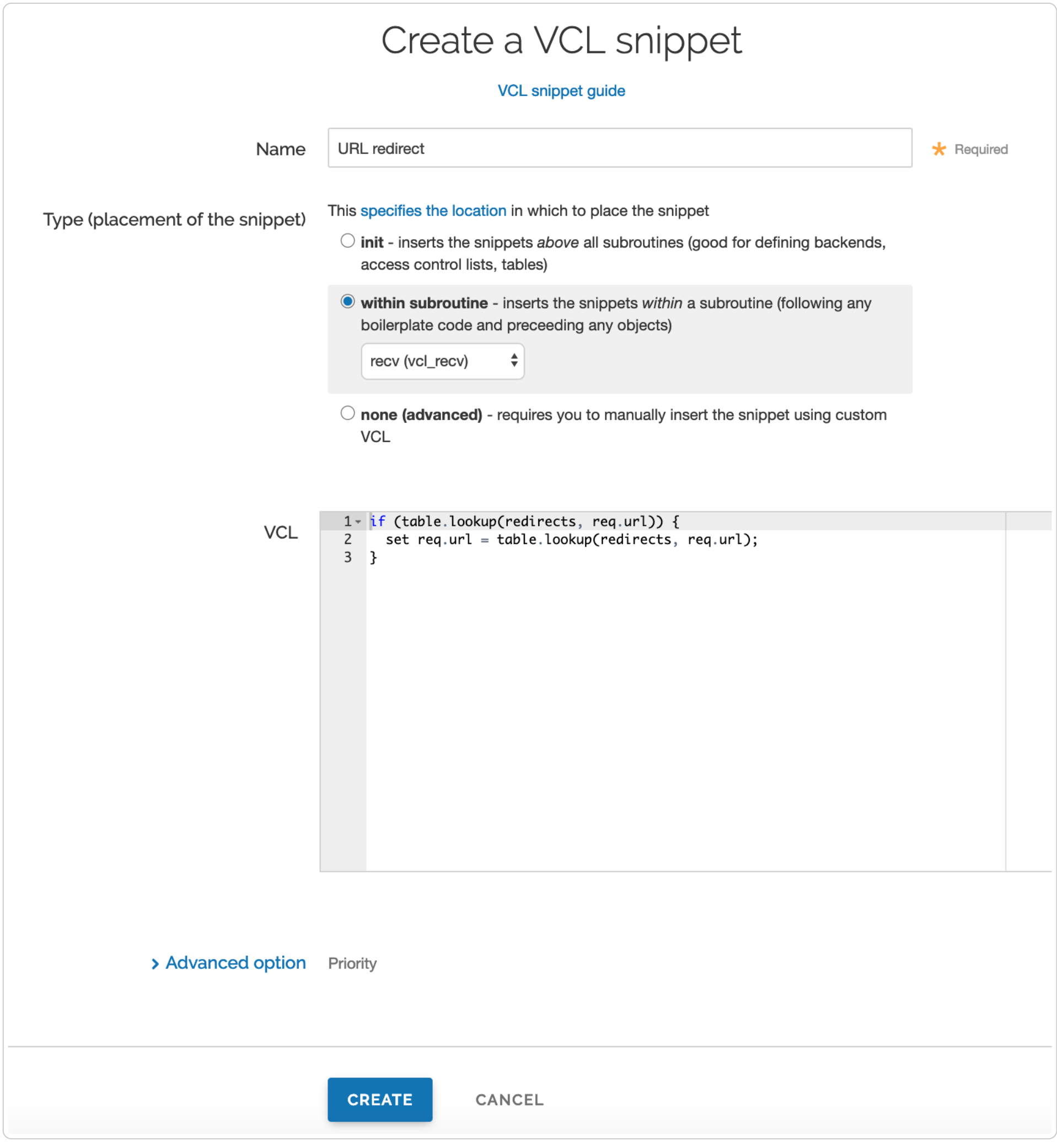
9. Click **Create** to create the snippet.

Using a dictionary with VCL snippets

Once you've created a dictionary, you can start using it.

To start using your dictionary with VCL Snippets, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL snippets** link. The VCL Snippets page appears.
5. Click **Create snippet**. The Create a VCL snippet page appears.



6. In the **Name** field, enter an appropriate name (e.g., `URL redirect`).

7. From the **Type (placement of the snippet)** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **recv (vcl_recv)**.
9. In the **VCL** field, add a condition to use the table you created in [Creating a dictionary using VCL Snippets](). For example, if you need to rewrite your URL destination, you could use:

```
if (table.lookup(redirects, req.url)) {
  set req.url = table.lookup(redirects, req.url);
}
```

   where `table.lookup` checks the dictionary for the desired contents you want. The first parameter is the table being looked in and the second parameter is the key you're looking for. If the key exactly matches the second parameter, that value is returned. Be aware that regex doesn't work with a dictionary lookup.

10. Click **Create** to create the snippet.
11. Click the **Activate** button to deploy your configuration changes.

Editing a dictionary using VCL snippets

You can edit the dictionary name and the condition that was created using VCL Snippets in any unlocked service version:

1. Find a [dictionary]() associated with an unlocked version of your service.
2. Click the pencil icon next to the dictionary. You can now make changes to the name and the condition.
3. Click the **Update** button once you're finished with your changes.
4. Click the **Activate** button to activate the version you made the edits in and view the changes in your VCL.

Deleting a dictionary using VCL snippets

You can delete a dictionary using VCL Snippets by following the steps below. Keep in mind the [limitations]() involved when deleting a dictionary.

1. [Find a dictionary]() associated with an unlocked version of your service.
2. Click the trash can icon on the top right corner of the snippet
3. Click the **Confirm and delete** button.
4. Click the **Activate** button to deploy your configuration changes to the service version you're editing.

Private edge dictionaries

Private Edge Dictionaries store dictionary items that can't be listed or read via the web interface or the API.

Limitations and considerations

When creating private Edge Dictionaries, keep the following things in mind:

- You can create, read, update, and delete a private dictionary
- You cannot update the `write_only` attribute of a dictionary
- You can create, update, and delete items that belong to a private dictionary
- You cannot view items that belong to a private dictionary via the API
- Depending on how your [service]() is configured, data stored in private Edge Dictionaries can be sent in [headers]() and to [log streaming endpoints]().

> **WARNING**
>
> To edit or delete dictionary items in a private dictionary, you'll need to remember [the keys of the dictionary items]().

Creating a private dictionary container

To use a private dictionary container, start by creating an empty one within an unlocked version of a service.

Before a private Edge Dictionary can be manipulated, its private dictionary container must be associated with at least one service version that is not locked and not active so that the service becomes aware of the private dictionary's existence.

For example, if you were creating a `my_example_dictionary` private dictionary via the API, you would make an API call by running this command:

```
$ curl -X POST -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'name=my_example_dictionary&write_only=true' https://api.fastly.com/service/<service_id>/version/<version_number>/dictionary
```

which would return:

```
{
  "created_at": "2017-05-03T16:11:41+00:00",
  "deleted_at": null,
  "id": "<dictionary_id>",
  "name": "my_example_dictionary",
  "service_id": "<service_id>",
  "updated_at": "2017-05-03T16:11:41+00:00",
  "version": <version_number>,
  "write_only": true
}
```

You can start adding dictionary items after you've created the dictionary. Don't forget to activate the service when you're finished.

Viewing private dictionaries in VCL

The contents of private Edge Dictionaries are hidden in VCL. The private dictionary's metadata is displayed, as shown below:

```
table my_example_dictionary {
     # REDACTED dictionary content
     # last_updated: 2017-10-16 20:44:43
     # item_count: 2
     # digest: 8f92141234567890da30ba9cea7d98ef614
 }
```

What's next

Explore custom VCL examples for using edge dictionaries on Developer Hub.

IMPORTANT

Personal information, secrets, or sensitive data should not be included in dictionaries or incorporated into VCL. In addition, we do not maintain version histories of your dictionaries. Our Compliance and Law FAQ describes in detail how Fastly handles personal data privacy.

---

§

These articles describe configuration settings and changes you can make to your domains and origins when setting up Fastly services.

https://docs.fastly.com/en/guides/configuration#_domains-and-origins

Changing origins based on user location

Last updated: 2018-08-01

https://docs.fastly.com/en/guides/changing-origins-based-on-user-location

Fastly allows you to change origin servers based on the user's geographic location. This is useful when you need to serve different content to users who are in different locations. For example, you could change origin servers to serve a restricted version of your website to users in a different country.

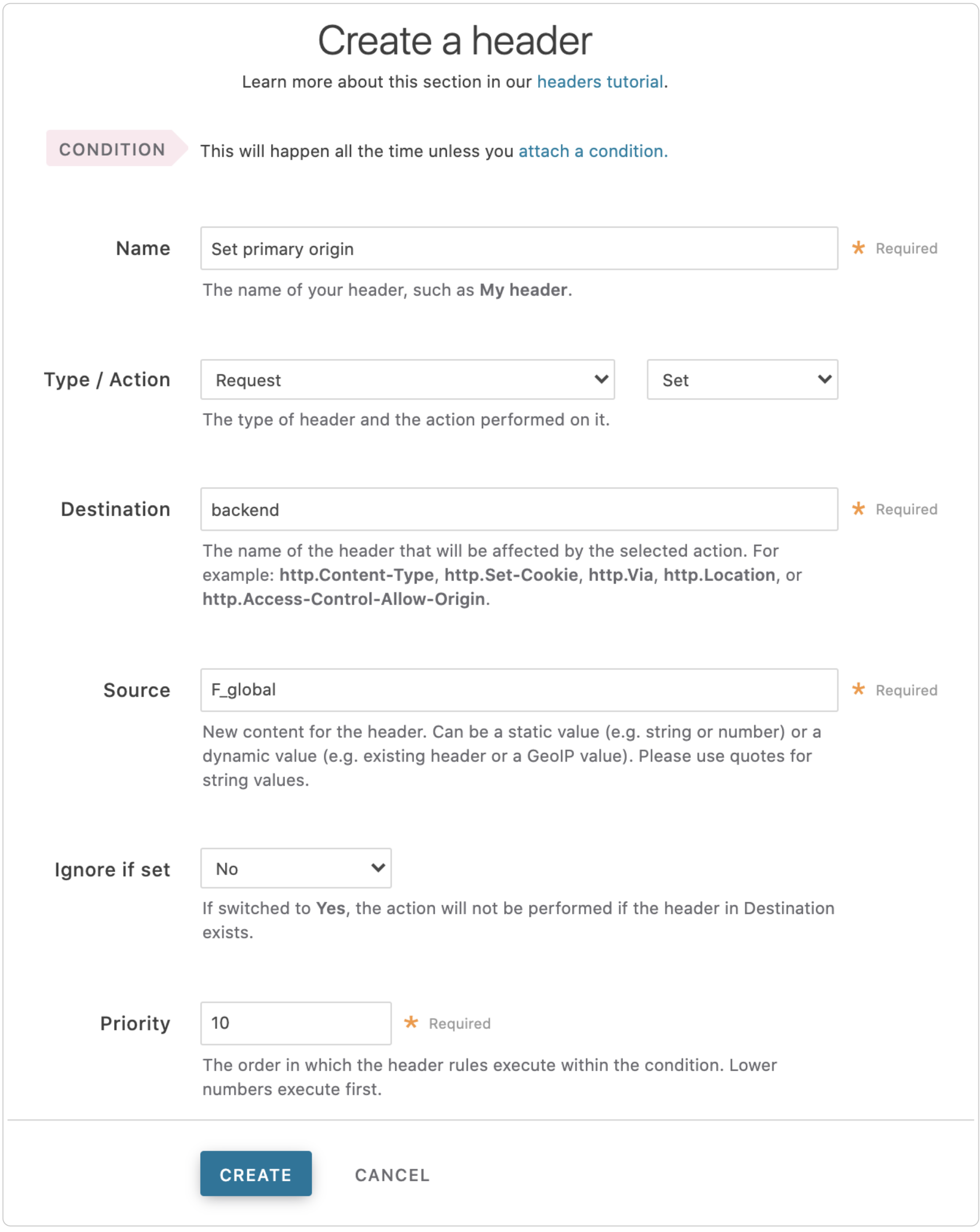
Using the web interface

You can use the web interface to create the headers and the condition.

Creating the header for the default origin server

First, create a header for the default origin server to serve content to the majority of users. Follow these instructions to create the header:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter the name of your header rule (for example, `Set default origin`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `backend`.
   - In the **Source** field, enter the name of the origin server you want to serve content to the majority of users (here it's `F_global`). Preview the VCL to find the name of the origin server.
   - From the **Ignore if set** menu, select **No**.

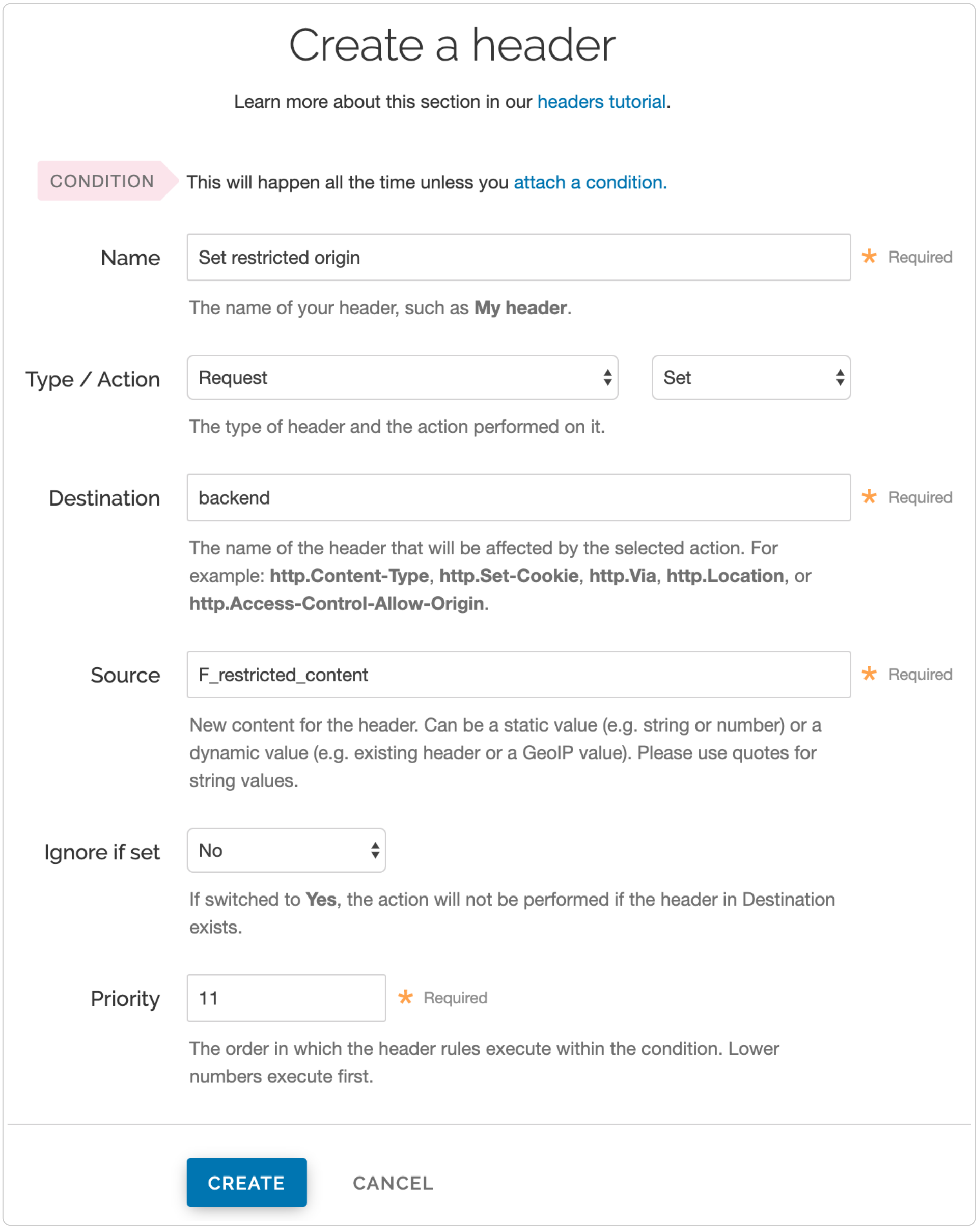
- In the **Priority** field, enter `10`.

7. Click the **Create** button.

Creating the header for the restricted origin server

Now, create a header for the restricted origin server to serve content to the users residing in the countries specified in the condition. Follow these instructions to create the header:

1. Click the **Content** link. The Content page appears.
2. Click the **Create header** button. The Create a header window appears.



3. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter the name of your header rule (for example, `Set restricted origin`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `backend`.
   - In the **Source** field, enter the name of the restricted origin server you want to serve content to the users residing in the countries specified in the condition (here it's `F_restricted_content`). Preview the VCL to find the name of the origin server.
   - From the **Ignore if set** menu, select **No**.

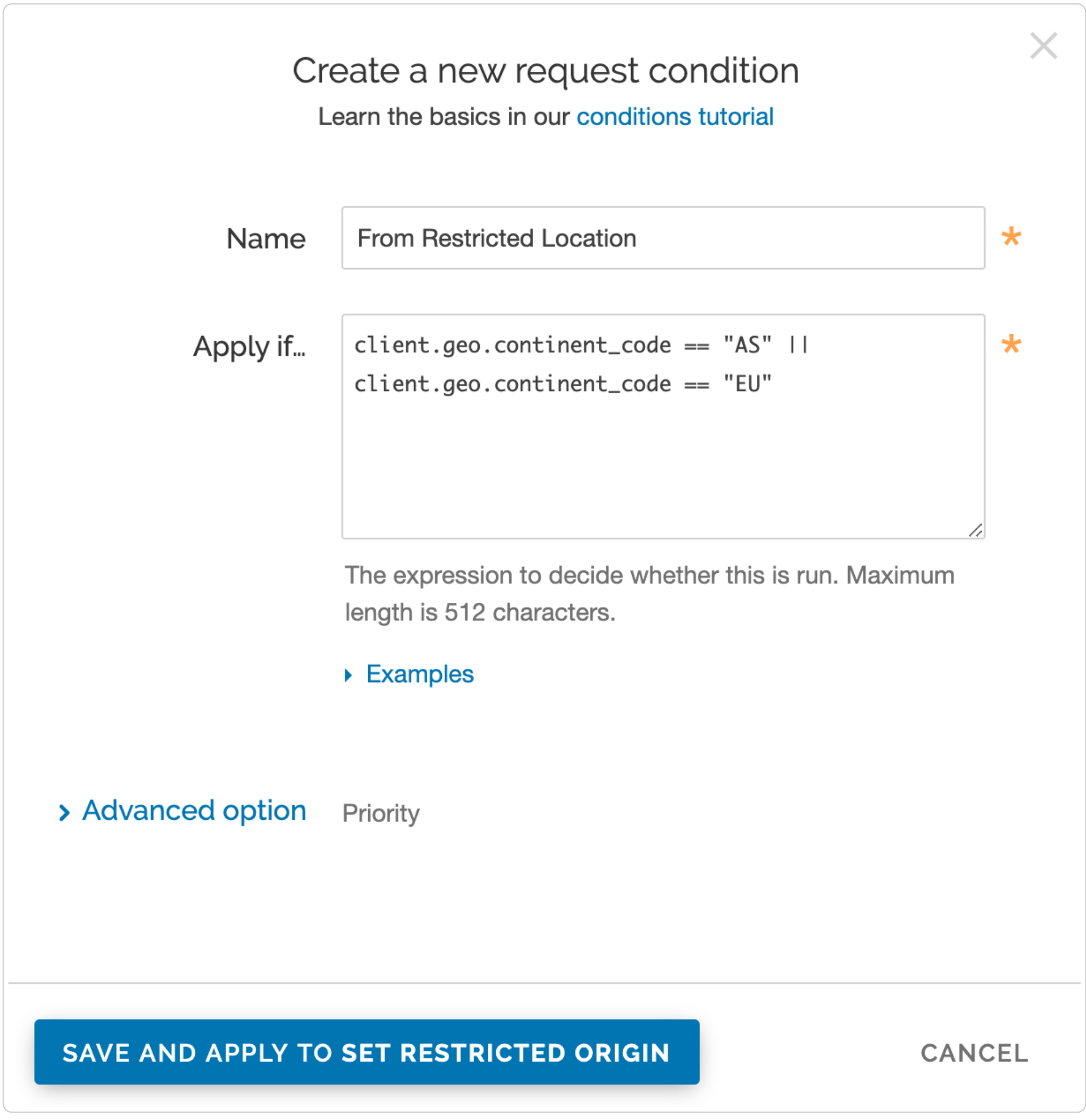
- In the **Priority** field, enter `11`.

4. Click the **Create** button.

Creating a condition for the restricted origin header

Finally, create a condition for the restricted origin header. The condition checks the geolocation header. If the user's geolocation matches a location specified in the condition, Fastly uses the restricted origin server. Follow these instructions to create the condition:

1. Click the **Content** link. The Content page appears.
2. In the Headers section, click the **Attach a condition** link next to the **Set restricted origin** header. The Create a new request condition window appears.



3. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter a descriptive name for the new condition (for example, `From Restricted Location`).
   - In the **Apply if** field, enter a request condition. For example, to send all users in Asia and Europe to the restricted origin server, enter `client.geo.continent_code == "AS" || client.geo.continent_code == "EU"`. See Geolocation-related VCL features for more information.

4. Click the **Save and apply to** button.

5. Click the **Activate** button to deploy your configuration changes.

Using custom VCL

If you'd prefer not to use the web interface, you can use custom VCL to configure your service to change origin servers based on the user's geographic location. Use the following VCL as a starting point:

```
# default conditions
set req.backend = F_global;

# Use restricted content if the user is in Asia, France or Germany
if (client.geo.continent_code == "AS" || client.geo.country_code == "FR" || client.geo.country_code == "DE") {
  set req.backend = F_restricted_content;
}
```

Failover configuration

Last updated: 2021-02-19

https://docs.fastly.com/en/guides/failover-configuration

This guide describes how to configure a failover origin server. A failover (backup) server ensures you can maintain availability of your content if your primary server is not available.

Before you begin

Before you configure failover, keep in mind the following:

- To configure a failover origin server you must make sure you have health checks configured for your primary server. If you configure your failover server but don't configure health checks on the primary server, the failover won't work properly if your primary server stops responding.
- Many customers configure load balancing at the same time they configure failover functionality. Our guide on configuring load balancing can show you how.

Configuring a failover origin server

Once you've confirmed health checks are configured, you must:

1. Turn off automatic load balancing on both the primary origin server and the server that will become your failover.
2. Create headers that configure both the primary and failover origin servers.
3. Create a header condition that specifies exactly when to use the failover server.

Turn off automatic load balancing

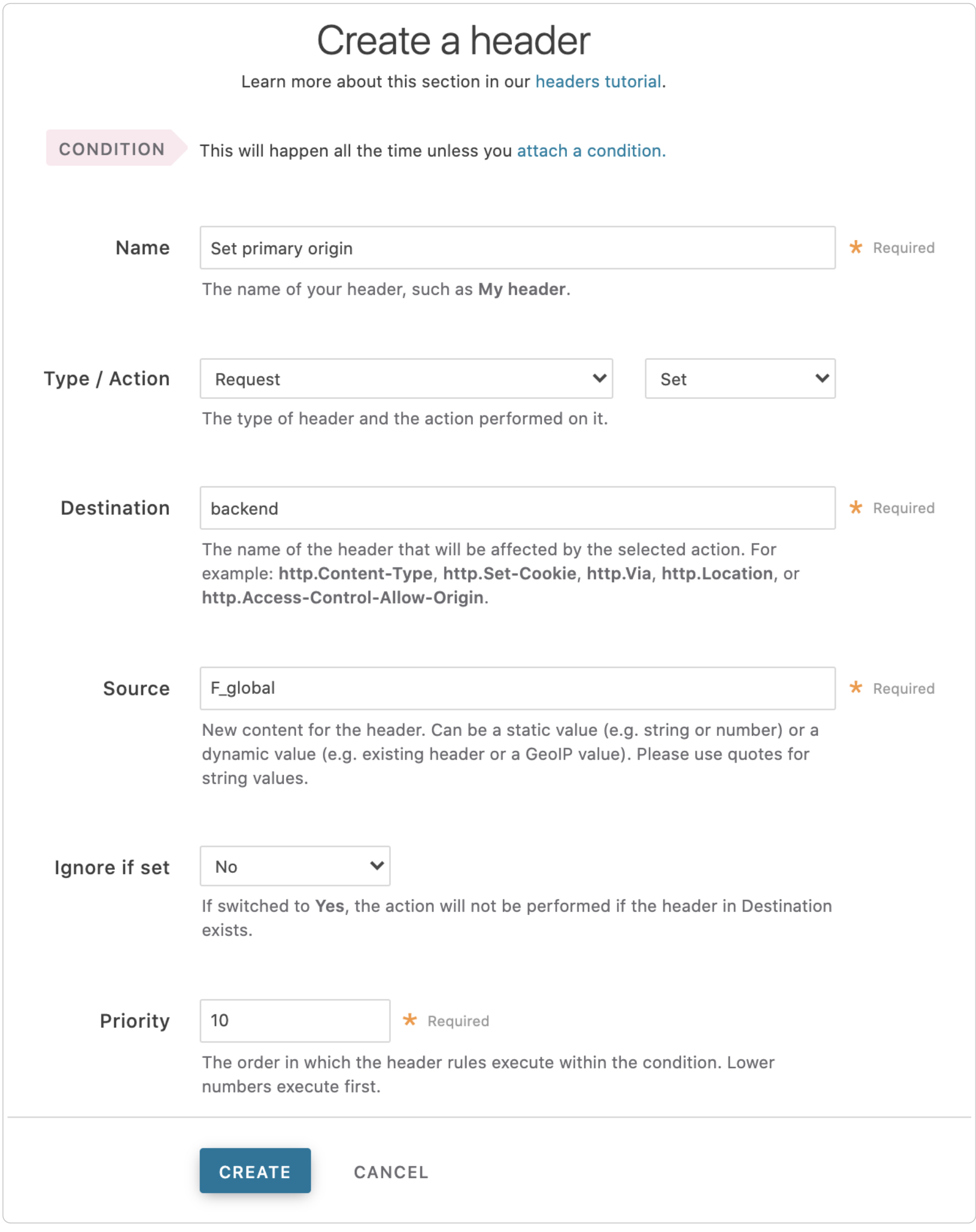
To configure a failover origin server you must turn off automatic load balancing for both the server that will act as your primary origin server and the server that will become your failover origin server.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the name of the origin server you want to configure. The Edit this host page appears.
6. From the **Auto load balance** menu, select **No**.
7. Click the **Update** button to apply the changes.

Configure the primary and failover origin servers

Once you've turned off automatic load balancing, create two new request headers, one each for your primary and failover servers.

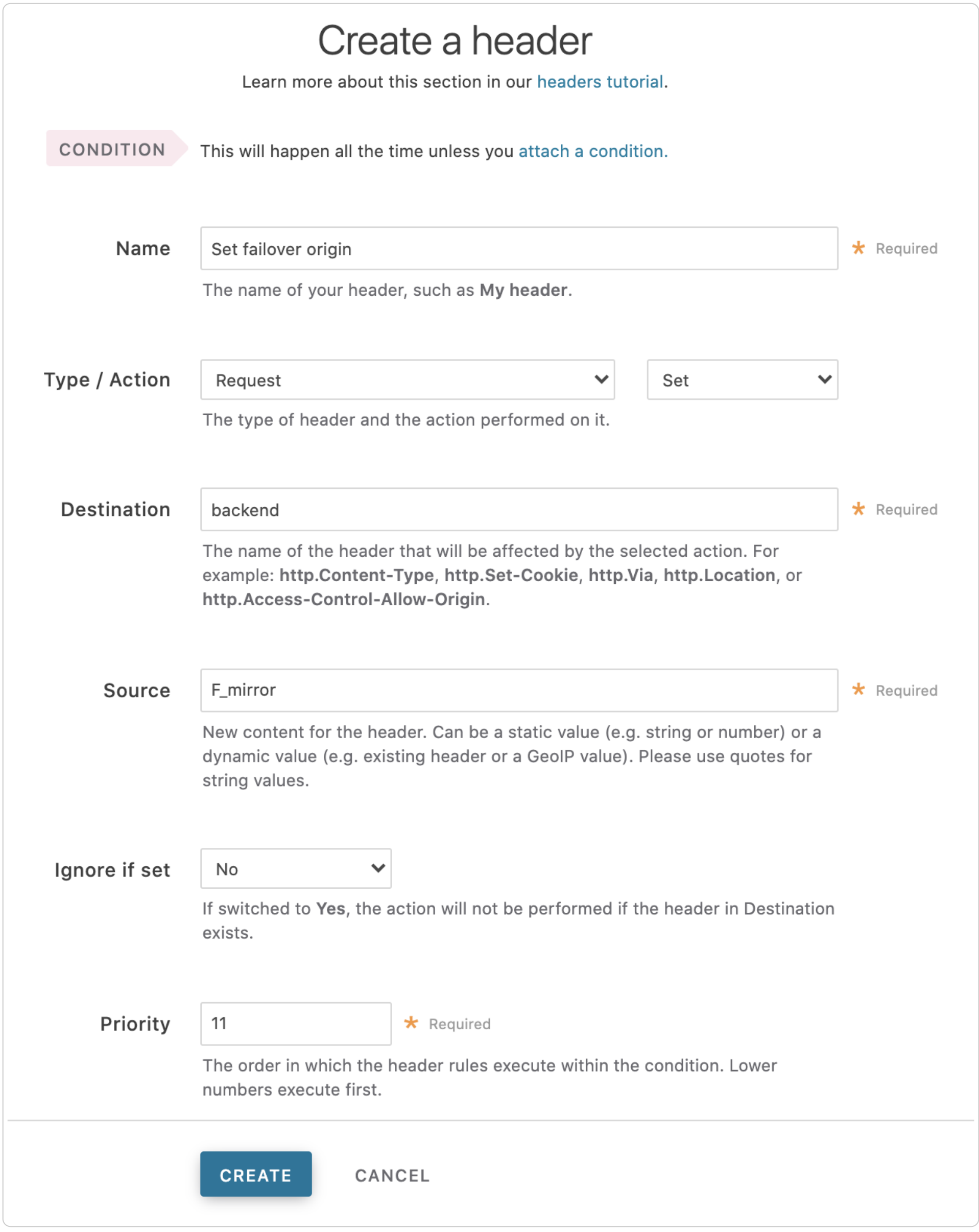
1. Click the **Content** link. The Content page appears.
2. Click the **Create header** button to create the first request header for your primary server. The Create a header window appears.



3. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a descriptive name for the header. This name is displayed in the Fastly web interface.

- From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
- In the **Destination** field, enter the name of the header that will be affected by the selected action.
- In the **Source** field, enter the name of your primary server. Preview the VCL to find the name of the server.
- Leave the **Ignore if set** and **Priority** controls at their default settings.

4. Click the **Create** button to create the first header. A new header appears on the Content page.
5. Click the **Create header** button to create a second request header for your failover server. The Create a header window appears.

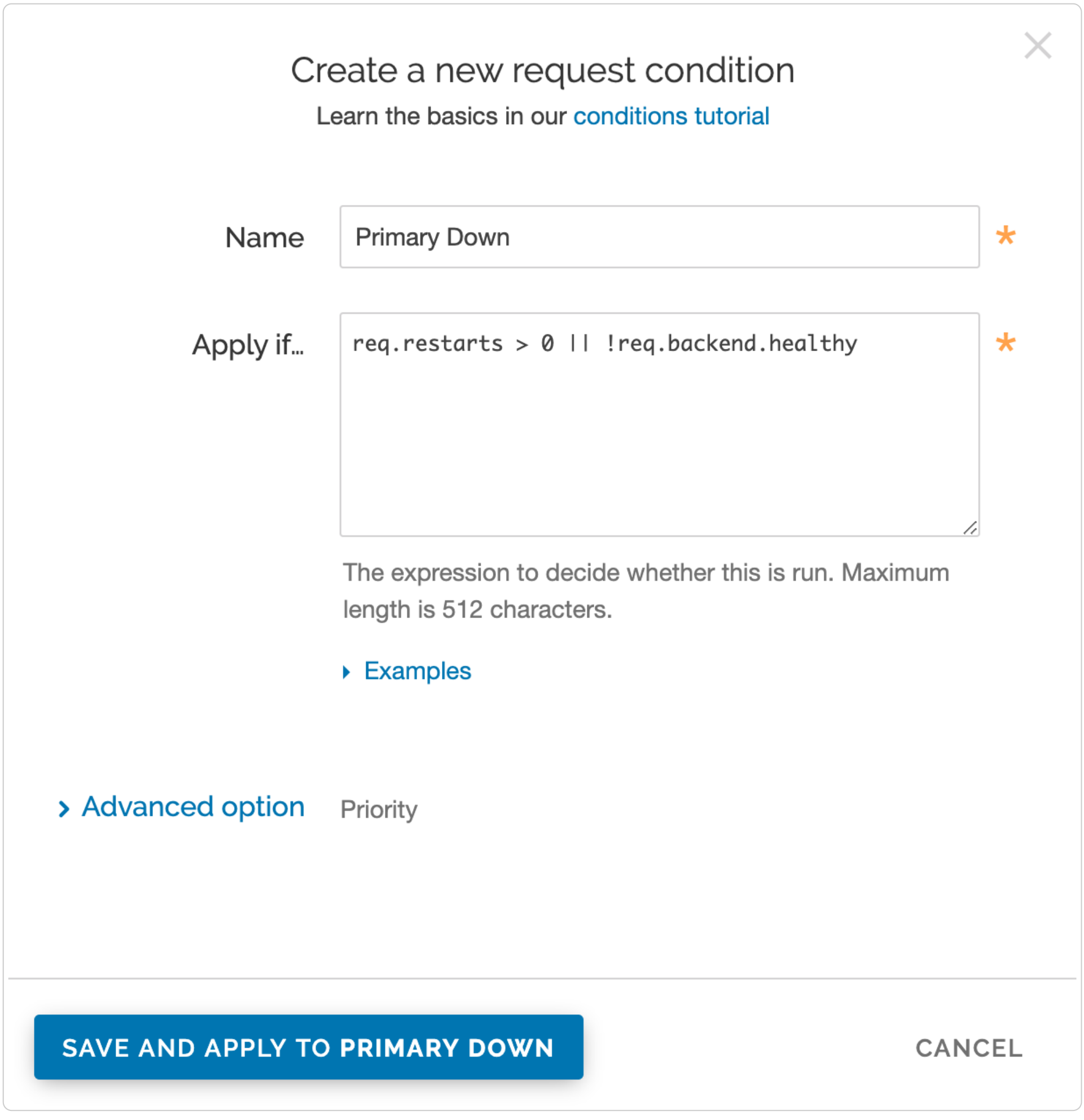


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a descriptive name for the header. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter the name of your failover server. Preview the VCL to find the name of the server.
   - In the **Source** field, enter where the new content for the header comes from.
   - Leave the **Ignore if set** control at the default setting.
   - In the **Priority** field, enter a number at least one higher than the priority you set on the primary server's request header. For example, if you left the first header's priority set to the default, `10`, you would set the second header's priority to `11` or higher.
7. Click the **Create** button to create the second header. A new header appears on the Content page.

Specify when to use the failover server

Once you've configured your primary and failover servers, create an associated header condition that specifies exactly when the failover server should be used.

1. On the **Content** page, click the **Attach a condition** link next to the new header you just created for the failover origin server.
The Create a new request condition window appears.



2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter a descriptive name for the new condition (for example, `Primary Down`).
   - In the **Apply if** field, enter the appropriate request condition that will be applied. For example, `req.restarts > 0 || !req.backend.healthy` would tell the system only to use the failover server if the number of restarts is more than 0 or the origin is unhealthy.
3. Click the **Advanced Options** link.
4. In the **Priority** field, enter `11`.
5. Click the **Save and apply to** button to create the new condition for the header.
6. Click the **Activate** button to deploy your configuration changes.

IPv6 support

Last updated: 2021-10-06

https://docs.fastly.com/en/guides/ipv6-support

Fastly has integrated IPv6 into its technology stack. By enabling IPv6, visitors on IPv6 connections can access your websites and applications. This can be done without any changes to your backend infrastructure.

Enabling IPv6

To enable traffic to be served over IPv6 and IPv4 addresses, follow the instructions below appropriate to your CNAME record.

If you're using one of our TLS products, you can find your CNAME records in the **HTTPS and network** tab, under **DNS details**. If the name ends in `map.fastly.net`, it is a customer-specific hostname. Otherwise, it is a Fastly-shared hostname.

> **NOTE**
>
> Fastly doesn't support IPv6 connections to origin servers.

Switching to dualstack for Fastly-shared hostnames

You can enable IPv6 dualstack (IPv4 and IPv6) functionality for your hostname by prefixing your CNAME record with `dualstack`. For example, if you have traffic on an IP address pool nicknamed `j.sni` (which supports a minimum TLS version of 1.2, a maximum TLS version of 1.3, does not support 0RTT, and offers HTTP/2), then you could use the following dualstack options:

- `dualstack.j.sni.global.fastly.net` (dualstack global map)
- `dualstack.j.sni.us-eu.fastly.net` (dualstack NA/EU)

If you are using `m.ssl` (which supports a minimum and maximum TLS version of 1.2 and HTTP/1.x only), then you could use the following dualstack options:

- `dualstack.m.ssl.global.fastly.net` (dualstack global map)
- `dualstack.m.ssl.us-eu.fastly.net` (dualstack NA/EU)

> **TIP**
>
> For more information on updating your CNAME record, see our instructions on updating your CNAME record with your DNS provider.

Enabling IPv6 on customer-specific hostnames

If you have a customer-specific hostname, contact support@fastly.com and we'll provide you with a parallel IPv6 map or enable dualstack on your current one. By default, maps will be HTTP/2 enabled and have a global billing region set. Be sure to specify any required changes when having a new map created.

Enabling Anycast IPv6 addresses for apex domains

If you use our Anycast IPv4 addresses for apex domains, contact Fastly Support and we'll provide you with the appropriate Anycast IPv6 addresses.

Geolocation features for IPv6

Fastly's geolocation features work with IPv6 addresses.

VCL variable

You can track whether a request came in as an IPv6 request with the `req.is_ipv6` VCL variable as well as by the IPv6 format itself when logging `%h`.

Testing IPv6

> **NOTE**
>
> If you're using our free shared domain to serve HTTPS traffic, check out our alternate instructions, for testing IPv6 instead.

Once you're up and running with IPv6, test IPv6 by entering a dig command in a terminal application to make sure your map returns AAAA records. For example, you can type something similar to this:

```
$ dig www.example.com AAAA +short
```

where `www.example.com` is the domain that you're testing.

Your output should appear similar to the following:

```
2606:2800:220:1:248:1893:25c8:1946
```

You can also use a tool like What's my DNS and choose the AAAA option to see how clients around the world are resolving to your CNAME record.

Performance implications

Enabling IPv6 shouldn't negatively impact performance. Most modern clients implement an approach called Happy Eyeballs to connect over either IPv4 or IPv6, whichever is faster. Happy Eyeballs chooses IPv6 over IPv4 when all else is equal.

Maintaining separate HTTP and HTTPS requests to origin servers

Last updated: 2018-10-03

https://docs.fastly.com/en/guides/maintaining-separate-http-and-https-requests-to-backend-servers

It is common to use the same origin web application to serve both HTTP and HTTPS requests and let the application determine which actions to take to secure communications depending on the incoming protocol. Fastly allows users to set this up to preserve this functionality within their servers. To set Fastly up to send HTTP requests to the non-secure service and HTTPS requests to the secure service, configure two origins, one each for the secure and non-secure ports, then set up the conditions under which requests will be sent there.

Create multiple origins

Begin by configuring the same origin address with a different port as a separate origin server. Follow the instructions for working with hosts. You'll add specific details about the non-secure server (port `80`) when you fill out the **Create a host** fields:

- In the **Name** field, enter a name for the non-secure server (for example, `Server Name (plain)`).
- In the **Address** field, enter the address of the non-secure server (for example, `server.example.com`).
- In the **Transport Layer Security (TLS)** section, set **Enable TLS?** to **No**.

Follow the instructions for working with hosts to create another origin server, this time for your secure server. You'll add specific details about the secure server (port `443`) when you fill out the **Create a host** fields:

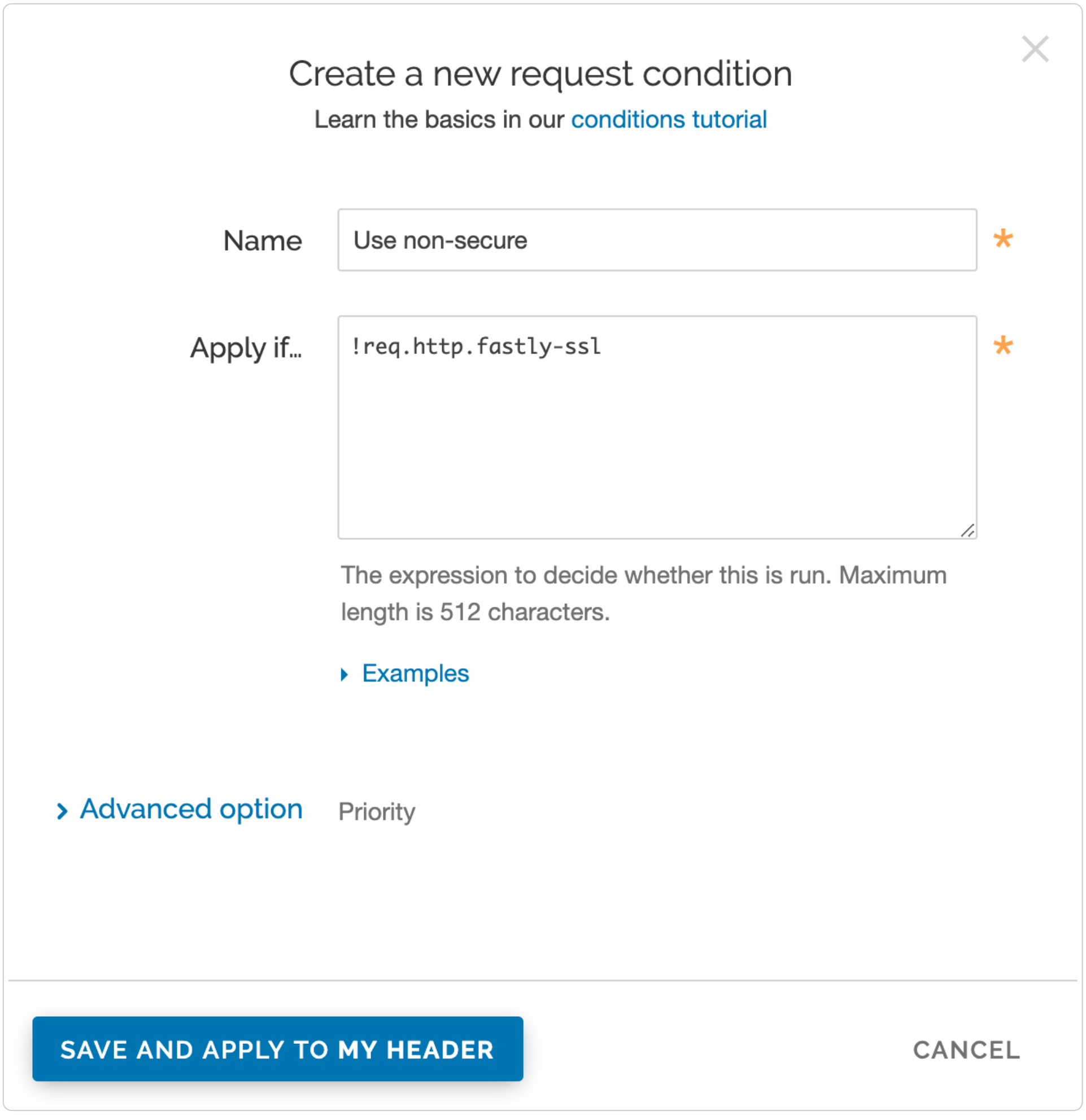
- In the **Name** field, enter a name for the non-secure server (for example, `Server Name (secure)`).
- In the **Address** field, enter the address of the non-secure server (for example, `server.example.com`).
- In the **Transport Layer Security (TLS)** section, leave the **Enable TLS?** default set to **Yes**.

Conditionally send traffic to origins

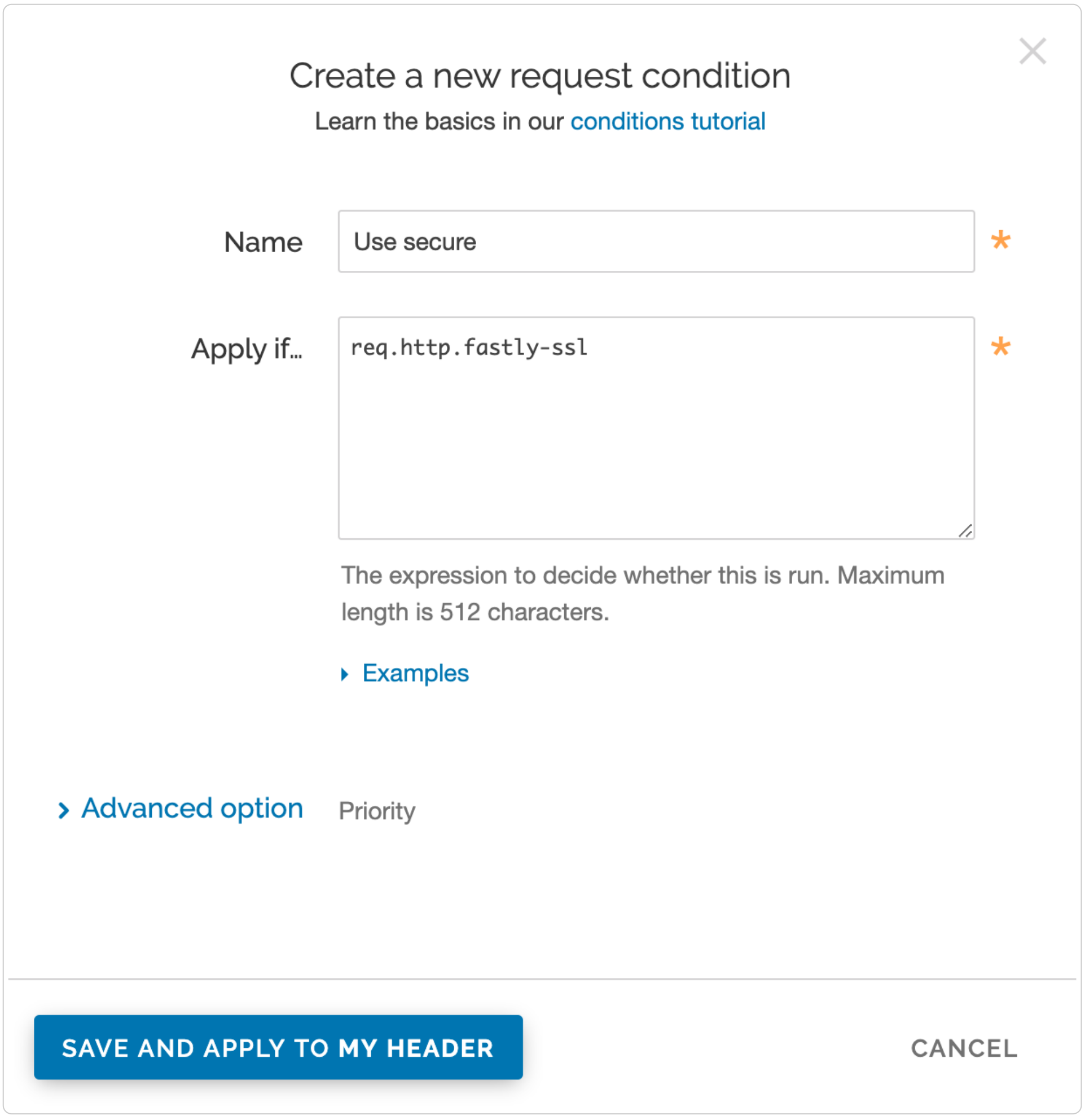
To conditionally determine which server receives secure and non-secure requests, Fastly relies on the presence or absence of a specific header when the backend is selected. When an incoming connection is received over TLS, Fastly sets the `req.http.fastly-ssl` header to determine which server to use.

Set a condition for this header on each origin by following the steps below.

1. On the **Origins** page, click the **Attach a condition** link next to the name of the non-secure server. The Create a new request condition window appears.



2. Fill out the **Create a new request** fields as follows:
   - In the **Name** field, enter the name of the condition specifying use of the non-secure server (for example, `Use non-secure`).
   - In the **Apply if** field, enter `!req.http.fastly-ssl`.
   - Leave the priority set to its default value.
3. Click the **Save and apply to** button to create the new condition.
4. On the **Origins** page, click the **Attach a condition** link next to the name of the secure server. The Create a new request condition window appears.



5. Fill out the **Create a new request condition** window as follows:
   - In the **Name** field, enter the name of the condition specifying use of the secure server (for example, `Use secure`).
   - In the **Apply if** field, enter `req.http.fastly-ssl`.
   - Leave the priority set to its default value.
6. Click the **Save and apply to** button to create the new condition.
7. Click the **Activate** button to deploy your configuration changes.

Routing assets to different origins

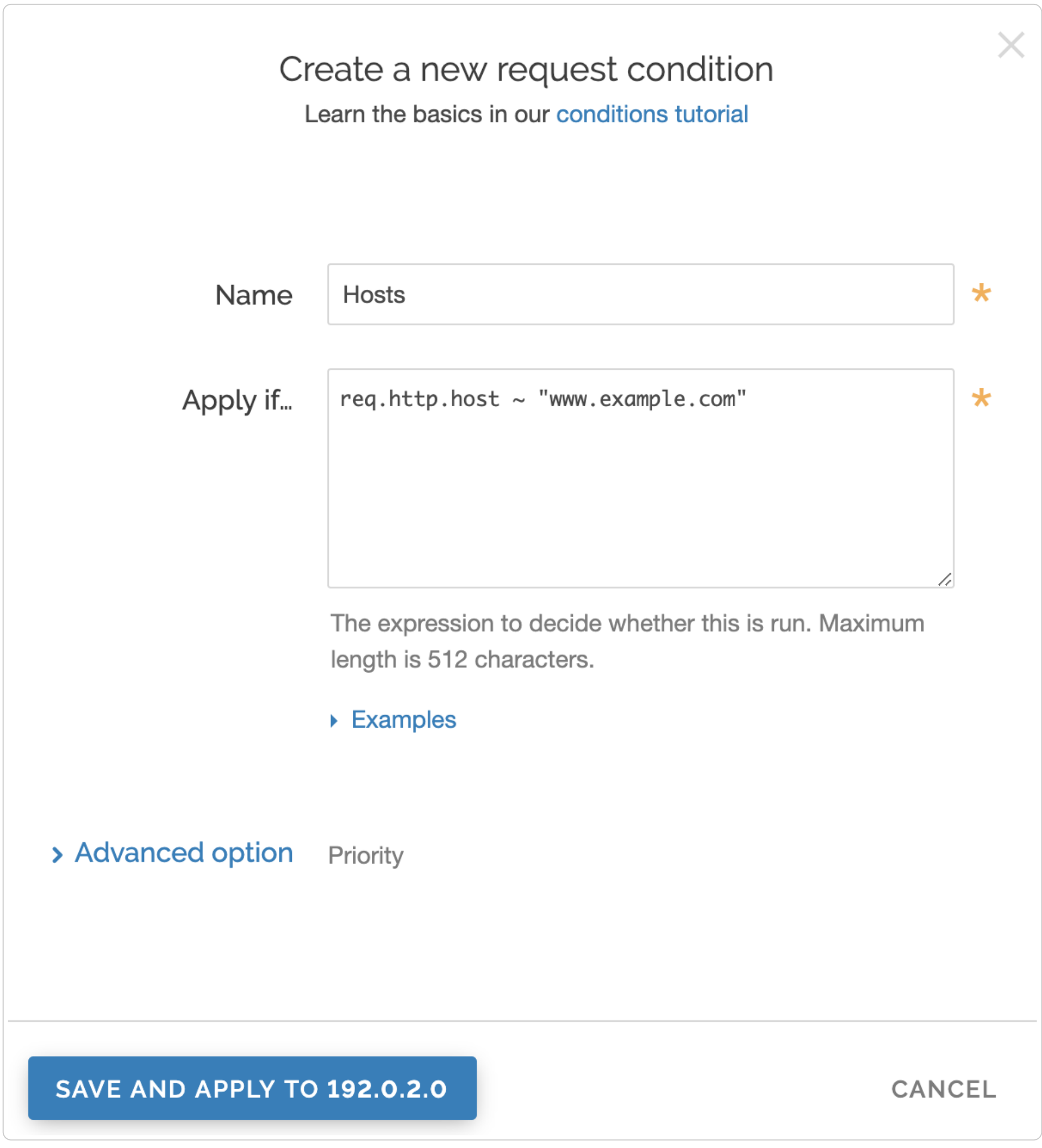
Last updated: 2019-05-23

https://docs.fastly.com/en/guides/routing-assets-to-different-origins

Some customers have assets stored on multiple origin servers and want to route various requests to specific, different servers based on criteria they supply (e.g., asset type, file directory, Host header). Fastly offers customers the ability to set conditions on their origins, which simply adds an if statement block to your VCL.

Basic setup: Create conditions for each origin

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the **Attach a condition** link to the right of the name of an origin server. The Create a new request condition window appears.



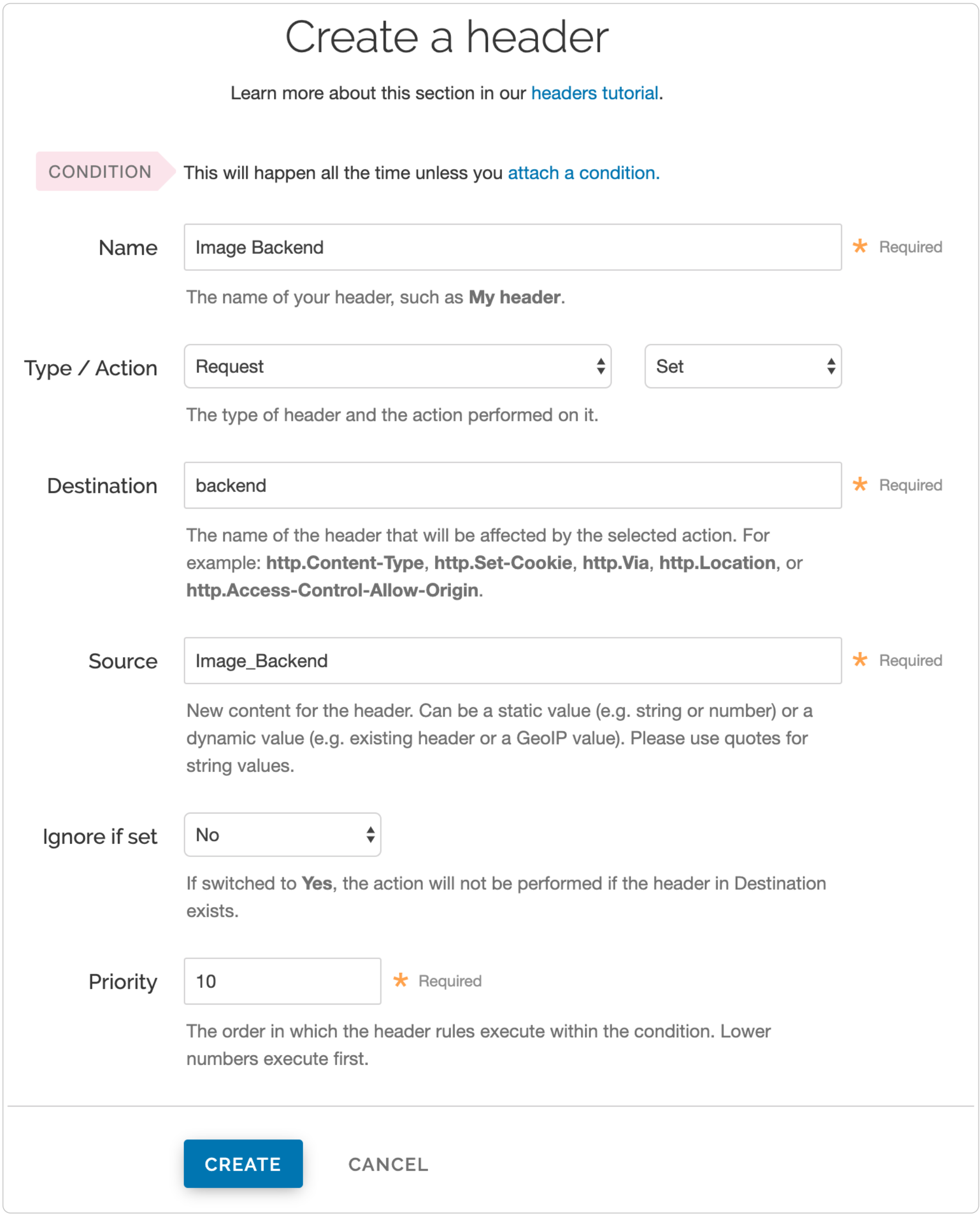
6. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter a human-readable name for the condition.
   - In the **Apply if** field, enter the conditions that you want to apply to your origin server. For example, for hosts, you could enter `req.http.host ~ "www.example.com"`. Or, for content-type / URL, you could enter `req.url ~ ".(jpg|png|gif)($|\?)"`.

7. Click the **Save and apply to** button. The new condition appears on the Origins page.
8. Click the **Activate** button to deploy your configuration changes.

Backup setup: Create a header

What if you have a condition already assigned to your origin? Although you can group request conditions on the origin with an 'and' or 'or' clause, there can only ever be one condition rule attached to that origin. If you want to separate your request conditions instead of grouping them, you can use header rules to route assets to different origins instead.

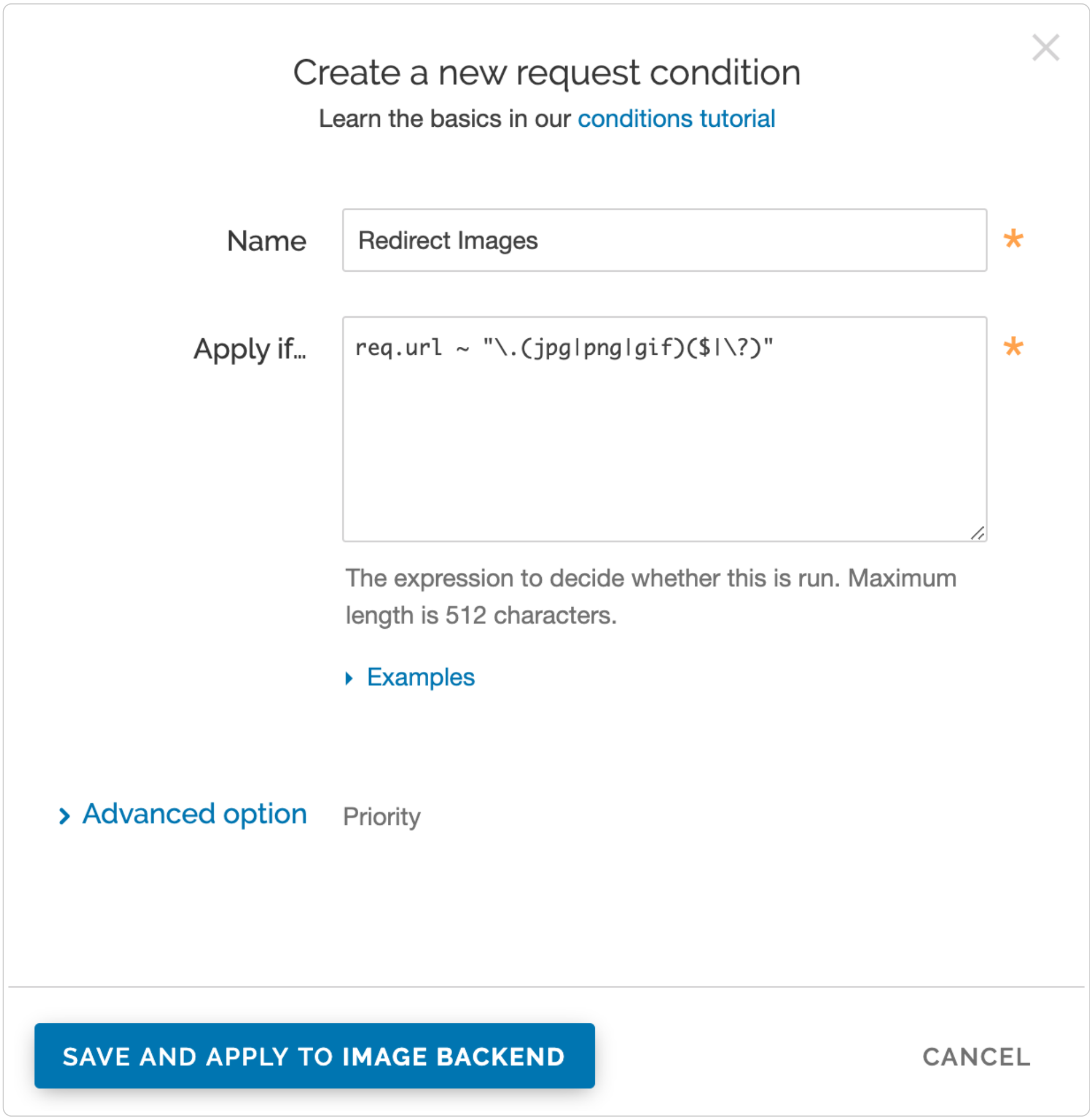
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Image Backend` (or any meaningful, preferred name).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `backend`.
   - In the **Source** field, enter `Image_Backend`. (This should match the name of your global origin server. You can see the exact name if you look at your VCL. Click on the **VCL** button at the top of the page.)
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.

7. Click the **Create** button. The new header appears on the Content page.

8. On the **Content** page, click the **Attach a condition** link next to the header you just created. The Create a new request condition window appears.



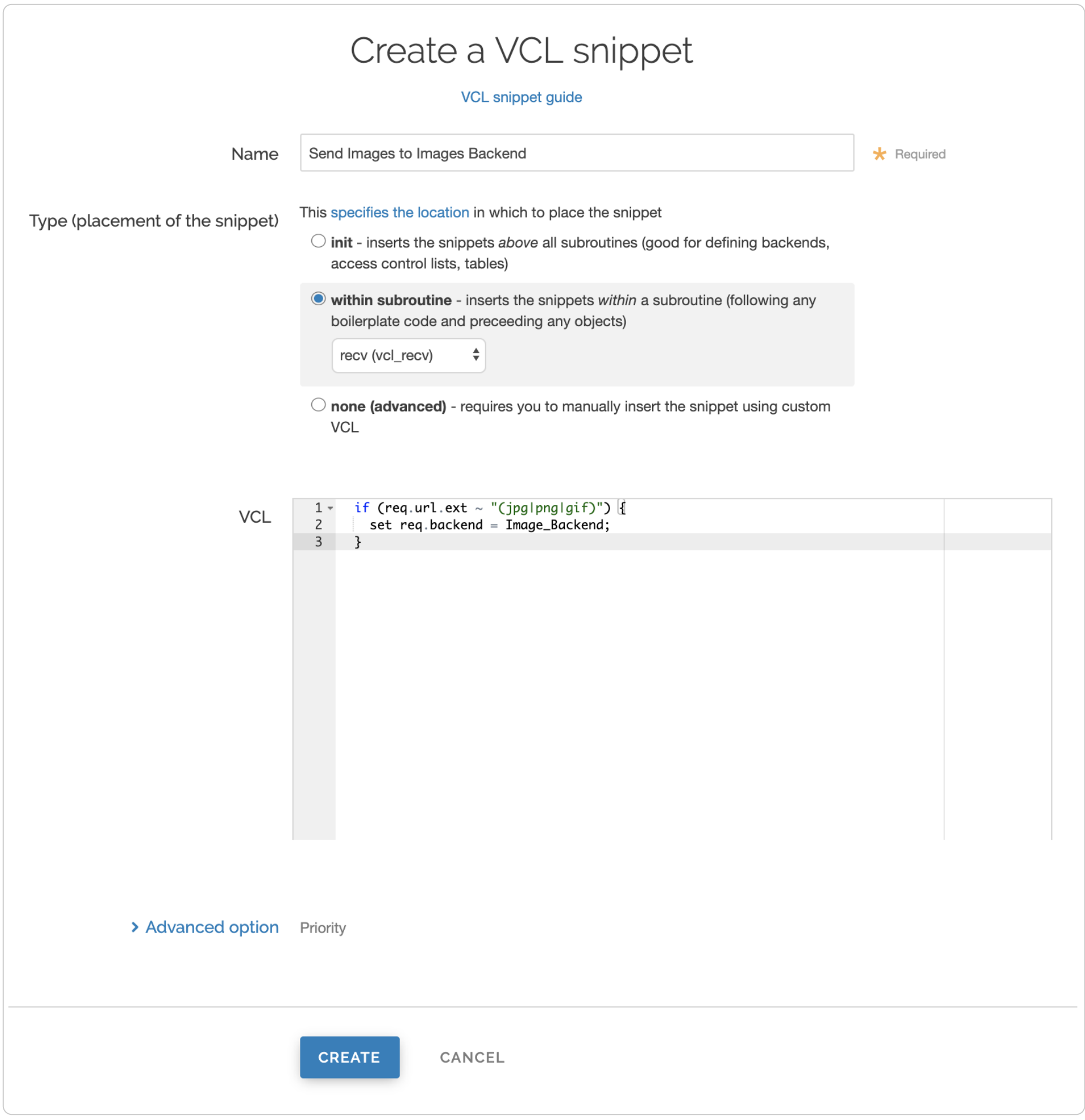
9. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter `Redirect Images` (or any meaningful, preferred name).
   - In the **Apply if** field, enter `req.url ~ "\.(jpg|png|gif)($|\?)"`.
10. Click the **Save and apply to** button. The condition appears on the Content page.
11. Click the **Activate** button to deploy your configuration changes.

> **TIP**
>
> Our about guide provides more information about working with conditions.

Use VCL Snippets to specify an origin

You can also use VCL Snippets to specify an origin. Once you've created your origin, you can conditionally route traffic to it.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click **Create Snippet**. The Create a VCL snippet page appears.



6. In the **Name** field, enter an appropriate name (e.g., `Send Images to Images Backend`).
7. From the **Type** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **recv (vcl_recv)**.
9. In the **VCL** field, add the following condition:

```
if (req.url.ext ~ "(jpg|png|gif)") {
  set req.backend = Image_Backend;
}
```

10. Click **Create** to create the snippet.

11. Click the **Activate** button to deploy your configuration changes.

Setting up redundant origin servers

Last updated: 2018-10-02

https://docs.fastly.com/en/guides/setting-up-redundant-origin-servers

Sometimes you want to set up two different origin servers, one as a primary and one as a backup in case the primary becomes unavailable. You can do this via the web interface or using custom VCL.

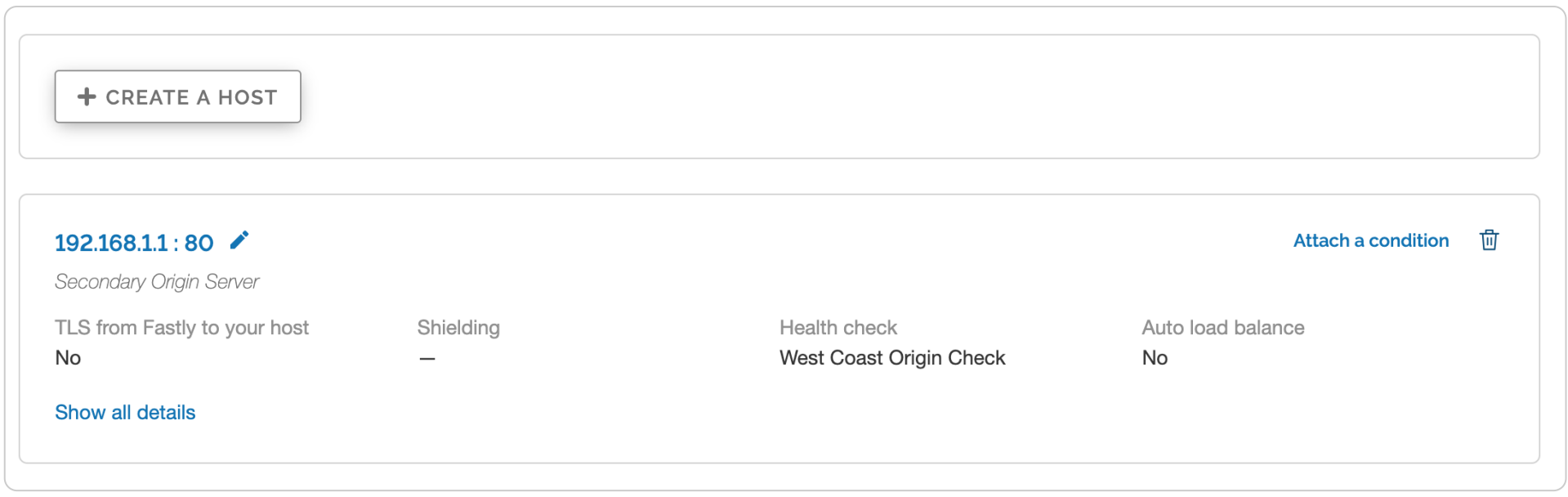
NOTE

Each Fastly service can be configured with up to five origin servers. Contact sales@fastly.com to enable more than five origin servers per service in your account.

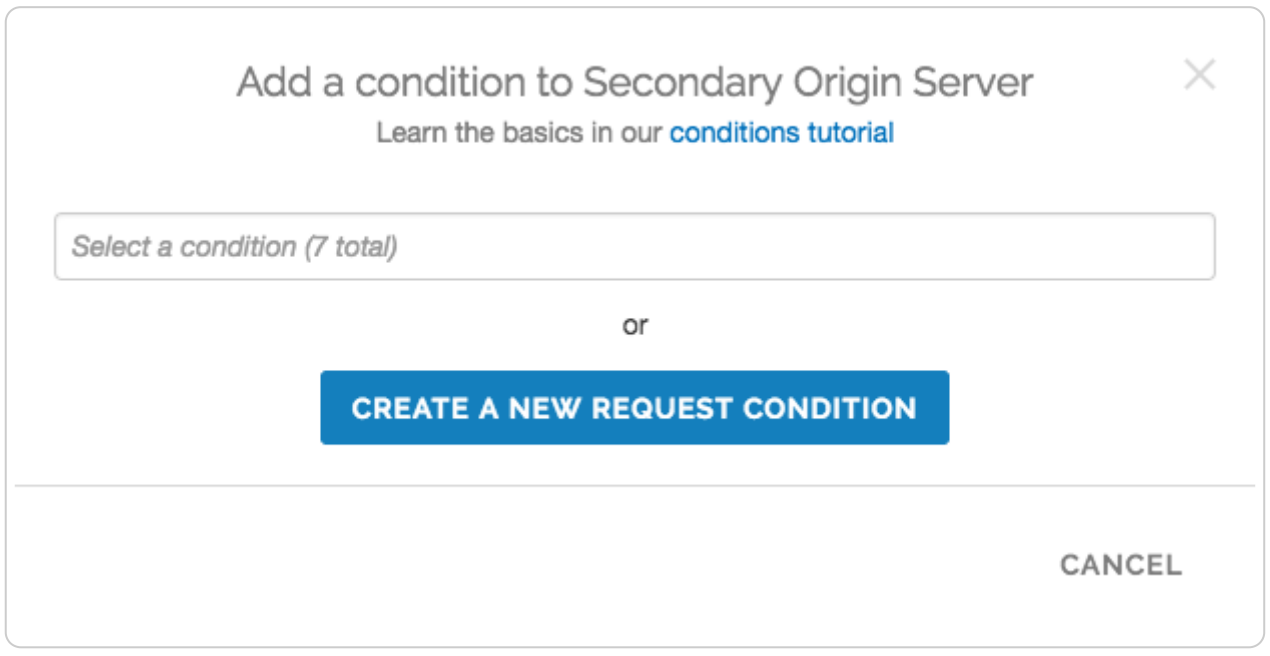
Using the web interface

Set up redundant origins via the web interface using these steps.

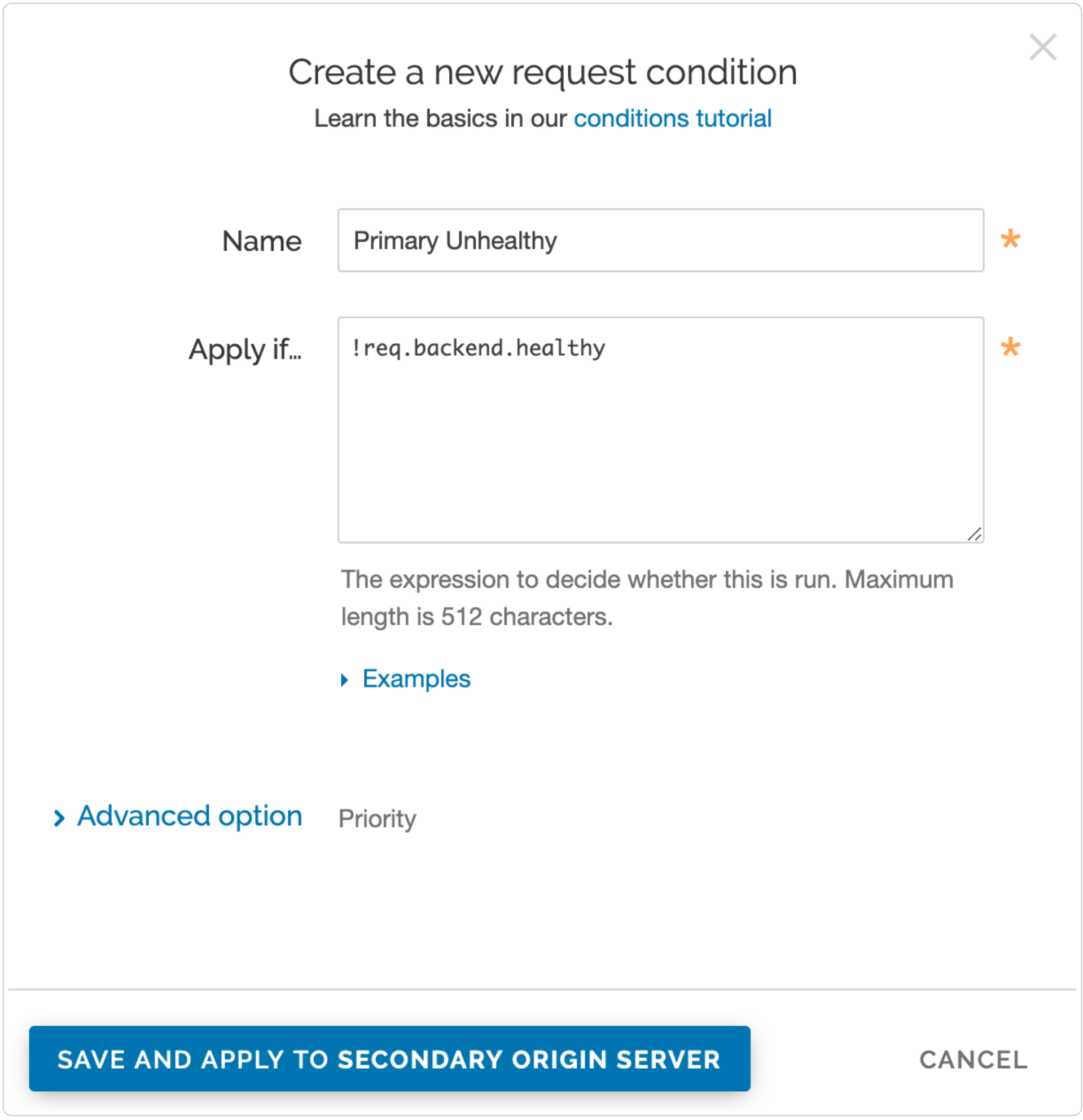
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. In the **Health Checks** area, define a health check and assign it to the primary origin server.
6. In the **Hosts** area, find your secondary origin server and click the **Attach a condition** link.



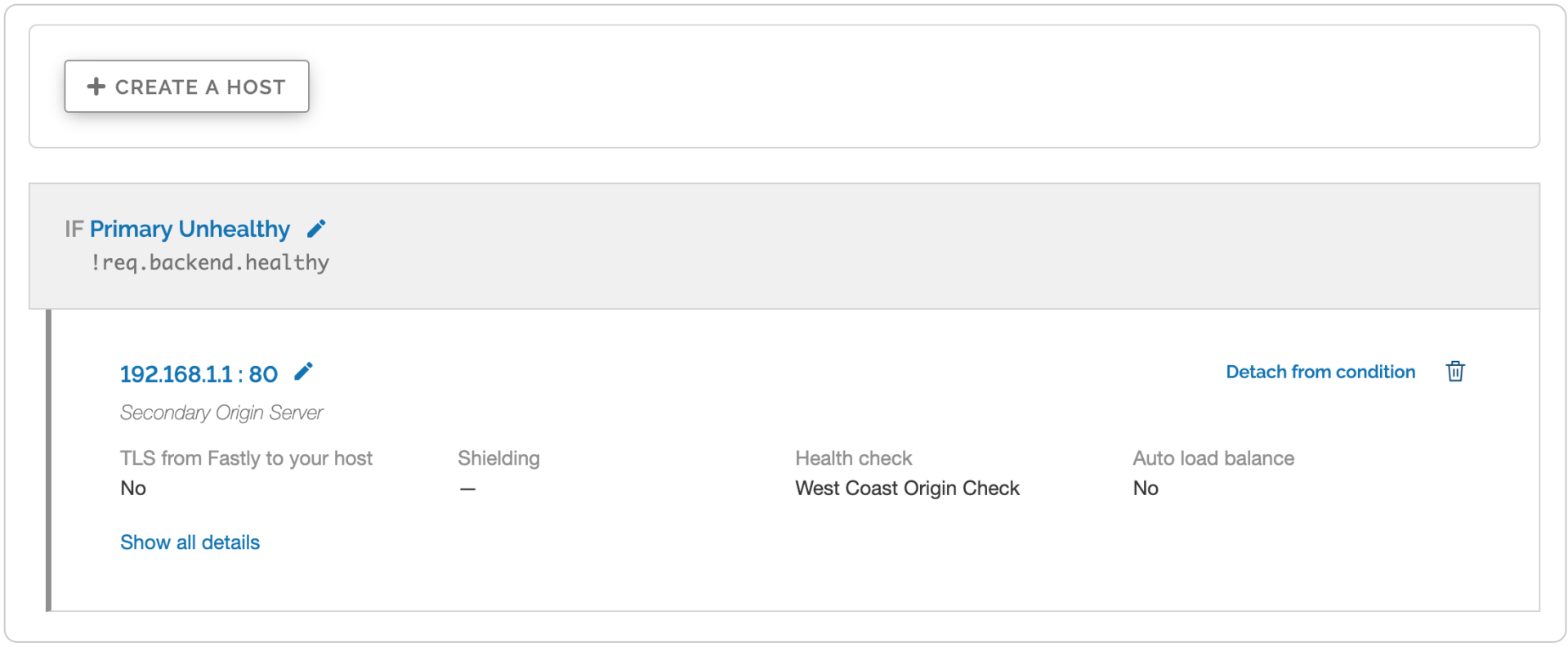
The Add a condition window appears.



7. Click **Create a new request condition**. The Create a new request condition window appears.



8. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter the name of your request condition (for example, `Primary Unhealthy`).
   - In the **Apply if** field, enter `!req.backend.healthy`.
9. Click the **Save and apply to** button. The Hosts area now displays the condition that must be met (Primary Unhealthy) in order for your secondary origin server to begin being used.



Once you've added the condition to your secondary origin server, the VCL generated by Fastly will reflect the new condition.

10. Preview the VCL, and confirm the following snippets appear in `vcl_recv`:

```
# default conditions
set req.backend = F_primary;
```

```
# Request Condition: primary unhealthy Prio: 10
if (!req.backend.healthy) {
  set req.backend = F_secondary;
}
#end condition
```

Using custom VCL

Set up redundant origins with custom VCL using these steps.

1. In the Fastly web interface, define a health check and assign it to the primary origin server.
2. Copy the boilerplate VCL from our guide on mixing Fastly VCL with custom VCL, and paste it into a new file.
3. Replace the `vcl_recv` sub with:

```
sub vcl_recv {
#FASTLY recv
  set req.backend = F_<primary_origin>;
  if (!req.backend.healthy) {
    set req.backend = F_<secondary_origin>;
  }
  if (req.method != "HEAD" && req.method != "GET" && req.method != "FASTLYPURGE") {
    return(pass);
  }
  return(lookup);
}
```

   To find the exact backend names, view the generated VCL.

4. Upload your VCL file.

Specifying an override host

Last updated: 2022-03-01

https://docs.fastly.com/en/guides/specifying-an-override-host

If you want to rewrite the Host header being sent to your origin regardless of the Host used in the initial request, specify an override host. Use this if you have multiple domains tied to a service and want them all served by the same origin, if the domain your origin is expecting is different than one specified in your Fastly service, or if the Host header being sent to your origin is different from the Host used in the initial request. You most likely won't need to use this feature.

You can override the Host header being sent to your origin by specifying the domain name of your override Host on the **Settings** page for a specific service or by specifying a host on the **Origins** page for a specific Host.

Here are some examples of when to use an override host:

- When using backends such as Amazon S3, Google Cloud Storage, or Heroku, you want to ensure you use the proper Host header so these providers know how to route requests directly to your content. Each provider uses the Host header to associate requests with your account's storage location. For example, if you set up your origin using Amazon S3, you send the name of your S3 bucket as your Host header. Amazon is set up so that it only accepts Host headers that have the same name as the bucket hosting your content. A request to `your-domain.com` must be re-written to `<BUCKET>.s3.<REGION>.amazonaws.com`, or else the request is denied.
- You have a service that contains three sites: `www.abc.com`, `www.myexample.com`, and `www.mysite.com` and you have one origin. You can have the same origin respond to each domain by overriding the Host header to one accepted by your origin, for example, `origin.example.com`. The result will be that a request to `www.abc.com`, `www.myexample.com`, or `www.mysite.com` returns content from `origin.example.com`.

> **NOTE**
>
> If you've initially set an override Host globally and then switch the configuration to set an override Host per origin, this will temporarily increase cache MISS and origin traffic because the Host header that Fastly was using for the cache lookup was changed from the client's original Host header. Once a new object is cached with the new host, cache HIT will be served.

Overriding a host at the origin level

You can add an override Host per origin if you're using an origin that requires a specific hostname to be passed to it. Once you've added a host, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. In the **Hosts** area, click the pencil icon next to the Host you want to edit. The Edit this Host page appears.
6. In the **Override host** field, enter the hostname of your override Host header based on the origin you're using. The value in this field will take precedence over anything you've set using the global override Host configuration. For example:
   - If you're using Amazon S3 as your origin, enter `<BUCKET>.s3.<REGION>.amazonaws.com`.
   - If you're using Google Cloud Storage as your origin, enter `<BUCKET>.storage.googleapis.com`.

   > **TIP**
   >
   > To see other examples of Host headers for third-party services, refer to our developer documentation on overriding the Host header.
7. Click the **Update** button. The new override Host appears under the **Show all details** field of the **Override host** section and a code block is added to the origin definition in your VCL that will look similar to the following:

```
   Backend F_Host_1 {
      .host = "..."; # IP or hostname
      .host_header = "example.com";
      .always_use_host_header = true;
       ...
    }
```

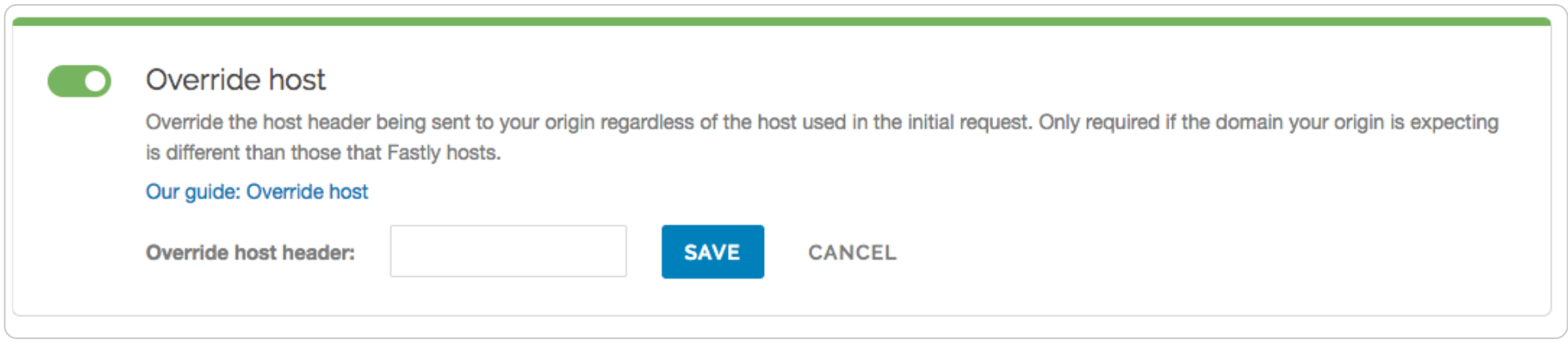
8. Click the **Activate** button to deploy your configuration changes.

Overriding a host globally

> **NOTE**
>
> Use the global override if you only have one backend. If you do use the global override, be sure to read all the caveats below.

You can globally override a Host if your service has multiple domains to serve but they are all same assets (e.g., `assets1.example.com` and `assets2.example.com`) and you want to normalize them (e.g., `assets.example.com`). To globally override a host, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **Override host** switch. The Override host header field appears.



6. In the **Override host header** field, enter the hostname of your override Host based on the origin you are using:
   - If you are using Amazon S3 as your origin, enter `<yourbucket>.s3.amazonaws.com`.
   - If you are using Google Cloud Storage as your origin, enter `<your bucket name>.storage.googleapis.com`.
7. Click the **Save** button. The new override host header appears in the Override host section.
8. Click the **Activate** button to deploy your configuration changes.

Caveats about using the global override host

There are situations when you may not want to use an override host:

- **Forcing TLS and enabling HSTS.** You may experience problems if you enable this setting along with the force TLS and enable HSTS setting. Instead of enabling this setting, create a new request setting and specify the override host in the advanced options.
- **Using multiple origins.** When you specify a Host override, you're specifying what hostname is actually sent to your origin. If you have a service with two different origins and each origin requires a different hostname, specifying a Host override for all requests results in one origin not returning valid responses. If you specify a default hostname that matches only one of the origins, then no content is returned from the other origin requests.

  > **NOTE**
  >
  > If you want to serve content from multiple backends, you should conditionally route to them. Refer to Routing assets to different origins for more information.

- **Shielding is enabled.** If you enable a Host override along with shielding and the specified override host doesn't match to a domain within the service, the shield won't route the request properly and an error of 500 is expected. Refer to Shielding for more information.

  > **NOTE**

To ensure consistent behavior of Fastly customer services and origins, we normalize the host header's value to all lowercase in the `vcl_hash` function. This means that no matter how your site's domain name is capitalized in the request, the hash function will behave predictably. This does not apply to any other parts of the URL, which remain case-sensitive.

Using Fastly with apex domains

Last updated: 2021-11-05

https://docs.fastly.com/en/guides/using-fastly-with-apex-domains

Some customers use only their second-level or apex domain (e.g., `example.com` rather than `www.example.com`) as their canonical domain. Due to limitations in the DNS specification, we don't recommend placing a CNAME record at the apex domain or using the CNAME Flattening features offered by some DNS providers (e.g., ALIAS or ANAME).

Instead, we offer anycast IP addresses for content that must be hosted on a second-level or apex domain. Our anycast options allow you to add A (IPv4) or AAAA (IPv6) records that point your apex domain at Fastly, prioritizing either performance or cost, depending on the option you choose.

IMPORTANT

Because anycast addressing methods don't offer Fastly as much flexibility in routing requests, our anycast options may not be as performant as our CNAME-based system. We recommend using our CNAME-based system for as much content as possible, particularly for large resources or streaming video.

TIP

Our guide on generating redirects at the edge discusses how you can redirect all traffic for apex domains to WWW subdomains.

Anycast options

Fastly offers the following anycast options to all paid customers.

Global anycast option

Fastly offers anycast IP addresses that allow you to use our entire global network to route requests to the nearest Fastly POP (from a network perspective), without regard to the billing region in which that POP resides.

Choose this option to prioritize traffic routing performance and to avoid restricting traffic to specific POPs. We'll provide you with a list of anycast IP addresses, which you then enter into your DNS records.

Billing Zone anycast options

IMPORTANT

Billing Zone anycast options are part of a limited availability release. For more information, see our product and feature lifecycle descriptions.

Fastly offers anycast IP addresses that allow you to prioritize where requests get routed based on the cost of its travel through groups of specific billing regions called *zones*.

Choose one of these options to prioritize cost savings over routing performance:

- **Maximum Billing Zone (MBZ) 100:** This option includes only the Fastly POPs located in the North America and Europe billing regions.
- **Maximum Billing Zone (MBZ) 200:** This option includes all Fastly POPs in MBZ 100 as well as those in the billing regions of Asia, Australia and New Zealand, and South America. It does not include POPs in the South Korea, India, and Africa billing regions.

Availability versus routing accuracy

> **IMPORTANT**
>
> Fastly prioritizes service availability over routing accuracy. Fastly will not offer invoice credits for traffic delivered from Fastly POPs outside of intended billing zones when that traffic is intentionally diverted to preserve the integrity or performance of Fastly services, for scheduled maintenance, or when the traffic routing changes are outside of Fastly's control.

Because most anycast routing on the public internet is outside of our direct control, we cannot guarantee traffic routed to us will never arrive at POPs in higher billing zones. When routing changes on the public internet shift your traffic to higher priced Fastly POPs, however, we will make our best effort to investigate the cause and work with all parties involved to correct any problems discovered.

In addition to factors outside of our control, Fastly performs traffic engineering on a regular basis to ensure the availability of all Fastly services during incidents and scheduled maintenance to our network. When necessary, we may temporarily choose to route traffic to POPs outside of the chosen Billing Zones.

If you observe traffic being served from Fastly POPs outside of the intended Maximum Billing Zone, then before opening a support ticket:

- Check the Fastly Service Status page. Verify that there are no events that would explain the temporary rerouting of your traffic.
- Check the DNS records for your impacted domain names. Only the Fastly provided Billing Zone anycast IP addresses should be present in the DNS records of your impacted domain names.

If neither of the above issues explain your observed Fastly POP routing issue, open a support ticket within 30 days of the Fastly invoice date that reflects the billed traffic in question by emailing support@fastly.com.

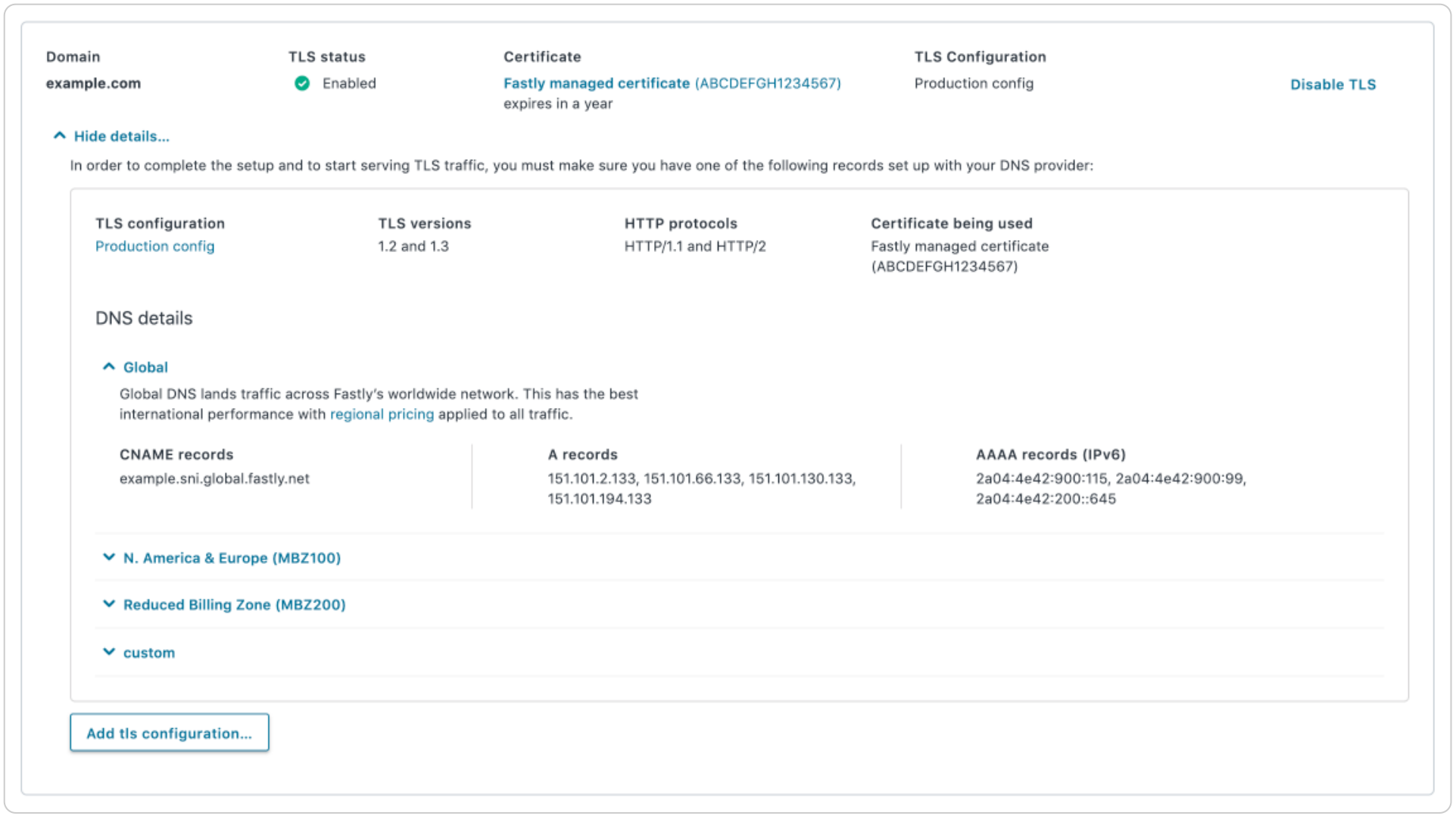
Finding anycast IP addresses

When you choose a Billing Zone anycast option, we'll provide you with the list of anycast IP addresses for you to enter into your DNS records. Fastly will use these addresses to serve traffic only from the POPs included in the zones listed above, even if those POPs are unlikely to give the best performance for any given request.

When you have TLS configured

If TLS is configured for your Fastly service, you can obtain your global anycast IP addresses in the Fastly web interface by following these steps:

1. Log in to the Fastly web interface and click the **Secure** link. The TLS domains page appears, displaying any domains for which you have TLS enabled.
2. Click the **More details** link for the domain you would like to route to Fastly via the global anycast option. The domain's details page appears.



The A Records section contains the global anycast IP addresses for you to enter into your DNS records of this domain.

When you don't have TLS configured

If you don't have TLS configured for the domain you would like to use with the global anycast option or would like to use one of the new Billing Zone anycast options, you can request your anycast IP addresses by contacting support@fastly.com. We'll provide you with the anycast IP addresses appropriate for the option you choose so you can enter those addresses into your DNS records. Be sure to enter all of them to ensure maximum availability and reliability. We don't charge extra for these options, however, you must be using one of Fastly's paid plans (with or without a contract) to take advantage of them.

> **TIP**
>
> Fastly's billing regions can be found on our pricing page. We announce changes to these regions via our network status page.

Apex domain problems and their workarounds

The DNS instructions in RFC1034 (section 3.6.2) state that, if a CNAME record is present at a node, no other data should be present. This ensures the data for a canonical name and its aliases cannot be different. Because an apex domain requires NS records and usually other records like MX to make it work, setting a CNAME at the apex would break the "no other data should be present" rule.

In general, the problem with apex domains happens when they fail to redirect to their www equivalents (`example.com` points nowhere instead of pointing to `www.example.com`). Two workaround options exist:

- only use Fastly for API or AJAX calls, images, and other static assets (e.g., serve `example.com` yourself and CNAME to Fastly for assets at `assets.example.com`).
- redirect from the apex domain to the version proxied by Fastly (e.g., redirect any requests for `example.com` to `www.example.com`).

Neither workaround, however, is ideal.

> **IMPORTANT**
>
> To use TLS with an apex domain, explicitly add it to a certificate using one of our TLS products. For wildcard entries on our shared certificates, add an apex domain as a separate SAN entry.

Using Fastly's global POP network

Last updated: 2020-05-19

https://docs.fastly.com/en/guides/using-fastlys-global-pop-network

Fastly operates a domain name system (DNS) service specifically written to optimize getting your traffic to Fastly. This optimization automatically routes your traffic to the nearest Fastly POP (in terms of network proximity) on our global network and reroutes around internet outages and other disturbances.

Directing your traffic through Fastly's global network

To take advantage of Fastly's global network, use a fully qualified domain name (FQDN) when you create domains in the Fastly web interface and make sure you've properly configured your CNAME DNS records.

You can use an apex domain (e.g., example.com instead of www.example.com) as your canonical domain and still take advantage of Fastly's global network by pointing your apex DNS record at IPs on Fastly's anycast network.

Limiting traffic to a subset of POPs

Fastly allows you to configure traffic routing choices that best suit your specific needs by prioritizing a subset of POPS through which your traffic travels. For example, you can specify that Fastly prioritizes the use of only North American and European Union (EU) POPs. For more information, read about our Billing zone anycast options.

§

These articles describe configuration settings and changes you can make to your headers when setting up Fastly services.

https://docs.fastly.com/en/guides/configuration#_headers

Adding or modifying headers on HTTP requests and responses

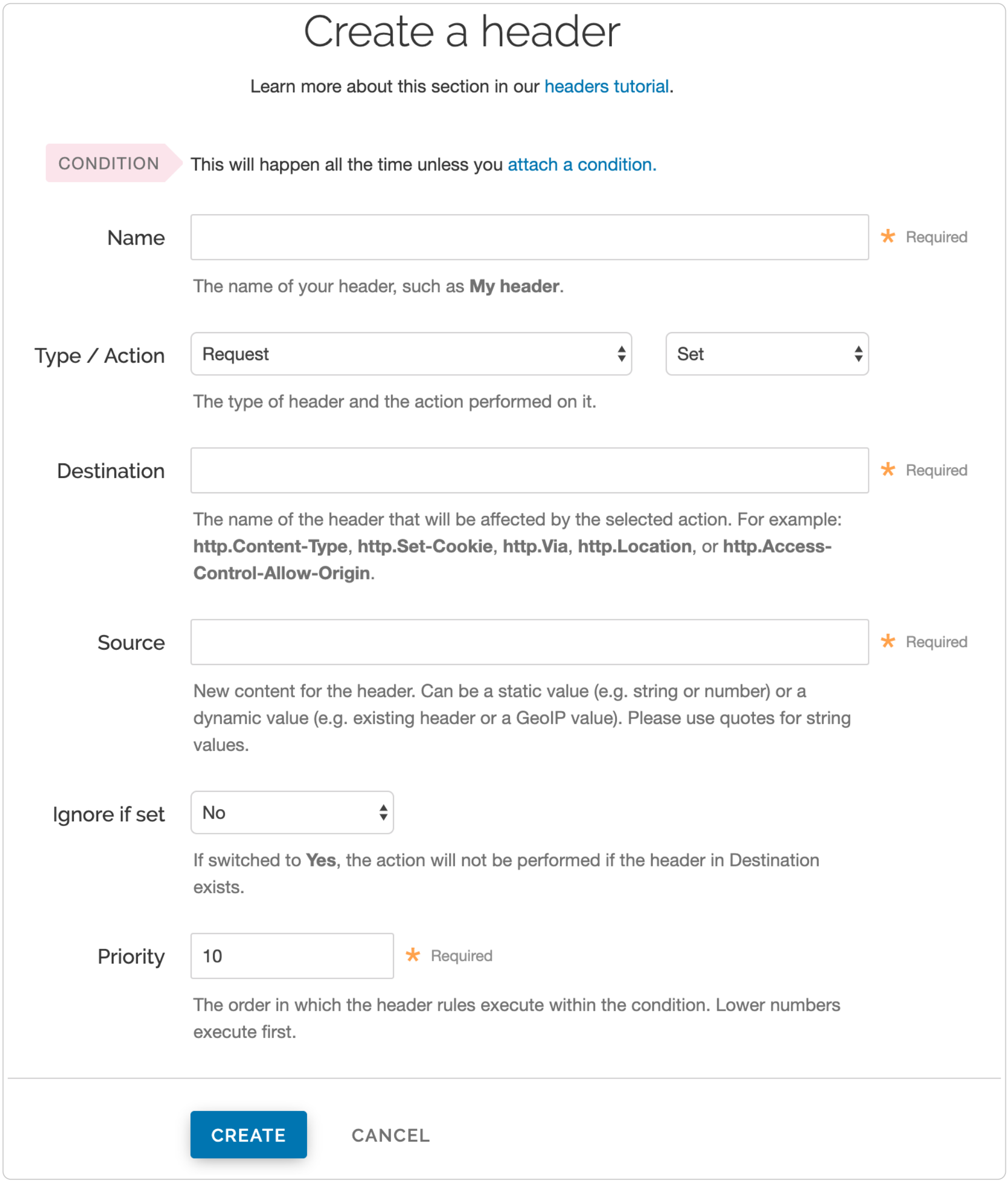
Last updated: 2018-08-16

https://docs.fastly.com/en/guides/adding-or-modifying-headers-on-http-requests-and-responses

HTTP header fields are components of the header section of request and response messages in the Hypertext Transfer Protocol (HTTP). They define the operating parameters of an HTTP transaction. When you create and configure headers, you can determine how you want your content served to your users. The following steps show you how to add and edit headers.

Create new headers

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



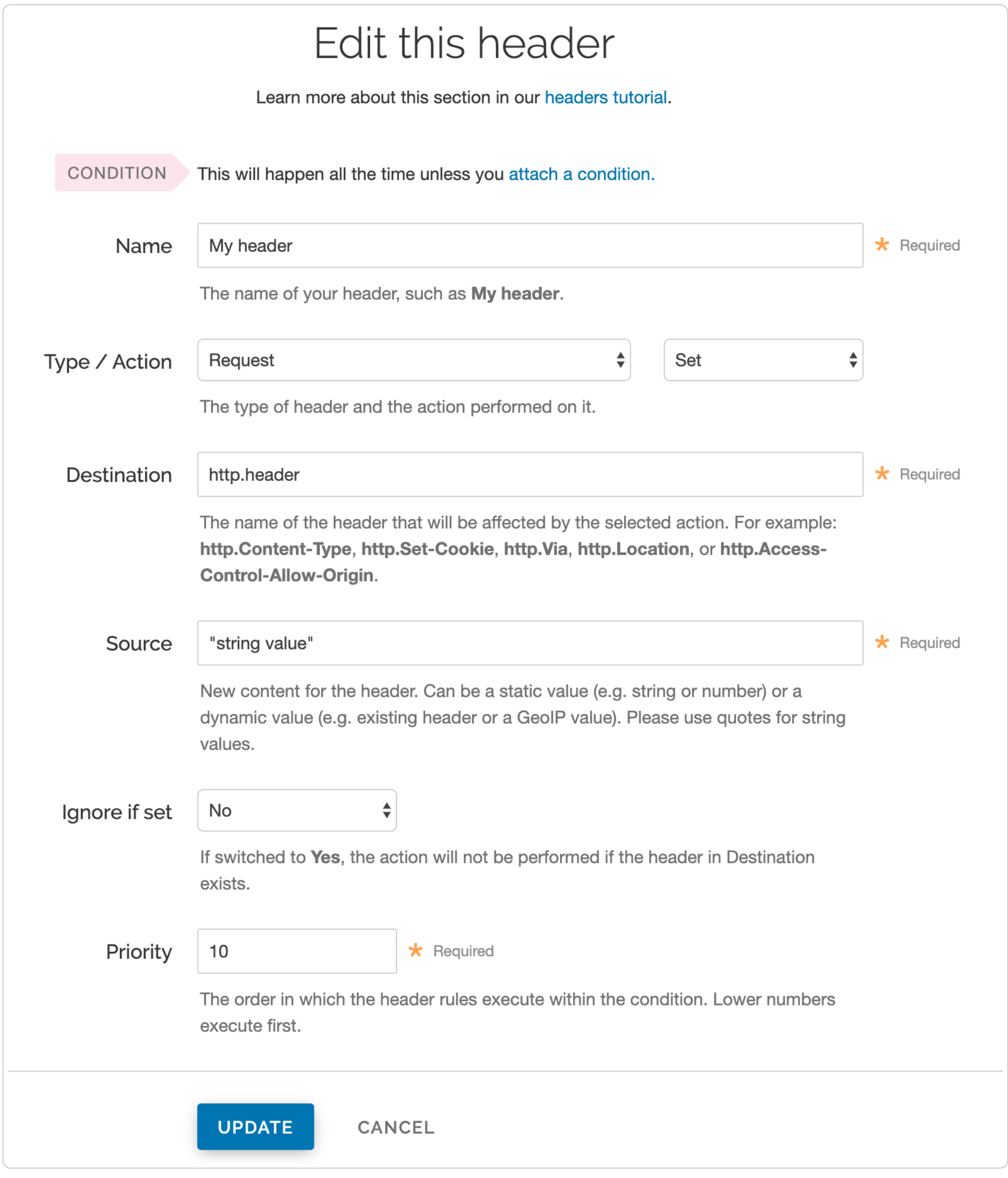
6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter the name of your header rule (for example, `My header`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter the name of the header affected by the selected action.
   - In the **Source** field, enter where the content for the header comes from.
   - From the **Ignore if set** menu, select **No** if you want the header in the **Destination** field modified or select **Yes** if you don't want it modified.
   - In the **Priority** field, enter the order the header rules execute.

   The Field description table below provides additional details about each of these controls.

7. Click the **Create** button.
8. Click the **Activate** button to deploy your configuration changes.

Edit headers

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the name of the header you want to edit. The Edit this header page appears.



6. Fill out the **Edit this header** fields as follows:
   - In the **Name** field, enter the name of your header rule (for example, `My header`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter the name of the header affected by the selected action.
   - In the **Source** field, enter where the content for the header comes from.
   - From the **Ignore if set** menu, select **No** if you want the header in the **Destination** field modified or select **Yes** if you don't want it modified.
   - In the **Priority** field, enter the order the header rules execute.
7. Click the **Update** button.
8. Click the **Activate** button to deploy your configuration changes.

Field description table

This table describes what each field in the Header window means:

Field	Description
Name	The Name field specifies a memorable word or phrase that allows you to recognize and remember a particular Header rule.
Type	The Type menu can be set to **Request**, **Response**, or **Cache**. Selecting **Request** modifies the request coming from the user, and this will carry through to the request that gets sent to your origin server. Selecting **Response** affects the HTTP response that is sent back to the user. Selecting **Cache** affects the HTTP response that your origin server returns before it gets stored on Fastly servers, meaning whatever changes you make there will be remembered on a cache hit.
Action	The Action menu can be set to **Set**, **Append**, **Delete**, **Regex**, and **Regex All**. Selecting **Set** (the default) will write a value into the header (potentially overwriting it, if it already exists). Selecting **Append** will add a value onto the end of a header or set it if it doesn't exist. Selecting **Delete** will remove a header. When selected, it hides the Source field in the Header window. Selecting **Regex** allows you to perform a find and replace on specific text and is based on a regular expression you type in. When selected, the Regex and Substitution controls appear in the Header window. Selecting **Regex All** allows you to perform the same function as **Regex** but it performs a find and replace multiple times. When selected, the Regex and Substitution controls appear in the Header window.
Destination	The Destination field determines the name of the header that is going to be affected by our Action. Because header rules can be used to affect more than just HTTP headers, your input to this field should be formatted like this: `http.Header-Name`.
Source	The Source field is available on Set, Append, Regex, and Regex All actions. This field becomes hidden in the Header window when you select Delete from the Action menu. It determines where the new content for the header comes from. There are a plethora of options for Source. The simplest is a static string such as `"My Static String"` (including the quotes). Other options include `client.ip`, `req.http.Another-Header`, and `client.geo.city`. See the list of Common Sources below for more common sources of new content.
Regex	The Regex field only appears in the Header window when you select Regex or Regex All from the Action menu. It allows you to perform a find and replace on specific text and is based on a regular expression that you type in.
Substitution	The Substitution field only appears in the Header window when you select the Regex and Regex All from the Action menu. It replaces the text that was removed by the regex expression with the text you typed in the Substitution field.
Ignore if set	By default this is set to No, which means that if the header you are modifying already exists, it will be modified.
Priority	The Priority field determines the order in which the header rules execute (e.g., a priority of 1 means the header rule executes first). This can be important if you set headers and then set other headers based on the earlier ones.

Common sources of new content

Name	Valid Types	Description
`req.http.Fastly-Client-IP`	Request, Cache, Response	The true IP address of the client.
`client.ip` and `client.identity`	Request, Cache, Response	The client IP address. These variables are available, but may not always display the source IP address. For instance, they may show the edge node IP when shielding is enabled. For the true client IP address, use `req.http.Fastly-Client-IP`. **IMPORTANT:** In some cases, client IP data may be considered sensitive. Make sure you protect the sensitive IP data you stream or store.
`server.identity`	Request, Cache, Response	A unique identifier for the Fastly server processing the request.
`server.region`	Request, Cache, Response	The region in which the Fastly server resides.
`server.datacenter`	Request, Cache, Response	The data center in which the Fastly server resides.
`req.url`	Request, Cache, Response	The URL of the HTTP Request from the client.
`req.http.*`	Request, Cache, Response	The headers from the HTTP Request, access as: `req.http.HeaderName`
`beresp.status`	Cache	The status returned from the origin server.
`beresp.http.*`	Cache	The headers from the origin's HTTP Response, access: `beresp.http.HeaderName`
`resp.status`	Response	The status that is going to be returned to the client.
`resp.http.*`	Response	The headers in the HTTP Response to be returned to the client, access: `resp.http.HeaderName`
`client.geo.*`	Request, Cache, Response	Geolocation values for the client's IP.

Enabling cross-origin resource sharing (CORS)

Last updated: 2021-02-23

https://docs.fastly.com/en/guides/enabling-cross-origin-resource-sharing

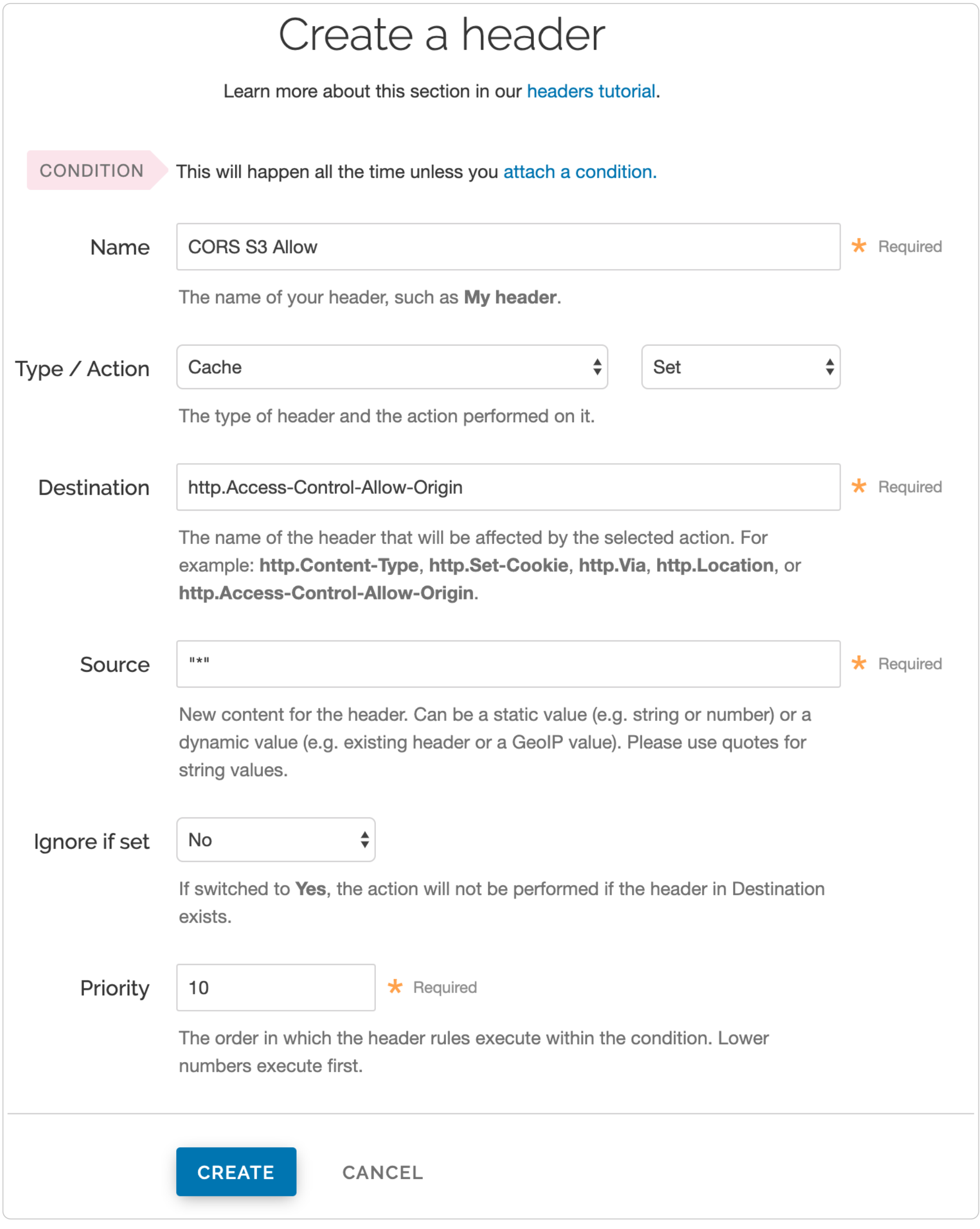
Enabling Cross-Origin Resource Sharing (CORS) allows a server to indicate that other origins can request sub-resources, like scripts and stylesheets, from it. These origins might use a different scheme (HTTP vs HTTPS) or an entirely different domain or port. This guide describes how to add an `Access-Control-Allow-Origin` header, which is sufficient for simple scenarios, and is often useful when using static bucket providers like Amazon S3 and Google Cloud Storage as an origin.

IMPORTANT

We recommend only enabling CORS when you have sub-resources on your server that you want other origins to load, especially if you allow requesting code from *any* origin, as it makes things less secure.

To enable CORS, set up a custom HTTP header for your service by following the steps below.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:

- In the **Name** field, enter a descriptive name for the new header (e.g., `CORS S3 Allow`). This name is displayed in the Fastly web interface.
- From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
- In the **Destination** field, enter `http.Access-Control-Allow-Origin`.
- In the **Source** field, enter `"*"` to allow requesting code from any origin or specify a single origin location (https://example.com). If specifying an origin, ensure CORS rules are applied in `vcl_deliver`.
- Leave the **Ignore if set** menu and the **Priority** field set to their default values.

7. Click the **Create** button. The new header appears on the Content page.
8. Click the **Activate** button to deploy your configuration changes.

> **IMPORTANT**
>
> Objects already cached won't have this header applied until you purge them.

Test it out

Running the command:

```
$ curl -I example.tld/path/to/resource
```

should include similar information to the following in your header:

```
1 Access-Control-Allow-Origin: http://example.tld
2 Access-Control-Allow-Methods: GET
3 Access-Control-Expose-Headers: Content-Length, Connection, Date...
```

> **TIP**
>
> `Access-Control-Allow-Methods` and `Access-Control-Expose-Headers` are examples of additional headers you can add with CORS. For more information about these headers, see MDN Web Docs.

Removing headers from backend response

Last updated: 2018-08-16

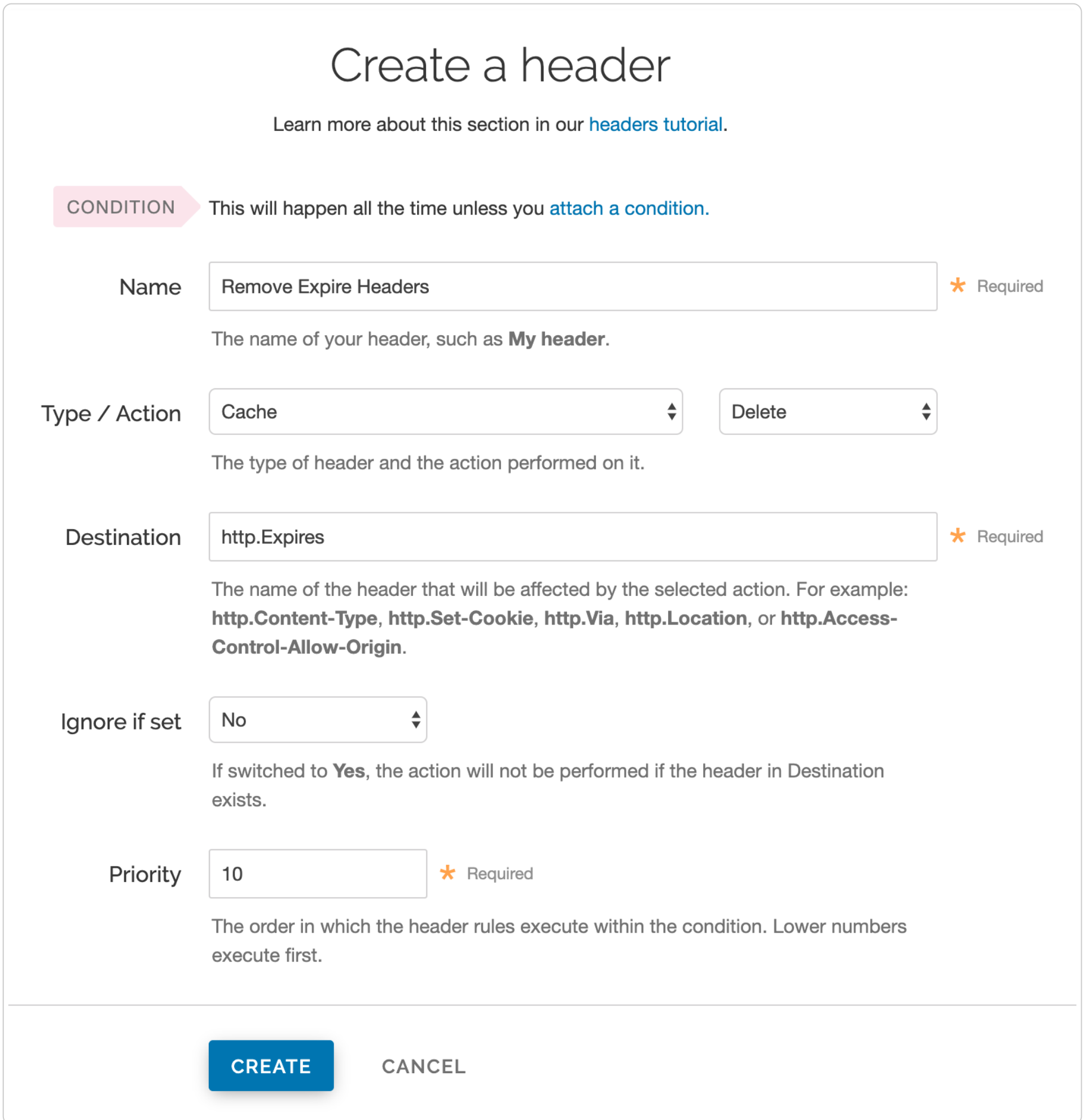
https://docs.fastly.com/en/guides/removing-headers-from-backend-response

You can remove headers from any backend response. This may be necessary if your application automatically sets headers. For example, Drupal can set the following Expires and Cache-Control headers to prevent caching:

```
1 Expires: Sun, 19 Nov 1978 05:00:00 GMT
2 Last-Modified: Wed, 18 Jul 2012 18:52:16 +0000
3 Cache-Control: no-cache, must-revalidate, post-check=0, pre-check=0
```

To remove a header from the backend response, add a new header as follows:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a descriptive name for the header rule (e.g., `Remove Expire Headers`).
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Delete**
   - In the **Destination** field, enter the name of the header (e.g., `http.Expires`).
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
7. Click the **Create** button.
8. Click the **Activate** button to deploy your configuration changes.

> **TIP**
>
> You may also be interested in our information on setting content type based on file extension.

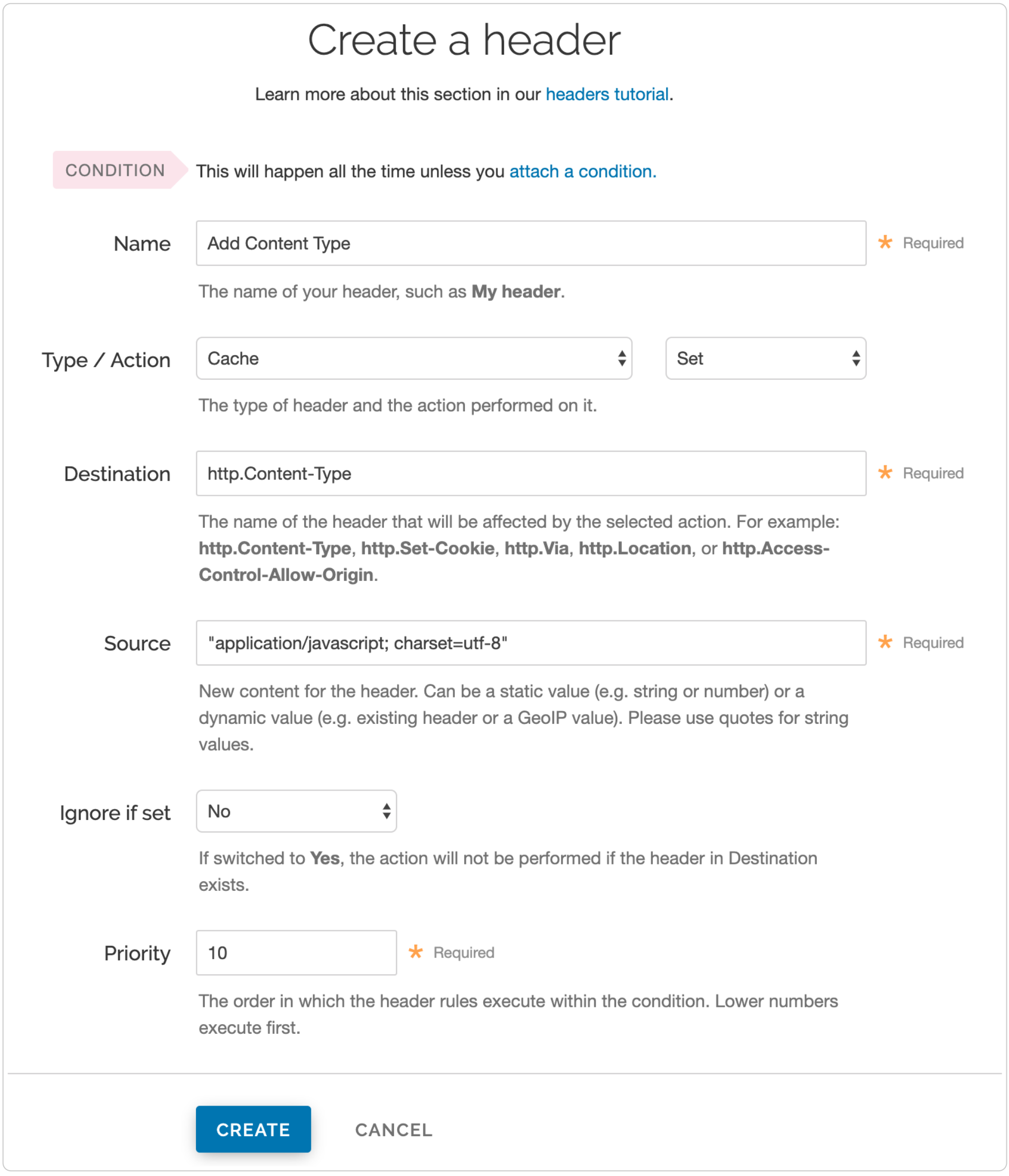
Setting Content Type based on file extension

Last updated: 2018-08-16

https://docs.fastly.com/en/guides/setting-content-type-based-on-file-extension

In some situations you may want to override the content type that a backend returns. To do that you will need to create a new header object and an associated condition.

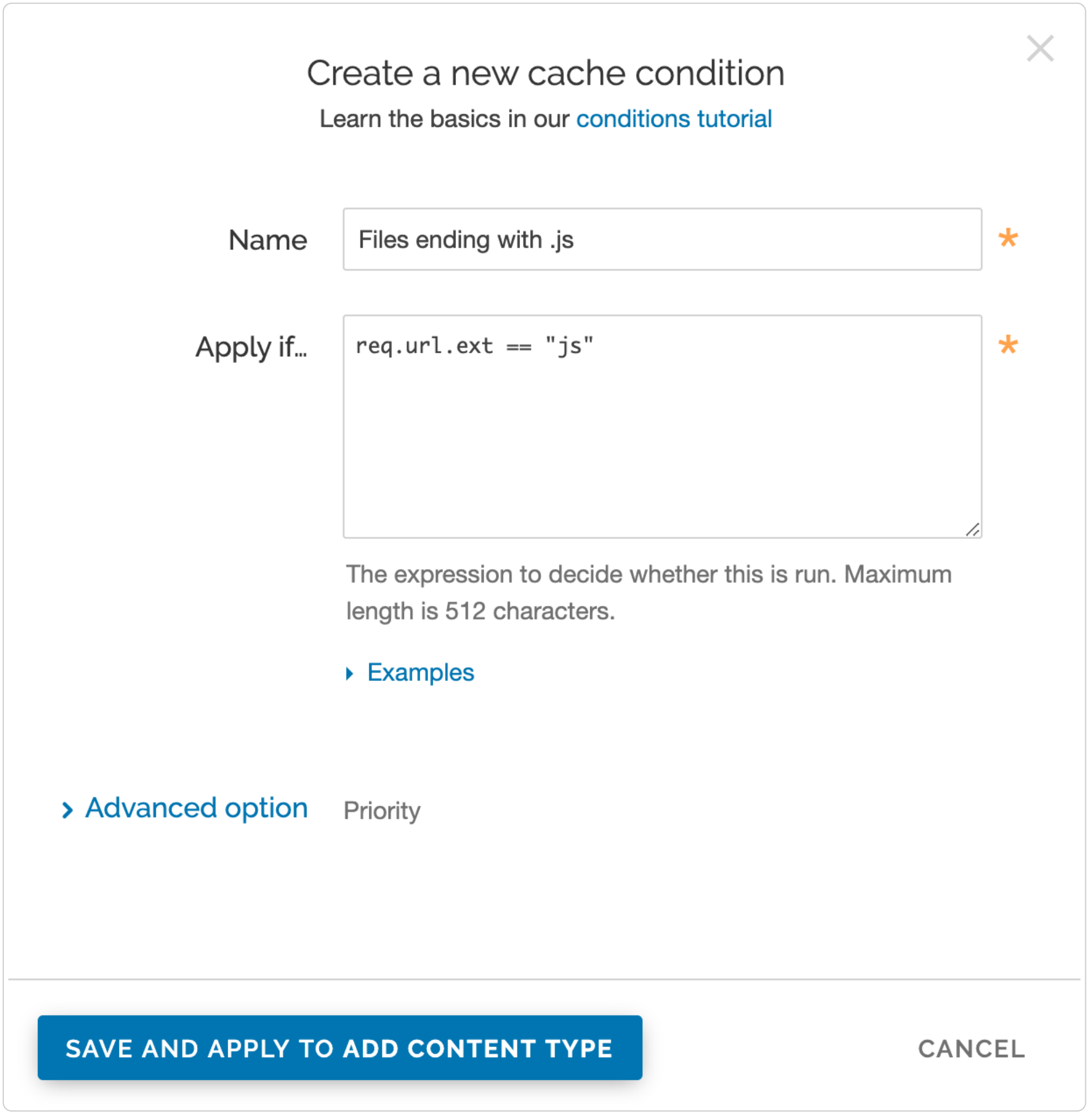
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter an appropriate name (e.g.,`Add Content Type`).
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Content-Type`.
   - In the **Source** field, enter the content type you want to match, such as `"application/javascript; charset=utf-8"`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
7. Click the **Create** button.

Once you have created the header object, apply a condition. Otherwise, that particular object is applied to all requests.

1. Click the **Attach a condition** link to the right of the new header name. The Create a new cache condition window appears.



2. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, enter a descriptive name, such as `Files ending with .js`.
   - In the **Apply if** field, enter the condition that matches your request, such as `req.url.ext == "js"` (to match the request for files ending in .js).
3. Click the **Save and apply to** button to create the new condition.
4. Click the **Activate** button to deploy your configuration changes.

TIP

You may also be interested in our guide to Removing headers from backend response.

Understanding the X-Timer header

Last updated: 2018-08-01

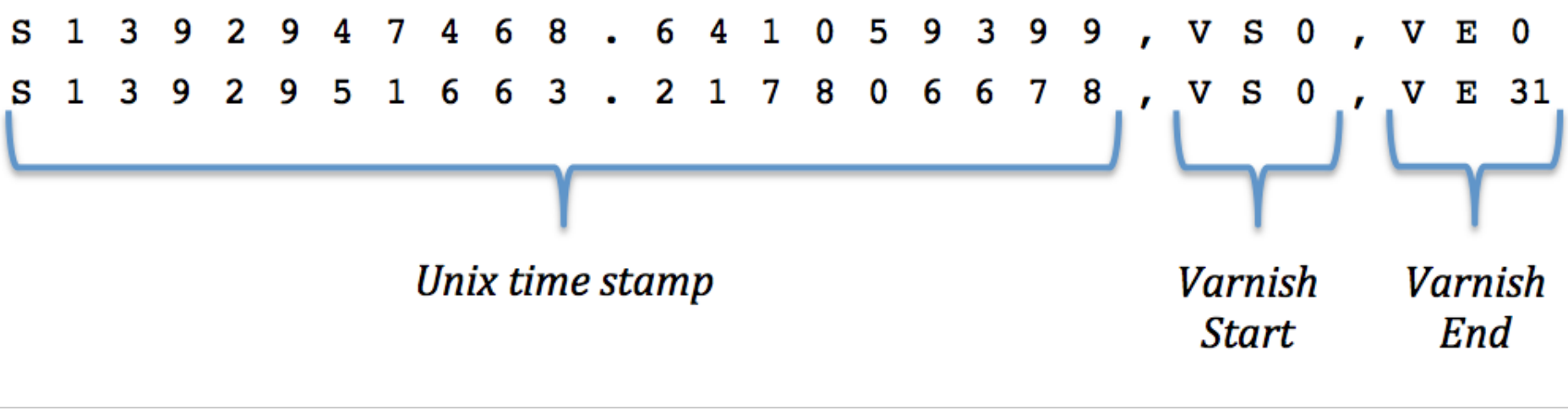
https://docs.fastly.com/en/guides/understanding-the-xtimer-header

If you look at the raw headers returned with a response from a Fastly cached asset, you will notice some extra headers tacked on. One in particular is X-Timer. This header provides timing information about the journey of a request from end to end.

Here are two examples of X-Timer headers:

- `S1392947468.641059399,VS0,VE0` (a cache HIT)
- `S1392951663.217806578,VS0,VE31` (a cache MISS)

Let's break these headers down into their parts, separated by commas, and examine what each part means.



The first section of the header, starting with `S`, represents a Unix timestamp of the start of the request on our edges.

The next section, `VS` or "varnish start," represents the start of the varnish part of the request's journey. This should always be 0 (we've got to start counting somewhere).

And the last section, `VE` or "varnish end," represents the sum of the length of the trip. For cache HITs, the length of the trip will nearly always be 0 (not actually zero, but less than a millisecond is rounded down). For cache MISSs, the number represents the number of milliseconds it took to retrieve the data from your origin server and send the response back to the requester. In the example above, it took 31ms to retrieve the data.

TIP

You can use VCL functions and variables to control dates and times.

---

§

These articles provide basic instructions for and examples of setting up and beginning to use the Fastly Image Optimizer.

https://docs.fastly.com/en/guides/configuration#_image-optimization

About Fastly Image Optimizer

Last updated: 2021-03-04

https://docs.fastly.com/en/guides/about-fastly-image-optimizer

Fastly's Image Optimizer (Fastly IO) manipulates and transforms your images in real time and caches optimized versions of them. When an image is requested from your origin server, Fastly IO can perform one or more transformation tasks before serving and caching the optimized version. For example, you can resize, adjust quality, crop and trim, serve responsive images, and more.

WARNING

Only send image content through Fastly IO. Non-image content can't be optimized using it, but will still be counted and charged as an image optimization request, which may cost you more.

Before you begin

Contact sales to request access to Fastly IO. Be sure to include the Service ID of the service for which image optimization should be enabled.

Enable shielding for your origins. Services that use image optimization are required to enable shielding. Our guide to enabling shielding provides more information on how to set this up.

When setting up shielding, be sure to choose a shield location as geographically close to your image's origin as possible. Read our guidance on choosing a shield location for more information. Also, take special note of the step immediately following your shielding location selection in that guide. If the Host header for the service has been changed from the default, you must ensure the new hostname is added to the list of domains.

Setting up image optimization

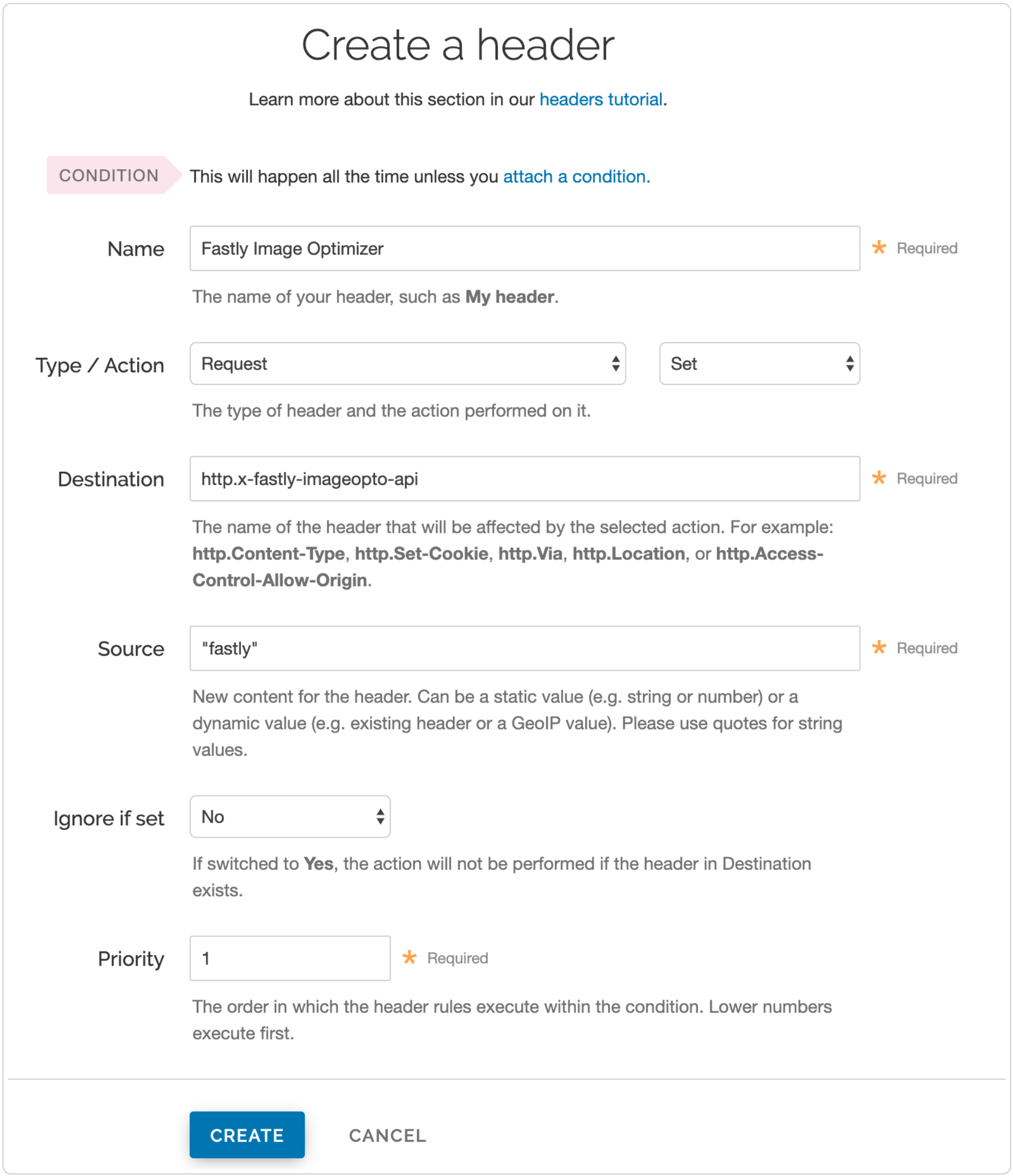
Once we've enabled it for your service, you can set up image optimization by following this process:

1. Add a header.
2. Create a request condition.
3. Test an image.

Add the Fastly Image Optimizer header

Once image optimization has been activated for your service ID and confirmed via email, configure your service by adding the Fastly Image Optimizer header.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



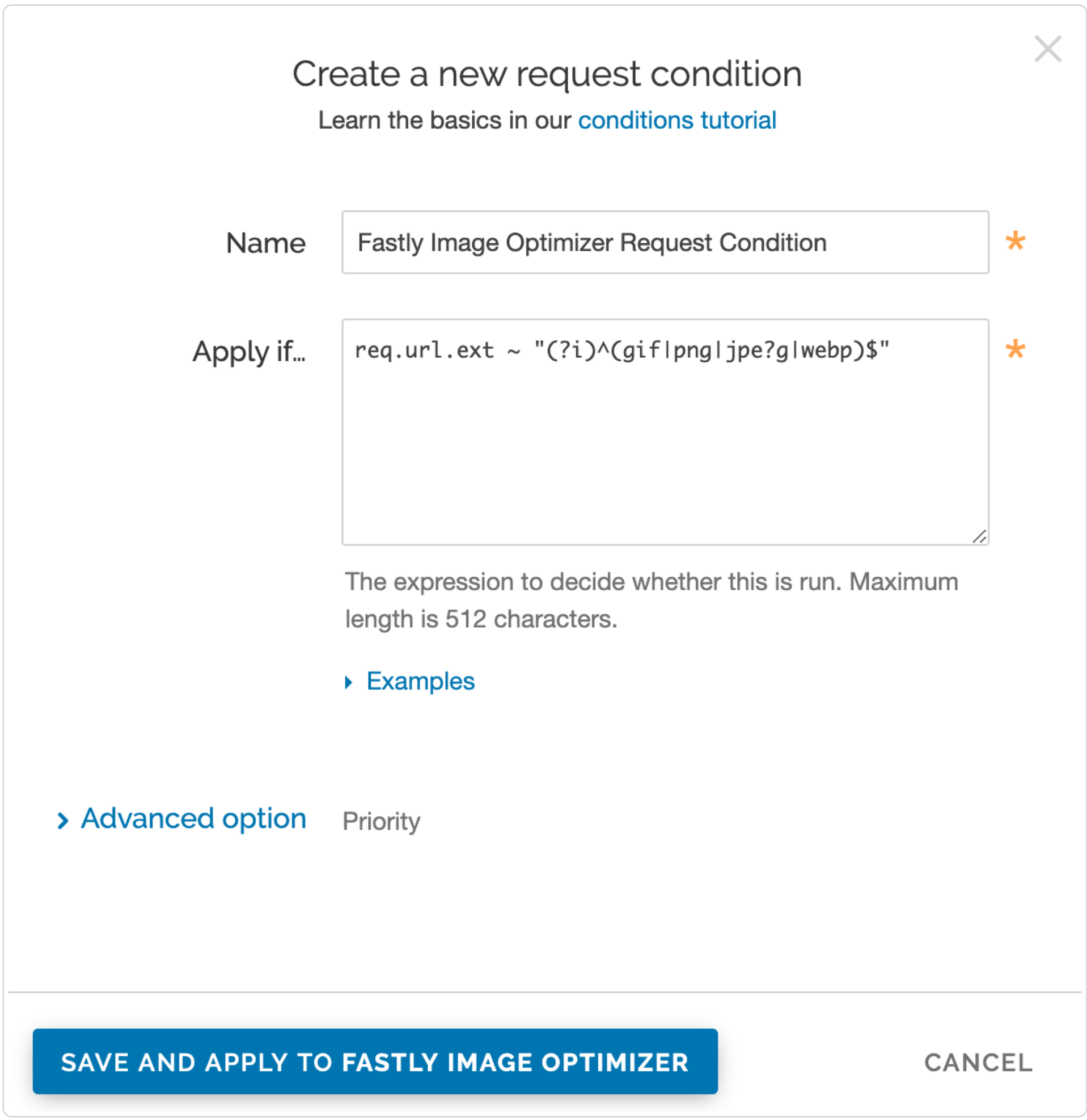
6. Fill out the **Create a header** window as follows:
   - In the **Name** field, enter `Fastly Image Optimizer`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.x-fastly-imageopto-api`.
   - In the **Source** field, enter `"fastly"`. By default, the Fastly Image Optimizer removes any additional query string parameters that are not part of our image API. If your source image requires delivery of additional query string parameters from origin then enter `"fastly; qp=*"` instead.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `1`.
7. Click **Create** to create the new header.

> **TIP**
>
> For more help with adding or modifying headers, see our guide.

Create a request condition

To ensure only your image assets are routed via the Fastly Image Optimizer, create a request condition.

1. Click the **Attach a condition** link next to the `Fastly Image Optimizer` header. The Add a condition window appears.
2. Click the **Create a new request condition** button. The Create a new request condition window appears.



3. Fill out the **Create a new request condition** window as follows:
   - In the **Name** field, enter a descriptive name for the new condition (for example, `Fastly Image Optimizer Request Condition`).
   - In the **Apply if** field, enter the appropriate request condition. For example, `req.url.ext ~ "(?i)^(gif|png|jpe?g|webp)$"` will send all files with gif, png, jpg, jpeg, and webp extensions via the Fastly Image Optimizer. Likewise, `req.url ~ "^/images/"` will send all files in the `images` directory via the Fastly Image Optimizer.
4. Click the **Save and apply to** button to create the new condition for the header.

> **TIP**
>
> For more help using conditions, see our guide.

Confirm everything is working

Once you've activated your changes, check to see if the Fastly Image Optimizer is processing your image request by typing the following command on the command line:

```
$ echo -n "Image Width: " ; curl -sI https://www.fastly.io/image.jpg?width=200 | grep -i "Fastly-Io-Info:" | cut -d' ' -f6 |
cut -d= -f2 | cut -dx -f1
```

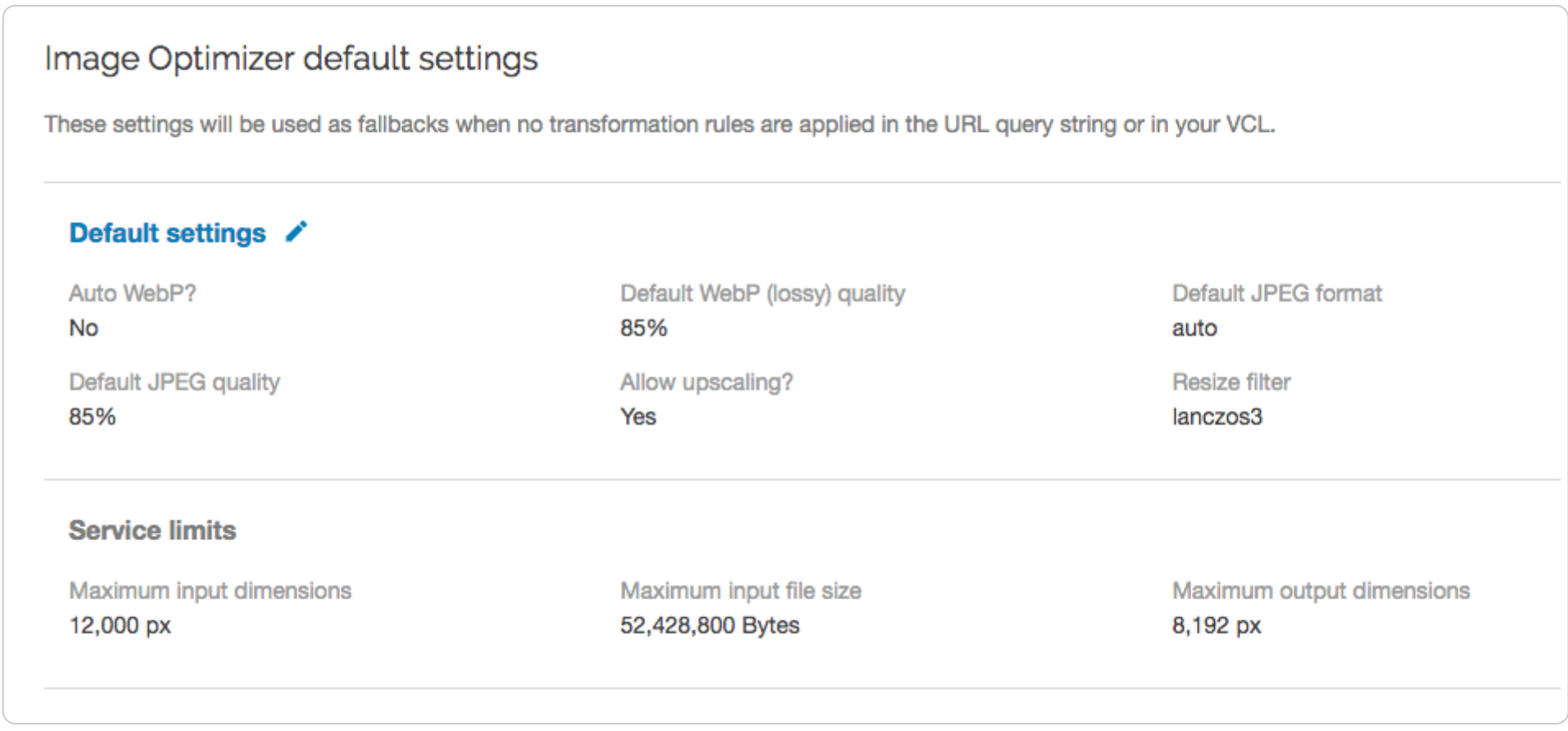
Replace `https://www.fastly.io/image.jpg?width=200` with the full image URL and width of the image you're testing.

The command line output will display the image's width, which should match the width API parameter you added to your image. For example, the output might be:

```
Image Width: 200
```

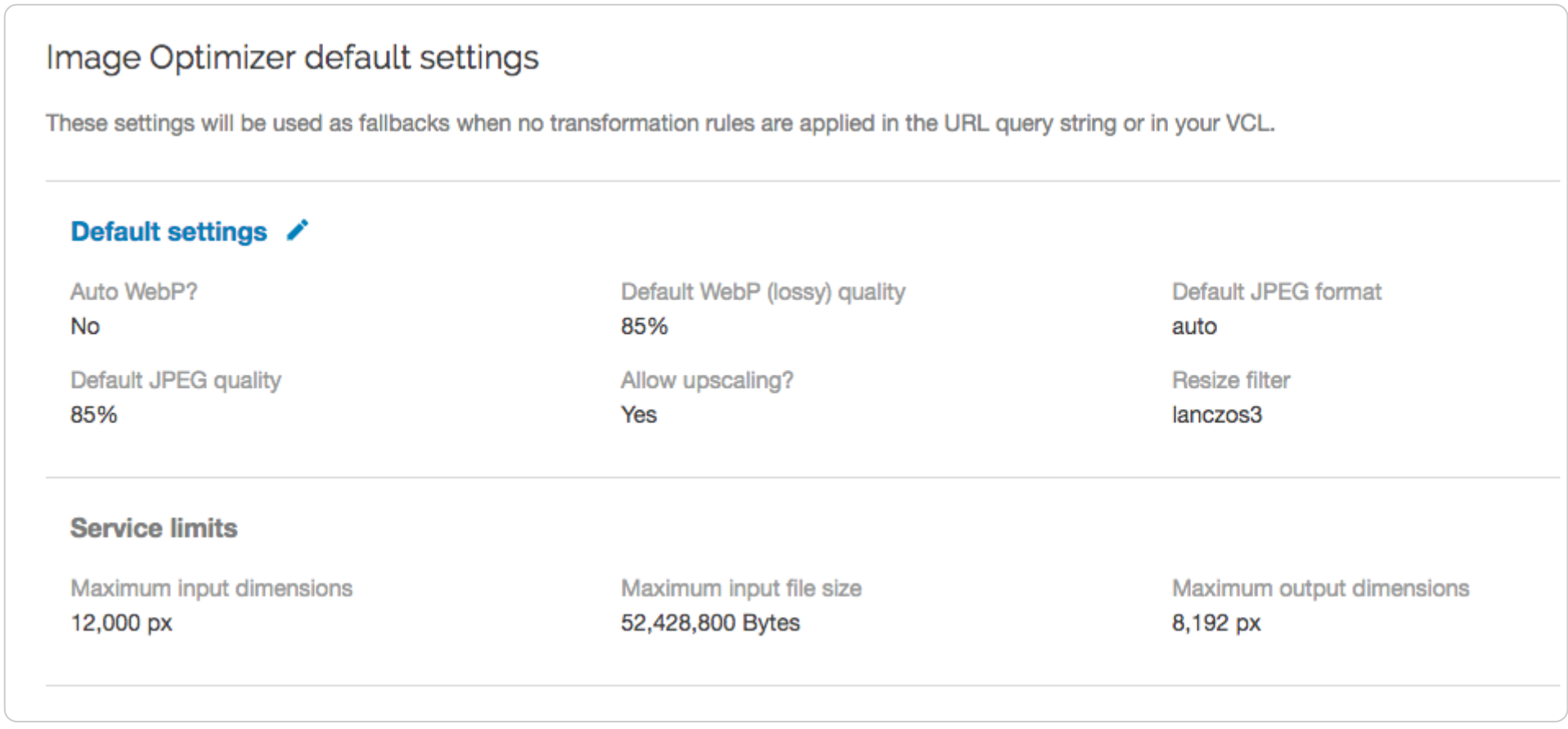
Configuring default image settings

The Fastly Image Optimizer supports a variety of image formats and applies specific settings to all images by default. Use the Fastly web interface to review and adjust the default settings as appropriate. Changes to other image settings, including most image transformations, require issuing API calls.

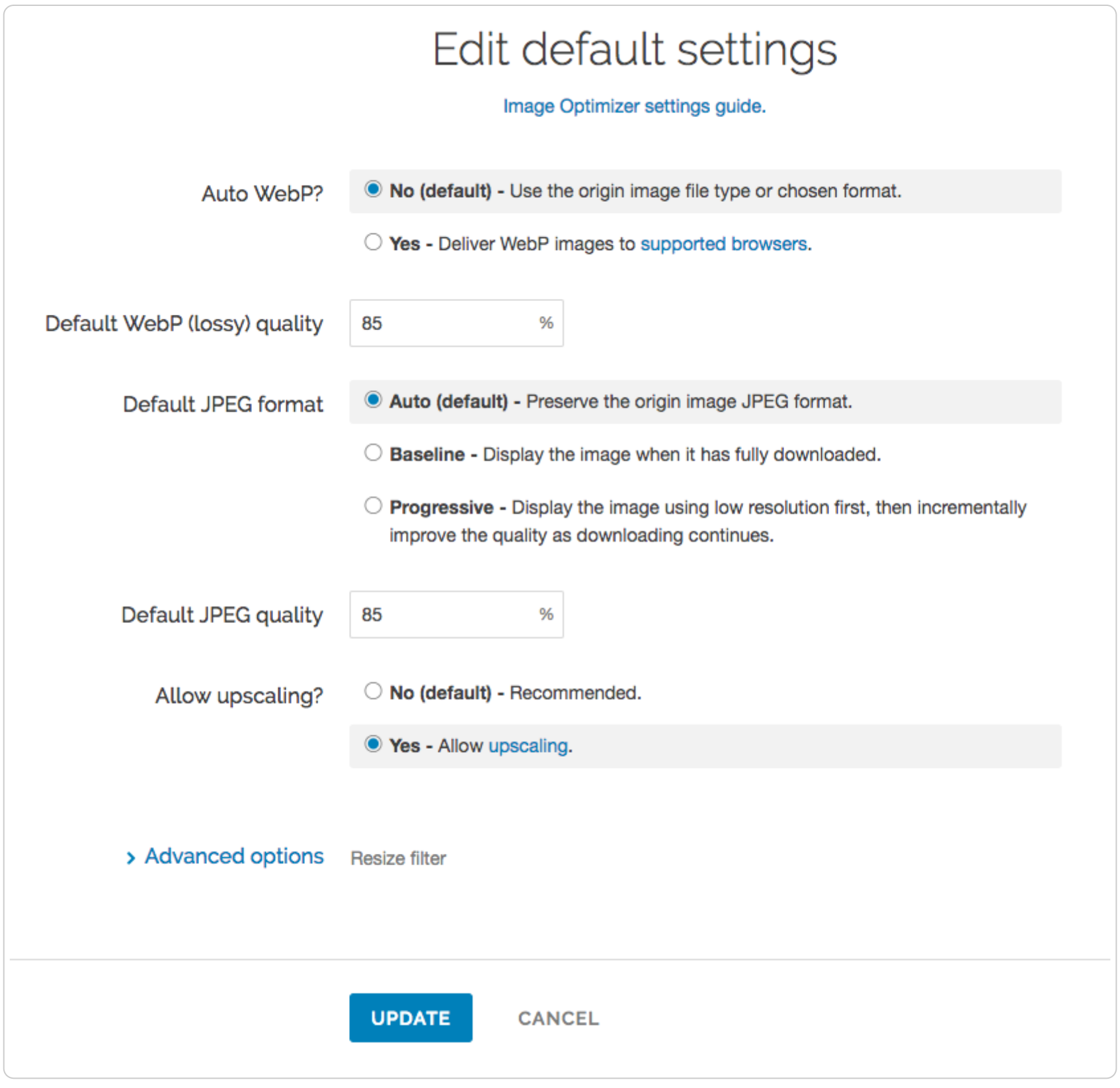


To review and edit the default image settings via the web interface, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Image optimization** link. The Image Optimizer default settings appear.



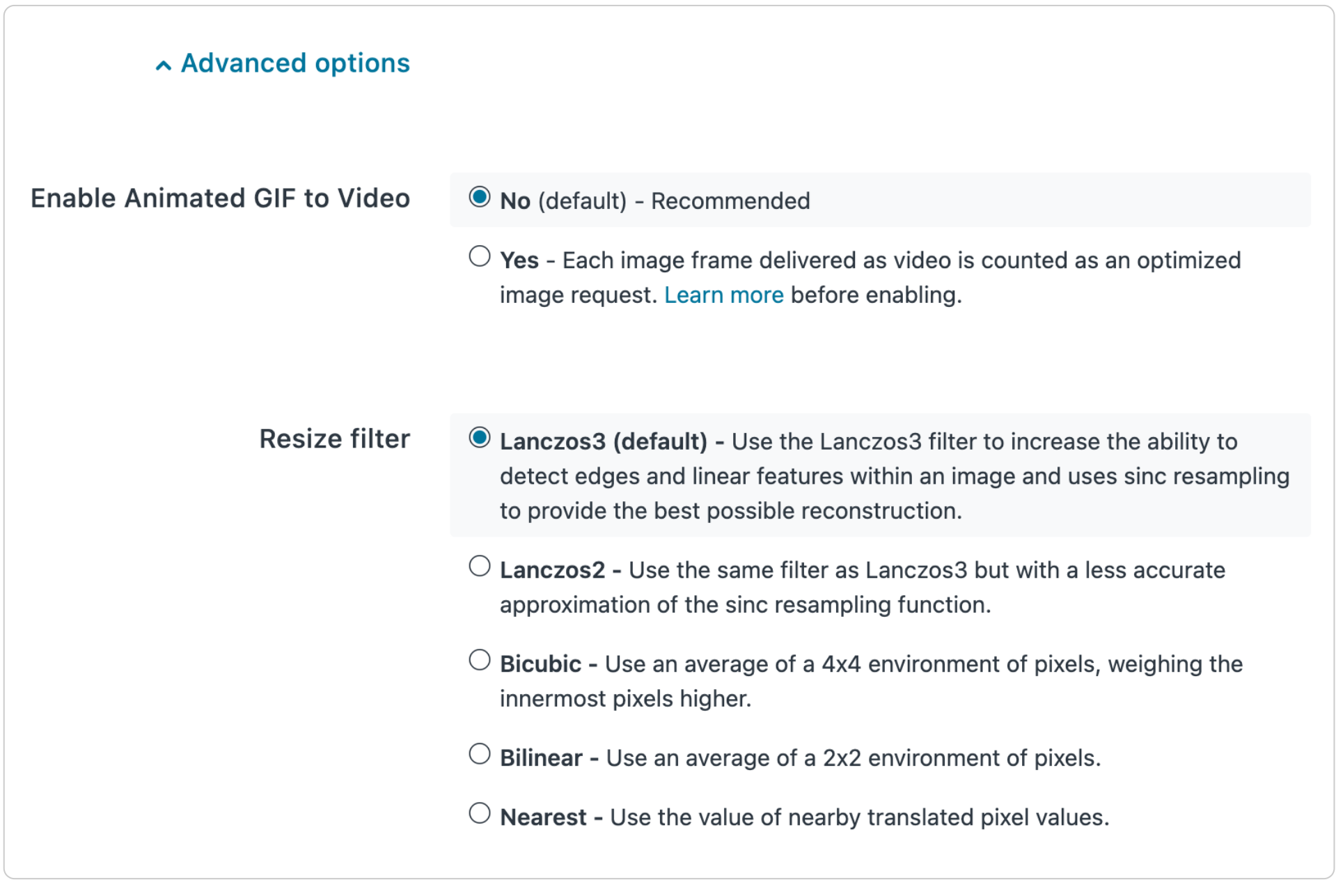
5. Click the pencil icon next to the **Default settings**. The Edit default settings window appears.



6. Adjust the **Edit default settings** as follows:
   - From the **Auto WebP** controls, leave the settings at their default or select **Yes** to convert images to the WebP format in browsers that support it. When you use the default setting, **No**, Fastly uses the image file type instead.

- In the **Default WebP (lossy) quality** field, leave the settings at their default or enter the compression level for lossy file-formatted images. Fastly uses `85` for the default quality but you can specify any whole number between `1` and `100`.
- From the **Default JPEG format** controls, leave the settings at their default or select the JPEG type to use when delivering the image. By default, Fastly sets the JPEG type to **Auto** to deliver images with the output type matching the input type. You can also select **Baseline** to display the image line by line starting from top left and going to the bottom right, or **Progressive** to display a blurry image that becomes clear as it loads.
- In the **Default JPEG quality** field, leave the settings at their default or enter the compression level for quality of lossy file formats. Fastly uses `85` for the default quality but you can specify any whole number between `1` and `100`.
- From the **Allow upscaling** controls, leave the settings at their default or select **Yes** to return images larger than the original source file so they can fit the requested dimensions.

7. Click the **Advanced options** link. The Resize filter controls appear.



8. Adjust the **Advanced options** settings as follows:
   - From the **Enable Animated GIF to Video** controls, leave the settings at their default or select **Yes** to enable Animated GIF to video functionality. Each video frame will be counted and charged as an image optimizer request.
   - From the **Resize filter** controls, select the image quality filter to use when resizing and generating new images to use a higher or lower number of pixels. By default, Fastly uses the **Lanczos3** filter. You can also choose **Lanczos2**, **Bicubic**, **Bilinear**, and **Nearest**.

Using advanced image settings

To go beyond the basic image optimization and transformation settings in the Fastly web interface, you must change your existing image URLs by adding Fastly API query string parameters. For example, if your image source existed at `http://www.example.com/image.jpg`, you would need to add `?<PARAMETER=VALUE>` to the end of the URL to create the proper query string structure for Fastly to transform the image.

Our Fastly Image Optimizer API describes each of the available image transformations in detail and includes the exact API pattern you can add to URLs, along with a description and example of how to use each parameter and its values. These examples perform transformations and optimizations on our `www.fastly.io/image.jpg` URL so you can see exactly how they work before you change your image URLs. Additionally, it provides details you should know before you start adding Fastly image transformation URL API query string parameters to your existing image URLs.

Debugging

Running into problems? See our details on image optimization debugging for more information.

This article describes a product that may use third-party cloud infrastructure to process or store content or requests for content. For more information, see our cloud infrastructure security and compliance program.

§

These articles provide information about distributing requests across multiple servers to optimize resource use and avoid overloading any single resource.

https://docs.fastly.com/en/guides/configuration#_load-balancing

About Dynamic Servers

Last updated: 2018-07-27

https://docs.fastly.com/en/guides/about-dynamic-servers

Fastly's Load Balancer allows you to create pools of origin servers that you dynamically manage using Fastly's Dynamic Servers feature to distribute and direct incoming requests. The benefits include:

- support for any infrastructure deployments including any type of server instances and one or more data centers, regions, or cloud providers
- high availability of web applications when using health checks
- compatibility with TLS termination, HTTP/2, and IPv6
- server stickiness without requiring a cookie-based approach
- implement any number of request routing rules/conditions to select a pool of origin servers

To set up Dynamic Servers, you attach a pool to a service, then add versionless origin servers that are stored separately from your VCL configuration. You can use the Fastly API to programmatically add, remove, and update pools and origin servers.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

How Dynamic Servers work

Like Edge Dictionaries and ACLs, Dynamic Servers have two major components: the pool and origin servers within it. Pools act as containers for origin servers that store the hostnames or IP addresses of servers to which incoming requests can be directed. Each pool is attached to a version of a service, but origin servers are versionless and any changes will become effective immediately.

When Dynamic Servers might be useful

Dynamic Servers might be useful for organizations that need to load balance requests among origin servers. They can be used to:

- evaluate new server instance types and new software deployments.
- independently scale individual microservices.

More specifically, they can be used as:

- **Local Server Load Balancer (LSLB)** where they are used to balance requests among origin servers within in a single region, such as AWS EC2 instances in the US East region, or within a single data center or on-premises location.

- **Global Server Load Balancer (GSLB)** where they are used to load balance requests among origin servers across any geographically distributed infrastructure deployments such as:
  - within multiple regions of an Infrastructure as a Service (IaaS) provider (e.g., AWS, GCP, Microsoft Azure).
  - between multiple IaaS providers (e.g., AWS, GCP, Microsoft Azure).
  - as part of hybrid infrastructure deployments that include a combination of on-premises origin servers or data centers and IaaS providers.

Getting started

You'll need to follow these steps:

1. Create a pool in a working version of a service that's unlocked and not yet activated.
2. Add at least one origin server to the newly created pool, keeping in mind the limitations.
3. Activate the version of the service you associated with the pool.

Once the pool is created, properly associated, and filled with origin servers, it can be called in your service.

Limitations

Keep the following limitations in mind as you use Dynamic Servers:

- **Each Fastly service can be configured with up to five origin servers.** Origin servers count as origins for the purposes of these limits. Contact sales@fastly.com to enable more than five origin servers per service in your account.
- **Pools cannot be created with custom VCL.** If you create a pool using the API, you can use the API to make changes to it and use custom VCL to interact with it.
- **Pools need at least one enabled server entry.** Origin servers cannot be deleted when a pool only has one enabled entry left.
- **Origin server deletions are permanent.** If you delete an origin server, it is permanently removed from all service versions and cannot be recovered.
- **When you delete a pool, you'll only delete it from the service version you're editing.** Pools are tied to versions and can be cloned and reverted. When using pools, we want you to be able to do things like delete a pool from a current version of your service in order to roll back your configuration to a previous version using as few steps as possible.
- **Event logs don't exist for origin server changes.** If you use the API to add, update, or remove an origin server, there will be no record of it. The only record of a change will exist when you compare service versions to view the point at which the pool was associated with the service version in the first place.

Creating and using pools with Dynamic Servers

Last updated: 2018-04-25

https://docs.fastly.com/en/guides/creating-and-using-pools-with-dynamic-servers

Fastly's Load Balancer allows you to create pools of origin servers that you dynamically manage using Fastly's Dynamic Servers feature to distribute and direct incoming requests. A pool is responsible for balancing requests among a group of origin servers. In addition to load balancing, pools can be configured to attempt retrying failed requests. Pools have a quorum setting that can be used to determine when the pool as a whole is considered `up` in order to prevent problems following an outage as origin servers come back up.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

Creating a pool

To start using Dynamic Servers, you'll need to create an empty pool within a version of a service that's unlocked and not yet activated. Make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/version/<service_version>/p
ool -d 'name=<pool_name>&comment=<comment>'
```

The response will look like this:

```
{
    "between_bytes_timeout": "10000",
    "comment": "<comment>",
    "connect_timeout": "1000",
    "created_at": "2016-08-01T14:43:22+00:00",
    "deleted_at": null,
    "first_byte_timeout": "15000",
    "healthcheck": null,
    "id": "2IpWU5CGzPpbpGsABSDops",
    "max_conn_default": "200",
    "max_tls_version": null,
    "min_tls_version": null,
    "name": "<pool_name>",
    "quorum": "75",
    "request_condition": null,
    "service_id": "<service_id>",
    "shield": null,
    "tls_ca_cert": null,
    "tls_cert_hostname": null,
    "tls_check_cert": 1,
    "tls_ciphers": null,
    "tls_client_cert": null,
    "tls_client_key": null,
    "tls_sni_hostname": null,
    "type": "random",
    "updated_at": "2016-08-01T14:43:22+00:00",
    "use_tls": 0,
    "version": "<service_version>"
}
```

Be sure to activate the new version of the service you associated with the pool after adding at least one origin server.

> **NOTE**
>
> Each Fastly service can be configured with up to five origin servers. Contact sales@fastly.com to enable more than five origin servers per service in your account.

Viewing pools

To view a list of all pools attached to a particular version of a service, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/version/<service_version>/pool
```

The response will look like this:

```
[
{
    "between_bytes_timeout": "10000",
    "comment": "just my first pool",
    "connect_timeout": "1000",
    "created_at": "2016-08-01T14:43:22+00:00",
    "deleted_at": null,
    "first_byte_timeout": "15000",
    "healthcheck": null,
    "id": "2IpWU5CGzPpbpGsABSDops",
    "max_conn_default": "200",
    "max_tls_version": null,
    "min_tls_version": null,
    "name": "SP_Prod_Pool_1",
    "quorum": "75",
    "request_condition": null,
    "service_id": "<service_id>",
    "shield": null,
    "tls_ca_cert": null,
    "tls_cert_hostname": null,
    "tls_check_cert": 1,
    "tls_ciphers": null,
    "tls_client_cert": null,
    "tls_client_key": null,
    "tls_sni_hostname": null,
    "type": "random",
    "updated_at": "2016-08-01T14:43:22+00:00",
    "use_tls": 0,
    "version": "<service_version>"
}
]
```

To see information related to a single pool (in this example, `SP_Prod_Pool_1`) attached to a particular version of a service, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/version/<service_version>/pool/<pool_name>
```

The response will look like this:

```
{
    "between_bytes_timeout": "10000",
    "comment": "just my first pool",
    "connect_timeout": "1000",
    "created_at": "2016-08-01T14:43:22+00:00",
    "deleted_at": null,
    "first_byte_timeout": "15000",
    "healthcheck": null,
    "id": "2IpWU5CGzPpbpGsABSDops",
    "max_conn_default": "200",
    "max_tls_version": null,
    "min_tls_version": null,
    "name": "SP_Prod_Pool_1",
    "quorum": "75",
    "request_condition": null,
    "service_id": "<service_id>",
    "shield": null,
    "tls_ca_cert": null,
    "tls_cert_hostname": null,
    "tls_check_cert": 1,
    "tls_ciphers": null,
    "tls_client_cert": null,
    "tls_client_key": null,
    "tls_sni_hostname": null,
    "type": "random",
    "updated_at": "2016-08-01T14:43:22+00:00",
    "use_tls": 0,
    "version": "<service_version>"
}
```

Deleting a pool

Deleting a pool deletes the pool and all of its associated server entries. To delete a pool, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X DELETE https://api.fastly.com/service/<service_id>/version/<service_version
>/pool/<pool_name>
```

The response will look like this:

```
{
  "status":"ok"
}
```

Creating and using server entries with Dynamic Servers

Last updated: 2018-04-25

https://docs.fastly.com/en/guides/creating-and-using-server-entries-with-dynamic-servers

Fastly's Load Balancer allows you to create pools of origin servers that you dynamically manage using Fastly's Dynamic Servers feature to distribute and direct incoming requests. An origin server is an address (IP address or hostname) of a server to which the Dynamic Servers feature can forward requests. Fastly can then select any one of the origin servers based on a selection policy defined for the pool.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

Creating an origin server

To add an origin server to the pool, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/pool/<pool_id>/server -d 'a
ddress=<hostname_or_ip_address>'
```

The response will look like this:

```
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": false,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
```

NOTE

Each Fastly service can be configured with up to five origin servers. Contact sales@fastly.com to enable more than five origin servers per service in your account.

Viewing origin servers

To see information related to a single origin server, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/pool/<pool_id>/server/<hostname_or_
ip_address>
```

The response will look like this:

```
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": false,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
```

To view a list of all origin servers attached to a particular pool, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/pool/<pool_id>/servers
```

The response will look like this:

```
[
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": false,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
]
```

Enabling and disabling origin servers

You can enable or disable an origin server to control whether or not traffic is sent to it. Disabling an origin server allows you to remove it from the pool temporarily.

Enabling an origin server

Origin servers are enabled by default. To enable an origin server that has been disabled, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/pool/<pool_id>/server -d 'address=<hostname_or_ip_address>&disabled=false'
```

```
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": false,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
```

Disabling an origin server

To disable an origin server, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/pool/<pool_id>/server -d 'address=<hostname_or_ip_address>&disabled=true'
```


```
{
   "id": "6kEuoknxiaDBCLiAjKqyXq",
   "service_id": "<service_id>",
   "pool_id": "<pool_id>",
   "weight": "100",
   "max_conn": "200",
   "port": "80",
   "address": "<hostname_or_ip_address>",
   "comment": "",
   "disabled": true,
   "created_at": "2016-06-20T08:20:36+00:00",
   "updated_at": "2016-06-20T08:20:36+00:00",
   "deleted_at": null
}
```


Deleting an origin server

To permanently delete an origin server, make the following API call in a terminal application:

```
$ curl -vs -H "Fastly-Key: FASTLY_API_TOKEN" -X DELETE https://api.fastly.com/service/<service_id>/pool/<pool_id>/server/<server_id>
```

The response will look like this:


```
{
  "status":"ok"
}
```


NOTE

Pools must have at least one origin server. The API won't allow you to delete the last origin server in the pool.

Load-balancing configuration

Last updated: 2021-10-25

https://docs.fastly.com/en/guides/load-balancing-configuration

This guide describes how to automatically load balance between two or more origin servers. Load balancing distributes requests across multiple servers to optimize resource use and avoid overloading any single resource.

Before you begin

Before you configure load balancing, keep in mind the following:

- To prevent errors when shielding is enabled, all backends in the automatic load balancing group must use the same shielding location.
- Conditions on your origin server can directly change how automatic load balancing behaves. Be sure to review conditions behavior to ensure automatic load balancing works properly.
- Many customers configure failover at the same time they configure load balancing functionality. Our guide on configuring failover can show you how.

Enabling load balancing

To enable load balancing across two or more origin servers, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.

3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the name of the Host you want to edit. The Edit this Host page appears.
6. From the **Auto load balance** menu, select **Yes**.
7. In the **Weight** field, enter the percentage of the total traffic to send to the origin server.

> **TIP**
>
> When you specify a whole number in the **Weight** field, you specify the percentage of the total traffic to send to a specific origin server. Each origin server receives the percentage (<weight>/<total>) of the total traffic equal to the number you specify. For example, if you have two origin servers, A and B, setting the weight to 50 on both splits the traffic between them equally. Each origin server receives 50 percent of your total traffic. If you increase the weight on origin server A to 55 and decrease the weight on origin server B to 45, the percentage of traffic changes to 55 percent and 45 percent respectively.

8. Click the **Update** button.
9. Repeat steps 5, 6, 7, and 8 for each origin server you want to include in the automatic load balancing group.

> **NOTE**
>
> Each Fastly service can be configured with up to five origin servers. Contact sales@fastly.com to enable more than five origin servers per service in your account.

10. Click the **Activate** button to deploy your configuration changes.

Using conditions with load balancing

You can set conditions on origin servers or headers to change how load balancing works.

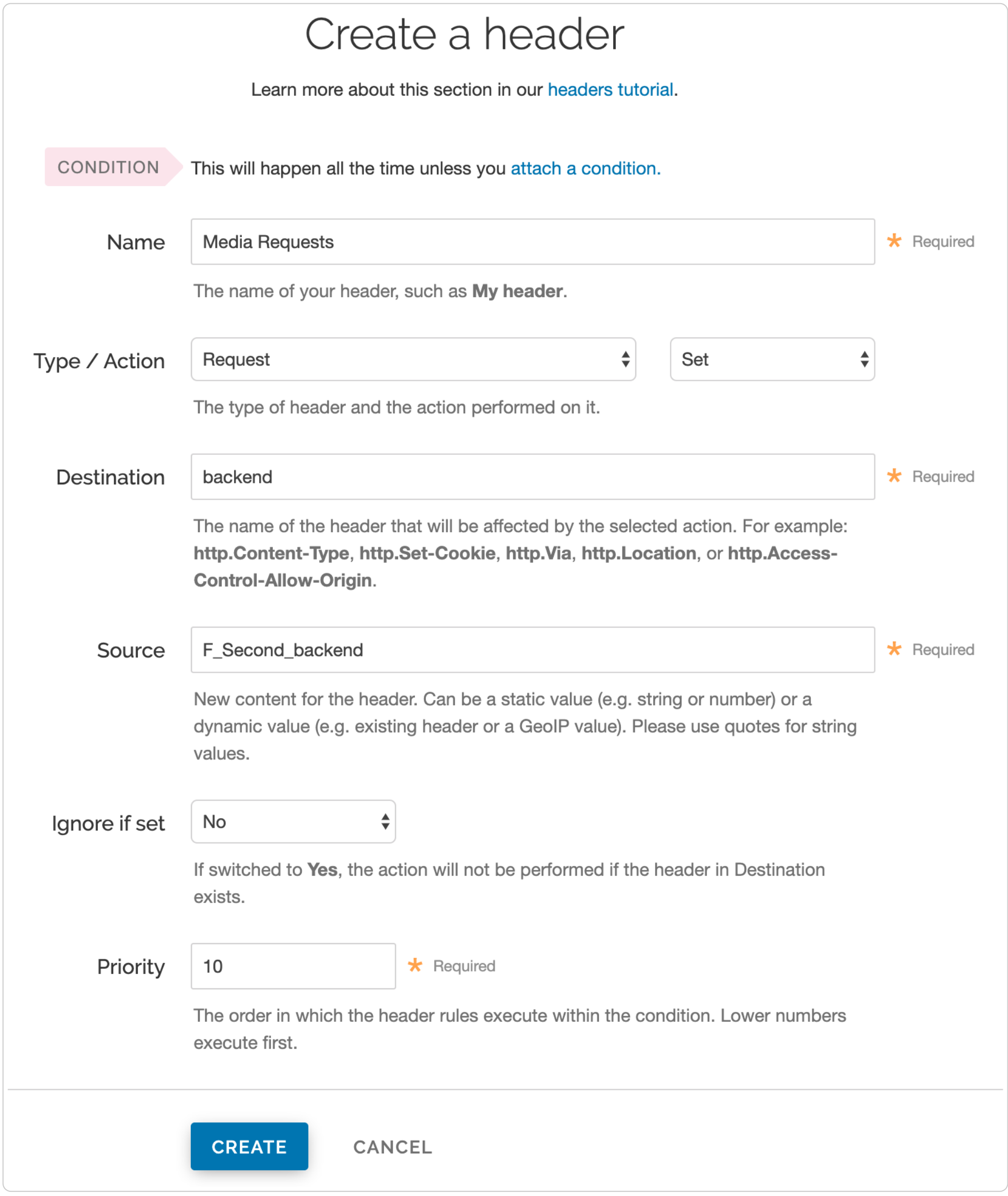
Setting conditions on origin servers

When you set conditions on origin servers, you can potentially change how automatic load balancing works. The load balancing autodirector groups servers together based on like conditions, giving you the flexibility to effectively create subsets of the autodirector by assigning a condition to one group of origins and another condition to another set of origins. If each group of origin servers has a different condition that affects load balancing, the auto load function will not randomly load balance between the different servers.

Setting conditions on headers

Conditions can also be assigned to a server through a header. For example, you have three servers called F_Fastly, F_Second_backend, and F_Third_backend and want all URLs with a certain prefix to default to the second server. First, you'd create a header.

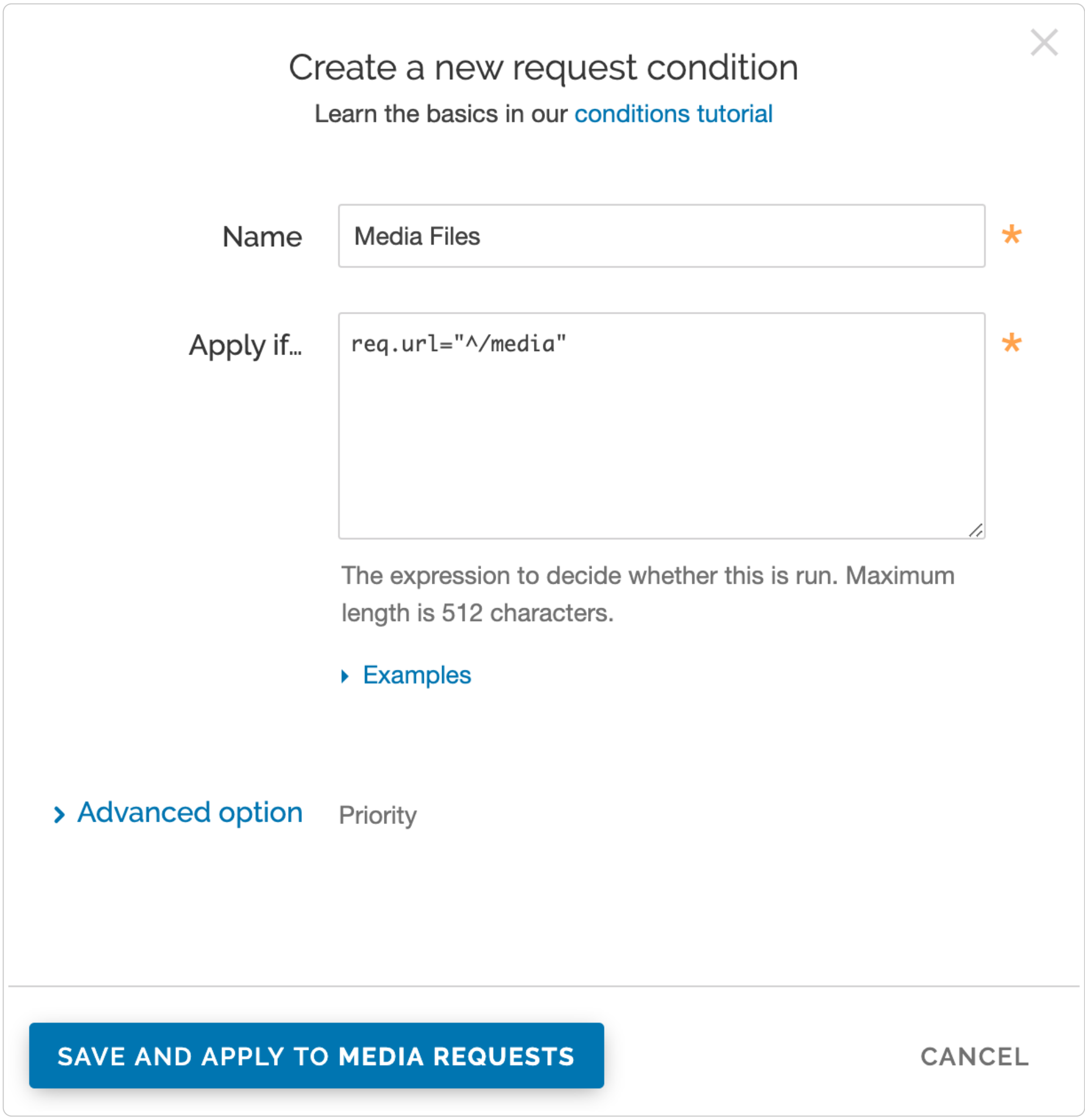
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. In the **Headers** area, click the **Create header** button to create a new header. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a descriptive name for the new header (for example, `Media Requests`).
   - From the **Type** menu, select **Request** and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter the name of the header that will be affected by the action (for example, `backend`).
   - In the **Source** field, enter the name of the origin server the content for this header comes from (for example, `F_Second_backend`).
   - Leave the **Ignore if set** and **Priority** fields set to their default settings.
7. Click **Create**.

After the header is created, you'd create a new condition to apply if the URL matches the desired prefix.

1. In the **Headers** area, click the **Attach a condition** link next to the name of the new header you just created. The Create a new request condition window appears.



2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter a descriptive name for the new condition (for example, `Media Files`).
   - In the **Apply if** field, enter the request condition that will be applied (for example, `req.url="^/media"`).
   - Leave the priority set to its default value.
3. Click the **Save and apply to** button to create the new condition for the header.
4. Click the **Activate** button to deploy your configuration changes.

The generated VCL below illustrates the autodirector set for all three servers. Within the section **sub vcl_recv**, the default origin server is set to the autodirector and, if the media condition is met, requests are forwarded to the second server. If the condition is not met, requests are forwarded to one of the three servers at random.

```
director autodirector_ random {
  {
    .backend = F_Second_backend;
    .weight  = 100;
  }{
    .backend = F_Third_backend;
    .weight  = 100;
  }{
    .backend = F_Fastly;
    .weight  = 100;
  }
}

sub vcl_recv {
#--FASTLY RECV CODE START
  if (req.restarts == 0) {
    if (!req.http.X-Timer) {
      set req.http.X-Timer = "S" time.start.sec "." time.start.usec_frac;
    }
    set req.http.X-Timer = req.http.X-Timer ",VS0";
  }

  # default conditions
  set req.backend = autodirector_;

  # end default conditions

  # Request Condition: Media files Prio: 10
  if (req.url ~ "^/media") {

    # Header rewrite Media Requests : 10
    set req.backend = F_Second_backend;
  }
  #end condition

#--FASTLY RECV CODE END
}
```

§

These articles describe how to adjust the performance of Fastly's services beyond standard configuration methods.

https://docs.fastly.com/en/guides/configuration#_performance

Enabling automatic gzipping

Last updated: 2022-02-16

https://docs.fastly.com/en/guides/enabling-automatic-gzipping

Fastly supports gzip data compression to speed up information transmission. The Fastly gzip feature dynamically fetches content from origin, compresses it, and then caches it. There are two ways to enable gzip:

- **Based on file extension.** Enable the default gzip policy to compress content in files with the following extensions: `css js html eot ico otf ttf json svg`.
- **Based on content type.** Set up an advanced gzip policy to customize the content and conditions for compression.

Compression can reduce the size of responses which reduces the bandwidth you use and thus can reduce your total bill.

IMPORTANT

Fastly provides support for Brotli compression as part of a limited availability release. To enable this feature for your account, contact support@fastly.com. For more information, see our product and feature lifecycle descriptions.

Enabling gzip

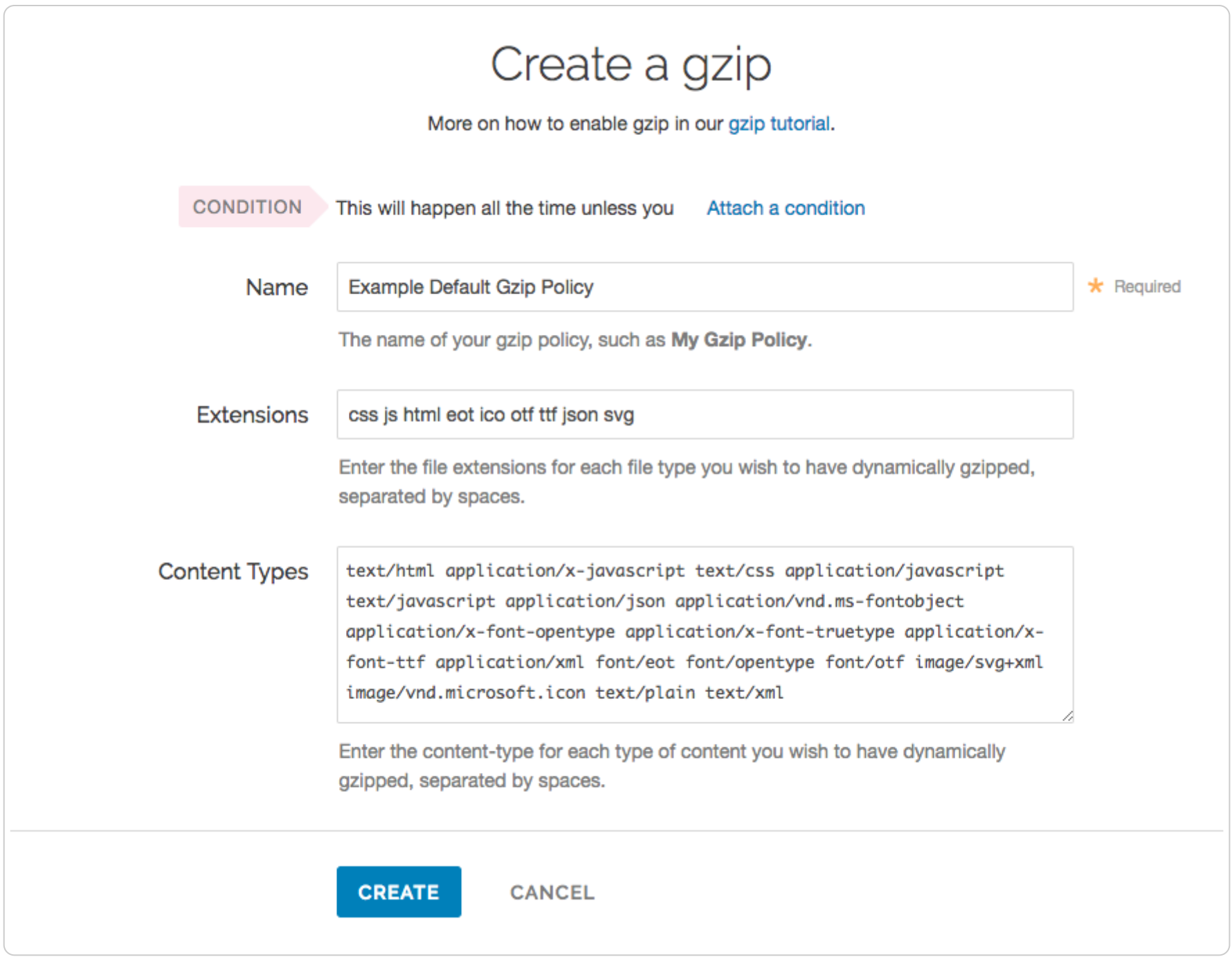
To dynamically gzip cacheable content based on file extension or content-type, follow the steps below to enable the default gzip policy:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Default gzip policy** switch to enable gzip.
6. Click the **Activate** button to deploy your configuration changes.

Setting up an advanced gzip policy

To customize the content that's compressed and the conditions under which this compression occurs, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Set up advanced gzip** button. The Create a gzip page appears.
6. Click the **Override these defaults** link. Additional gzip fields appear.



7. Fill out the **Create a gzip** fields as follows:
   - In the **Name** field, enter an arbitrary name for your new gzip rule.
   - In the **Extensions** field, enter the file extension for each file type to be dynamically gzipped, separated by spaces. Only enter the three- or four-letter string representing the file extension.

- In the **Content Types** field, enter the content-type for each type of content to dynamically gzip, separated by spaces. Do not use regular expressions.

8. Click the **Create** button. The new gzip policy appears.

9. Click the **Activate** button to deploy your configuration changes.

> **TIP**
>
> Because gzip is one of the most common reasons to vary output based on a request header, Fastly will normalize the value of `Accept-Encoding` on incoming requests. For more information, see our documentation on the Accept-Encoding header.

Limitations and considerations

This feature has the following limitations and considerations you should take into account:

- The automatic gzip feature can only compress cacheable content.
- Content already compressed at origin will be cached as-is. We will not attempt to re-compress it.
- As a best practice, we recommend compressing content at the origin whenever possible to improve performance and reduce costs (e.g., cloud egress charges).

Enabling HTTP/3 for Fastly services

Last updated: 2022-03-29

https://docs.fastly.com/en/guides/enabling-http3-for-fastly-services

This guide describes how to enable HTTP/3 for your Fastly services.

About HTTP/3

HTTP/3 uses a web transport protocol standard called QUIC that you can offer to end user clients as an optional upgrade from HTTP/1.1 and HTTP/2 connections. End user clients will initially connect to your Fastly service using an earlier version of HTTP and will only attempt to use HTTP/3 if the Fastly service offers it.

Unlike the TCP transport protocol used by earlier versions of HTTP, QUIC requires that all connections be secured and uses TLS 1.3 to secure them. Like TLS 1.3, QUIC also supports the optional 0-RTT feature to help reduce the latency of resumed connections.

Prerequisites

To use HTTP/3 on Fastly's edge cloud services, you will need:

- a Fastly user account assigned the role of superuser
- relevant domains you will be using for HTTP/3 added to a properly configured Fastly service
- access privileges to modify DNS records for the domains you will be using

Limitations and key behaviors

- Fastly currently only supports HTTP/3 for end user connections, as that is where we expect the primary benefits of these protocols will be seen. We do not support HTTP/3 between Fastly and your origin servers.
- The QUIC transport protocol used by HTTP/3 requires all connections to be secured using TLS 1.3. This means that when you configure domains to offer HTTP/3, you are also configuring them offer TLS 1.3 to clients using HTTP/1.1 or HTTP/2.

Enabling HTTP/3 for Fastly services

To enable HTTP/3 for your Fastly services, start by configuring HTTP/3 on your domains and then configure your services to offer HTTP/3 on client connections.

Configuring HTTP/3 on your domains

To configure HTTP/3 on a new domain, refer to Setting up TLS 1.3 for a new domain.

To configure HTTP/3 on an existing domain, complete the following:

1. Log in to the Fastly web interface and click the **Secure** link. The TLS domains page appears, displaying any domains for which TLS has been or can be enabled.
2. Find the domain on which you plan to offer HTTP/3 and use the **TLS configuration and DNS details** column to verify whether HTTP/3 has been enabled.
   - If the column has a value of either `HTTP/3 & TLS v1.3` or `HTTP/3 & TLS v1.3 + 0RTT`, then the domain already supports HTTP/3. Continue to the next section to offer HTTP/3 support for traffic to that domain from your service.
   - If the column does not display an HTTP/3 value, follow the steps to enable TLS 1.3 for this domain before proceeding to the next section.

Offering HTTP/3 from your Fastly service

HTTP/3 is designed as an optional upgrade to end user client connections. This means that end user clients will initially connect to your Fastly service using an earlier version of HTTP and will only attempt to use HTTP/3 if the Fastly service offers it.

Use the **HTTP/3** switch to configure your service to offer HTTP/3:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **HTTP/3** switch to configure your service to offer HTTP/3.
6. Click the **Activate** button to deploy your configuration changes.

If you are configuring your service using VCL directly (either via regular VCL snippets or custom VCL methods), you can use the following VCL in the receive (`vcl_recv`) sub-routine to configure your service to offer HTTP/3:

```
h3.alt_svc();
```

Sending your HTTP/3 traffic to Fastly

Unless you use Fastly's dedicated IP addresses, then as a final step to enabling HTTP/3, you must ensure the DNS records of your domains are routing users to the correct HTTP/3 enabled Fastly addresses by using one of the following CNAMEs:

0RTT enabled?	IPv4 and IPv6 dual stack routing	IPv4-only routing
Yes	`dualstack.n.sni.global.fastly.net`	`n.sni.global.fastly.net`
No	`dualstack.m.sni.global.fastly.net`	`m.sni.global.fastly.net`

Testing client compatibility with your Fastly service

If you want test a browser or other client for HTTP/3 support, Fastly has created a publicly available HTTP/3 test page to verify client support. Using the client, navigate to https://http3.is and the resulting page will let you know whether or not HTTP/3 was successfully used to request the page. If the request did not use HTTP/3, the page will also let you know which IETF draft versions are currently being used by Fastly. You can use that information to update your client or client configurations to use one of those supported versions.

Keep in mind that even if correctly configured, the first request from a browser (or its first request after a time-out period set by the browser's developers) to any HTTP/3 enabled web site will always use a lower version of HTTP because it has not yet seen the HTTP/3 offer. If you don't see the HTTP/3 success page on your first request, be sure to perform a reload in the browser to give it an opportunity to use HTTP/3 for subsequent requests.

Serving HTTP/3 traffic

Once you've configured your services appropriately, requests made to the domains on those services should be capable of supporting HTTP/3 if they are being made from a client that supports these protocols.

If the QUIC connections and HTTP/3 requests are successful, the clients will continue using them for all subsequent requests and connections for those domains. Should problems occur with the QUIC connections or HTTP/3 requests, clients are expected to automatically fall back to a standard HTTP/1.1 or HTTP/2 connection over TCP. You should verify that your chosen client implements this fallback if this is a priority for you.

Monitoring your HTTP/3 traffic

You can monitor HTTP/3 requests using Fastly's Real-Time Log Streaming feature and the Historic stats. We have also added a number of VCL variables specifically related to HTTP/3 and QUIC.

Disabling HTTP/3

You can disable HTTP/3 support on a Fastly service by either activating a previous service version where HTTP/3 is not enabled or by creating a new service configuration version and disabling the HTTP/3 switch before activation.

Failure modes with large files

Last updated: 2020-12-18

https://docs.fastly.com/en/guides/failure-modes-with-large-objects

If you haven't enabled Segmented Caching, you may encounter the following failure modes when working with large objects.

TIP

Large objects can cause inefficiencies in content distribution networks. The Segmented Caching feature can help resolve these large object inefficiencies. Fastly recommends enabling Segmented Caching on services that will be serving large objects. Without Segmented Caching enabled, object size limits for your account depend on when you become a Fastly customer:

- If you created your account on or after June 17, 2020 and haven't enabled Segmented Caching, your Fastly services have a maximum object size of 20 MB.
- If you created your account prior to June 17, 2020 and haven't enabled Segmented Caching, your Fastly services have a maximum cacheable object size of 2 GB for requests without Streaming Miss or 5 GB for requests with Streaming Miss.

Maximum object size limits

If the response from the origin has a `Content-Length` header field that exceeds the maximum object size, Fastly will generate a `503 Response object too large` response to the client. If no `Content-Length` header field is returned, Fastly will start to fetch the response body. If, while fetching the response body, we determine that the object exceeds the maximum object size, we will generate a status `503 Response object too large` response to the client.

If no `Content-Length` header field is present and the Streaming Miss feature is enabled, Fastly will stream the content back to the client. However, if while streaming the response body Fastly determines that the object exceeds the maximum cacheable object size, it will terminate the client connection abruptly. The client will detect a protocol violation, because it will see its connection close without a properly terminating 0-length chunk.

Origin read failures

If reading the response body from the origin fails or times out, the problem will be reported differently depending on whether or not you've enabled Streaming Miss to act when the object is fetched. Without Streaming Miss, a 503 response will be generated. With Streaming Miss, however, it is already too late to send an error response since the header will already have been sent. In this case, Fastly will abruptly terminate the client connection and the client will detect a protocol violation. If the response was chunked, the client will see its connection close without a properly terminating 0-length chunk. If Content-Length was known, the client will see the connection close before the number of bytes given.

HTTP/2 server push

Last updated: 2021-12-15

https://docs.fastly.com/en/guides/http2-server-push

HTTP/2 server push allows you to set up rules that enable Fastly to pre-emptively load and then send responses to an HTTP/2-compliant client before that client requests them. You can initiate an HTTP/2 server push via a response header or VCL function.

Server push with the `link` response header

Fastly recognizes `link` headers with the preload keyword sent by an origin server and pushes the designated resource to a client. For example, this `link` response header triggers an HTTP/2 push:

```
link: </assets/jquery.js>; rel=preload; as=script
```

We support multiple `link` headers and multiple assets in one `link` header:

```
link: </assets/jquery.js>; rel=preload; as=script, </assets/base.css>; rel=preload; as=style
```

Additional attributes used in the `link` header can further control server push and how the header itself is handled. If no additional attributes are included, the `link` header will trigger server push and be forwarded to the client:

```
link: </assets/jquery.js>; rel=preload; as=script
```

If used with the `nopush` directive, the header will *not* trigger a push and will be passed as is to the client:

```
link: </assets/jquery.js>; rel=preload; as=script; nopush
```

If used with the `x-http2-push-only` directive, the header will trigger a server push but will be subsequently removed and not forwarded to the client:

```
link: </assets/jquery.js>; rel=preload; as=script; x-http2-push-only
```

The attributes can be mixed and matched if needed:

```
link: </assets/jquery.js>; rel=preload; as=script, </assets/base.css>; rel=preload; as=style; nopush, </assets/main.css>; re
l=preload; as=style; x-http2-push-only
```

Link headers and Amazon S3 buckets

If you're using an Amazon Simple Storage Service (S3) bucket as your origin server, you can still use `link` headers by applying a cache setting condition like this one:

```
set beresp.http.Link = beresp.http.x-amz-meta-Link
```

Server push with the `h2.push()` function

Server push can also be triggered with the `h2.push()` VCL function. The asset to be pushed is passed to the function as a parameter. For example:

```
sub vcl_recv {
#FASTLY recv

  if (fastly_info.is_h2 && req.url ~ "^/index.html")
  {
    h2.push("/assets/jquery.js");
  }
}
```

The `h2.push()` function triggers server push as soon as it's called, which removes the need for a `link` header to arrive with a server response. This means assets can be pushed to the client before the response for the request that triggered the push is received from the server, accelerating their delivery.

Making query strings agnostic

Last updated: 2018-08-01

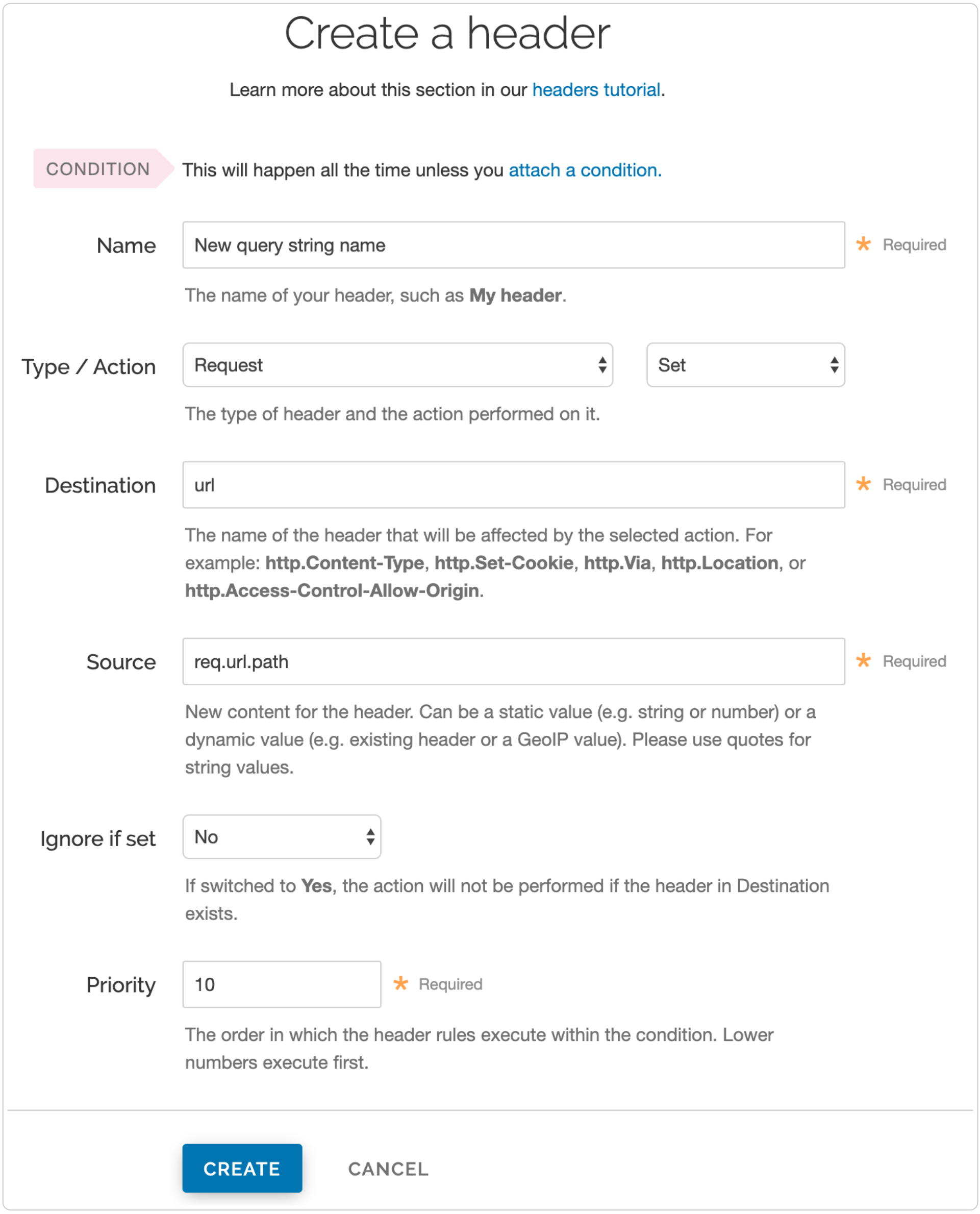
https://docs.fastly.com/en/guides/making-query-strings-agnostic

Under normal circumstances, Fastly would consider these URLs different objects that are cached separately:

- http://example.com
- http://example.com?asdf=asdf
- http://example.com?asdf=zxcv

It is possible, however, to have them all ignore the query string and return the same cached resource.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a description for the header (e.g., `New query string name`).
   - From the **Type** menu, select **Request**, and from **Action** menu, select **Set**.
   - In the **Destination** field, enter `url`.
   - In the **Source** field, enter `req.url.path`.
   - From the **Ignore if set** menu, select **No**.
   - Set the **Priority** field to whatever priority you want.

7. Click the **Create** button to create the new header. The new header you created appears on the Content page.

8. Click the **Activate** button to deploy your configuration changes.

The request will be sent to the origin as a URL without the query string.

For more information about controlling caching, see our documentation on cache freshness.

Precision Path

Last updated: 2021-09-17

https://docs.fastly.com/en/guides/precision-path

Fastly can automatically detect and, in real time, route around transient connection problems that occur when fetching content from your origin servers or when delivering content to end users from Fastly's Edge Cloud. Fastly uses different, automated techniques depending on where we observe a connection problem. This page describes two capabilities that Precision Path uses to automatically improve fetching and delivery of content to your users: origin connection rerouting and edge connection rerouting.

Origin connection rerouting

It's critical for Fastly to maintain high performance and reliable access to your origin services to fetch new or updated content when needed. The inability to do so, even temporarily, can result in 5xx HTTP server errors being returned to end users, regardless of how resilient and performant connections may be. Based on historical observations of our worldwide network, we estimate a significant portion of 5xx errors served to end users are due to the impact of transient internet performance issues (commonly called *internet weather*) on the connections between Fastly's point of presence (POP) locations and our customers' origin services. Origin rerouting is designed to address this.

How origin connection rerouting works

Our origin connection rerouting capability monitors Fastly's connections to your origin services for signs of internet weather and, if detected, automatically attempts to route around them so they don't impact your services. The vast majority of the time you won't notice changes to those connections in your Fastly services because there will be no issues to route around. However, when the default route from one of our POPs across the internet to your origins experiences a severe enough problem, it will automatically test other available routes to your origin and, if any successful alternatives are found, will select the most viable alternative route with which to re-establish the connection. All of this will happen before TCP timeouts occur on the end user connection, thereby avoiding 5xx errors being returned to end users.

Requirements

Precision Path's origin connection rerouting only works for origin connections using TLS. If your Fastly service has not been configured to use a TLS connection between Fastly and your origin server, you can enable TLS by following the instructions in our guide on working with hosts.

No special requirements are necessary to take advantage of Fastly's origin connection rerouting. This capability is enabled by default for all Fastly services using TLS for origin and shielding connections.

Many Fastly customers protect their origin services from security threats by implementing either a firewall service or IP address access control lists (ACLs). While this is a good security practice, you could be inadvertently blocking a subset of Fastly IP addresses from accessing your origins if these configurations aren't updated regularly.

To successfully implement origin connection rerouting on your Fastly services, ensure you aren't blocking any Fastly IP addresses. Fastly provides an API for you to obtain a complete list of the IPv4 and IPv6 address ranges owned by Fastly. Any firewalls or ACLs protecting your origin services should be updated to ensure all these IP address ranges are allowed to connect to your origins.

Monitoring

Fastly's real-time log streaming feature can be used to monitor for origin connection rerouting events that may occur on your Fastly services. The following VCL variables can be used to monitor origin connection rerouting activity:

- `beresp.used_alternate_path_to_origin` - This boolean value indicates whether or not the request to origin was made over an alternate route selected by the origin connection rerouting mechanism. Counting the number of true values for this variable over any given time period in your logs will indicate how many times this origin rerouting mechanism has been triggered.

- `beresp.backend.src_ip` - This variable indicates which Fastly source IP was used to make the request to origin. For most connections, this will be one of the most frequently used Fastly server IP addresses used for default routes across the Internet. If the origin rerouting mechanism was triggered by a problem with the original connection attempt over the default route, then this value will show an alternate Fastly server IP address from a pool of IPs reserved for alternate routes between that POP and your origin.

- `beresp.backend.alternate_ips` - This variable lists all the possible source IP addresses available to the origin rerouting mechanism for the Fastly server handling this origin connection. This information can be used to identify the set of valid Fastly IP addresses from which you may see connection attempts to your origin from this Fastly server.

Edge connection rerouting

When delivering content from Fastly to your end users, Fastly tracks the health of every TCP connection. When we observe connection-impacting degradation (e.g., congestion), we automatically switch delivery to a new TCP connection to route around the issue. This automatic mitigation is enabled by default on all of our POPs and applies to all Fastly traffic. No additional configuration is required.

No special requirements are necessary to take advantage of Fastly's edge connection rerouting. All traffic delivered from Fastly's POPs is assessed for connection issues and the fast path failover feature is applied only when necessary.

Support

For more information on Precision Path or its individual features, contact support@fastly.com.

Serving stale content

Last updated: 2021-04-21

https://docs.fastly.com/en/guides/serving-stale-content

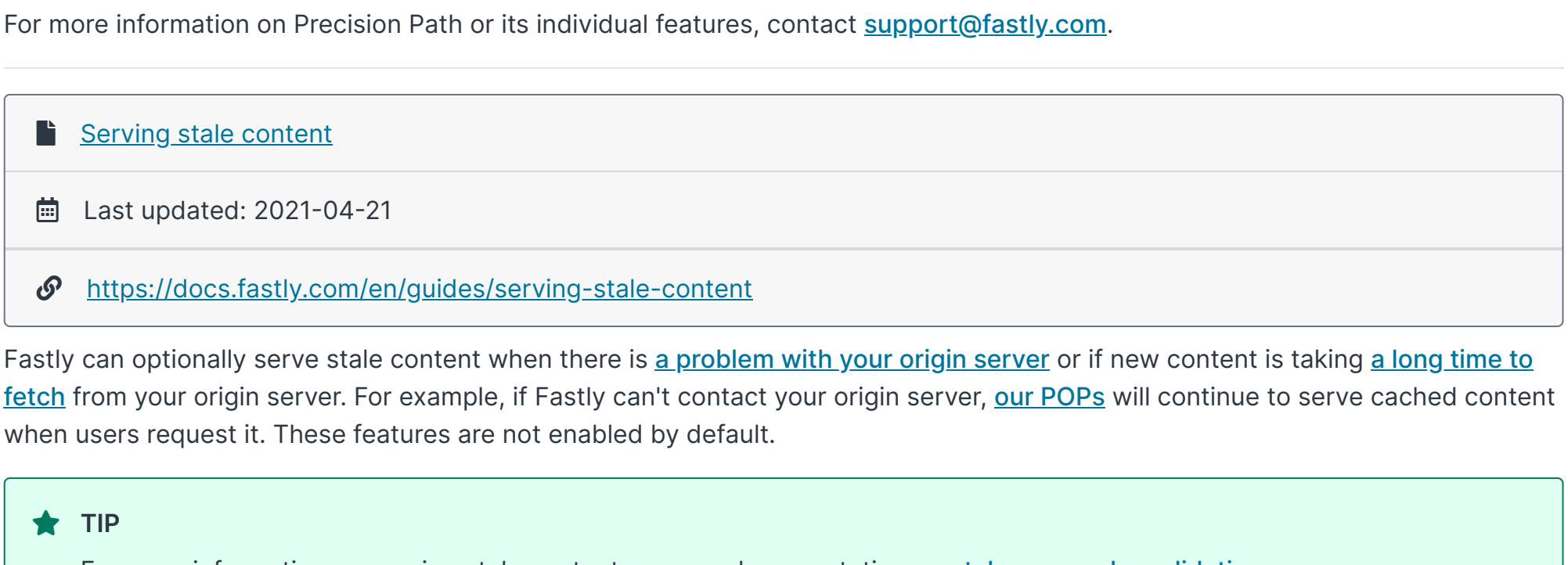
Fastly can optionally serve stale content when there is a problem with your origin server or if new content is taking a long time to fetch from your origin server. For example, if Fastly can't contact your origin server, our POPs will continue to serve cached content when users request it. These features are not enabled by default.

TIP

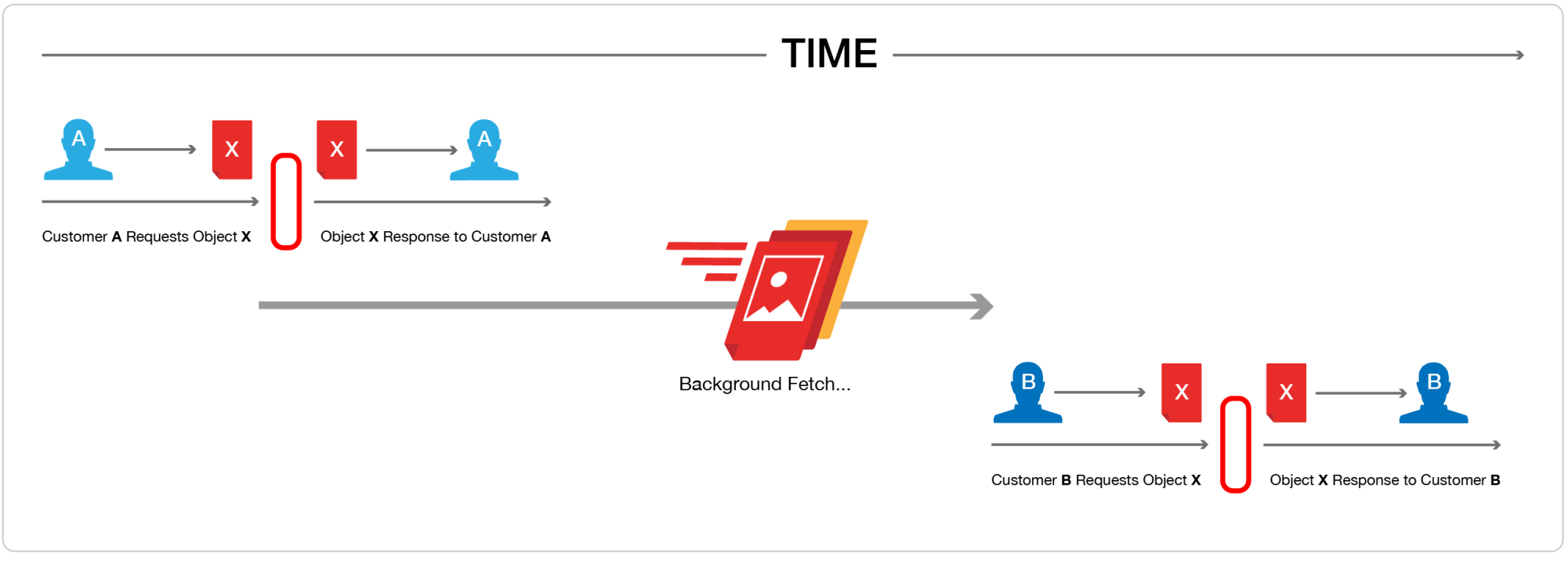
For more information on serving stale content, see our documentation on staleness and revalidation.

Serving old content while fetching new content

Certain pieces of content can take a long time to generate. Once the content is cached it will be served quickly, but the first user to try and access it will pay a penalty.



This is unavoidable if the cache is completely cold, but if this is happening when the object is in cache and its TTL is expired, then Fastly can be configured to show the stale content while the new content is fetched in the background.



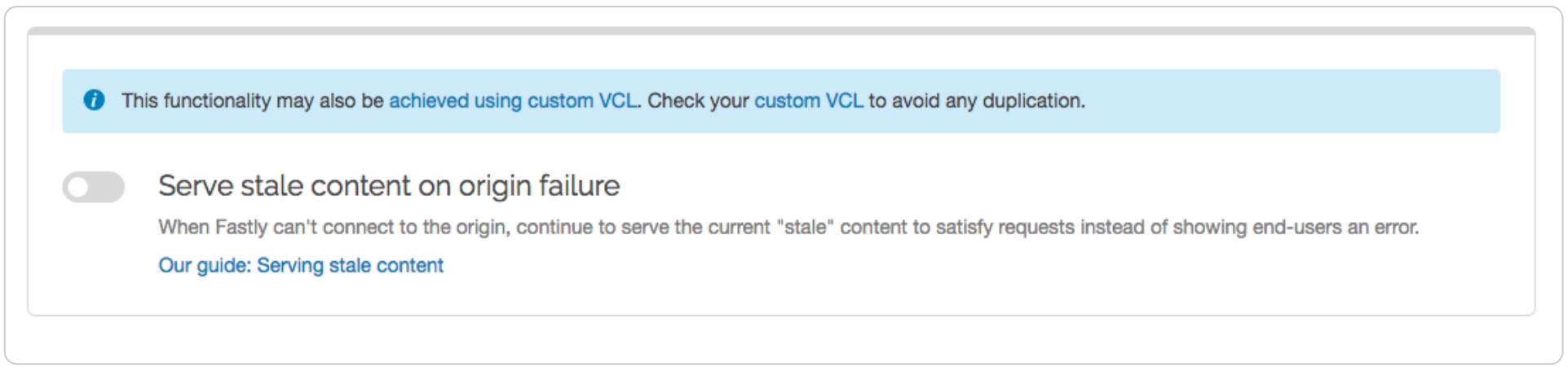
Fastly builds on the behavior proposed in RFC 5861 "HTTP Cache-Control Extensions for Stale Content" by Mark Nottingham, which is under consideration for inclusion in Google's Chrome browser.

Enabling serve stale

> **NOTE**
>
> If you've already enabled serving stale content on an error via custom VCL, adding this feature via the web interface will set a different stale TTL. To avoid this, check your custom VCL and remove the `stale-if-error` statement before enabling this feature via the web interface.

To enable serving stale content on an error via the web interface for the default TTL period (43200 seconds or 12 hours), follow the steps below.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.



5. Click the **Serve stale** switch to automatically enable serving stale content for the default TTL period of 43200 seconds (12 hours).
6. Click the **Activate** button to deploy your configuration changes.

Manually enabling serve stale

You can manually enable serving stale content by adding a `stale-while-revalidate` or `stale-if-error` directive to either the Cache-Control or Surrogate-Control headers in the response from your origin server. For example:

```
Cache-Control: max-age=600, stale-while-revalidate=30
```

will cache some content for 10 minutes and, at the end of that 10 minutes, will serve stale content for up to 30 seconds while new content is being fetched.

Similarly, this statement:

```
Surrogate-Control: max-age=3600, stale-if-error=86400
```

instructs the cache to update the content every hour (3600 seconds) but if the origin is down then show stale content for a day (86400 seconds).

Alternatively, these behaviors can be controlled from within VCL by setting the following variables in vcl_fetch:

```
set beresp.stale_while_revalidate = 30s;
```

```
set beresp.stale_if_error = 86400s;
```

Interaction with grace

Stale-if-error works exactly the same as Varnish's grace variable such that these two statements are equivalent:

```
set beresp.grace = 86400s;
```

```
set beresp.stale_if_error = 86400s;
```

However, if a grace statement is present in VCL it will override any `stale-if-error` statements in any Cache-Control or Surrogate-Control response headers.

Setting `beresp.stale_if_error` either via header or via VCL does nothing on its own. In order to serve stale, follow the instructions below.

Serving stale content on errors

In certain situations where your origin server becomes unavailable, you may want to serve stale content. These instructions provide an advanced configuration that allows all three possible origin failure cases to be handled using VCL.

In the context of Varnish, there are three ways an origin can fail:

- The origin can be marked as unhealthy by failing health checks.
- If Varnish cannot contact the origin for any reason, a 503 error will be generated.
- The origin returns a valid HTTP response, but that response is not one we wish to serve to users (for instance, a 503).

The custom VCL shown below handles all three cases. If the origin is unhealthy, the default serve stale behavior is triggered by `stale-if-error`. In between the origin failing and being marked unhealthy, Varnish would normally return 503s. The custom VCL allows us to instead either serve stale if we have a stale copy, or to return a synthetic error page. The error page can be customized. The third case is handled by intercepting all 5XX errors in `vcl_fetch` and either serving stale or serving the synthetic error page.

> **WARNING**
>
> Do not purge all cached content if you are seeing 503 errors. Purge all overrides stale-if-error and increases the requests to your origin server, which could result in additional 503 errors.

Although not strictly necessary, health checks should be enabled in conjunction with this VCL. Without health checks enabled, all of the functionality will still work, but serving stale or synthetic responses will take much longer while waiting for an origin to timeout. With health checks enabled, this problem is averted by the origin being marked as unhealthy.

The custom VCL shown below includes the Fastly standard boilerplate. Before uploading this to your service, be sure to customize or remove the following values to suit your specific needs:

- `if (beresp.status >= 500 && beresp.status < 600)` should be changed to include any HTTP response codes you wish to serve stale/synthetic for.
- `set beresp.stale_if_error = 86400s;` controls how long content will be eligible to be served stale and should be set to a meaningful amount for your configuration. If you're sending `stale_if_error` in Surrogate-Control or Cache-Control from origin, remove this entire line.
- `set beresp.stale_while_revalidate = 60s;` controls how long the `stale_while_revalidate` feature will be enabled for an object and should be set to a meaningful amount for your configuration. This feature causes Varnish to serve stale on a cache miss and fetch the newest version of the object from origin in the background. This can result in large performance gains on objects with short TTLs, and in general on any cache miss. Note that `stale_while_revalidate` overrides `stale_if_error`.

That is, as long as the object is eligible to be served stale while revalidating, `stale_if_error` will have no effect. If you're sending `stale_while_revalidate` in Surrogate-Control or Cache-Control from origin, remove this entire line.

- `synthetic {"<!DOCTYPE html>Your HTML!</html>"};` is the synthetic response Varnish will return if no stale version of an object is available and should be set appropriately for your configuration. You can embed your HTML, CSS, or JS here. Use caution when referencing external CSS and JS documents. If your origin is offline they may be unavailable as well.

```
sub vcl_recv {
  /* if shielding is enabled, the below code is required */
  if (fastly.ff.visits_this_service != 0) {
    set req.max_stale_while_revalidate = 0s;
  }

#FASTLY recv

  if (req.method != "HEAD" && req.method != "GET" && req.method != "FASTLYPURGE") {
    return(pass);
  }

  return(lookup);
}

sub vcl_fetch {
  /* handle 5XX (or any other unwanted status code) */
  if (beresp.status >= 500 && beresp.status < 600) {

    /* deliver stale if the object is available */
    if (stale.exists) {
      return(deliver_stale);
    }

    if (req.restarts < 1 && (req.method == "GET" || req.method == "HEAD")) {
      restart;
    }

    /* else go to vcl_error to deliver a synthetic */
    error beresp.status;
  }

  /* set stale_if_error and stale_while_revalidate (customize these values) */
  set beresp.stale_if_error = 86400s;
  set beresp.stale_while_revalidate = 60s;

#FASTLY fetch

  if ((beresp.status == 500 || beresp.status == 503) && req.restarts < 1 && (req.method == "GET" || req.method == "HEAD"
)) {
    restart;
  }

  if (req.restarts > 0) {
    set beresp.http.Fastly-Restarts = req.restarts;
  }

  if (beresp.http.Set-Cookie) {
    set req.http.Fastly-Cachetype = "SETCOOKIE";
    return(pass);
  }

  if (beresp.http.Cache-Control ~ "private") {
    set req.http.Fastly-Cachetype = "PRIVATE";
    return(pass);
  }

  /* this code will never be run, commented out for clarity */
  /* if (beresp.status == 500 || beresp.status == 503) {
     set req.http.Fastly-Cachetype = "ERROR";
     set beresp.ttl = 1s;
     set beresp.grace = 5s;
     return(deliver);
  } */

  if (beresp.http.Expires || beresp.http.Surrogate-Control ~ "max-age" || beresp.http.Cache-Control ~ "(s-maxage|max-ag
e)") {
    # keep the ttl here
  } else {
    # apply the default ttl
    set beresp.ttl = 3600s;
  }

  return(deliver);
}

sub vcl_hit {
#FASTLY hit

  if (!obj.cacheable) {
```

```
    return(pass);
  }
  return(deliver);
}

sub vcl_miss {
#FASTLY miss
  return(fetch);
}

sub vcl_deliver {

#FASTLY deliver
  return(deliver);
}

sub vcl_error {
#FASTLY error

  /* handle 503s */
  if (obj.status >= 500 && obj.status < 600) {

    /* deliver stale object if it is available */
    if (stale.exists) {
      return(deliver_stale);
    }

    /* otherwise, return a synthetic */

    /* include your HTML response here */
    synthetic {"<!DOCTYPE html><html>Replace this text with the error page you would like to serve to clients if your or
igin is offline.</html>"};
    return(deliver);
  }

}

sub vcl_pass {
#FASTLY pass
}

sub vcl_log {
#FASTLY log
}
```

Why serving stale content may not work as expected

Here are some things to consider if Fastly isn't serving stale content:

- **Cache:** Stale objects are only available for cacheable content.
- **VCL:** Setting `req.hash_always_miss` or `req.hash_ignore_busy` variable to `true` invalidates the effect of `stale-while-revalidate`.
- **Shielding:** If you don't have shielding enabled, a POP can only serve stale on errors if a request for that cacheable object was made through that POP before. We recommend enabling shielding to increase the probability that stale content on error exists. Shielding is also a good way to quickly refill the cache after performing a purge all.

- **Requests:** As traffic to your site increases, you're more likely to see stale objects available (even if shielding is disabled). It's reasonable to assume that popular assets will be cached at multiple POPs.
- **Least Recently Used (LRU):** Fastly has an LRU list, so objects are not necessarily guaranteed to stay in cache for the entirety of their TTL (time to live). But eviction is dependent on many factors, including the object's request frequency, its TTL, the POP from which it's being served. For instance, objects with a TTL of 3700s or longer get written to disk, whereas objects with shorter TTLs end up in transient, in-memory-only storage. We recommend setting your TTL to more than 3700s when possible.
- **Purges:** Whenever possible, you should purge content using our soft purge feature. Soft purge allows you to easily mark content as outdated (stale) instead of permanently deleting it from Fastly's caches. If you can't use soft purge, we recommend purging by URL or using surrogate keys instead of performing a purge all.

Streaming Miss

Last updated: 2021-12-16

https://docs.fastly.com/en/guides/streaming-miss

When fetching an object from the origin, the Streaming Miss feature ensures the response is streamed back to the client immediately and is written to cache only after the whole object has been fetched. This reduces the first-byte latency (the time that the client must wait before it starts receiving the response body).

TIP

Fastly recommends enabling Segmented Caching on services that will be serving large resources. Without Segmented Caching enabled, the resource size limits for your account depend on when you become a Fastly customer:

- If you created your account on or after June 17, 2020 and haven't enabled Segmented Caching, your Fastly services have a maximum object size of 20 MB.
- If you created your account prior to June 17, 2020 and haven't enabled Segmented Caching, your Fastly services have a maximum cacheable object size of 2 GB for requests without Streaming Miss or 5 GB for requests with Streaming Miss.

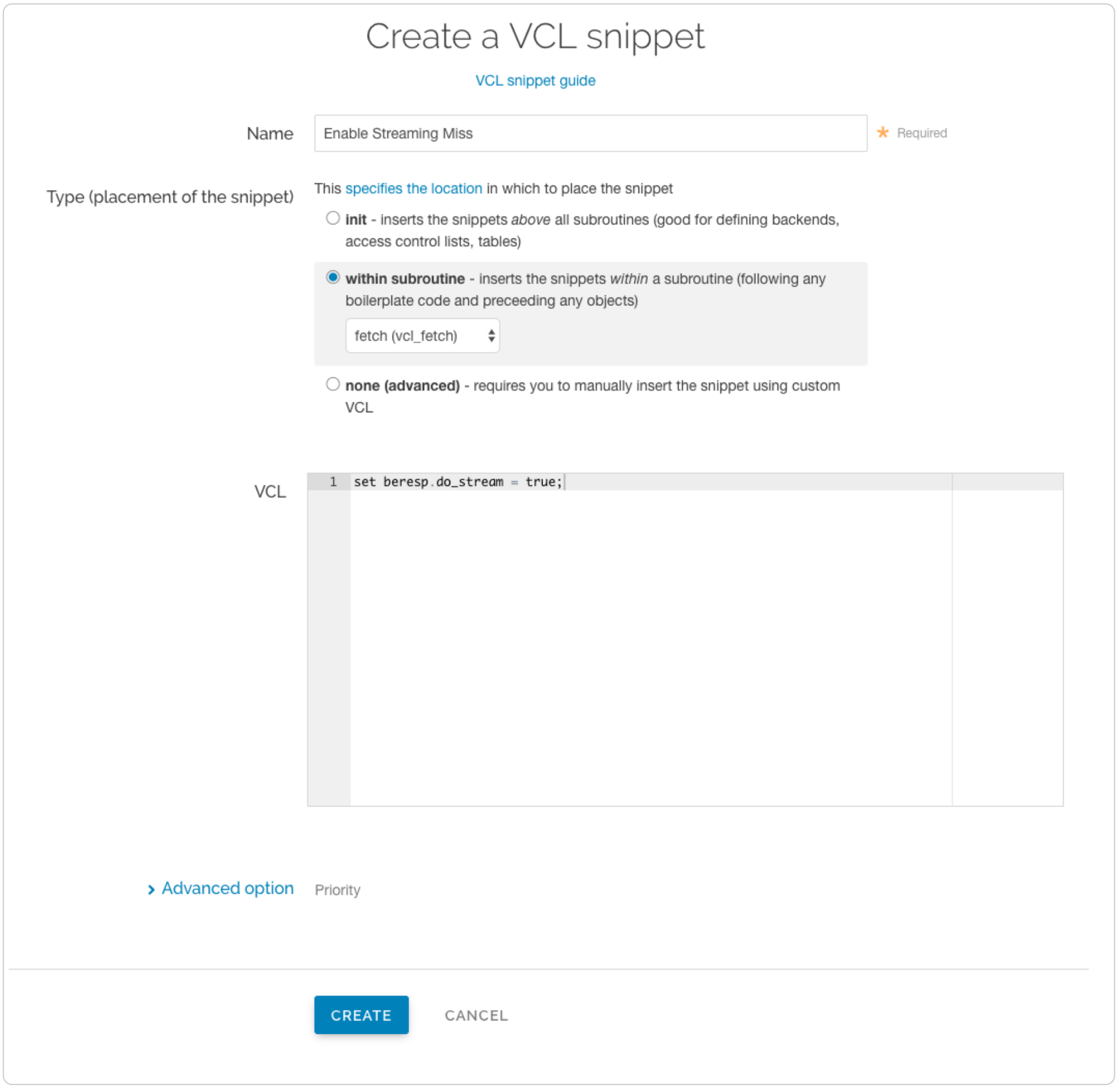
NOTE

If you enable Streaming Miss, be aware that if an error occurs while transferring the response body, Fastly cannot send an error because the headers are already sent to the client. All we can do is truncate the response.

Enable Streaming Miss by creating a VCL Snippet

Enable Streaming Miss by setting `beresp.do_stream` to `true` in `vcl_fetch` with a VCL Snippet using these steps.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the **Create your first VCL snippet**. The Create a VCL snippet page appears.

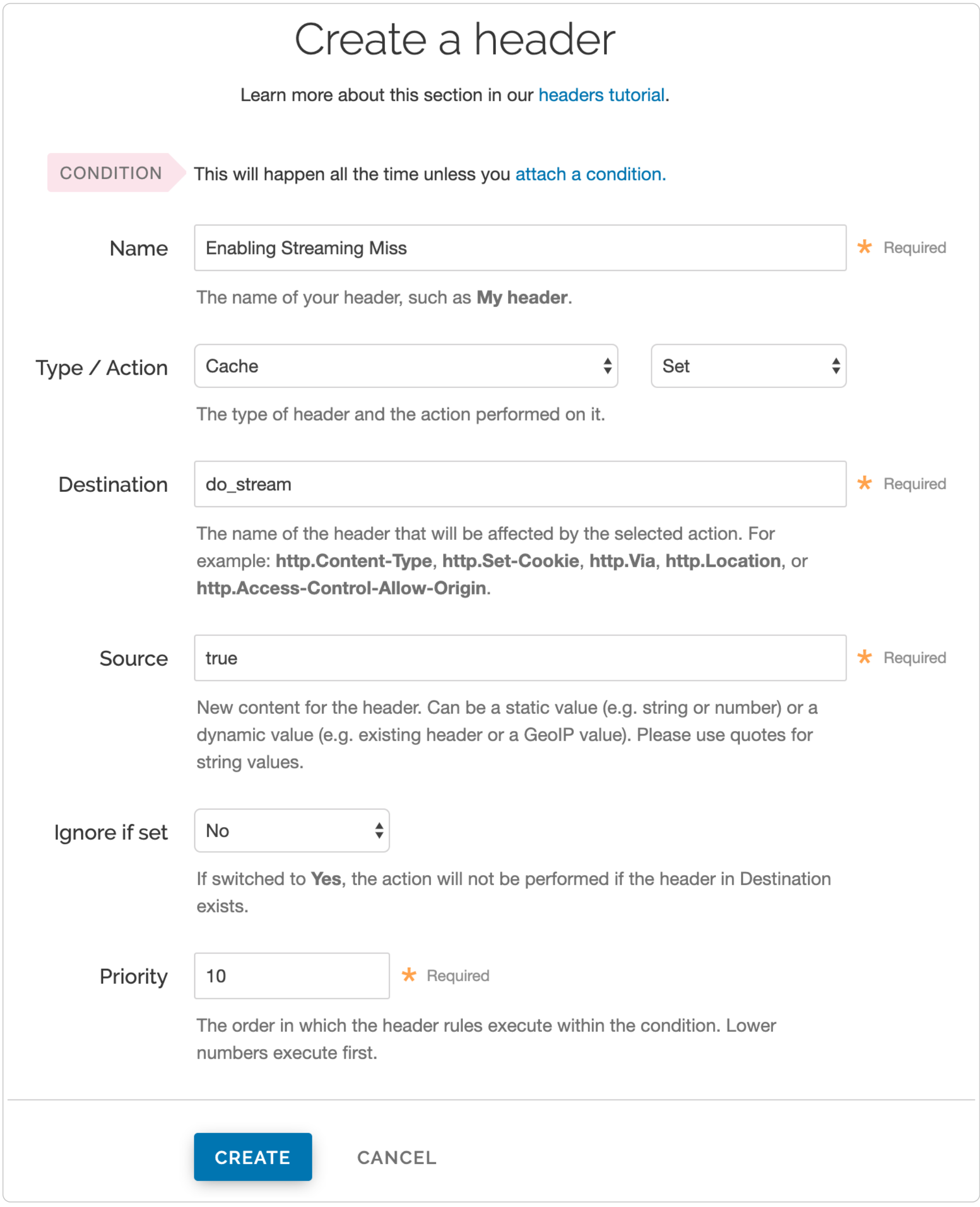


6. In the **Name** field enter an appropriate name (e.g., `Enabling Streaming Miss`).
7. From the **Type (placement of the snippets)** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **fetch (vcl_fetch)**.
9. In the **VCL** field, add `set beresp.do_stream = true;`.
10. Click **Create** to create the snippet.
11. Click the **Activate** button to deploy your configuration changes.

Enable Streaming Miss by creating a Fastly header

Enable Streaming Miss by setting `beresp.do_stream` to `true` in `vcl_fetch` with a Fastly header using these steps. .

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



6. Fill out the Create a header fields as follows:
   - In the **Name** field, enter the name of your header rule (for example, `Enabling Streaming Miss`).
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `do_stream`.
   - In the **Source** field, enter `true`.
   - From the **Ignore if set** menu, select **No** if you want the header in the **Destination** field modified or select **Yes** if you don't want it modified.
   - In the **Priority** field, enter the order the header rules execute.

7. Click the **Create** button.

8. Click the **Activate** button to deploy your configuration changes.

Streaming Miss limitations

There are several limitations to using Streaming Miss.

Streaming Miss is not available to HTTP/1.0 clients

If an HTTP/1.0 request triggers a fetch and the response header from the origin does not contain a Content-Length field, then Streaming Miss will be disabled for the fetch and the fetched object will be subject to the non-streaming-miss object size limit of 2GB. Without the client receiving the Content-Length, the client cannot distinguish the proper end of the download from an abrupt connection breakage anywhere upstream from it.

If an HTTP/1.0 request is received while a Streaming Miss for an object is in progress, the HTTP/1.0 request will wait for the response body to be downloaded before it will receive the response header and the response body, as if the object was being fetched without Streaming Miss.

Cache hits are not affected. An HTTP/1.0 client can receive a large object served from cache, just like an HTTP/1.1 client.

Streaming Miss is not compatible with on-the-fly gzip compression of fetched objects

Streaming Miss can handle large files whether or not they are compressed. On-the-fly compression of objects not already compressed is not compatible with Streaming Miss. Specifically, if VCL sets `beresp.gzip` to true, Streaming Miss will be disabled.

Streaming Miss is not compatible with ESI (Edge-Side Includes)

Responses processed through ESI, which dynamically inserts content into cached pages, cannot be streamed. Responses included from an ESI template also cannot be streamed. When ESI is enabled for a response or when a response is fetched using `<esi:include>`, then Streaming Miss will be disabled and the fetched object will be subject to the non-streaming-miss object size limit of 2GB.

---

§

These articles describe how to purge cache.

https://docs.fastly.com/en/guides/configuration#_purging

Authenticating URL purge requests via API

Last updated: 2021-12-20

https://docs.fastly.com/en/guides/authenticating-api-purge-requests

Fastly's URL purge feature allows you to purge individual URLs on your website. By default, authentication is not required to purge a URL with the Fastly API, but you can enable API token authentication in the Fastly web interface by adding a header or by using custom VCL.

WARNING

We recommend that all customers enable authentication for URL purge requests.

NOTE

All purge requests other than URL purges require authentication by default, as indicated in the API documentation.

Limitations and considerations

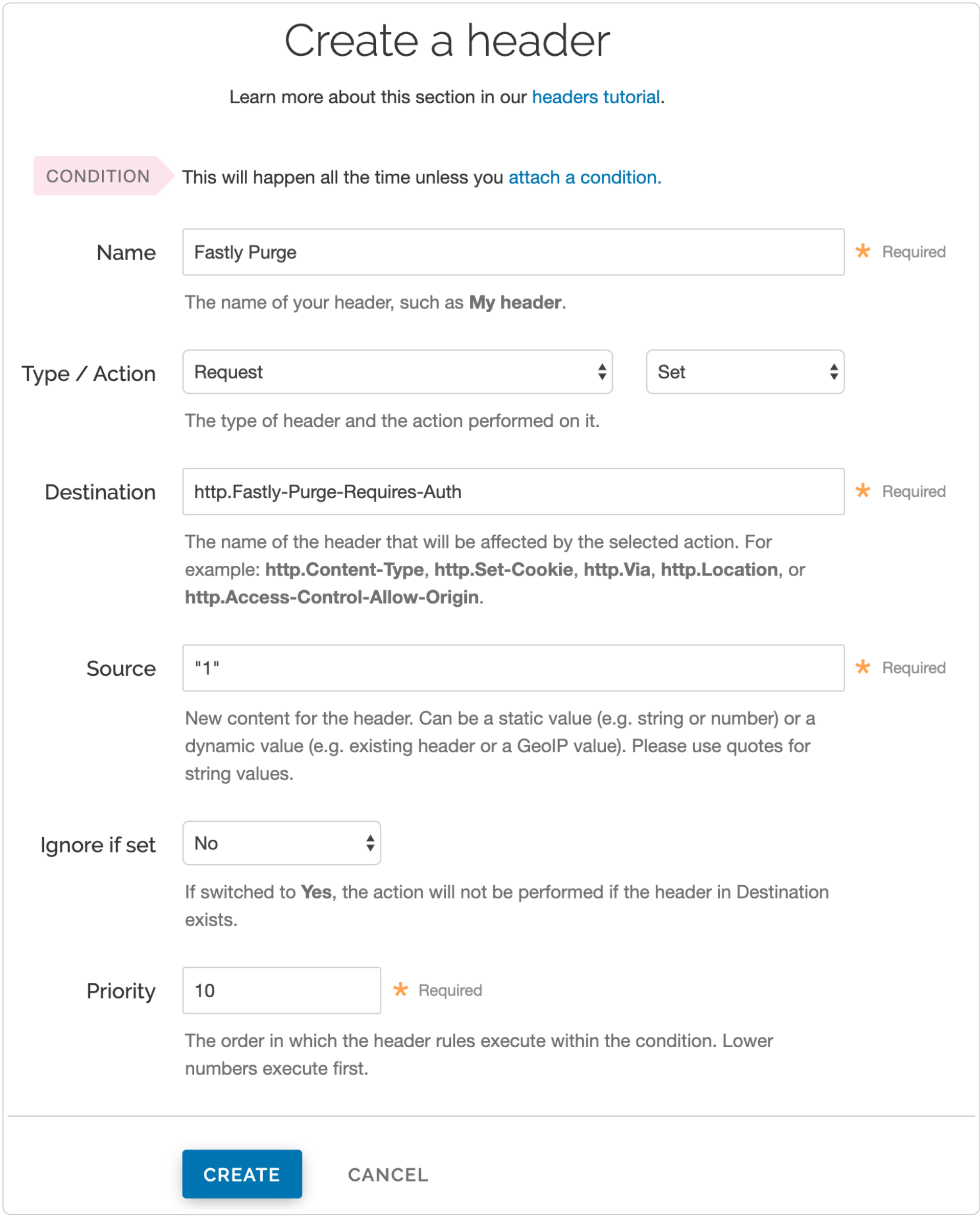
The solution outlined in this guide must be implemented on every service that requires authentication of URL purge requests. To enable purge authentication at the account level, email support@fastly.com.

Enabling authentication in the Fastly web interface

You can enable API token authentication for URL purge requests by adding a header and optionally attaching a condition in the Fastly web interface.

Adding the header

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header window appears.



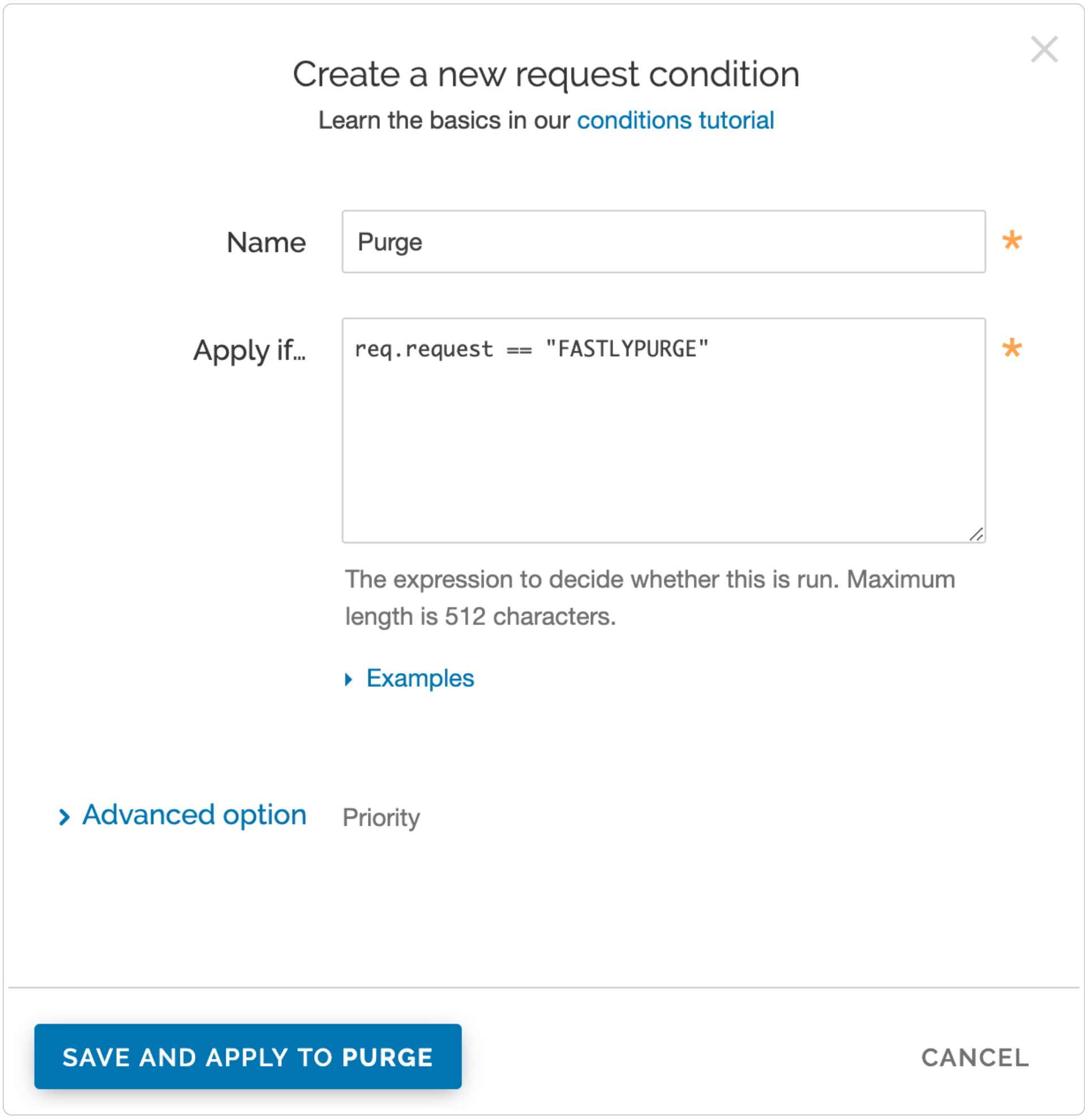
6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter the name of your header rule (for example, `Fastly Purge`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Fastly-Purge-Requires-Auth`.
   - In the **Source** field, enter `"1"`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.

7. Click the **Create** button.

Attaching a condition

Attaching the following condition is optional. Without the condition, the header you just created will be added to all requests. With the condition, the header will be added to purge requests only.

1. On the Content page, click the **Attach a condition** link to the right of your new header. The Create a new request condition window appears.



2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter a descriptive name for the new condition (for example, `Purge`).
   - In the **Apply if** field, enter `req.request == "FASTLYPURGE"`.
3. Click the **Save and apply to** button.
4. Click the **Activate** button to deploy your configuration changes.

Enabling authentication with VCL Snippets

You can also enable API token authentication for URL purge requests using VCL Snippets. Refer to the developer documentation for details on setting the `Fastly-Purge-Requires-Auth` header.

Purging URLs with an API token

After you've enabled API token authentication for URL purge requests, you'll need to provide your API token in the URL purge API request:

```
$ curl -X PURGE -H Fastly-Key:FASTLY_API_TOKEN https://www.example.com/
```

which would return this response:

```
{"status": "ok", "id": "1234567890"}
```

WARNING

If your website is not configured to use HTTPS, do not use the Fastly API to purge URLs. Doing so could expose your API token since the data in transit will not be encrypted.

Purging a URL via the web interface

Last updated: 2021-12-20

https://docs.fastly.com/en/guides/purging-a-url

Fastly provides several levels of cache purging. You can use the *Purge URL* option to purge a single URL via the web interface.

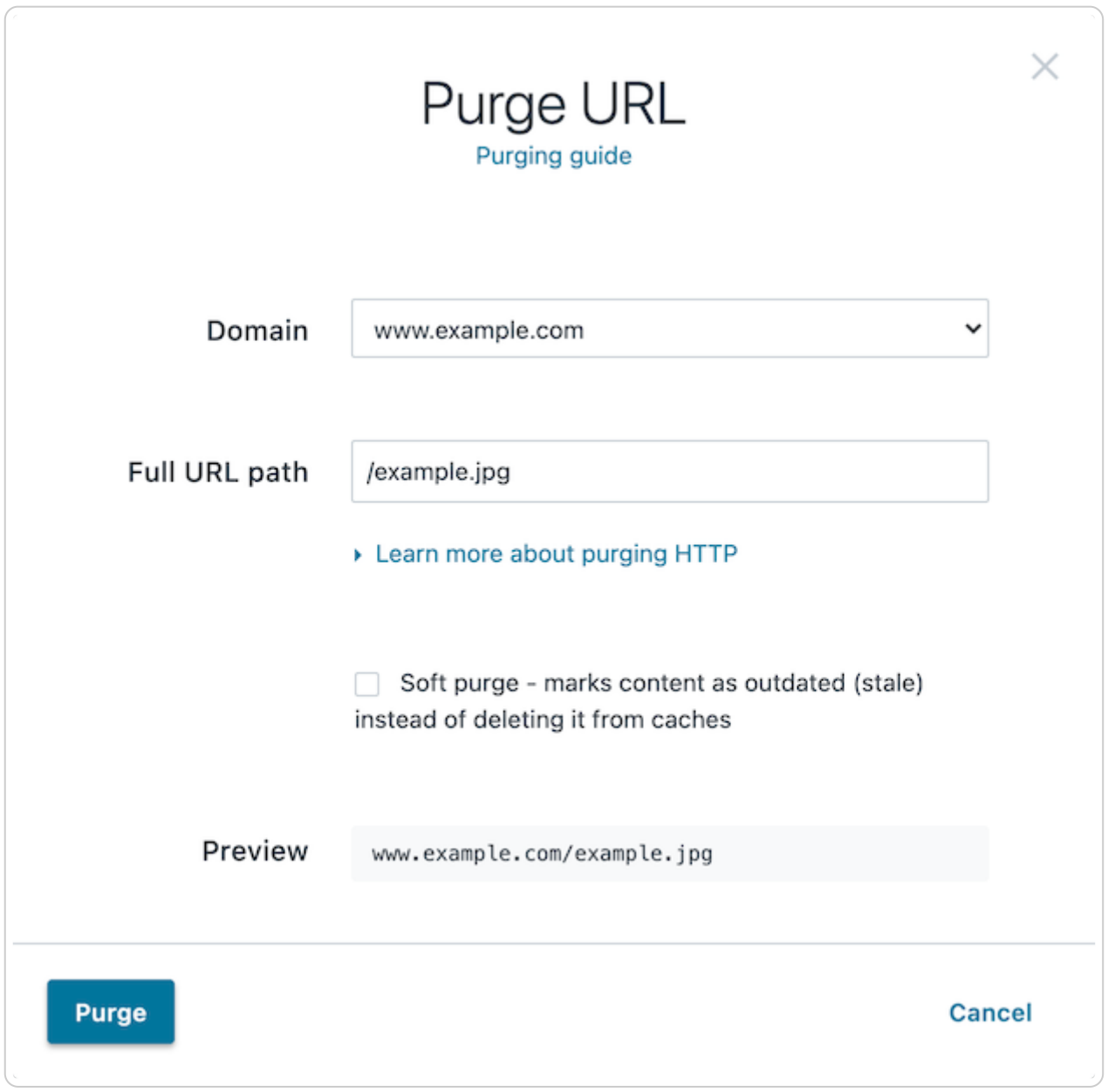
Before you begin

If you're new to the concept of purging, we recommend familiarizing yourself with our purging basics guide.

Purging a URL

To purge a single URL, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. From the **Purge** menu, select **Purge URL**. The Purge URL window appears.



4. From the **Domain** menu, select the domain on which your content resides. If the domain you select is a wildcard domain (e.g., `*.example.com`) the Subdomain field will appear.
5. If the **Subdomain** field appears, enter the subdomain to purge for the wildcard domain you've selected (e.g., `www`).
6. In the **Full URL path** field, enter the path to the content you'll be purging (e.g., `/example.jpg`). The Preview field displays the URL that will be purged.
7. Optionally select the Soft purge checkbox to mark your content as outdated instead of deleting it from cache.
8. Click the **Purge** button.

What's next

Explore the other purging methods available with Fastly such as purging with surrogate keys and purging all content.

Purging all content via the web interface

Last updated: 2021-12-06

https://docs.fastly.com/en/guides/purging-all-content

Fastly provides several levels of cache purging. You can use the *Purge all* option to purge all content under a service.

WARNING

Exercise caution when purging all content from cache, especially if you're seeing an increase in 503 errors. Purging all content overrides other caching-related settings that allow you to serve stale content and forces Fastly to retrieve that content again from your origin servers before it can be re-cached in Fastly POPs. This means that purging all content will temporarily cause your cache hit ratio to drop dramatically and cause an associated spike of traffic to your origin servers. Be certain your origins can handle that temporary spike in traffic until the cache repopulates.

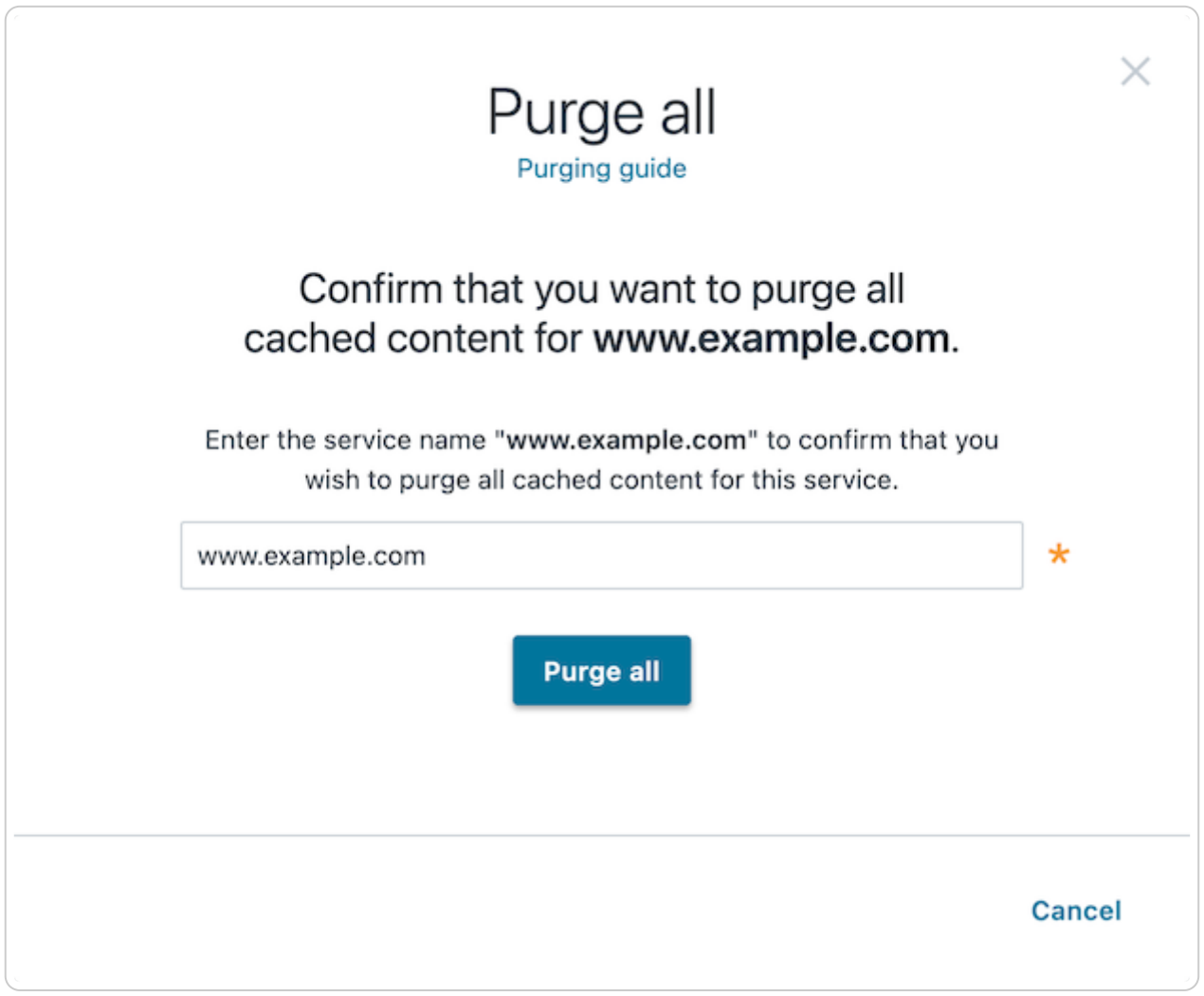
Before you begin

If you're new to the concept of purging, we recommend familiarizing yourself with our purging basics guide.

Purging all content

To instantly purge all content under your service, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. From the **Purge** menu, select **Purge all**. The Purge all window appears and displays the service name you'll be purging.



4. In the field of the **Purge all** window, confirm you want to purge all cached content on a service by entering the exact name of the service that appears (e.g., `An Example Service`).
5. Click the **Purge all** button.

What's next

Explore the other purging methods available with Fastly such as purging with surrogate keys and purging a URL.

Purging via API

Last updated: 2021-12-06

https://docs.fastly.com/en/guides/purging-via-api

The syntax for purging a service through the API can be found in the Purging section of the API documentation.

WARNING

We recommend that all customers enable authentication for URL purge requests via API.

When you purge using the API, the purge ID is returned in the response. For example:

```
{"status": "ok", "id": "1234567890"}
```

If the purge isn't working, send an email to support@fastly.com with the purge ID, the resource you're attempting to purge, and the type of purge (e.g., purge by surrogate key through the API).

What's next

Purging with surrogate keys

Last updated: 2021-12-06

https://docs.fastly.com/en/guides/purging-with-surrogate-keys

Fastly provides several levels of cache purging and choosing the right purging method is essential to keeping your website fast. While Fastly's purge all is a speedy way to invalidate your cache, it may temporarily increase your website's load time while the cache rebuilds. If you find yourself purging all cache on more than a weekly basis, consider using surrogate keys for more targeted purging.

Surrogate keys are unique identifiers that you assign to portions of content that share common traits, making them easier to process as groups in some way. Using the `Surrogate-Key` header, you can tag an object, such as an image or a blog post, with one or more keys. When you need to explicitly remove content from the Fastly edge cache, rather than allowing it to expire or to be evicted, you can use surrogate keys to purge groups of content selectively instead of purging all content.

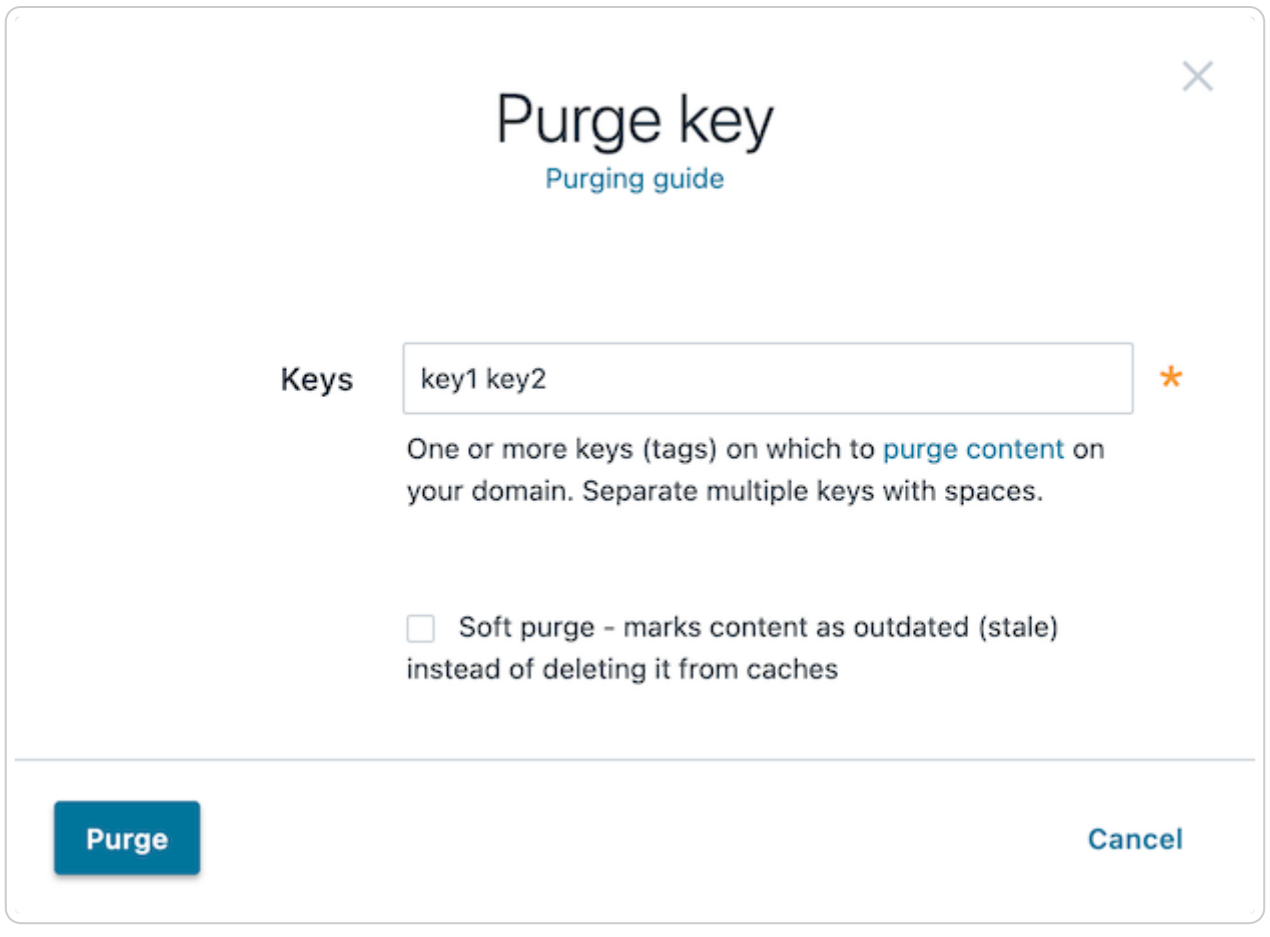
Before you begin

This guide assumes you're already familiar with how to work with the `Surrogate-Key` header to tag related pieces of cacheable content.

Purging objects with surrogate keys

You can use the Fastly web interface to manually purge objects via key:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. From the **Purge** menu, select **Purge key**. The Purge key window appears.



4. In the **Keys** field, enter one or more surrogate keys. Use spaces to separate multiple keys.
5. Optionally select the Soft purge checkbox to mark your content as outdated instead of deleting it from cache.
6. Click the **Purge** button.

You can also use our Purge API to purge objects via key.

What's next

Explore the other purging methods available with Fastly such as purging a URL and purging all content.

Setting Surrogate-Key headers for Amazon S3 origins

Last updated: 2018-08-16

https://docs.fastly.com/en/guides/setting-surrogate-key-headers-for-amazon-s3-origins

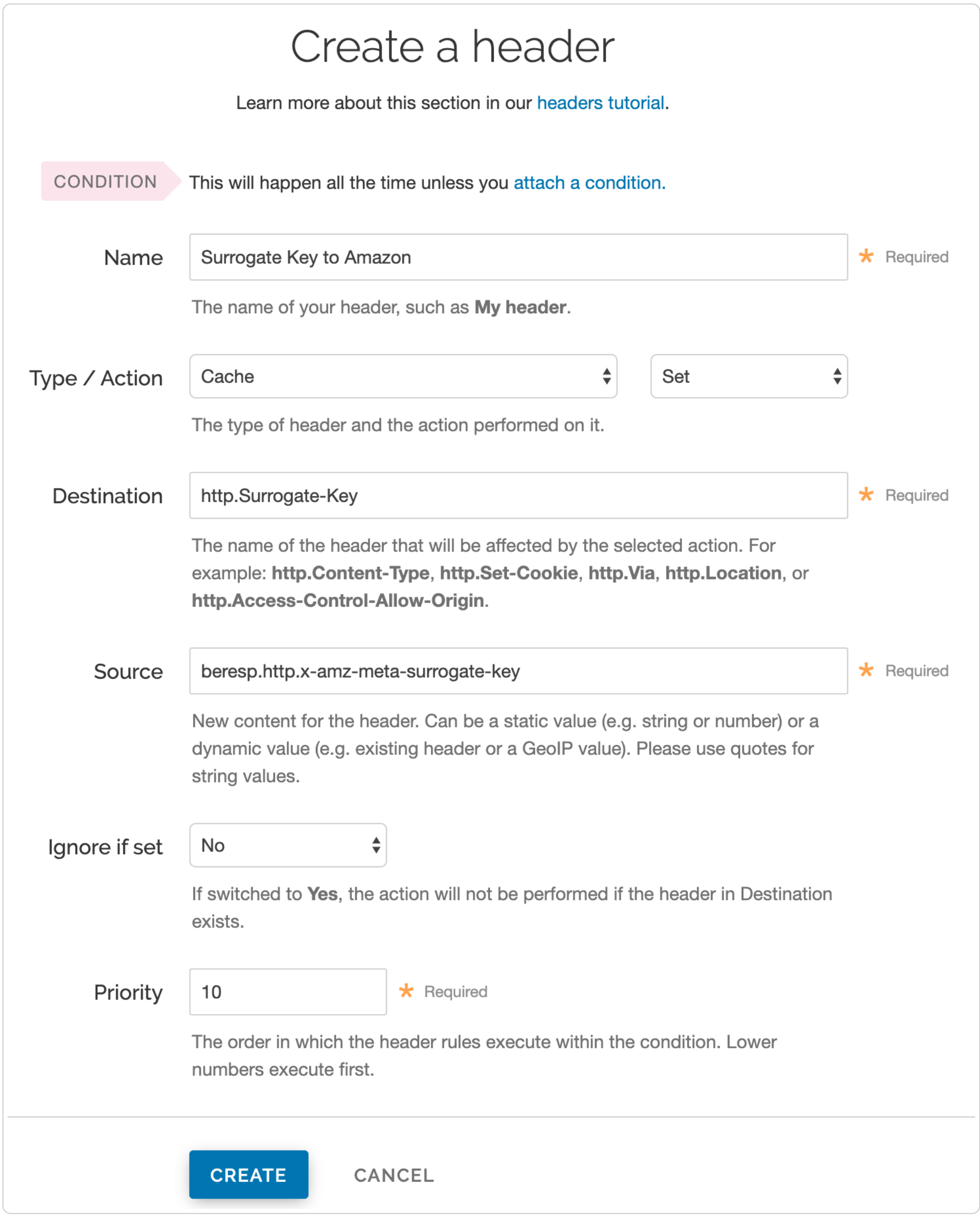
You can mark content with a surrogate key and use it to purge groups of specific URLs at once without purging everything, or purging each URL singularly. On the Amazon S3 side, you can use the `x-amz-meta-surrogate-key` header to mark your content as you see fit, and then on the Fastly side set up a Header configuration to translate the S3 information into the header we look for.

IMPORTANT

Pay close attention to the capitalization. Amazon S3 only accepts all lowercase header names.

Follow these instructions to set Surrogate-Key headers for Amazon S3 origin servers:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a human-readable name for the header. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Surrogate-Key`.
   - In the **Source** field, enter `beresp.http.x-amz-meta-surrogate-key`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.

7. Click the **Create** button to create your header.

8. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> There are several limitations to surrogate keys. See the surrogate key limitations section for more information.

Soft purges

Last updated: 2021-12-17

https://docs.fastly.com/en/guides/soft-purges

Fastly provides a Soft Purge feature that allows you to mark content as outdated (*stale*). Stale objects remain available to use in some circumstances while Fastly fetches a new version from origin, unlike objects invalidated by a *hard* purge which are made immediately unusable. You can purge by URL or by surrogate key using Soft Purge.

Before using Soft Purge, we recommend you implement one of the following methods to verify a stale object should temporarily remain in cache during revalidation:

- Set up `ETag` or `Last-Modified` headers for relevant content on your origin servers.
- Configure `stale_while_revalidate` to serve stale content and fetch the newest version of the object from origin in the background. If you choose this revalidation method, you must also configure `stale_if_error` at the same time.

To enable Soft Purge, add a `Fastly-Soft-Purge` request header (such as `Fastly-Soft-Purge: 1`) to any single URL or key-based purge.

To purge the URL `www.example.com` with Soft Purge, you would issue the following command:

```
$ curl -X PURGE -H "Fastly-Soft-Purge:1" http://www.example.com
```

To purge a surrogate key with Soft Purge, you would issue the following command:

```
$ curl -X POST -H 'Fastly-Soft-Purge:1' -H 'Fastly-Key: __API_KEY__' -H 'Accept: application/json' https://api.fastly.com/se
rvice/<SID>/purge/<S-Key>
```

> **TIP**
>
> Purge all requests cannot be done as a soft purge and will always immediately invalidate all cached content associated with the service. To do a soft purge all, consider applying a constant surrogate key tag (e.g., `all`) to all objects.

Wildcard purges

Last updated: 2019-07-15

https://docs.fastly.com/en/guides/wildcard-purges

Wildcard purging allows you to flush the cache of all pages under a directory branch or URL path; for example, you want to empty the cache of all pages under your "/service" path. Having to purge each URL one by one using the Fastly API or via the Fastly app is not very efficient.

Although Fastly does not have a specific wildcard purge function, you can implement the same behavior by making a small configuration change using surrogate keys. Surrogate keys allow you to tag a group of objects with a keyword (key) and then purge multiple pieces of content at once with it via the web interface or via custom VCL.

> **IMPORTANT**
>
> Purging will only apply to new objects as they're being put into the cache after you set up configuration changes. It will not apply to objects already in the cache when this configuration is being applied.

To purge content based on wildcard paths, follow the steps below.

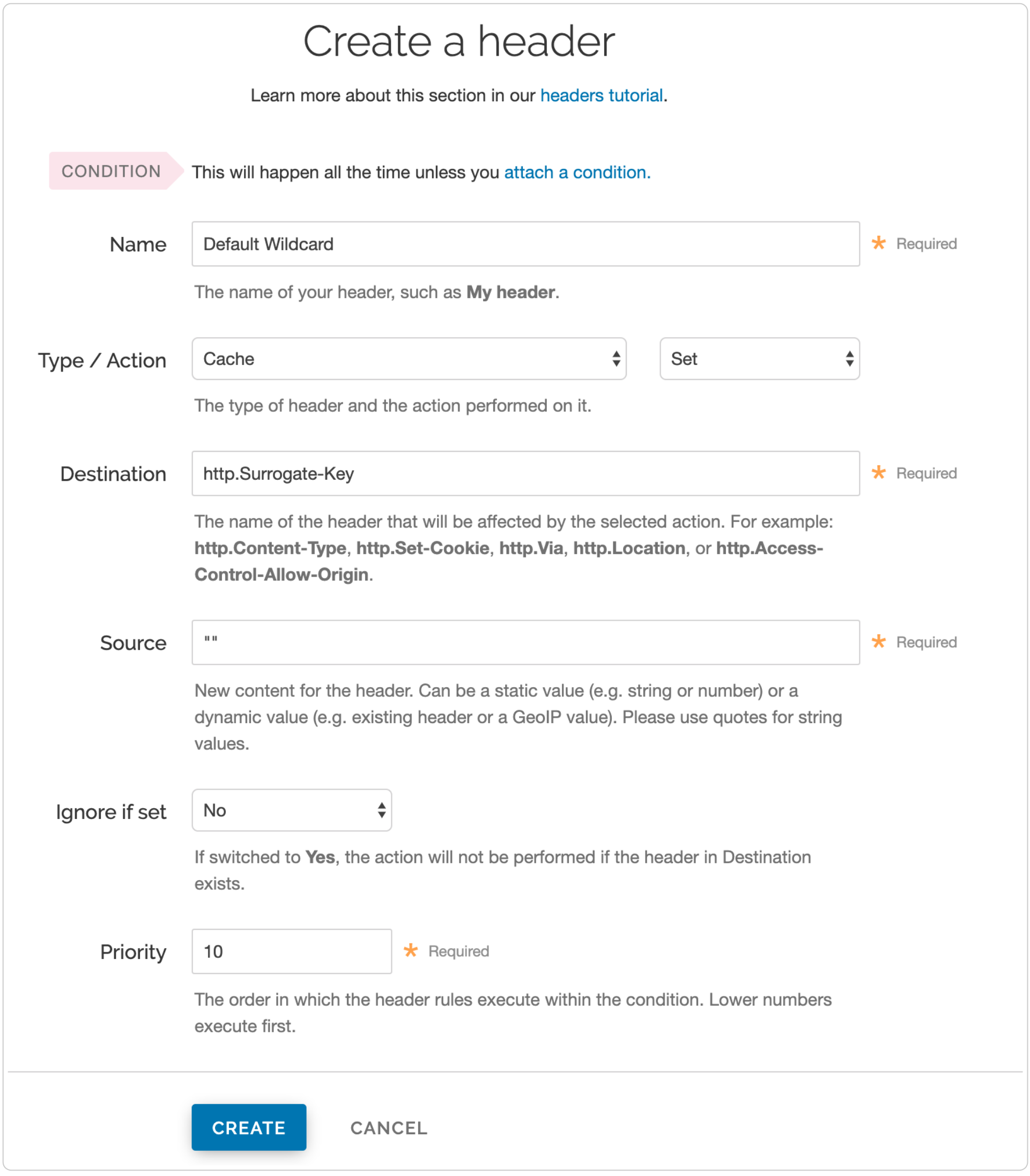
Via the web interface

To purge content based on wildcard paths via the web interface, follow the steps below.

Create a default wildcard header

We set a default wildcard so that we have the flexibility to append other surrogate keys to a URL path.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.

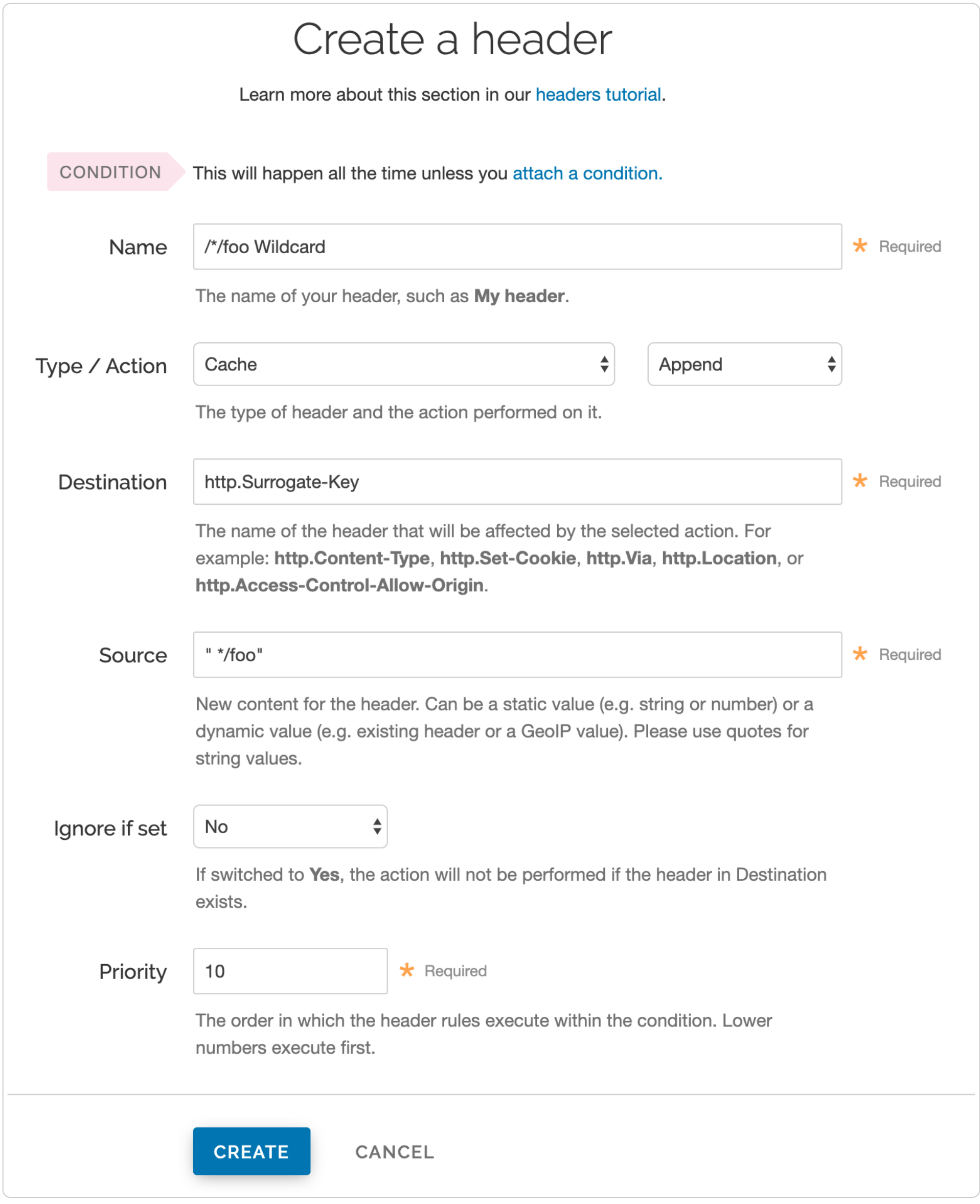


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Default Wildcard`. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache** and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Surrogate-Key`.
   - In the **Source** field, enter `""`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
7. Click the **Create** button. A new header appears in the Headers area of the Content page.

Create headers for each wildcard path being purged

Next, create a header for each of the wildcard paths you need the ability to purge. For instance, you want to purge the wildcard path `/*/foo`.

1. Click the **Create header** button to create another new header.



2. Fill out the **Create a header** fields as follows:
   - In the **Description** field, enter `/*/foo Wildcard`. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Append**.
   - In the **Destination** field, enter `http.Surrogate-Key`.
   - In the **Source** field, enter `" */foo"`. There is a space before the asterisk in the Source field, which is important when appending multiple surrogate keys to a URL.

- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, enter `20`.

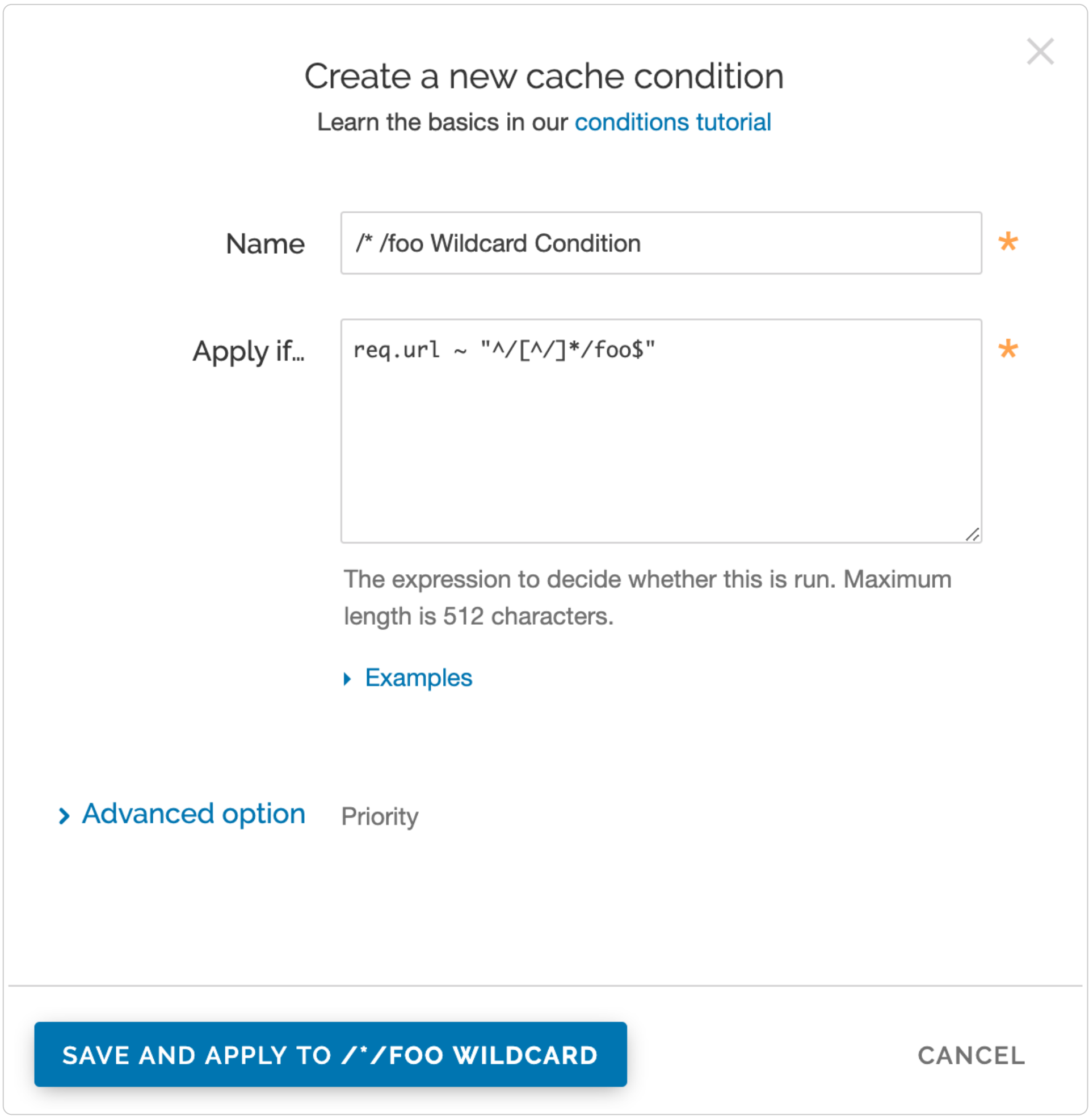
3. Click the **Create** button. A new header appears in the Headers area of the Content page.

Notice the Action is set to Append to add to the default wildcard surrogate key. The Priority is set to 20 so that the Default Wildcard header is executed first and then the wildcard path appends.

Create conditions for each wildcard path being purged

Finally, create a condition for each of the wildcard paths you need the ability to purge.

1. Click the **Attach a condition** link next to the wildcard path header name. The Create a new cache condition window appears.



2. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, enter `/*/foo Wildcard Condition`.
   - In the **Apply if** field, enter `req.url ~ "^/[^/]*/foo$"`.
3. Click the **Save and apply to** button to create the new condition.

What does the condition mean? In the Apply if field above, the first `"^"` and `"$"` tells Fastly to look for the following pattern:

- Start from the first slash after the request Host header.
- There should be one directory.
- It should be followed by the path `/foo` ending the URL.

Some examples would be `/a/foo`, `/bar/foo`, and `/c/foo`. You could also remove the first `"^"` and `">"$"` to allow the condition to be more general so that the pattern can occur in the middle of a URL path.

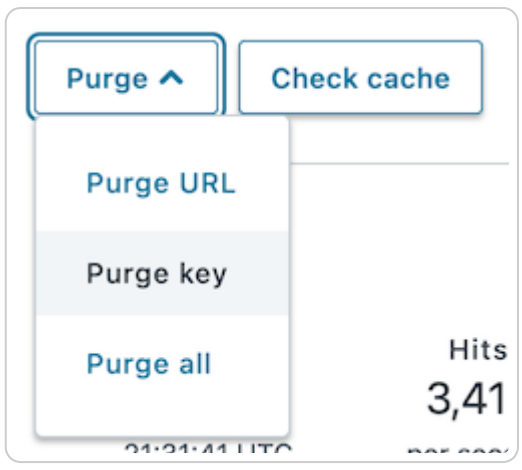
Some other examples for URL wildcard conditions:

Apply if field	Matched pattern
`req.url ~ "/[^/]*/foo"`	`/delta/wow/a/foo/neat/cool/img.gif`
`req.url ~ "^/.*/foo$"`	`/a/b/c/d/e/f/foo`

Purge the wildcard

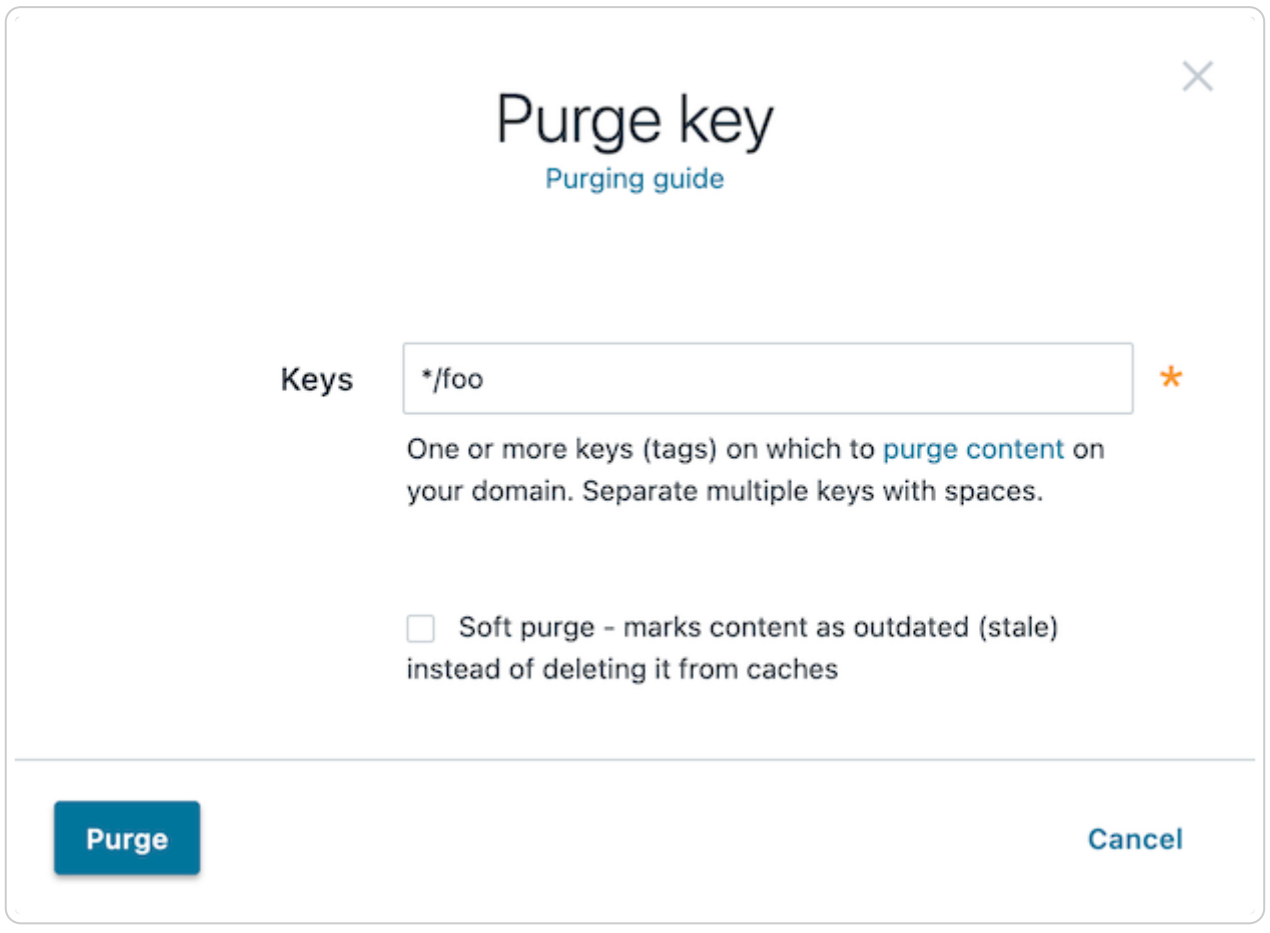
Ready to purge that wildcard? You can do this through the web interface using the steps below.

1. Log in to the Fastly web interface.
2. From the **Purge** menu, select **Purge Key**.



The Purge Key window appears.

3. In the **Keys** field, enter the surrogate key you want to purge. Continuing with our example, you would enter `*/foo` without the quotes that were entered in the Source field of the New Header window above.



4. Click the **Purge** button.

Via custom VCL

To purge content based on wildcard paths via custom VCL, follow the steps below.

1. Add the following code to the VCL template:

```
sub construct_skey {
  if (req.url.path ~ "^(((((/[^/]+)?/[^/]+)?/[^/]+)?/[^/]+)?/[^/]+)") {
    # This prevents us from doing this twice when shielding
    if (std.strstr(beresp.http.Surrogate-Key, re.group.1)) {
      return;
    }

    if (!re.group.2) {
      set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-Key " ", "")
        + re.group.1;
      return;
    }

    if (!re.group.3) {
      set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-Key " ", "")
        + re.group.1 + " " + re.group.2;
      return;
    }

    if (!re.group.4) {
      set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-Key " ", "")
        + re.group.1 + " " + re.group.2 + " " + re.group.3;
      return;
    }

    if (!re.group.5) {
      set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-Key " ", "")
        + re.group.1 + " " + re.group.2 + " " + re.group.3 + " " + re.group.4;
      return;
    }
    set beresp.http.Surrogate-Key = if(beresp.http.Surrogate-Key, beresp.http.Surrogate-Key " ", "")
      + re.group.1 + " " + re.group.2 + " " + re.group.3 + " " + re.group.4 + " " + re.group.5;
  }
}
```

2. Call the subroutine in `vcl_fetch`:

```
sub vcl_fetch {
call construct_skey;
}
```

3. Check your success by curling an object not already in cache with the `Fastly-Debug:1` header to expose the surrogate keys. For example:

```
$ curl -svo /dev/null http://www.example.com/test/test2/file3.txt -H Fastly-Debug:1
* Trying 192.0.2.0...
* Connected to www.example.com (192.0.2.0) port 80 (#0)
> Host: www.example.com
> User-Agent: curl/7.43.0
> Accept: */*
> Fastly-Debug:1
>
< HTTP/1.1 200 OK
< Server: Apache
< Content-Type: text/plain
< Surrogate-Key: /test /test/test2 /test/test2/file3.txt
< Via: 1.1 varnish
< X-Backend-IP: 203.0.113.0
< Cache-Control: max-age=31536000, stale-while-revalidate=31536000, stale-if-error=31536000
< Content-Length: 19
< Accept-Ranges: bytes
< Date: Fri, 29 Jan 2016 21:30:08 GMT
< Via: 1.1 varnish
< Age: 1035
< Connection: keep-alive
< Fastly-Debug-Path: (D cache-sjc3123-SJC 1454103008) (F cache-sjc3134-SJC 1454101973) (D cache-den6026-DEN 1454101973) (F cache-den6027-DEN 1454101973)
< Fastly-Debug-TTL: (H cache-sjc3123-SJC - - 1035) (M cache-den6026-DEN - - 0)
< Fastly-Debug-Digest: b43bd38cf940e1669c2927c8662660e5170758053dda42e772ce3fc34ee57fc1
< X-Served-By: cache-den6026-DEN, cache-sjc3123-SJC
< X-Cache: MISS, HIT
< X-Cache-Hits: 0, 1
< Vary: Accept-Encoding
<
{ [19 bytes data]
* Connection #0 to host www.example.com left intact
```

In the above example, the `< Surrogate-Key: /test /test/test2 /test/test2/file3.txt` headers show the addition of the three surrogate keys.

Via the API

You can also use our key-based purging via the API to perform wildcard purging using an HTTP request:

```
POST /service/<Fastly Service ID>/purge/*/foo
Fastly-Key: FASTLY_API_TOKEN
```

This will purge any content that was associated with the `"*/foo"` surrogate key according to the setup in your header rules. Additional syntax for purging a service through the API can be found in the Purging section of the API documentation.

Working with surrogate keys

Last updated: 2021-12-06

https://docs.fastly.com/en/guides/working-with-surrogate-keys

Surrogate keys are unique identifiers that you assign to groups of content for processing. While there are many use cases for surrogate keys, one of the primary way for using them with Fastly is to make purging more efficient.

Surrogate keys allow you to selectively purge related content. Using the `Surrogate-Key` header, you can "tag" content with a key term, a string of any characters you want. When you want to purge content associated with that key, you then issue a key purge request and all of the objects associated with that key will be purged. This process makes it easier to cache and purge content that changes rapidly and unpredictably without having to purge your entire cache.

Before you begin

This guide assumes you're already familiar with the way content delivery networks (CDNs) work in general, and the way Fastly's CDN works in particular. We also recommend reviewing our guide on adding headers on HTTP requests and responses.

Understanding the `Surrogate-Key` header

To understand how surrogate keys work, especially in the context of content purging, you need to understand what a Surrogate-Key header is and how it behaves.

About the `Surrogate-Key` header

Any time your origin responds to a request for content, the response it sends will include bits of code called *headers* in front of the actual body of the response. Those headers are used to give additional detail and provide context for the body of the response itself. HTTP headers sent as part of a response typically look something like this:

```
HTTP/1.1 200 OK
Content-Type: text/html
Connection: keep-alive
...
```

You can control that additional information and therefore control how content is served by adding to or modifying the headers your server responds with. The `Surrogate-Key` header is one of them. It's what categorizes or "tags" the content as part of a specific group. The tag you choose is called a *key* and it describes the common trait each element of a content group shares.

For example, suppose you have a website where you publish taco recipes. You know that entire categories of recipes will need to be purged, so you thoughtfully include these recipe categories as keys in the `Surrogate-Key` header. The response your server sends might look more like this:

```
HTTP/1.1 200 OK
Surrogate-Key: veggie seasonal central-mexico
Content-Type: text/html
...
```

The addition of the Surrogate-Key header there tells you that the content is tagged with specific identifiers that categorize it for future use. This response contains three surrogate keys: `veggie`, `seasonal`, and `central-mexico`. When Fastly receives a response like this, we use the surrogate keys to create a mapping from each key to the cached content, then we strip out the `Surrogate-Key` header so it's not included in the response to your readers.

> **TIP**
>
> You can't use duplicate surrogate keys. For example, if you tried to use `foo`, `bar`, and `foo`, the two `foo`s would be collapsed into a single instance of the term.

Creating relationships between keys and objects

One of the major advantages of surrogate keys is that they allow for a many-to-many relationship between keys and objects. An origin server's response can associate multiple keys with the object and the same key can be provided in different responses. Take a look at these two requests and responses:

```
GET /blog/healthy-taco-recipes HTTP/1.1
Host: www.tacolabs.com

HTTP/1.1 200 OK Content-Type: text/html
Content-Length: 1234
Surrogate-Key: mainpage low-carb
```

```
GET /recipes/low-carb-cheese-taco-shell HTTP/1.1
Host: www.tacolabs.com

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 2345
Surrogate-Key: low-carb cheese
```

In this example, there are two objects (`/blog/healthy-taco-recipes` and `/recipes/low-carb-cheese-taco-shell`) with three keys (`mainpage`, `low-carb`, and `cheese`). Two of the keys (`mainpage` and `cheese`) are associated with a single object and a third key (`low-carb`) is associated with both objects.

By using the `Surrogate-Key` header to associate keys with one or more objects, you can precisely control which objects are removed from cache during a purge. Consider the example presented above. Purging the `mainpage` key would remove only the `/blog/healthy-taco-recipes` object from the cache. On the other hand, purging the `low-carb` key would remove both the `/blog/healthy-taco-recipes` and `/recipes/low-carb-cheese-taco-shell` objects from the cache.

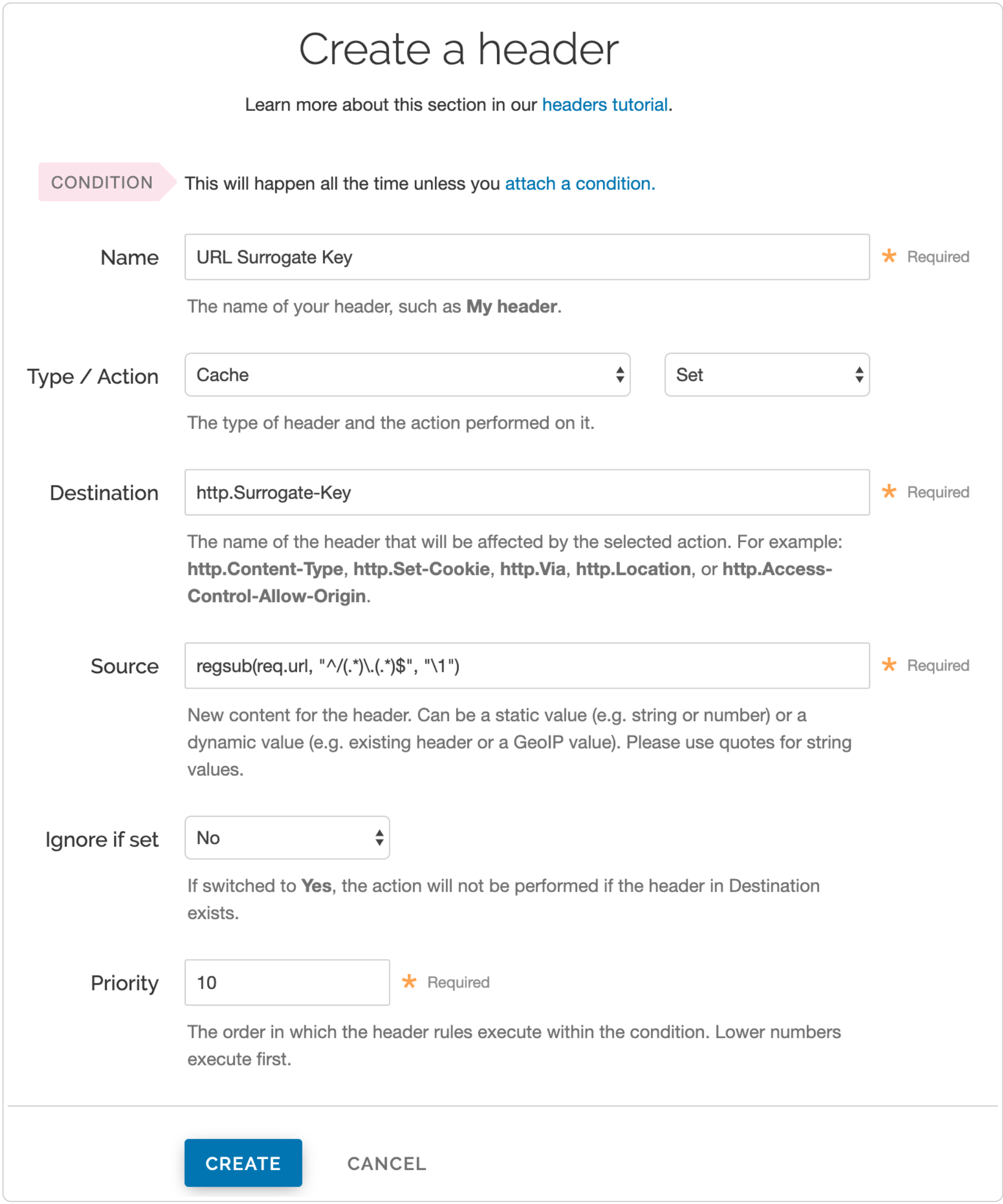
Wherever there's a relation between two different pieces of content, there might be a good reason to keep them categorized by using a surrogate key.

Generating and setting surrogate keys

There are two ways to set the `Surrogate-Key` header: by adding the header in the Fastly web interface, or by generating the keys with your own application. You can make your surrogate key associations on your own application server and include them in the HTTP response that you send to Fastly. This is the more useful method because you can assign exactly the keys that you want on every response.

However, if you want to set the `Surrogate-Key` header in the Fastly web interface you can do so following the steps below.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a human-readable name for the header. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Surrogate-Key`.
   - In the **Source** field, enter where the content for the header comes from.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
7. Click the **Create** button to create your header.

8. Click the **Activate** button to deploy your configuration changes.

Troubleshooting

You can check the surrogate keys for a URL by using the `Fastly-Debug: 1` header. See the instructions on using a Fastly-Debug header with curl for more information.

Limitations

The surrogate keys sent by your origin server can be as simple or complex as you need, subject to format limitations. Surrogate keys must be formatted as a single string without spaces. Spaces separate keys.

Surrogate keys are subject to size limitations. Individual surrogate keys may not exceed 1,024 bytes in length and `Surrogate-Key` header values (comprising one or more space-separated keys) may not exceed 16,384 bytes in length. If either of the key or key header value limits are reached while parsing a `Surrogate-Key` header, the key currently being parsed and all keys following it within the same header will be ignored.

What's next

Now that you understand how the `Surrogate-Key` header works and how to set it, learn how to purge objects with surrogate keys.

---

§

These articles describe configuration settings and changes you can make to requests when working with Fastly services.

https://docs.fastly.com/en/guides/configuration#_requests

Checking multiple backends for a single request

Last updated: 2017-05-23

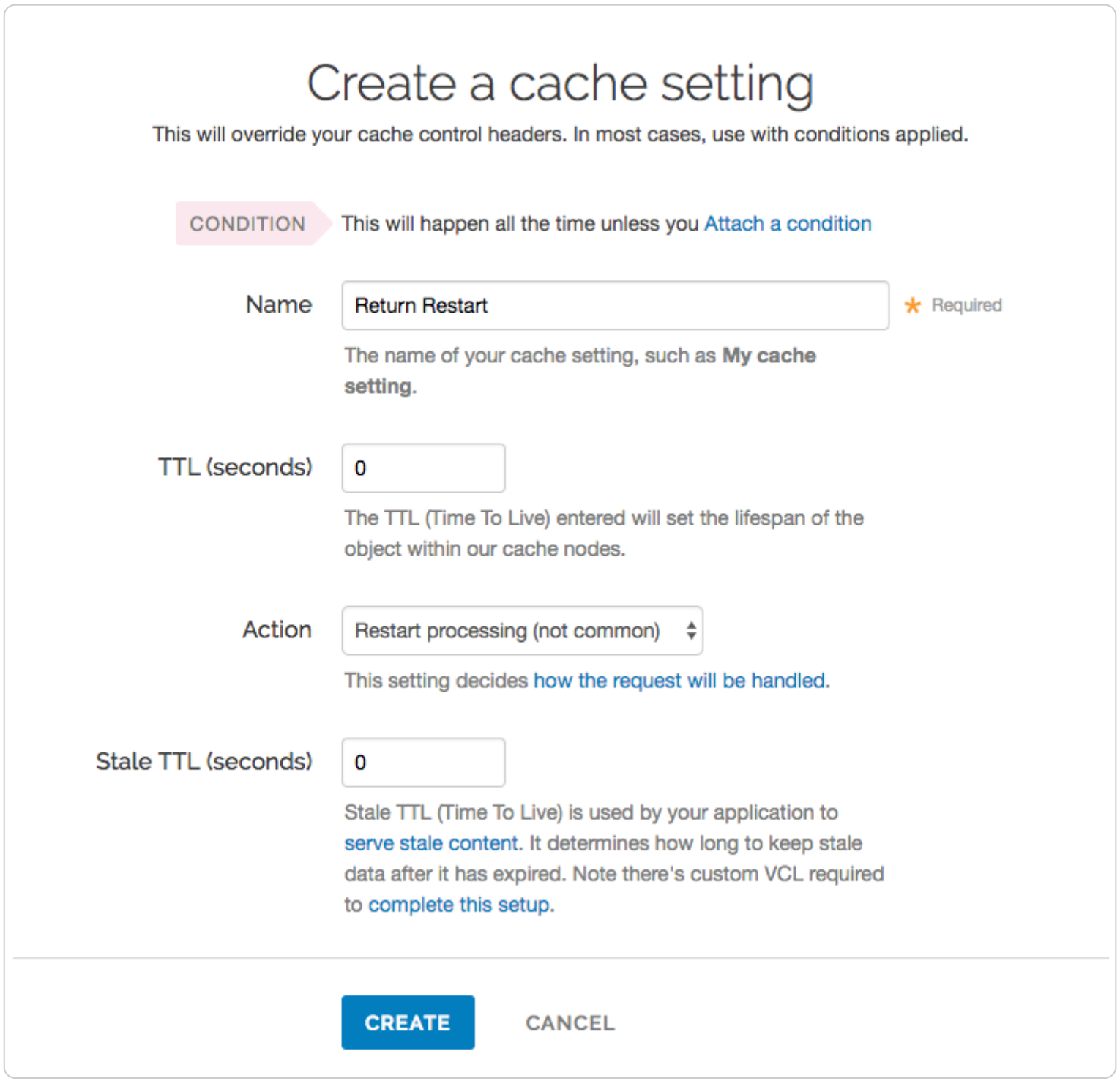
https://docs.fastly.com/en/guides/checking-multiple-backends-for-a-single-request

Using a restart is a good option to check multiple backends for a single request. This can be created using a cache setting rule and request headers.

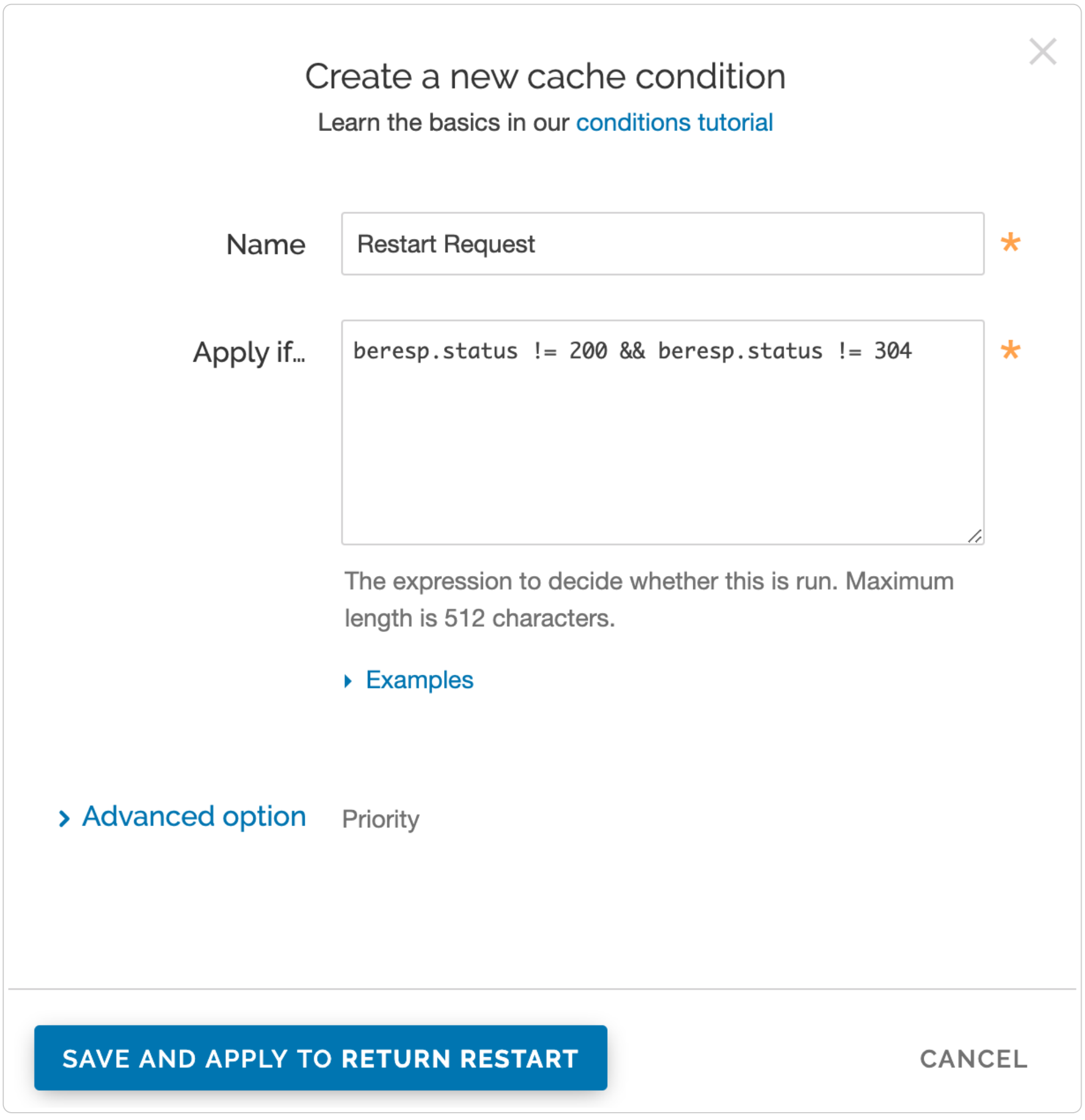
Create a new cache setting rule

Follow these steps to create a cache restart within vcl_fetch.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **Create cache setting** button. The Create a cache setting page appears.



6. Fill out the **Create a cache setting** fields as follows:
   - In the **Name** field, enter `Return Restart` (or any meaningful, preferred name).
   - In the **TTL (seconds)** field, enter `0`.
   - From the **Action** menu, select **Restart processing**.
   - In the **Stale TTL (seconds)** field, enter `0`.
7. Click the **Create** button. The new cache setting appears on the Settings page.
8. On the **Settings** page, click the **Attach a condition** link next to the cache setting you just created. The Create a new cache condition window appears.

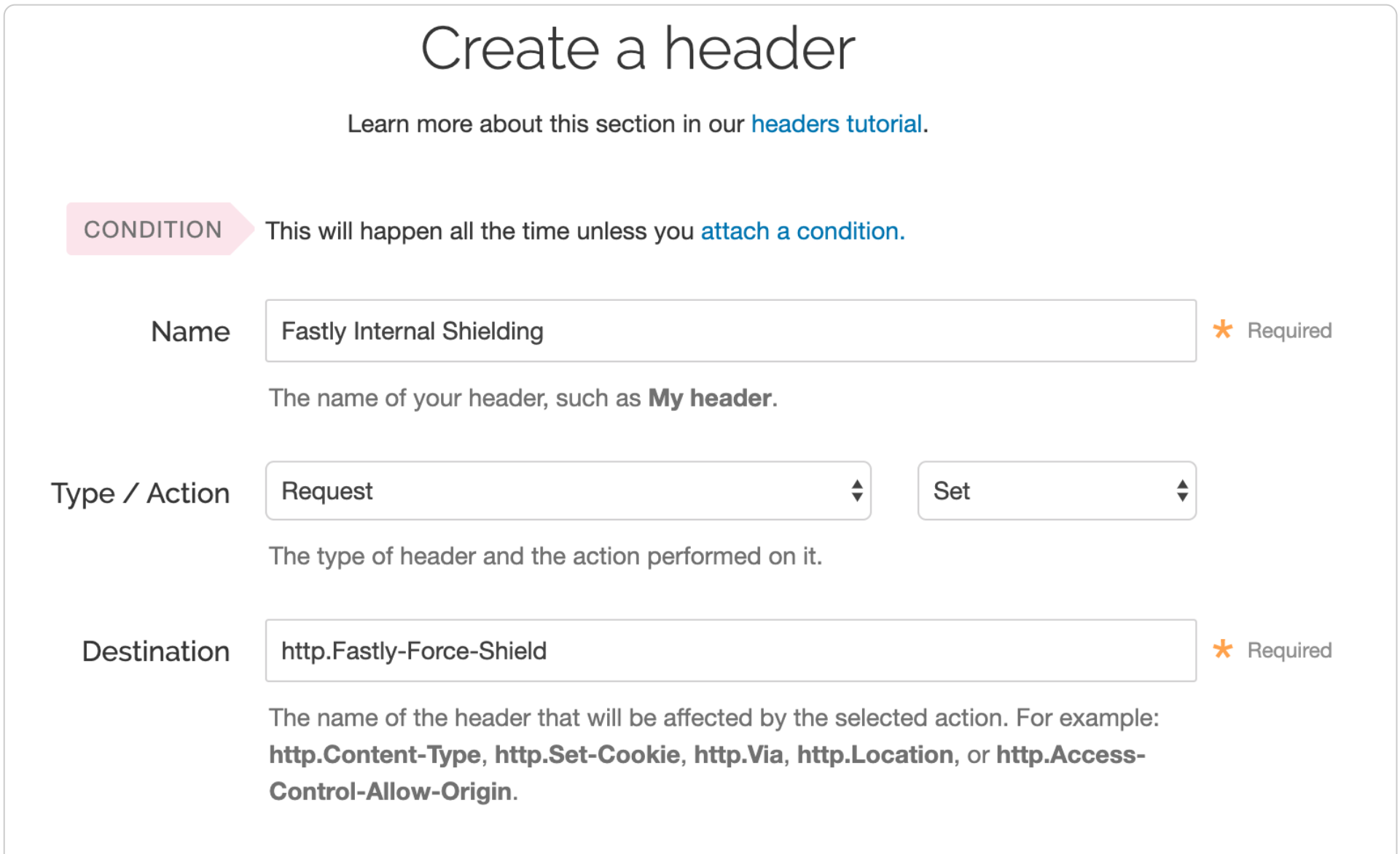


9. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, enter `Restart Request` (or any meaningful, preferred name).
   - In the **Apply if** field, enter `beresp.status != 200 && beresp.status != 304`.
10. Click the **Save and apply to** button to create the condition.

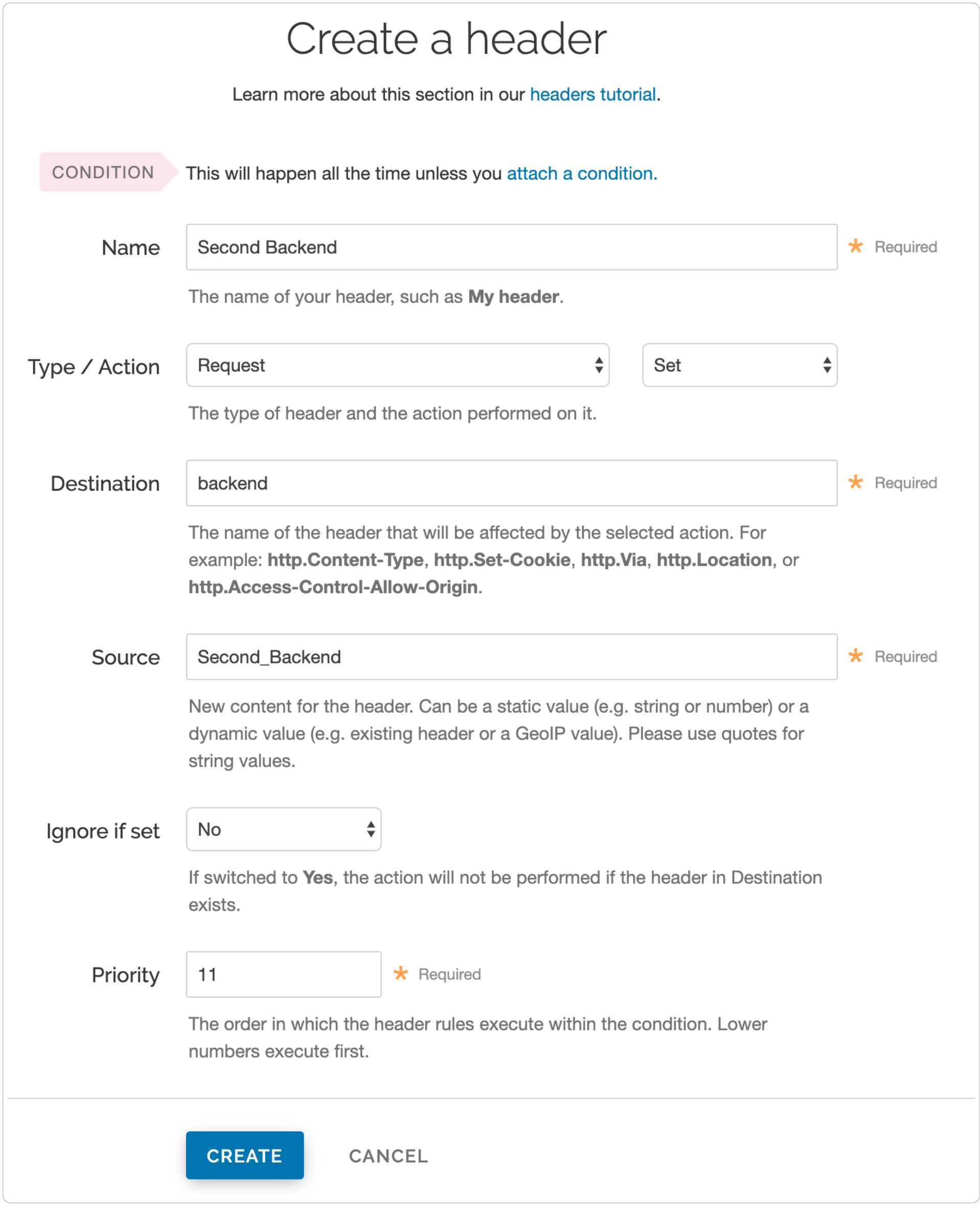
Create new request headers

Follow these steps to create a request header within vcl_recv.

1. Click the **Content** link. The Content page appears.
2. Click the **Create header** button. The Create a header page appears.



3. Fill out the **Create a new header** fields as follows:
   - In the **Name** field, enter `Fastly Internal Shielding` (or any meaningful, preferred name).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Fastly-Force-Shield`.
   - In the **Source** field, enter `"yes"`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
4. Click the **Create** button. The new header appears on the Content page.
5. Click the **Create header** button to create another header to switch to the next backend. The Create a header page appears.

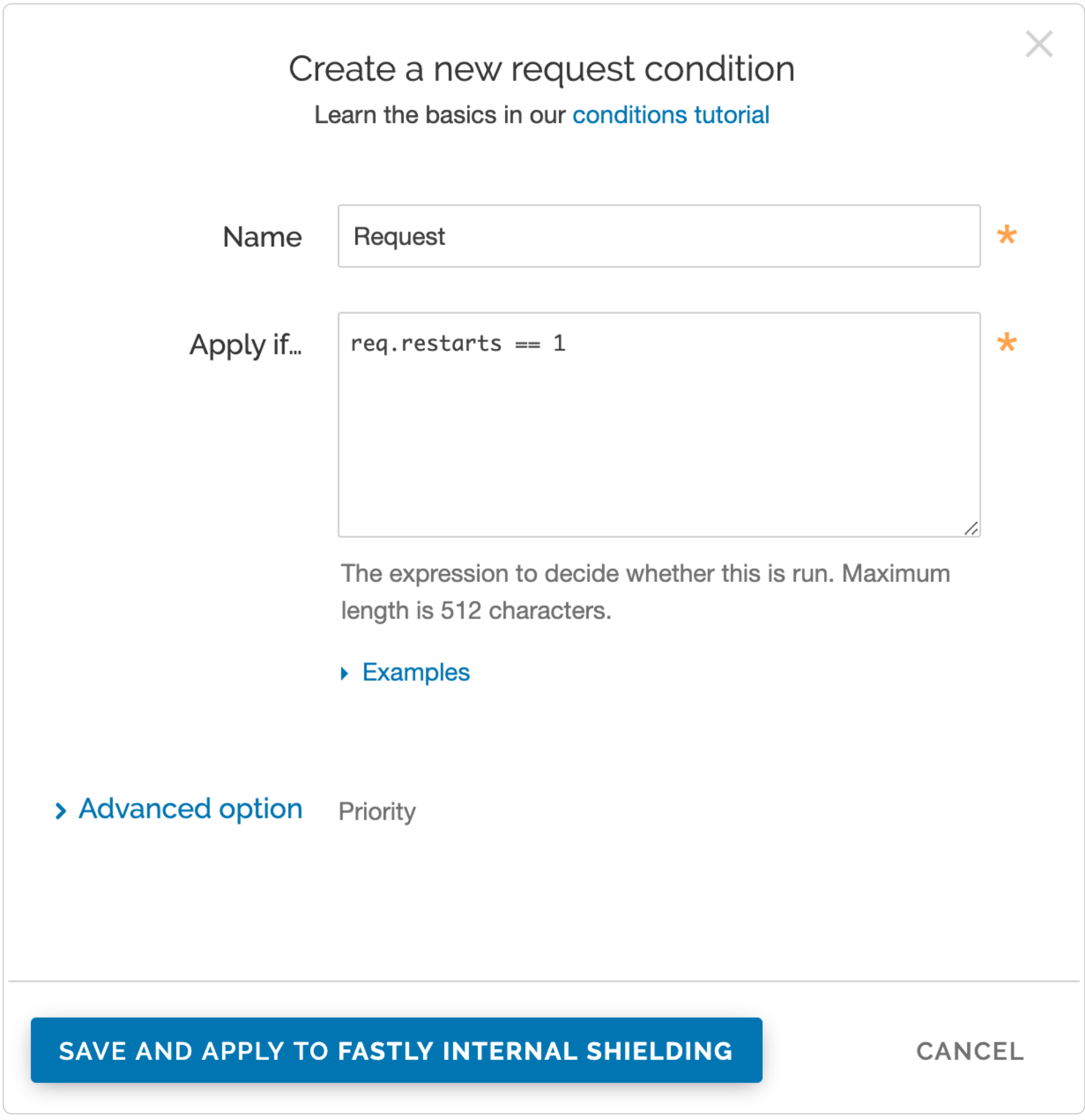


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Second Backend` (or any meaningful, preferred name).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `backend`.
   - In the **Source** field, enter `Second_Backend` (this should match the name of your other backend).
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `11`.
7. Click the **Create** button. The new header appears on the Content page.

Create new header conditions

Follow these steps to create conditions for the headers.

1. On the **Content** page, click the **Attach a condition** link next to one of the headers you just created. The Create a new request condition window appears.



2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, enter `Req.request` (or any meaningful, preferred name).
   - In the **Apply if** field, enter `req.restarts == 1`.
3. Click **Save and apply to**. The condition appears on the Content page.
4. Repeat steps 1-3 for the other header.
5. Click the **Activate** button to deploy your configuration changes.

Conditionally changing a URL

Last updated: 2018-08-01

https://docs.fastly.com/en/guides/conditionally-changing-a-url

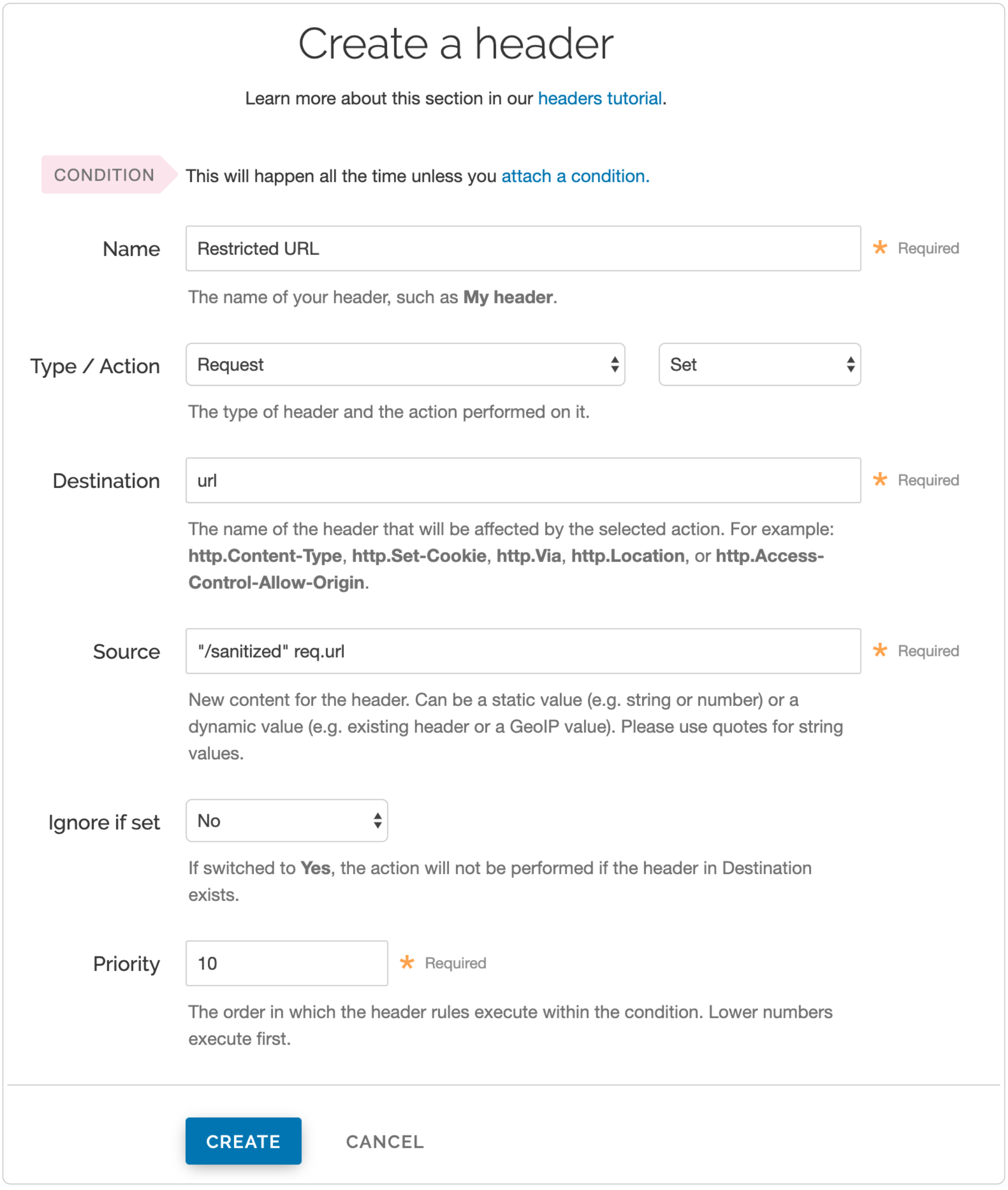
To conditionally change a URL based on the domain, include VCL that looks something like this:

```
if (req.http.host ~ "^restricted") {
  set req.url = "/sanitized" req.url;
}
```

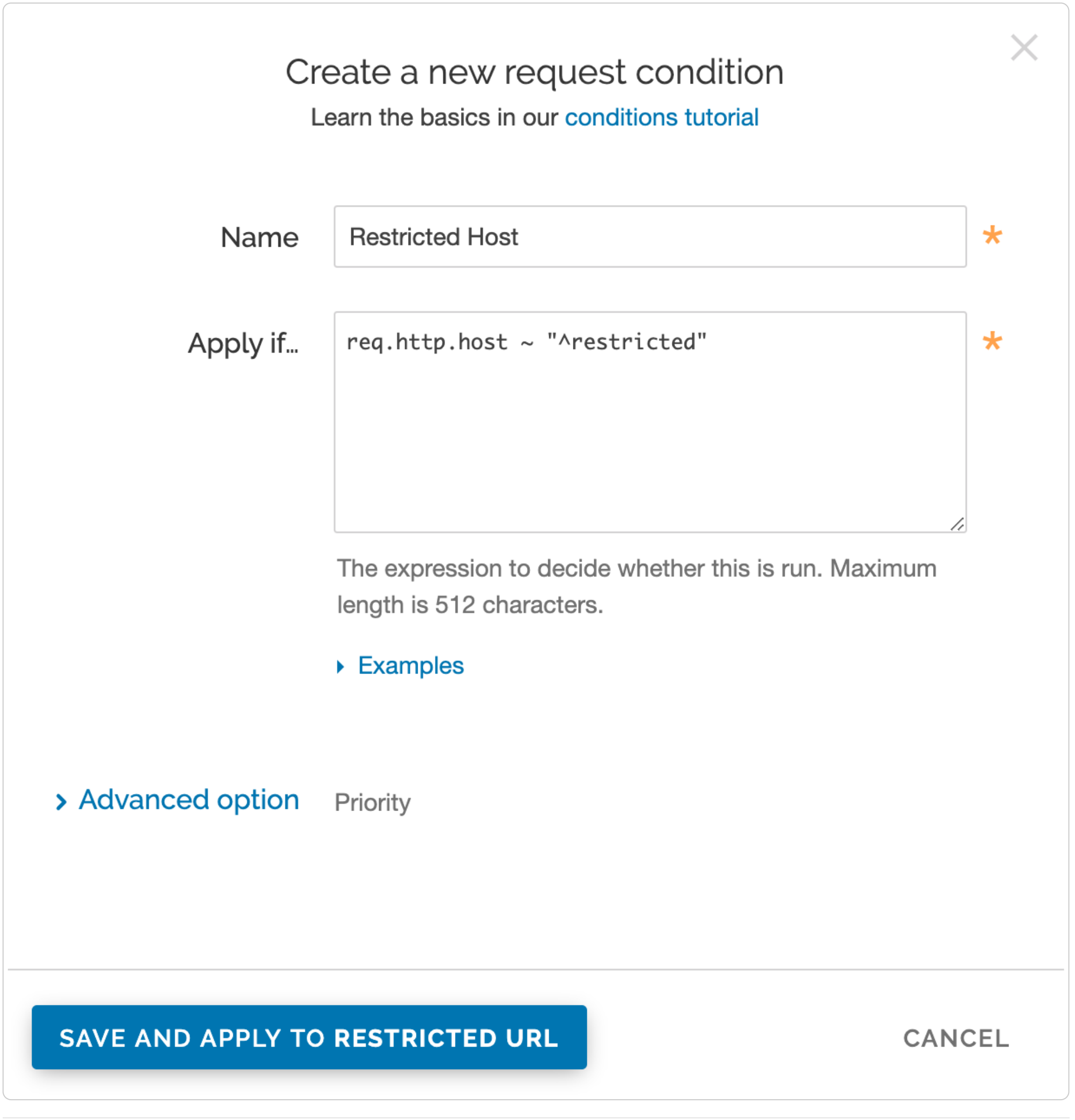
If you have shielding enabled, however, add the following code instead to avoid rewriting the URL twice:

```
if (req.http.host ~ "^restricted" && req.url !~ "^/sanitized") {
  set req.url = "/sanitized" req.url;
}
```

In Fastly's web interface, this VCL would be the equivalent of creating a new Header:



and then creating a request condition that restricts connections to that host:



How request settings are applied

Last updated: 2018-05-11

https://docs.fastly.com/en/guides/how-request-settings-are-applied

Requests settings are applied based on the Action you select in the Create a new request setting page. You can choose any one of the following settings:

- **Do nothing now** - Apply the request setting options, but don't force a lookup or a pass action. The request settings are applied as the system continues through the VCL logic.
- **Lookup (in cache)** - Immediately search the cache for content. If the content isn't found (a MISS), then send the request to the origin.
- **Pass (do not cache)** - Immediately send the request to the origin each time and ignore additional request configurations. See our info on understanding the different PASS action behaviors to learn more.

Manipulating the X-Forwarded-For header

Last updated: 2017-05-08

https://docs.fastly.com/en/guides/manipulating-the-x-forwarded-for-header

You can control what happens to the X-Forwarded-For HTTP header via the Create a new request setting page on the Requests settings area of the Settings page. From the X-Forwarded-For menu, select one of the following behaviors:

- **Append** - Appends the client IP to the X-Forwarded-For header.
- **Append All** - Appends the client IP (and edge-cache IP, in case of shielding) to the X-Forwarded-For header. Creates the header if it does not exist yet.
- **Clear** - Clears the X-Forwarded-For header.
- **Leave** - Leaves the X-Forwarded-For header as is, if it is present.
- **Overwrite** - Overwrites the X-Forwarded-For header with just the client IP.

For more information about requests and responses, see our tutorial.

§

These articles describe configuration settings and changes you can make to your response settings when setting up Fastly services.

https://docs.fastly.com/en/guides/configuration#_responses

Creating and customizing a robots.txt file

Last updated: 2018-08-16

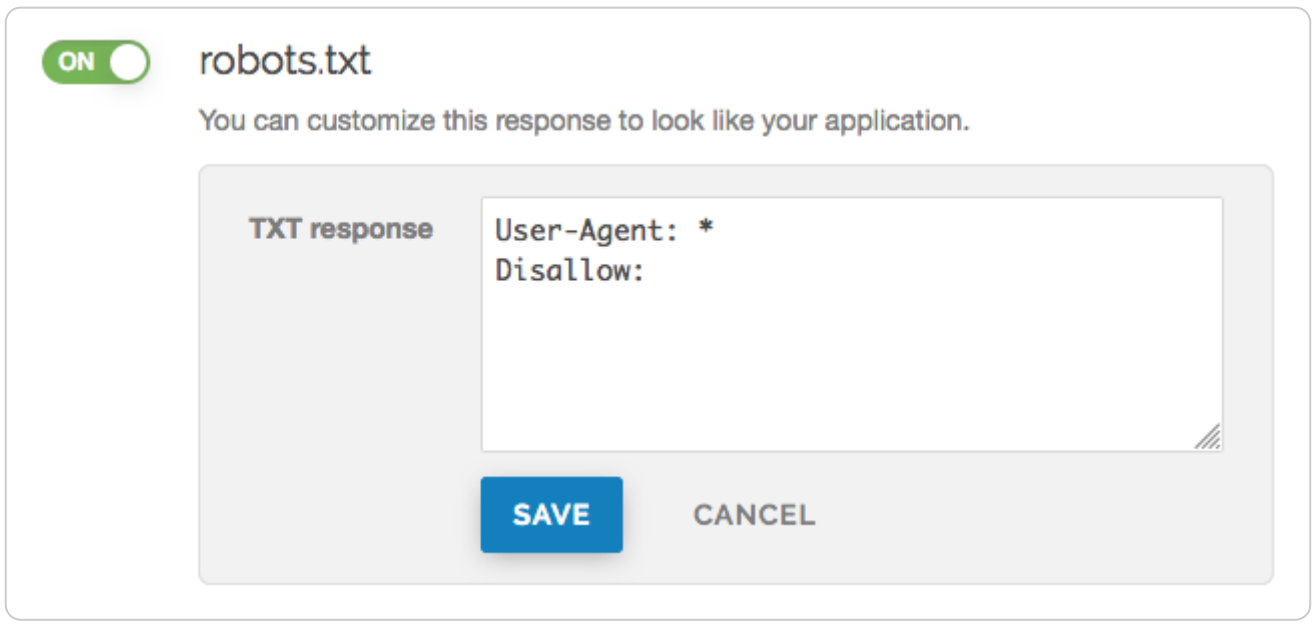
https://docs.fastly.com/en/guides/creating-and-customizing-a-robots-file

The robots.txt file tells web robots how to crawl webpages on your website. You can use Fastly's web interface to create and configure a robots.txt file. If you follow the instructions in this guide, Fastly will serve the robots.txt file from cache so the requests won't hit your origin.

Creating a robots.txt file

To create and configure your robots.txt file, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **robots.txt** switch to enable the robots.txt response.

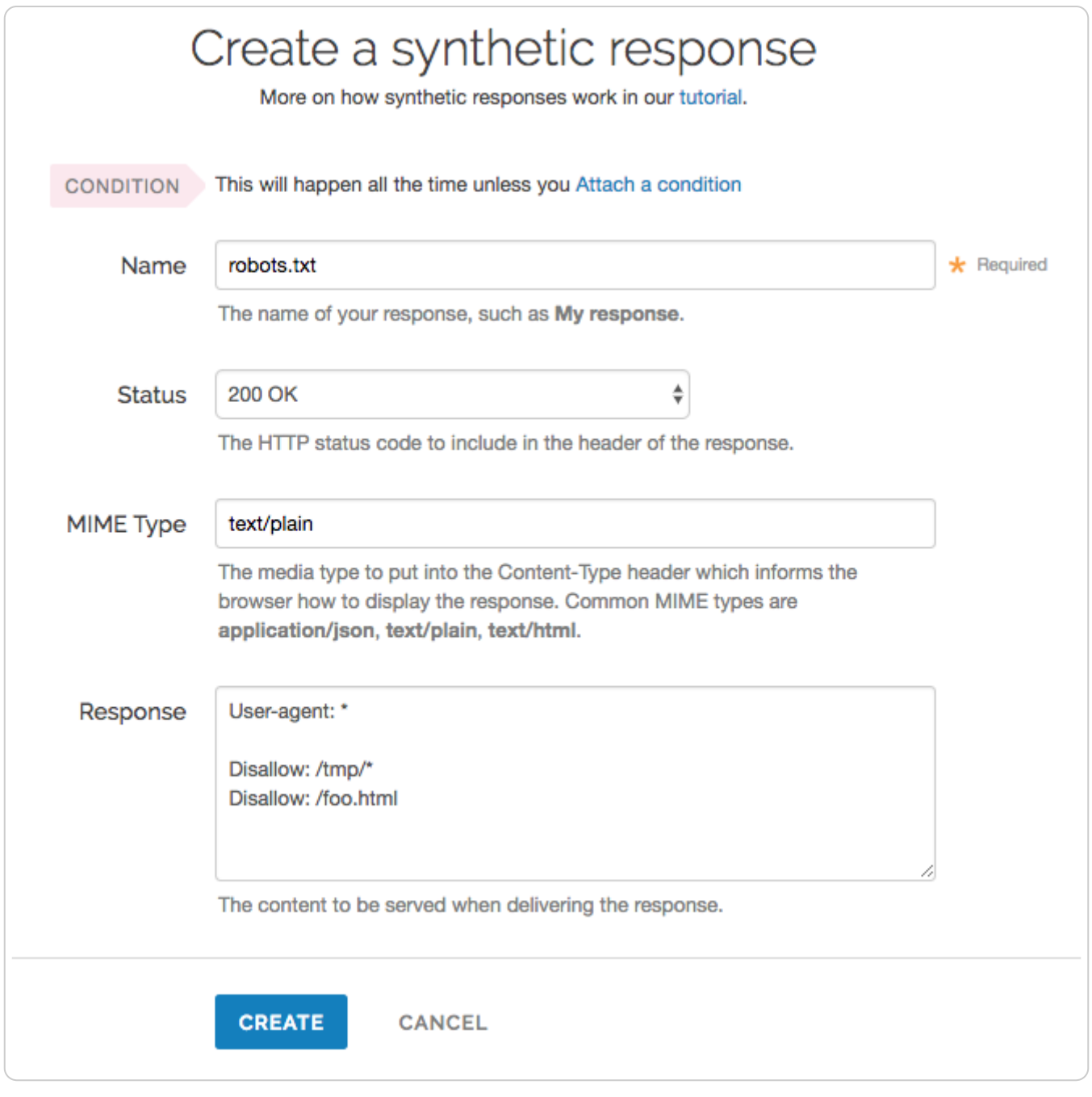


6. In the **TXT Response** field, customize the response for the robots.txt file.
7. Click the **Save** button to save the response.
8. Click the **Activate** button to deploy your configuration changes.

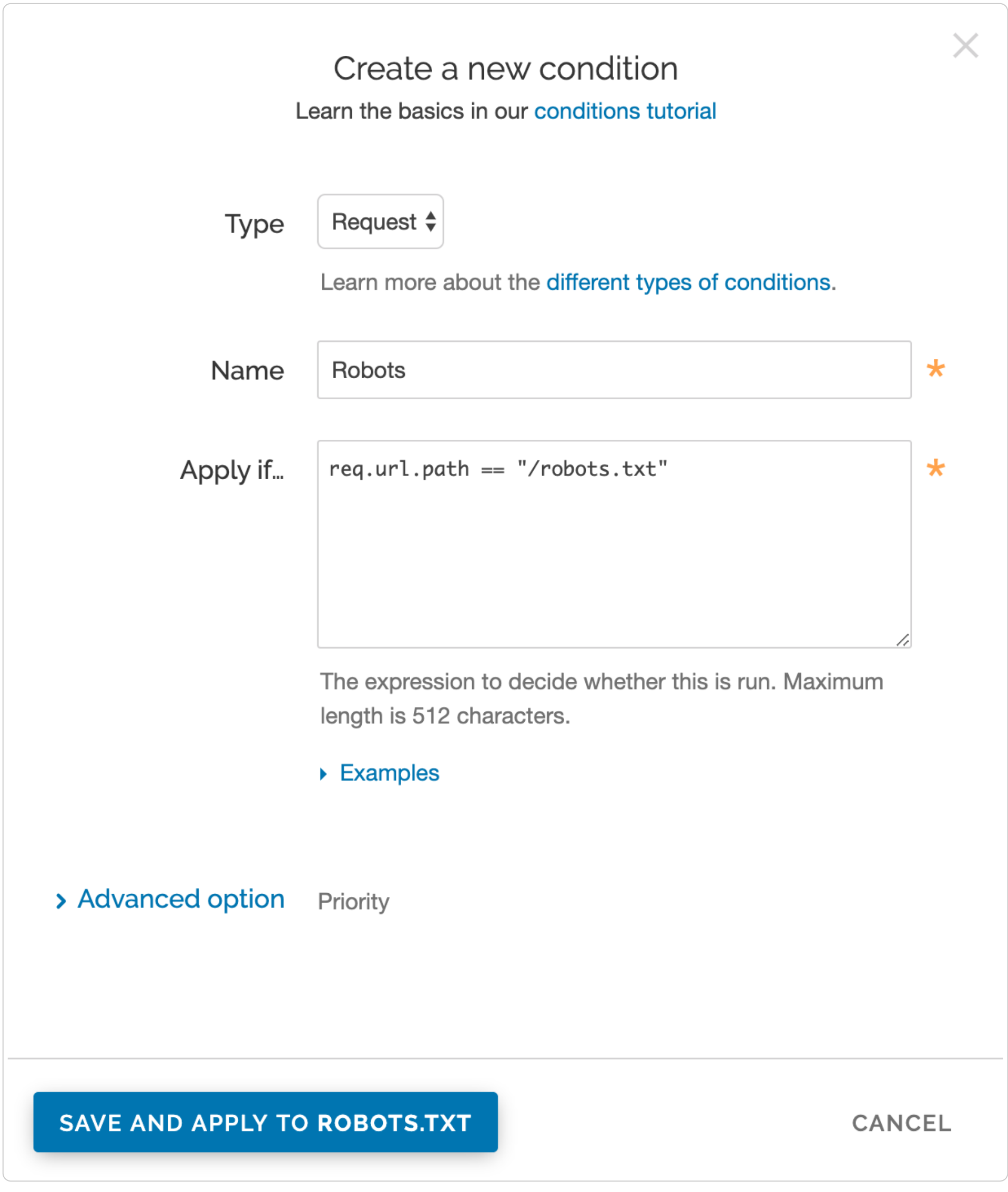
Manually creating and customizing a robots.txt file

If you need to customize the robots.txt response, you can follow the steps below to manually create the synthetic response and condition:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Set up advanced response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, enter an appropriate name. For example `robots.txt`.
   - Leave the **Status** menu set at its default `200 OK`.
   - In the **MIME Type** field, enter `text/plain`.
   - In the **Response** field, enter at least one User-agent string and one Disallow string. For instance, the above example tells all user agents (via the `User-agent: *` string) they are not allowed to crawl anything after `/tmp/` directory or the `/foo.html` file (via the `Disallow: /tmp/*` and `Disallow: /foo.html` strings respectively).
7. Click the **Create** button. Your new response appears in the list of responses.
8. Click the **Attach a condition** link to the right of the newly created response. The Create a new condition window appears.



9. Fill out the **Create a condition** fields as follows:

- From the **Type** menu, select the desired condition (for example, `Request`).
- In the **Name** field, enter a meaningful name for your condition (e.g., `Robots`).
- In the **Apply if** field, enter the logical expression to execute in VCL to determine if the condition resolves as true or false. In this case, the logical expression would be the location of your robots.txt file (e.g., `req.url.path == "/robots.txt"`).

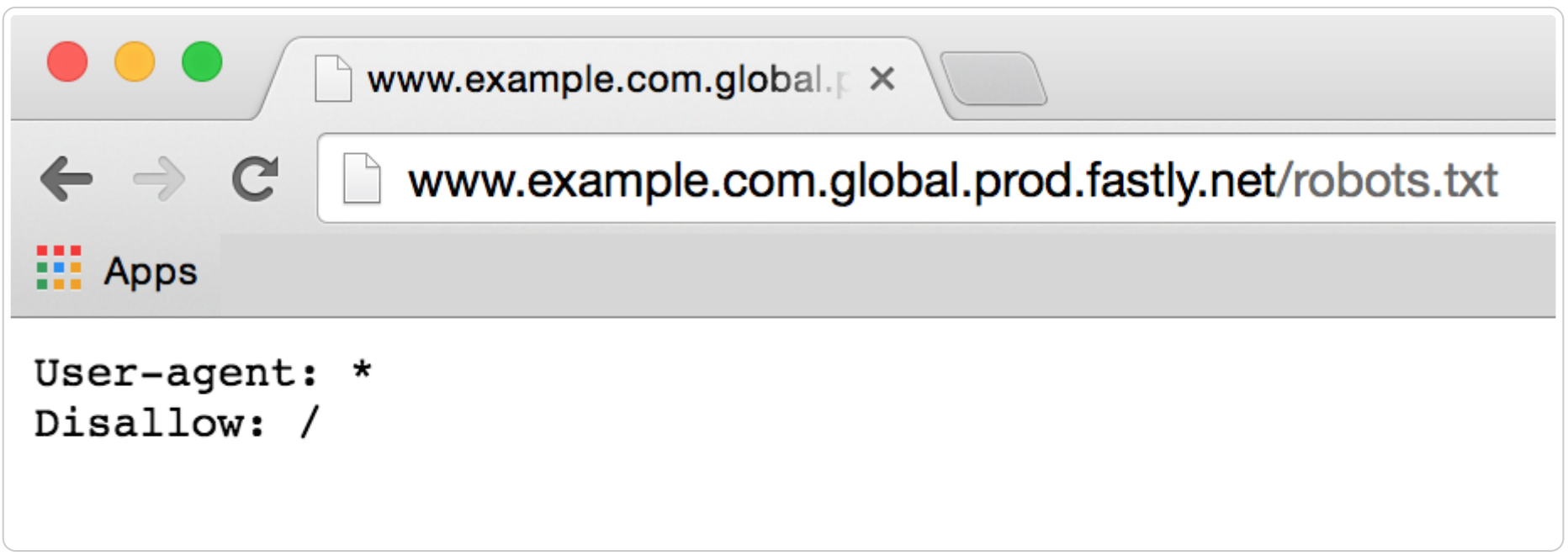
10. Click the **Save** button.
11. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> For an in-depth explanation of creating custom responses, check out our Responses Tutorial.

Why can't I customize my robots.txt file with global.prod.fastly.net?

Adding the `.global.prod.fastly.net` extension to your domain (for example, `www.example.com.global.prod.fastly.net`) via the browser or in a curl command can be used to test how your production site will perform using Fastly's services.

To prevent Google from accidentally crawling this test URL, we provide an internal robots.txt file that instructs Google's webcrawlers to ignore all pages for all hostnames that end in `.prod.fastly.net`.



This internal robots.txt file cannot be customized via the Fastly web interface until after you have set the CNAME DNS record for your domain to point to `global.prod.fastly.net`.

Creating error pages with custom responses

Last updated: 2019-04-22

https://docs.fastly.com/en/guides/creating-error-pages-with-custom-responses

The default error responses served by Fastly can be jarring for your users, especially when using Fastly for consumer applications. To mitigate this, consider configuring your service to present them with a custom page or a synthetic response when Fastly receives an error code from your origin.

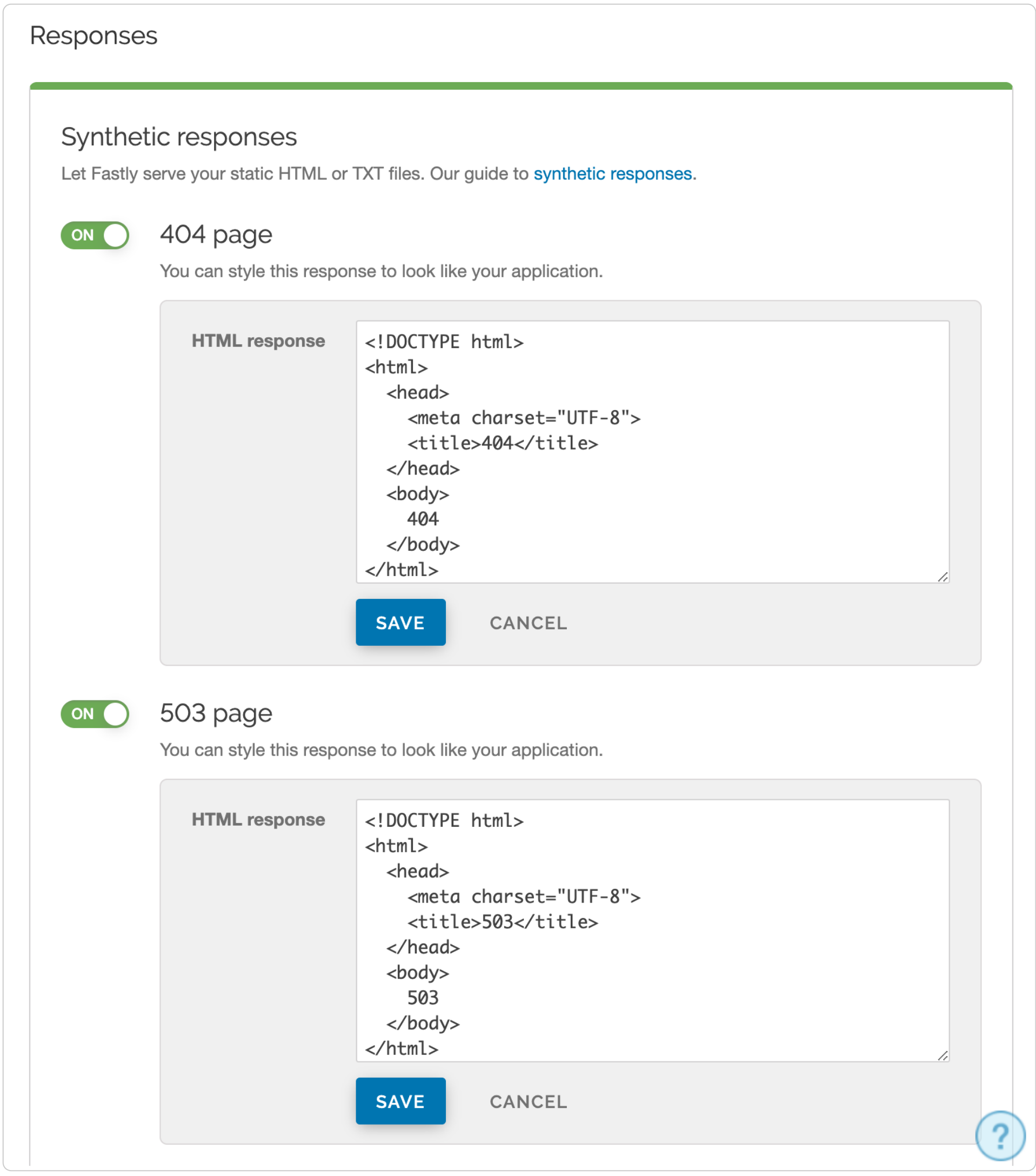
Fastly offers two quick configuration options for creating 404 and 503 error pages directly in the web interface, but you can also use the interface to create error pages for other status codes. If you're working with large blocks of content when styling your error pages, consider creating custom responses using VCL snippets instead.

> **TIP**
>
> Instead of an error message, Fastly can optionally serve stale content when there is a problem with your origin server. For more information, see our guide on serving stale content.

Creating error pages for 404 and 503 errors

To create error pages with custom responses for 404 and 503 errors, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. To create error pages with custom responses for 404 and 503 errors, click the **404 page** and **503 page** switches.



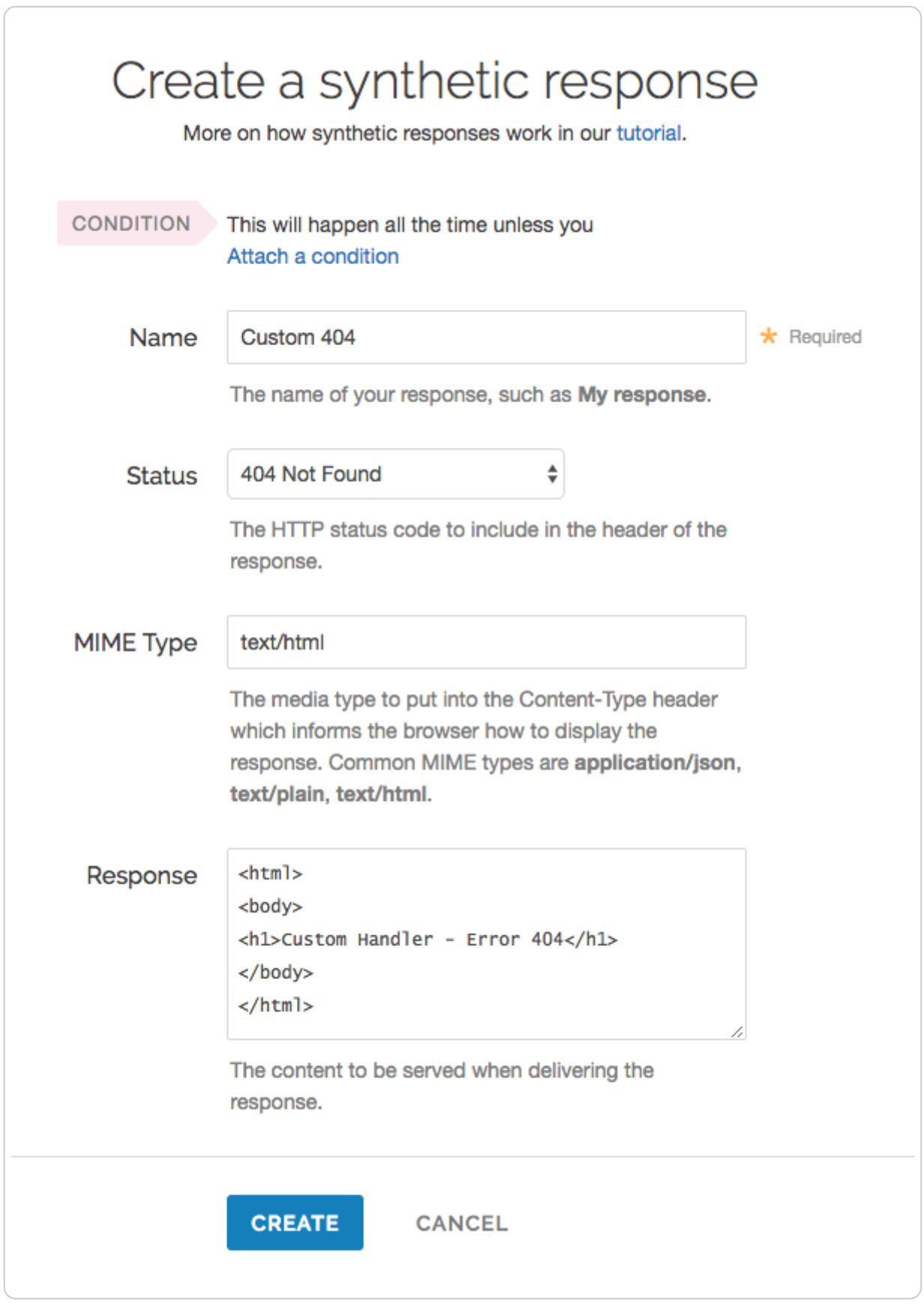
6. In the **HTML response** fields, customize the response for the 404 and 503 error pages.
7. Click the **Save** buttons to save the responses.
8. Click the **Activate** button to deploy your configuration changes.

Creating error pages for other status codes

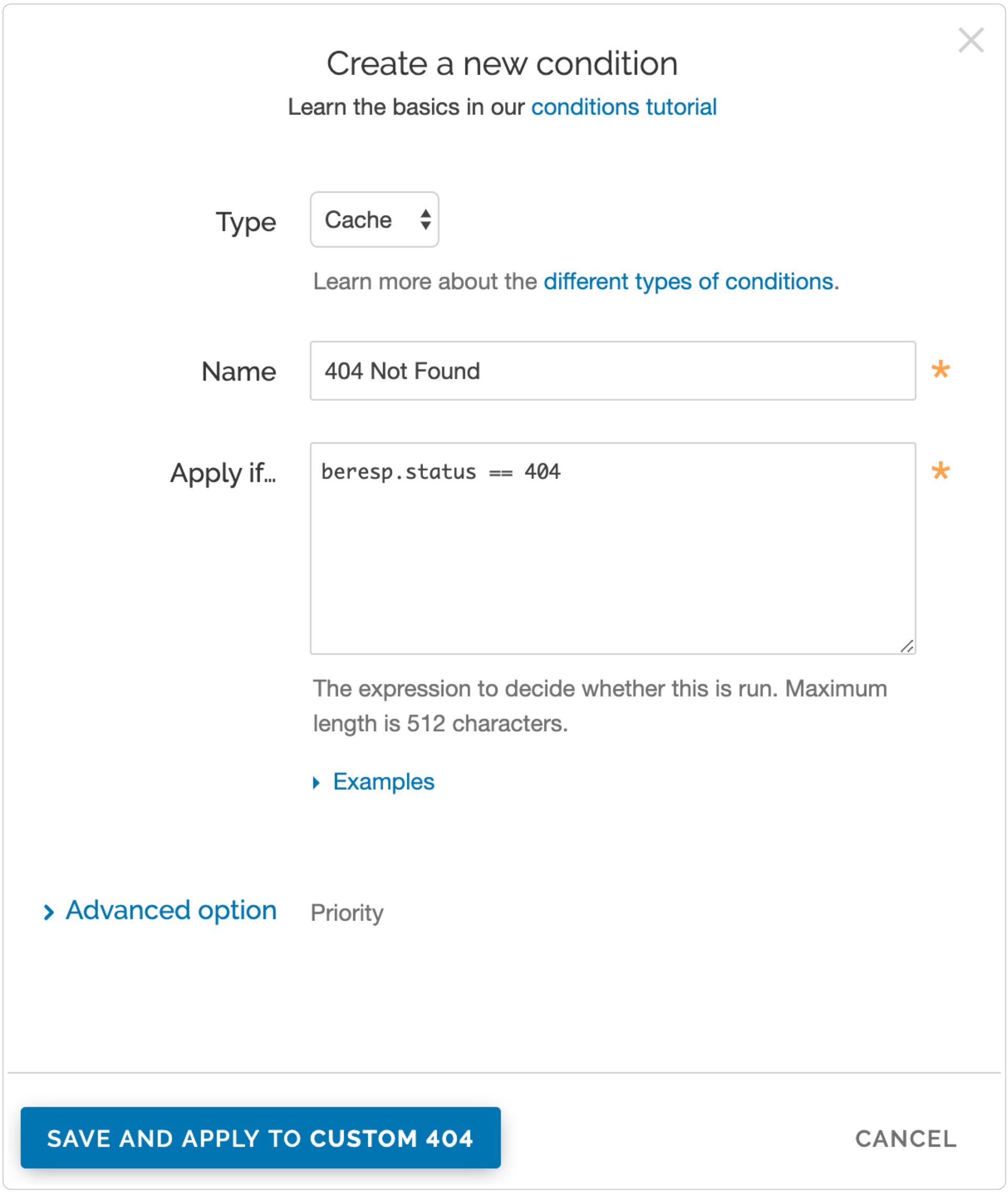
You can also create error pages for other HTTP status codes. We provide example HTML, but you can use any HTML you see fit. The response object will require that you use a condition in order for it to be served.

To create and configure an error page for an HTTP status code other than 404 or 503, follow the steps below to create the custom response and the condition under which it should be applied using the web interface:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Set up advanced response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, enter a name for the response you're creating (e.g., `Custom 404`).
   - From the **Status** menu, select the appropriate status (e.g., `404 Not Found`).
   - In the **MIME Type** field, specify the Content-Type of the response (e.g., `text/html`).
   - In the **Response** field, enter the content to be served when delivering a response.
7. Click the **Create** button. Your new response appears in the list of responses.
8. Click the **Attach a condition** link to the right of the name of your new response. The Create a new condition window appears.



9. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select the type of condition you're creating (e.g., `Cache`).
   - In the **Name** field, enter a name for the condition you're creating (e.g., `404 Not Found`).
   - In the **Apply if** field, enter the condition under which the new response occurs in the following format:

     `beresp.status == ###`

   where `###` equals the status condition you're creating the response for. For example, using the value of `beresp.status == 404` in the **Apply if** field here tells Fastly to use this response object whenever origin servers return a 404 status. (See the Conditions guides for more detailed information on conditions.)
10. Click the **Save and apply to** button. The condition is created and applied to the custom response object you made earlier.

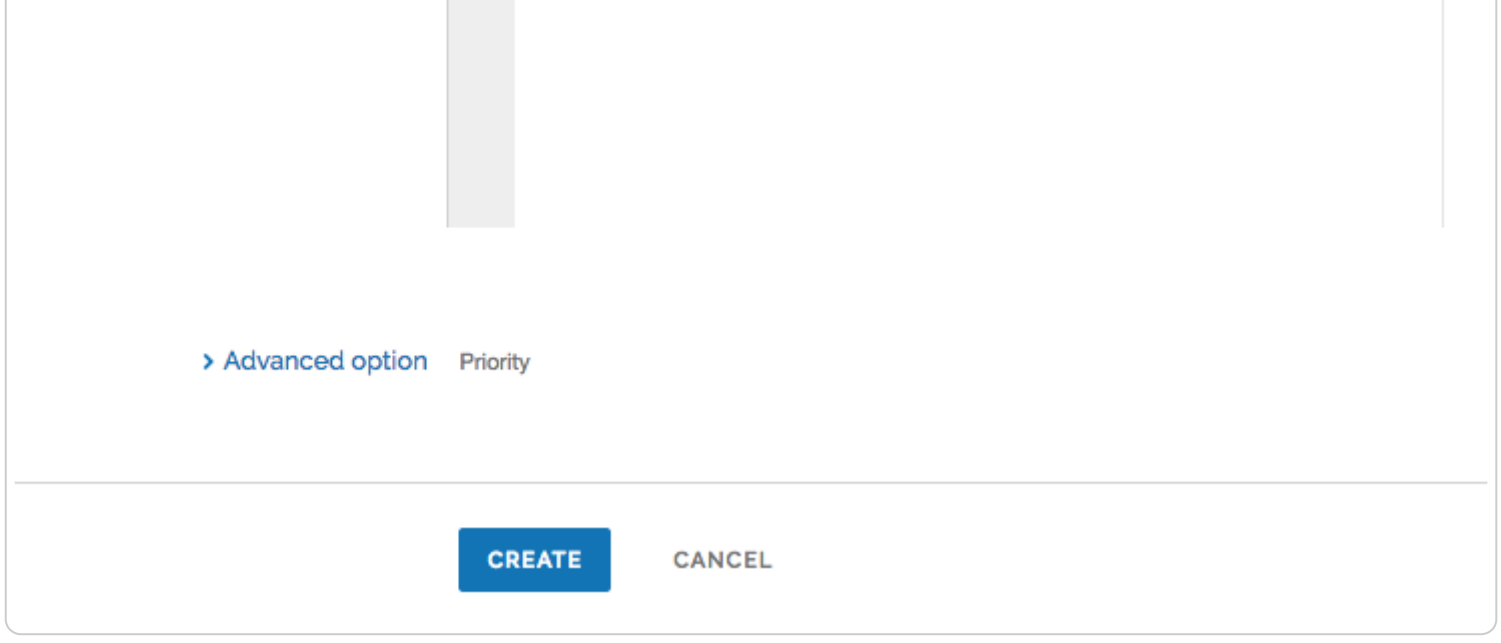
11. Click the **Activate** button to deploy your configuration changes. Fastly will now serve your custom HTML error page when required.

Creating custom responses using VCL Snippets

To create the custom response using VCL Snippets, create two separate snippets: one to trigger the condition for an internal Fastly error and the second to create the response to that error.

Create a VCL Snippet for a condition

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click **Create Snippet**. The Create a VCL snippet page appears.



6. In the **Name** field, enter an appropriate name (e.g., `Catch Error for Custom Response`).
7. From the **Type** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **fetch (vcl_fetch)**.
9. In the **VCL** field, add the following condition:

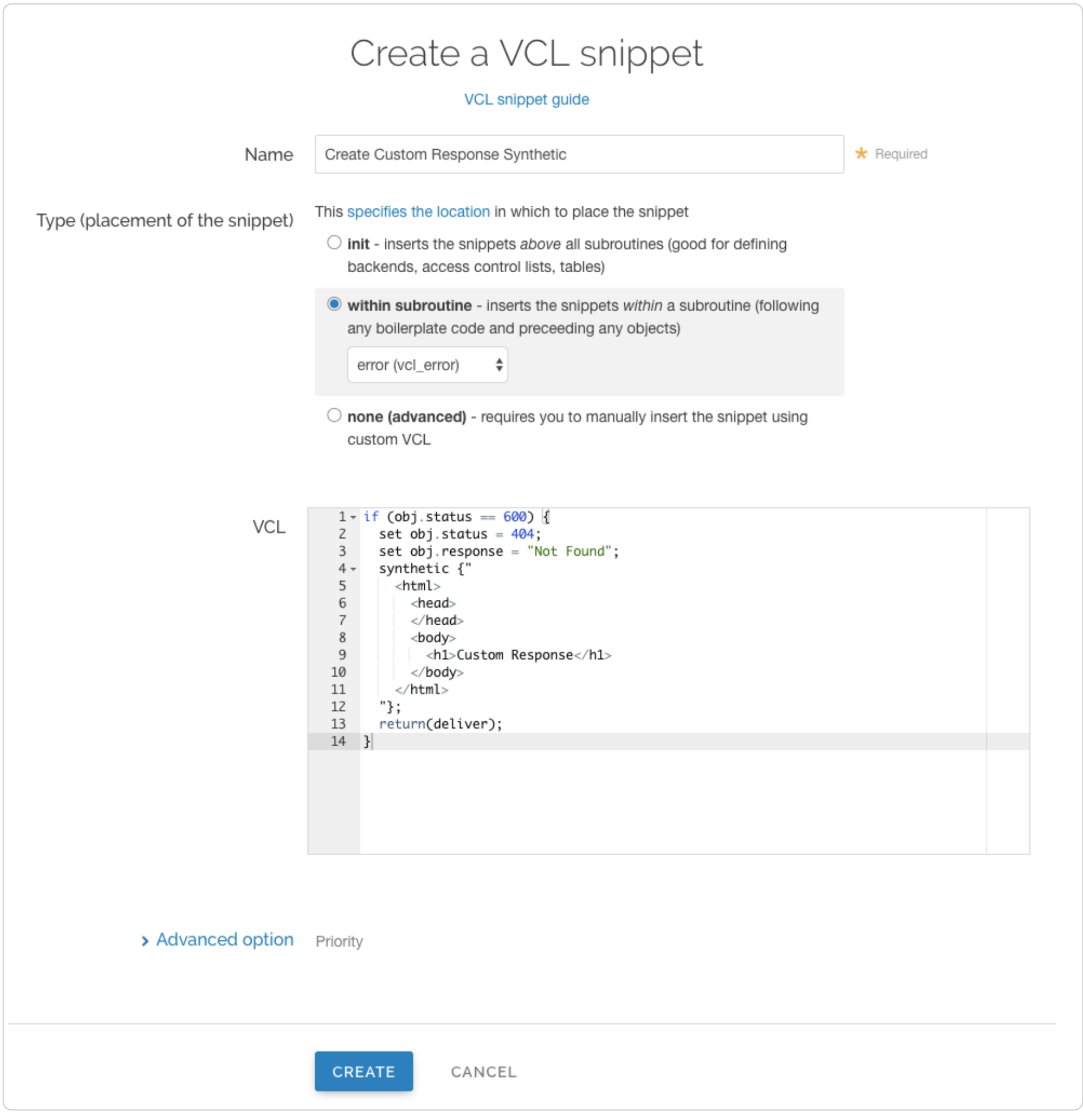
```
if (beresp.status == ###) {
  error 600 "### Custom Response"
}
```

where `###` is the status condition you're creating the response for. The error code used here, `600`, is a random number that doesn't conflict with standard HTTP error codes. Consider using custom error code numbers in the 600's or 700's to avoid confusion.

10. Click **Create** to create the snippet.

Create a VCL Snippet for a synthetic response

1. Click the **VCL Snippets** link. The VCL Snippets page appears.
2. Click **Create Snippet**. The Create a VCL snippet page appears.



3. In the **Name** field, enter an appropriate name (e.g., `Create Custom Response Synthetic`).
4. From the **Type** controls, select **within subroutine**.
5. From the **Select subroutine** menu, select **error (vcl_error)**.

6. In the **VCL** field, add the following condition:

```
if (obj.status == 600) {
  set obj.status = 404;
  set obj.response = "Not Found";
  synthetic {"
    <html>
      <head>
      </head>
      <body>
        <h1>Custom Response</h1>
      </body>
    </html>
  "};
  return(deliver);
}
```

replacing `Custom Response` with your custom, synthetic response. This VCL tells Fastly to respond with your custom response if a request for an object meets the condition you created in `vcl_fetch`.

> **NOTE**
>
> Synthetic responses don't have a character limit, but including them in the custom VCL file may push that file over its size limit.

7. Click **Create** to create the snippet.

8. Click the **Activate** button to deploy your configuration changes.

Generating HTTP redirects at the edge

Last updated: 2021-10-19

https://docs.fastly.com/en/guides/generating-http-redirects-at-the-edge

When users request information from your origin servers, you may want to redirect them for various reasons. For example, you may want to redirect them to pages that have been moved or updated since the last time they were requested. Or you may want to redirect their requests at the apex domain (e.g., `example.com`) level to a www subdomain (e.g., `www.example.com`) instead. You can send these redirects from the edge rather than having to go to origin. Redirecting traffic from one URL to another can be done by creating a synthetic response with the appropriate redirect status code and then creating a content rule with the proper Location header. Redirecting all traffic can be done by changing a single setting.

> **TIP**
>
> This guide describes how to create normal 301 (and 302) redirects from one URL to another and how to redirect apex domains to www subdomains. To automatically redirect *all* HTTP requests to HTTPS, follow the instructions in our guide forcing a TLS redirect instead. It describes an easier way to configure this.

Redirect traffic from one URL to another

Use these steps to redirect traffic from one URL to another. For example, to redirect a request for a page that has been moved since the last time it was requested. Begin by creating a synthetic response with the appropriate redirect status code. Then, create a content rule with the proper Location header.

Create a new response and condition

To generate redirects at the edge, start by creating a new response with the appropriate status code and a new condition describing when the response can be applied.

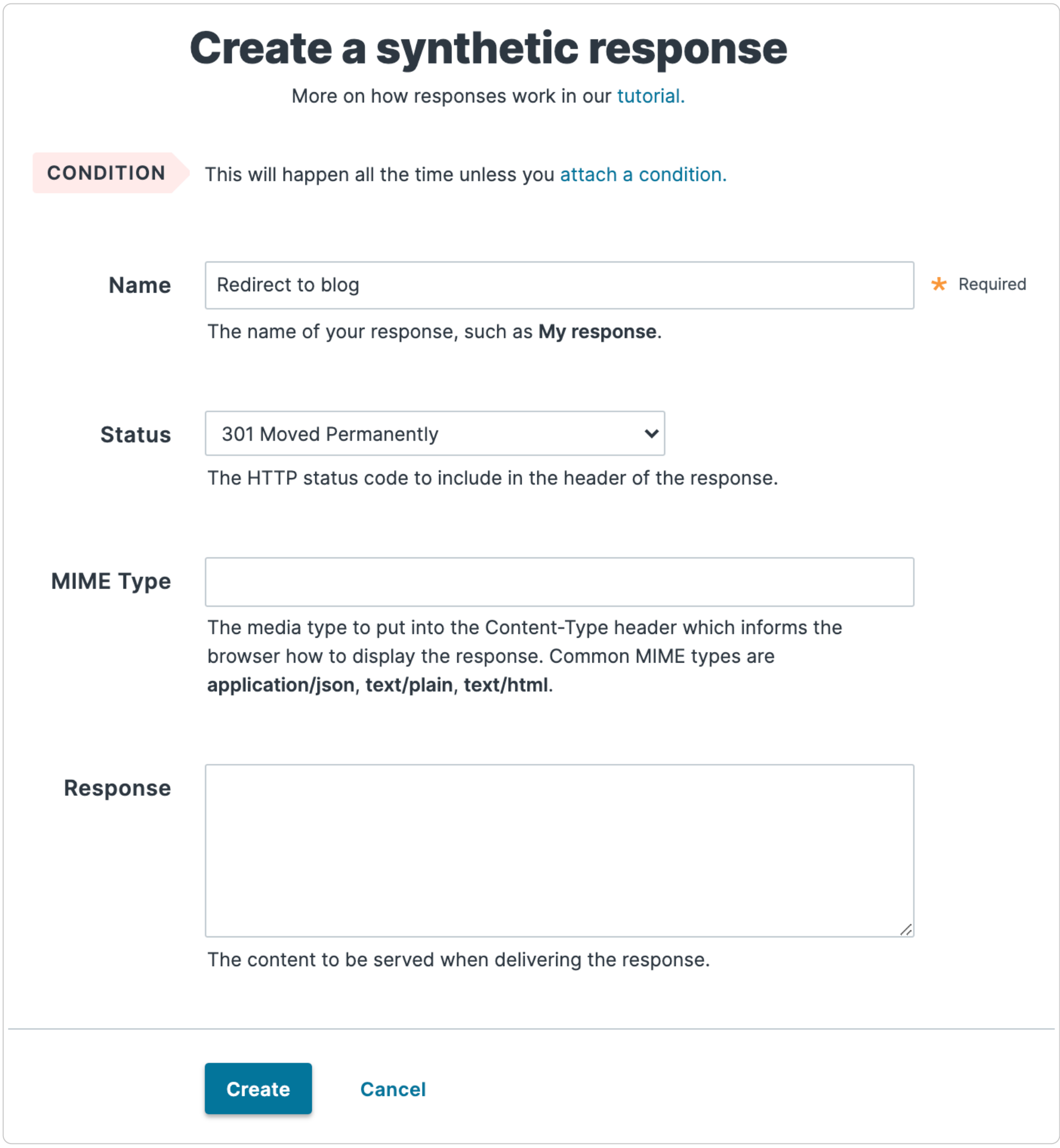
1. Log in to the Fastly web interface.

2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.

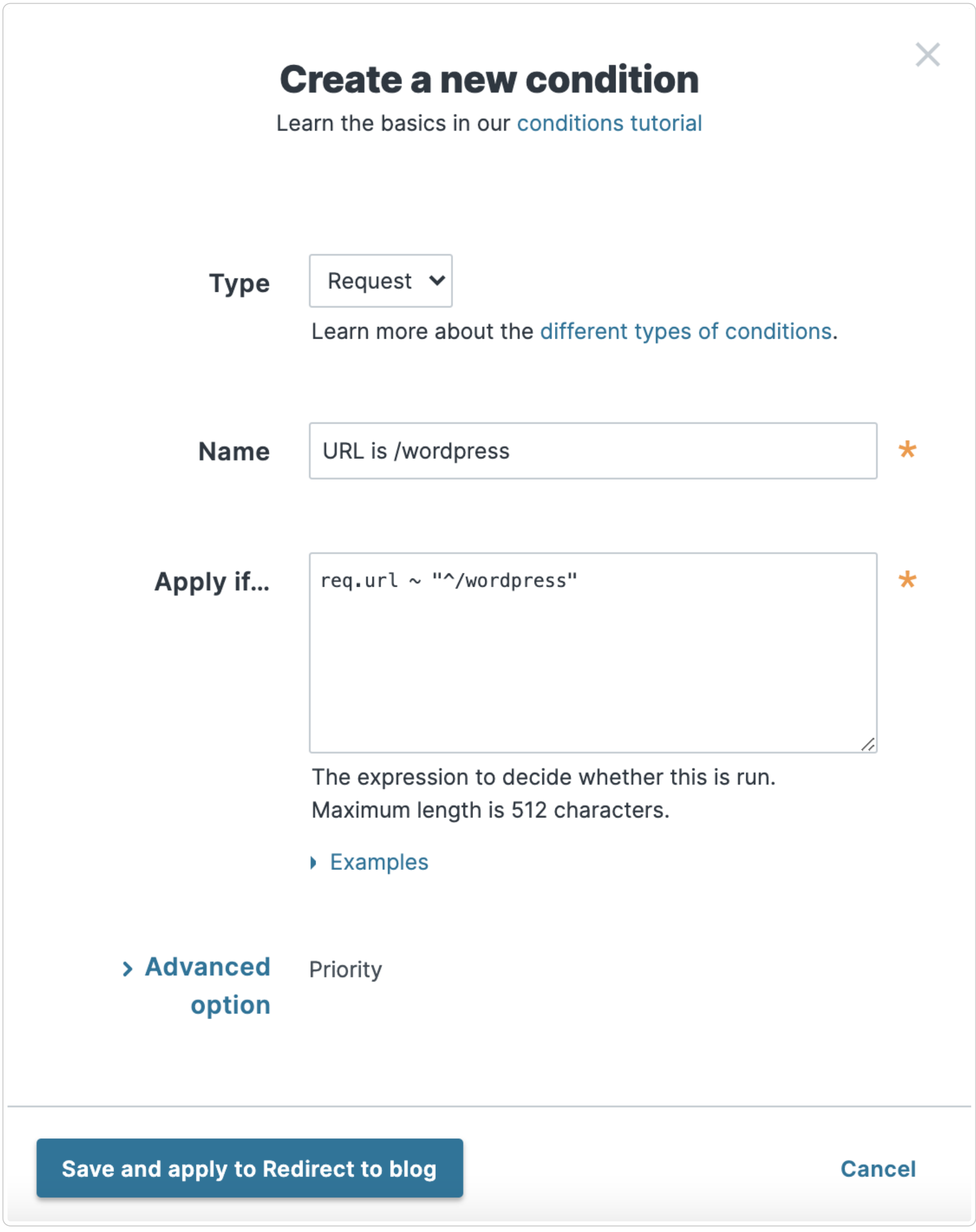
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.

4. Click the **Content** link. The Content page appears.

5. Click the **Set up advanced response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, enter a meaningful name for your response (e.g., `Redirect to blog`). This name is displayed in the Fastly web interface.
   - From the **Status** menu, select the HTTP status code that should be included in the header of the response (e.g., `301 Moved Permanently` or `302 Moved Temporarily` for redirections).
   - Leave the **MIME Type** field blank.
7. Click the **Create** button to create the new response.
8. On the **Content** page, click the **Attach a condition** link to the right of the new response you just created. The Create a new request condition window appears.

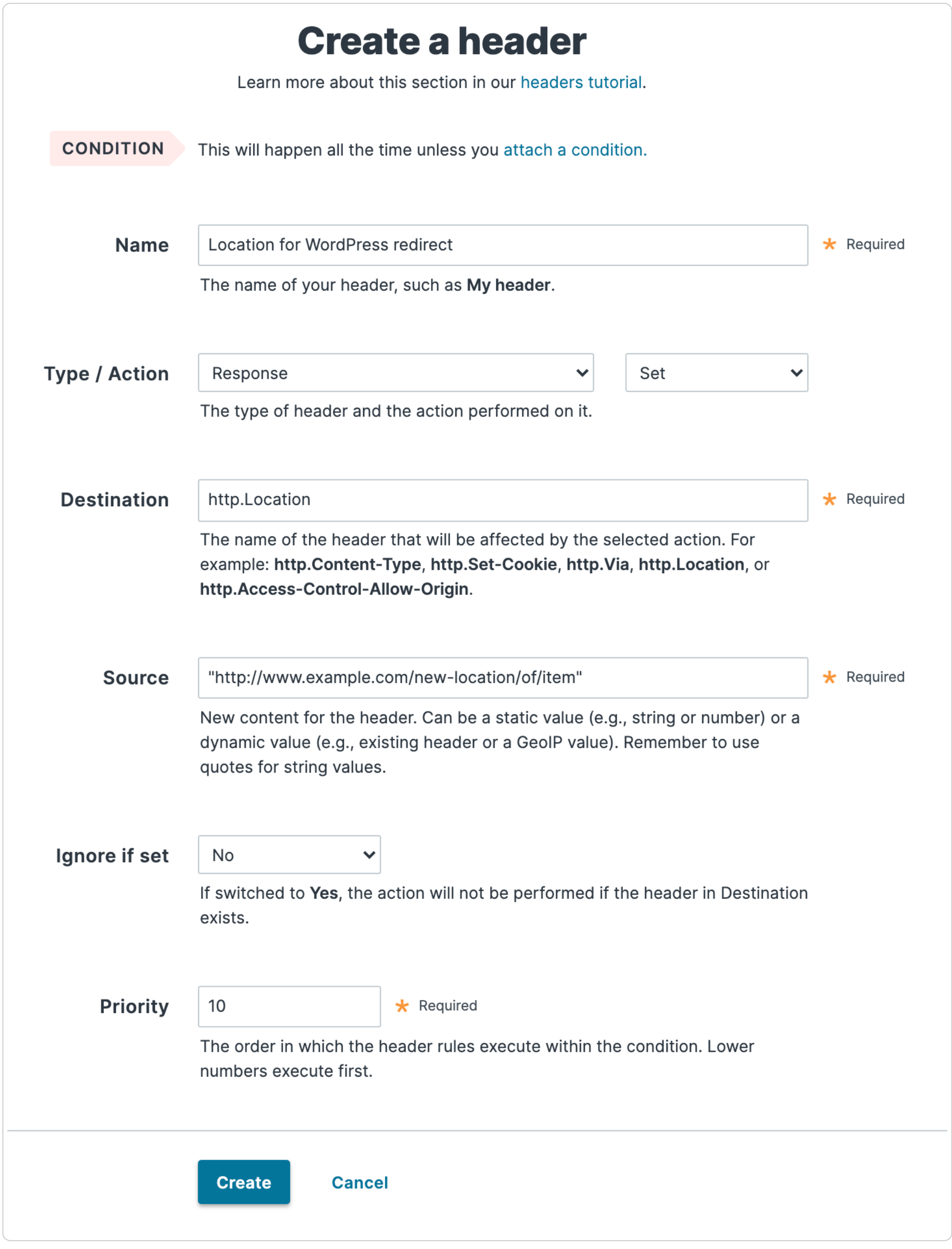


9. Fill out the **Create a new request condition** fields as follows:
   - From the **Type** menu, select the type of condition you want to create (e.g., `Request`).
   - In the **Name** field, enter a meaningful name for your condition (e.g., `URL is /wordpress`). This name is displayed in the Fastly web interface.
   - In the **Apply if** field, enter the logical expression to execute in VCL to determine if the condition resolves as True or False (e.g., `req.url ~ "^/wordpress"`).
10. Click the **Save and apply to** button to create the new request condition.

Create a new header and condition

Complete the creation of a synthetic redirect by creating a new header and condition that modifies that response by adding the Location header based on the status code and the matching URL. This ensures the redirect only applies when both of those are true.

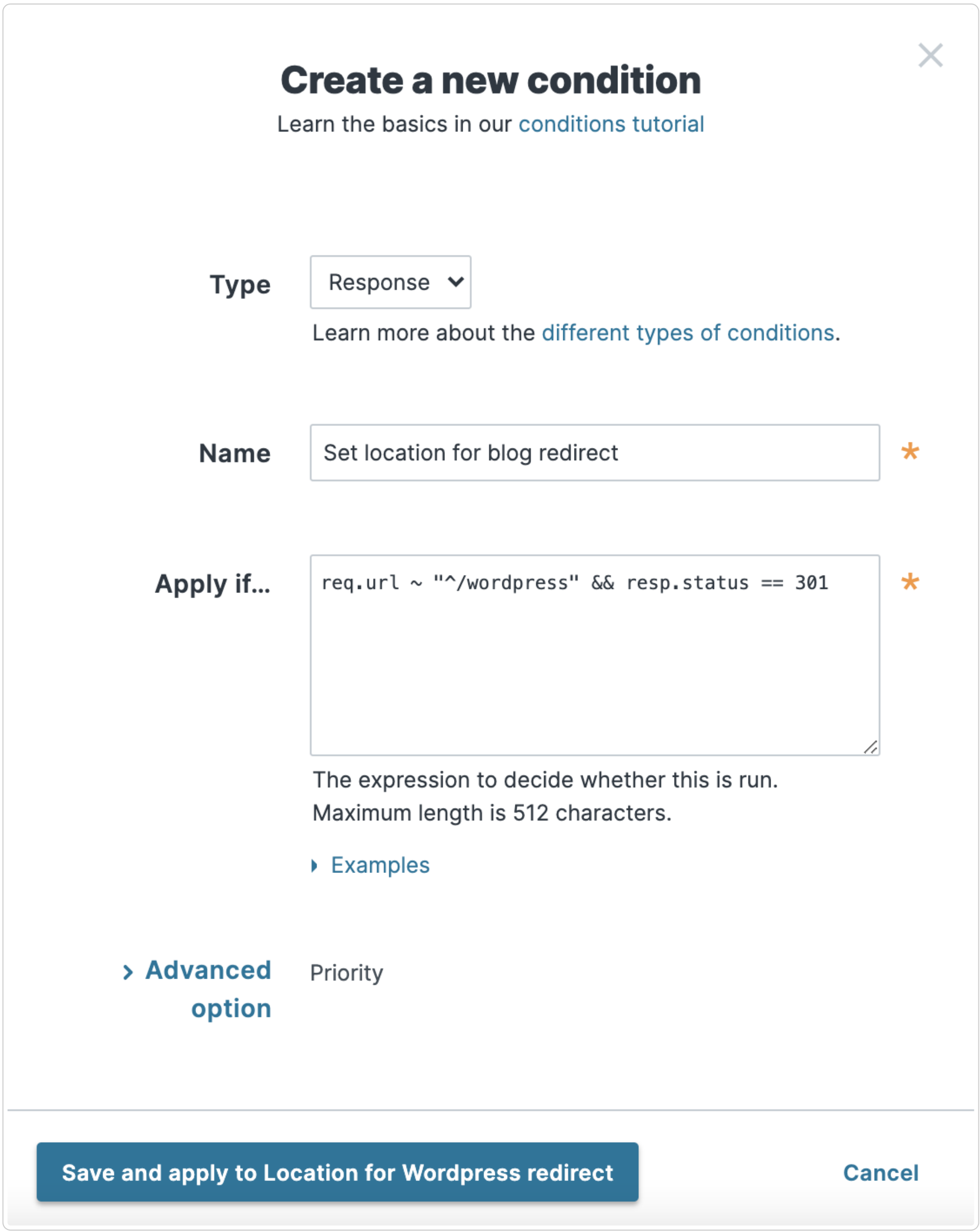
1. On the **Content** page, click the **Create header** button. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a meaningful name for your header (e.g., `Location for WordPress redirect`).

- From the **Type** menu, select **Response**, and from the **Action** menu, select **Set**.
- In the **Destination** field, enter `http.Location`.
- In the **Source** field, enter the source location of the new content (e.g., `"http://www.example.com/new-location/of/item"`).
- Leave the **Ignore if set** and **Priority** fields at their default settings.

3. Click the **Create** button to create the new header.
4. On the **Content** page, click the **Attach a condition** link to the right of the new header you just created.
5. Click **Create a new condition**. The Create a new condition window appears.



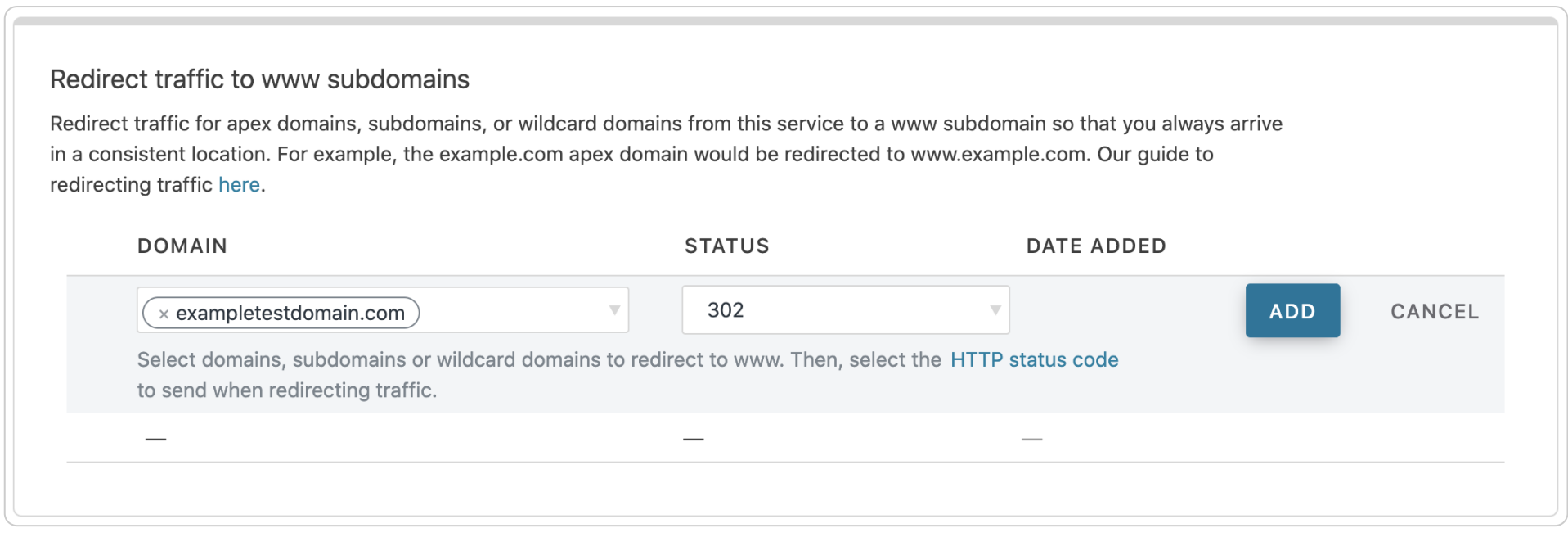
6. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select **Response**.
   - In the **Name** field, enter a meaningful name for your condition (e.g., `Set location for blog redirect`).
   - In the **Apply if** field, enter the logical expression to execute in VCL to determine if the condition resolves as true or false (e.g., `req.url ~ "^/wordpress" && resp.status == 301`). The `resp.status` needs to match the response code generated in the response above.
7. Click the **Save and apply to** button to create the new condition.
8. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> These responses use a custom status number >500. They will appear as errors on the Real-time stats page even though they are desired behavior.

Redirect traffic to www subdomains

Use the **Redirect traffic to www subdomains** setting to redirect traffic for apex domains, subdomains, or wildcard domains to a www subdomain so that you always arrive in a consistent location. For example, you could redirect requests from `example.com` to `www.example.com`.

1. If you haven't already done so, create a DNS record for the apex domain, subdomain, or wildcard domain you want to redirect. See our guide on apex domains for more information.
2. Log in to the Fastly web interface.
3. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
4. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
5. If you haven't already done so, add the apex domain, subdomain, or wildcard domain that you want to redirect. See our instructions on adding additional domains to your service for more information.
6. Click the **Settings** link. The Settings page appears.
7. In the **Redirect traffic to www subdomains** area, click **Add apex redirect**. The Domain and Status selection menus appear.



8. From the **Domains** menu, select the domain, subdomain, or wildcard domain you want to redirect. You can select more than one.
9. From the **Status** menu, select the HTTP status code to send when redirecting traffic from the domains you selected.
10. Click **Add**.
11. Repeat the steps for redirecting traffic to www subdomains for any other domains you want to redirect.
12. Click the **Activate** button to deploy your configuration changes.

Editing a www redirect

You can edit redirects at any time by following the steps below:

1. Navigate to the **Redirect traffic from Apex to www subdomains** area.
2. Hover your cursor over the entry you want to edit, then click the pencil icon that appears.
3. Edit the redirect by doing one of the following:
   - to add another domain to the redirect, select it from the **Domain** menu selections.
   - to remove an existing domain from the redirect, click the X icon to the left of the domain name in the **Domain** menu.
   - to change the status of the redirect for the selected domains, select a new one from the **Status** menu.
4. Click the **Save** button. The changes you make will be immediately applied to your configuration. If you can't save edits to your redirect, make sure at least one domain appears in the Domains menu. You can't [delete a www redirect] by deleting all the listed domains.

Deleting a www redirect

You can delete redirects at any time by following the steps below:

1. Navigate to the **Redirect traffic from Apex to www subdomains** area.
2. Hover your cursor over the entry you want to edit, then click the trash can icon that appears.
3. Click the **Confirm and delete** button.

Using custom VCL

You can also use [custom VCL] to redirect traffic from one URL to another. For more information, see our [redirects tutorial].

Responses tutorial

Last updated: 2018-08-16

https://docs.fastly.com/en/guides/responses-tutorial

Fastly allows you to create custom HTTP responses that are served directly from the cache without storing the page on a server. Responses are commonly used to serve small static assets that seldom change and maintenance pages that are served when origins are unavailable. This tutorial shows you how to create your own responses.

NOTE

We assume that you already know how to edit and deploy configurations using the [web interface]. If you are not familiar with basic editing using the application, see [our help guides] to learn more.

Creating a quick response

Fastly provides features that allow you to quickly enable and configure responses for a [robots.txt file] and [404 and 503 errors]. For more information, see our guides on [creating and customizing a robots.txt file] and [creating error pages with custom responses].

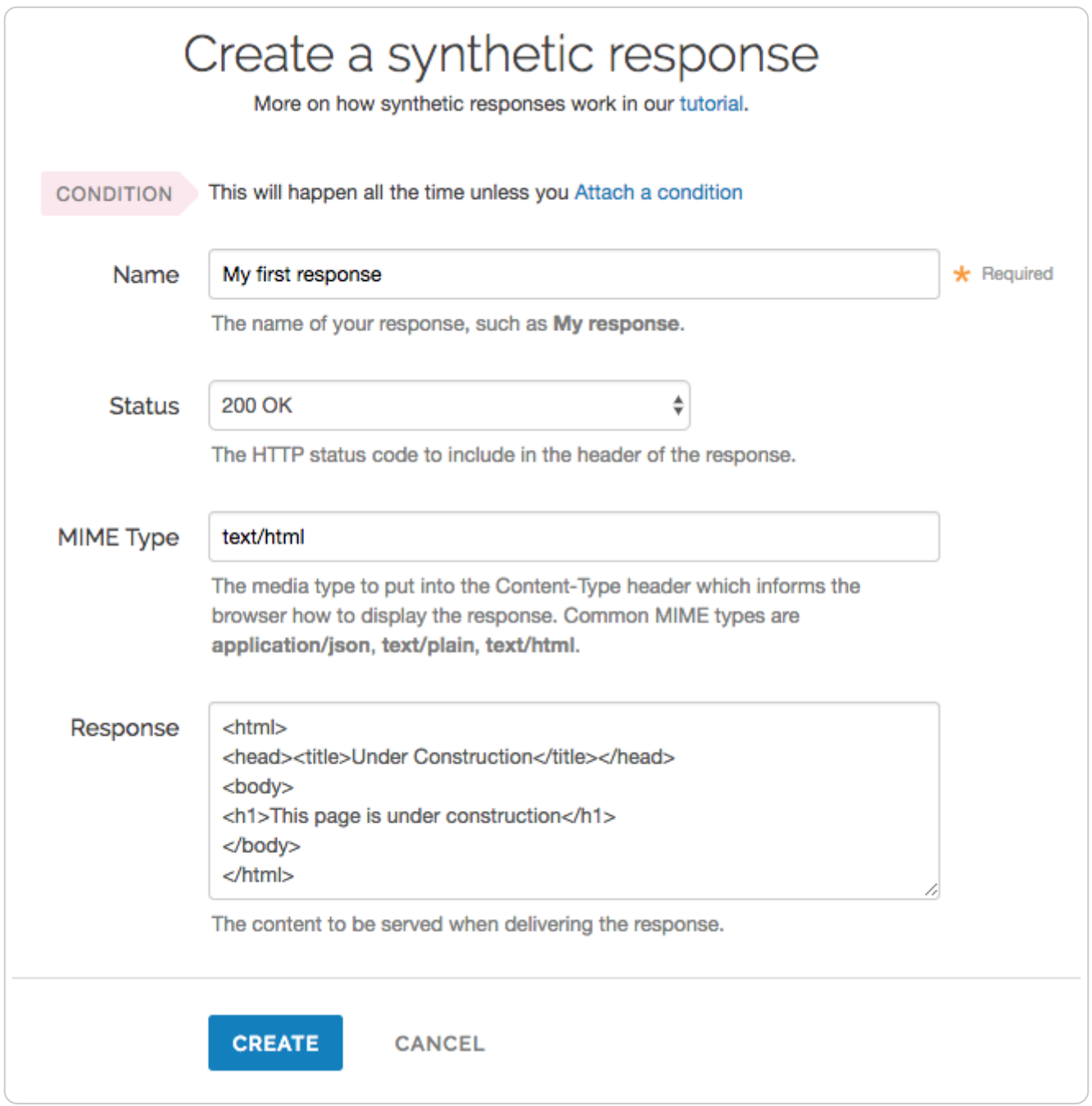
Creating an advanced response

You can create an advanced response to specify the HTTP status code, MIME type, and content of the response. An advanced response has three basic attributes:

- Status - An HTTP status code to include in the header of the response
- Response - The content to be served when delivering the response
- Description - A human readable identifier for the response

By setting these three attributes and adding a condition to the response, you can very quickly get one up and running on your service. To create an advance response, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Set up advanced response** button. The Create a synthetic response page appears.



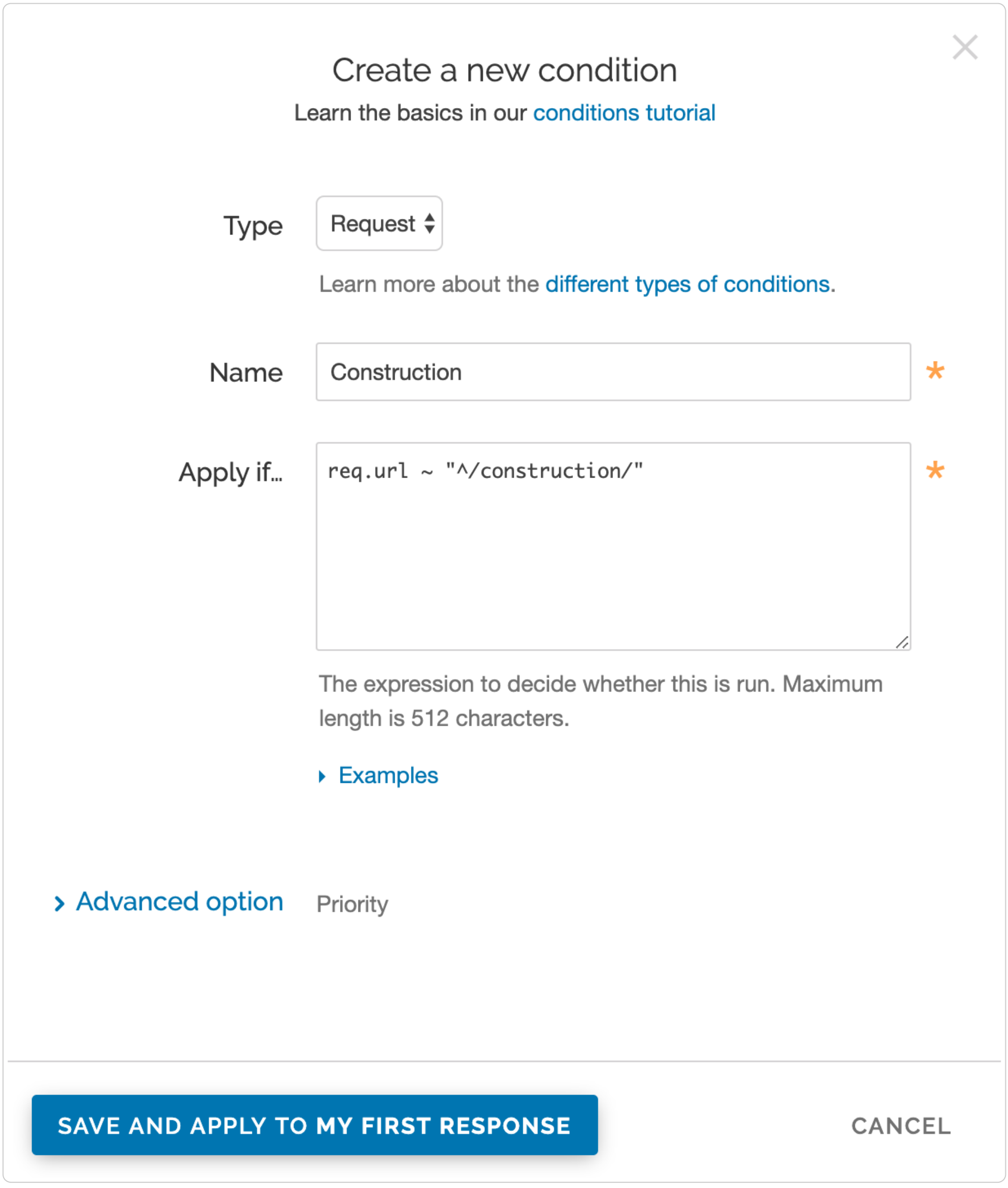
6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, enter a human-readable name for the response (e.g., `My first response`).
   - From the **Status** menu, select the appropriate status (e.g., `200 OK`).
   - In the **MIME Type** field, enter the content type of the response (e.g., `text/html`).
   - In the **Response** field, enter the response you want to appear when the conditions are met.
7. Click the **Create** button to create your custom response.

Your new response appears in the list of responses.

Adding conditions

To add a condition, follow the steps below:

1. Click the **Attach condition** link to the right of the new response. The Create a new condition window appears.



2. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select the type of condition you want to create.
   - In the **Name** field, enter a human-readable name for the condition so that it can be easily identified in the future.
   - In the **Apply if** field, enter the condition under which the new response occurs. The condition should take the following format: `req.url ~ "^/construction/"` equals the request condition you're creating the response for.
   - In the **Priority** field, enter a priority if needed. Condition priorities are only needed in "interesting" cases, and can usually be left at the default "10" for all response conditions.
3. Click the **Save and apply to** button.
4. Click the **Activate** button to deploy your configuration changes.

Fastly now serves your custom response page when the condition is met.

§

These articles describe setup and configuration guidelines for setting up live stream delivery or video on-demand.

https://docs.fastly.com/en/guides/configuration#_video

Adaptive bitrate playback URL guidelines

Last updated: 2018-09-04

https://docs.fastly.com/en/guides/adaptive-bitrate-playback-url-guidelines

Fastly's On-the-Fly Packaging (OTFP) service supports any directory structure you might use to store different quality levels of a video. To construct adaptive bitrate (ABR) playback URLs for a video, make directory paths to that video unique. Ensure all the files associated with a particular video (e.g., quality levels, subtitles) exist under a single directory.

IMPORTANT

If you aren't sure how to configure OTFP, contact support@fastly.com before making any changes.

For example, say you had a video called Example Video. Assuming you had multiple quality levels and associated files for Example Video, the following directory structure would provide the best start to constructing ABR playback URLs:

Directory path example	Description
`/foo/bar/example-video/`	Base folder unique to this video
`/foo/bar/example-video/480p_30fps.mp4`	Quality level 480p with 30 frames per sec with audio
`/foo/bar/example-video/720p_30fps.mp4`	Quality level 720p with audio with 30 frames per sec with audio
`/foo/bar/example-video/720p_60fps.mp4`	Quality level 720p with audio with 60 frames per sec with audio
`/foo/bar/example-video/1080p_30fps.mp4`	Quality level 1080p with audio with 30 frames per sec with audio
`/foo/bar/example-video/1080p_60fps.mp4`	Quality level 1080p with audio with 60 frames per sec with audio
`/foo/bar/example-video/4k_30fps.mp4`	Quality level 4k with audio with 30 frames per sec with audio

With this directory structure, the ABR playback URL for all videos in the base directory would follow this template:

```
http://example.com/path/to/dir/<video_id>/<quality_file1_name_wo_ext>,<quality_file2_name_wo_ext>,...,<quality_fileN_name_wo_ext>/master.<f4m|m3u8|mpd>
```

For example, the ABR playback URLs for Example Video in every format would be:

Format	Example URL
HDS	`http://example.com/foo/bar/example-video/480p_30fps,720p_30fps,720p_60fps,1080p_30fps,1080p_60fps,4k_30fps/master.f4m`
HLS	`http://example.com/foo/bar/example-video/480p_30fps,720p_30fps,720p_60fps,1080p_30fps,1080p_60fps,4k_30fps/master.m3u8`
MPEG-DASH	`http://example.com/foo/bar/example-video/480p_30fps,720p_30fps,720p_60fps,1080p_30fps,1080p_60fps,4k_30fps/master.mpd`

You can reduce the duplication in ABR playback URLs separating out the repeated prefix and suffix info as follows:

`<filename_prefix><filename_variable><filename_suffix_wo_ext>.mp4`

and the template would change to one of the following:

```
http://example.com/path/to/dir/<video_id>/<filename_prefix><quality_file1_variable_name_wo_ext>,<quality_file2_variable_name_wo_ext>,...,<quality_fileN_variable_name_wo_ext>,<filename_suffix_wo_ext>/master.<f4m|m3u8|mpd>
```

```
http://example.com/path/to/dir/<video_id>/<filename_prefix><quality_file1_variable_name_wo_ext>,<quality_file2_variable_name_wo_ext>,...,<quality_fileN_variable_name_wo_ext>/master.<f4m|m3u8|mpd>
```

```
http://example.com/path/to/dir/<video_id>/<quality_file1_variable_name>,<quality_file2_variable_name>,...,<quality_fileN_variable_name>,<filename_suffix_wo_ext>/master.<f4m|m3u8|mpd>
```

IMPORTANT

To use token validation with ABR manifest URLs, special modifications must be made using custom VCL. Contact support@fastly.com for assistance.

Collecting OTFP metrics

Last updated: 2018-09-04

https://docs.fastly.com/en/guides/collecting-otfp-metrics

Fastly allows you to collect and process On-the-Fly Packaging (OTFP) service metrics for analysis using a combination of custom VCL updates and specific log streaming settings. Once you've set up OTFP metrics collection through remote log streaming you can use any of a number of third-party and open source software options to aggregate your logging data for visualization and further analysis.

IMPORTANT

If you aren't sure how to configure OTFP, contact support@fastly.com before making any changes.

Upload custom VCL

1. Before uploading custom VCL, review the caveats of mixing and matching Fastly VCL with custom VCL.
2. Add the following custom VCL to your Fastly VCL:

```
sub vcl_deliver {
  # Identify Request type
  if (req.url.ext ~ "m3u8|ts|aac|webvtt") {
    set resp.http.Otfp-Format = "HLS";
  } else if (req.url.ext ~ "mpd|m4s") {
    set resp.http.Otfp-Format = "DASH";
  } else {
    set resp.http.Otfp-Format = "OTHER";
  }

  # Extract name-value pairs Otfp Info herder
  if (resp.http.X-Fastly-Otfp-Info) {
    set resp.http.Otfp-SS = regsub(resp.http.X-Fastly-Otfp-Info, ".*ss=(\S+).*", "\1");
    set resp.http.Otfp-SL = regsub(resp.http.X-Fastly-Otfp-Info, ".*sl=(\S+).*", "\1");
    set resp.http.Otfp-VL = regsub(resp.http.X-Fastly-Otfp-Info, ".*vl=(\S+).*", "\1");

    # Resolution (rs name-value) not available for audio-only segments
    if (resp.http.X-Fastly-Otfp-Info ~ ".*rs=(\S+).*") {
      set resp.http.Otfp-RS = re.group.1;
    } else {
      set resp.http.Otfp-RS = "-";
    }
  }
#FASTLY deliver

  return(deliver);
}
```

Create a logging endpoint

Follow the instructions to set up remote log streaming for your account and when creating your specific logging endpoint, set the **Format String** field to the following:

```
%h now.sec %r %>s %b resp.http.Otfp-Format resp.http.Otfp-SS resp.http.Otfp-SL resp.http.Otfp-VL resp.http.Otfp-RS
```

Control log file timing with a logging endpoint condition

To avoid excess log files, consider attaching a condition to the logging endpoint so logs are only sent when video segments are requested so that logging specifically exclude those files sent from Fastly's Origin Shield.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Logging** link. The logging page appears.
5. In the list of logging endpoints, find the endpoint you enabled when setting up remote log streaming, then click **Attach a condition**. The Create a new condition window appears.
6. Fill out the **Create a new condition** window as follows:
   - In the **Name** field, enter a human-readable name for the condition.
   - In the **Apply if** field, enter `resp.http.X-Fastly-Otfp-Info && !req.http.Fastly-FF`.
7. Click **Save and apply to**.

Analyze logging data

In addition to any Varnish variable, and a variety of Fastly's extensions to VCL, log files include the following video-specific fields:

- `ss` - video segment start presentation time in seconds
- `sl` - video segment duration in seconds
- `vl` - video duration in seconds
- `rs` - video track display resolution in pixels

You can use these fields to run queries for analysis and use what you discover to refine your video delivery settings.

Configuration guidelines for live streaming

Last updated: 2021-02-25

https://docs.fastly.com/en/guides/configuration-guidelines-for-live-streaming

The Fastly network can deliver live streams for any HTTP streaming technology, archived or recorded, on any public or private cloud storage service. When configuring VCL to deliver live streams, we recommend following these guidelines, which Customer Support can help you with.

Configure shielding

Configure shielding by designating a specific shield POP for your origin to ensure live streams remain highly available within the Fastly network. If your setup includes primary and alternate origins (e.g., for high profile live streams), be sure to select a shield POP close to each origin, one for each origin you define.

Configure video manifest and segment caching TTLs

In live streams, video manifests are periodically refreshed when new segments become available, specially for HLS. We recommend setting manifest file TTLs to less than half of the video segment duration, typically 1-2 seconds for 5-second video segments. For long DVRs and live-to-VOD transitions, set segment TTLs longer on shields and shorter on edge POPs such that they are served from memory (that is, less than 3600s).

The following VCL sample may help you implement different TTLs for video manifest and segments. It can also be added to your service using VCL Snippets:

```
sub vcl_fetch {
#FASTLY fetch

  # Set 1s ttls for video manifest and 3600s ttls for segments of HTTP Streaming formats.
  # Microsoft Smooth Streaming format manifest and segments do not have file extensions.
  # Look for the keywords "Manifest" and "QualityLevel" to identify manifest and segment requests.
  if (req.url.ext ~ "m3u8|mpd" || req.url.path ~ "Manifest") {
    set beresp.ttl = 1s;
    return (deliver);
  }
  else {
    if (req.url.ext ~ "aac|dash|m4s|mp4|ts" || req.url.path ~ "QualityLevel") {
      set beresp.ttl = 3600s;
      return (deliver);
    }
  }

  return (deliver);
}
```

Optionally, identify video manifests and segments using the MIME type.

Configure lower TTLs for errors

By default, Fastly honors the `Cache-Control` header from the origin to set TTLs for cacheable objects. However, origins may not send `Cache-Control` headers for non-200 or 206 HTTP status code responses. As a result, Fastly will only cache few status code responses with default TTLs configured, usually 3600s, to prevent large numbers of requests from hitting the origin. Uncacheable status code responses can be enabled for caching by setting `beresp.cacheable` flag to `true`.

For live streams, new video segments are added every few seconds. Typically, live stream transcoders are configured to generate 5s segments and manifests are refreshed after each new segment is available. Frequently, video players can make requests to segments not yet available or requests can return errors like 500 or 503 status codes. In such cases, status code responses should be made cacheable and should only be cached with TTLs small enough to give sufficient time for origins to recover (around 1s).

The following VCL sample may help you implement this and can also be added to your service using VCL Snippets:

```
sub vcl_fetch {
#FASTLY fetch

  # Set 1s ttl if origin response HTTP status code is anything other than 200 and 206
  if (!http_status_matches(beresp.status, "200,206")) {
    set beresp.ttl = 1s;
    set beresp.cacheable = true;
    return (deliver);
  }

  return (deliver);
}
```

Configure Streaming Miss

Configure Streaming Miss to reduce the time clients (players) must wait to begin downloading streams when Fastly's edge servers must fetch content from your origin. Streaming Miss should be enabled for video or audio objects only (these are sometimes called *chunks* or *segments*).

The following VCL sample may help you implement this. It can also be added to your service using VCL Snippets:

```
sub vcl_fetch {
#FASTLY fetch

  # Enable Streaming Miss only for video or audio objects.
  # Below conditions checks for video or audio file extensions commonly used in
  # HTTP Streaming formats.
  if (req.url.ext ~ "aac|dash|m4s|mp4|ts") {
    set beresp.do_stream = true;
  }

  return (deliver);
}
```

Configure automatic gzipping

Configure automatic gzipping for manifest files based on their file extension or content-type using the following table as a guide:

HTTP streaming format	file extension	content-type
Apple HLS	m3u8	application/x-mpegurl, application/vnd.apple.mpegurl
MPEG-DASH	mpd	application/dash+xml
Adobe HDS	f4m, bootstrap	application/f4m (for manifest), application/octet-stream (for bootstrap)
Microsoft HSS	N/A	application/vnd.ms-sstr+xml

Configure a CORS header

Configure a CORS header on your service to play audio or video content on a different domain.

Enable the experimental BBR congestion algorithm

The BBR TCP congestion control algorithm is an optional, TCP-related configuration that can help improve a client's experience. Unlike the default CUBIC congestion control algorithm, which is packet-loss-based and latency-insensitive, BBR is designed to maximize bandwidth while controlling latency.

WARNING

While expected to perform better than CUBIC (especially under transient packet losses), BBR is still a work-in-progress and implementing it may cause performance degradation for some users.

You can implement this algorithm by adding the following VCL to your service using VCL Snippets:

```
sub vcl_deliver {
#FASTLY deliver

  # set congestion algorithm for client requests
  if (!req.http.Fastly-FF && client.requests == 1) {
    set client.socket.congestion_algorithm = "bbr";
  }

  return(deliver);
}
```

TIP

TCP optimizations can be applied conditionally rather than applying them to all clients. For example, enable BBR only for clients within a specific ASN or ISP network like a mobile or wireless network.

Configure origin timeouts

Set appropriate origin timeouts to ensure new live stream segments are downloaded from origin in a timely manner. For example, for a live stream with 5s video segments, set the Origin Connect value to 1s and the First Byte and Between Bytes timeout values to 2s. Typically, these values should be configured such that Fastly can also retry another origin (if configured) before sending the

appropriate response on client requests.

Consider setting up failover (fallback) origins

Consider configuring your VCL to allow your origins to failover from high-profile primary streams to alternate streams in case of encoder failures or other issues (e.g., high resource utilization).

Configure real-time log streaming

For troubleshooting and debugging live streaming delivery issues, configure real-time log streaming and include TCP connection, caching, and different time-related metrics in `vcl_log`. For example, consider including:

- `fastly_info.state` (cache hits or misses)
- `client.socket.tcpi_rtt` (client round-trip time)
- `time.to_first_byte` (time from client request to the first byte being received)
- `time.elapsed` (time since the request started, which can be used to calculate response time or time-to-last-byte for both origin and clients)
- `client.as.number` and `client.as.name` (autonomous system number and name associated with client IP)
- `client.socket.tcpi_delta_retrans` (number of packets re-transmitted to the client)
- `client.socket.tcpi_snd_mss` (maximum segment size used to send responses to client)
- `client.requests` (number of requests on a connection so far)
- `client.socket.nexthop` (network path Fastly is sending the client response)
- `req.restarts` (number of request restarts typically indicates retry attempts)
- `server.datacenter` (the Fastly POP that served the request)
- `resp.http.content-length` and `resp.body_bytes_written` (actual bytes sent to client compared to what was expected to be sent)

These metrics can help you analyze throughput and may help you determine reasons a video player might switch quality levels during ABR playback.

Take advantage of surrogate key purging

All video segments and the manifest for a live stream can be purged using a single API call by using Fastly's surrogate key feature.

Manage live-to-VOD smoothly

Most encoders generate a separate video manifest when making the same live stream available for VOD. If your VOD manifest has the same URL as the live one, purge the live stream video manifest or wait for the caches to invalidate (as they will be set with low TTLs). If your setup archives the live stream as progressive mp4s, consider delivering them using Fastly's OTFP service.

> **NOTE**
>
> **Wowza integrations.** When configuring your Wowza origin server, be sure to select the Live HTTP Origin application type. If you select Live Edge, Wowza will always return a unique URL for manifest requests, resulting in extremely low cache hit.

Security

These articles provide information about the administrative, physical, and technical safeguards that protect the Fastly CDN service, as well as describe how to secure communications between Fastly and your origin servers and customers.

https://docs.fastly.com/en/guides/security

§

These articles describe how to restrict access to resources by allowing or blocking IP addresses with access control lists (ACLs).

https://docs.fastly.com/en/guides/security#_access-control-lists

About ACLs

Last updated: 2021-01-21

https://docs.fastly.com/en/guides/about-acls

Malicious actors can present themselves in a variety of ways on the internet. Automated tools can scrape information from your website, bots can probe your application for vulnerabilities, and hackers can exploit them. Using access control lists (ACLs) at the edge can help prevent the offending IP addresses they use from ever accessing your information resources.

When ACLs can be useful

Access control lists at the edge might be useful for:

- E-commerce companies preventing scraping from certain IP ranges
- Offices restricting access to their administrative portals
- Advertising technology companies blocking bad-actors at the edge
- Mobile applications accepting only calls from specific proxies or IP ranges
- System administrators restricting access to groups of backends from an office IP address or subnet range

How ACLs work

ACLs have two parts: an ACL container and the ACL entries within it. In combination, containers and entries allow you to store a list of permissions that Varnish will use to grant or restrict access to URLs within your services.

Once you attach an ACL container to a version of your service and that service is activated, the data in the container (the ACL entries) becomes *versionless*. This means that once your service is activated, any further changes to the data within, such as the addition of ACL entries, will become effective immediately.

Ways to create ACLs

To create an ACL at the edge and use it within your service, start by creating an empty ACL container and then add its entries in a working version of a service that's unlocked and not yet activated. You can create ACLs in several ways:

- **Via Fastly's web interface or API:** You can create your ACLs at the edge via the Fastly web interface or via the Fastly API. We recommend these options for most configurations that integrate websites or applications with an ACL at the edge.
- **Using custom VCL:** You can manually create an ACL using VCL. We recommend this option only if you have simple access control requirements and can hardcode a few IP addresses in your VCL. Manually created ACLs are versioned with your services and any changes to the ACL will require changes to your VCL.

Example ACL use

After you've used the Fastly API to create an ACL and add ACL entries, the VCL for the ACLs and ACL entries will be automatically generated, as shown below. For example, this VCL shows an ACL called `office_ip_ranges` has been created:

```
# This VCL is automatically generated when you create an ACL container and entries
# using the Fastly API. In this example, the ACL name is office_ip_ranges.
acl office_ip_ranges {
  "192.0.2.0"/24;                                # internal office
  "198.51.100.4";                                # remote VPN office
  "2001:db8:ffff:ffff:ffff:ffff:ffff:ffff";      # ipv6 address remote
}
```

Once created, you can add logic to interact with your ACL at the edge by uploading custom VCL. You could use the `office_ip_ranges` ACL as an allow list by uploading the following custom VCL:

```
sub vcl_recv {
  # block all requests to Admin pages from IP addresses not in office_ip_ranges
  if (req.url ~ "^/admin" && ! (client.ip ~ office_ip_ranges)) {
    error 403 "Forbidden";
  }
}
```

With this VCL, access to `/admin` is denied for everyone by default, but the IP addresses listed in the ACL are allowed to access `/admin` without restriction.

TIP

Because ACL entries have a boolean option for negation, you can specify whether or not an IP address is allowed (false or `0`) or blocked (true or `1`).

Limitations

When working with ACL containers and entries specifically, remember the following:

- **ACL entry changes via the API don't appear in the event logs.** If you use the API to add, update, or remove an ACL entry, there will be no record of it in the audit log or event log. The only record of a change will exist when you compare service versions and view the exact point at which the ACL was associated with the service version in the first place.
- **ACL entry deletions are permanent.** ACL entries are versionless. This means that if you delete an entry within an ACL container, that entry is permanently removed from all service versions and cannot be recovered.
- **ACL containers are limited to 1000 ACL entries.** If you find your containers approaching this entry limit, contact us. We may be able to help you figure out an even more efficient way to do things with your ACLs at the edge.
- **Deleted ACL containers are only removed from the service version you're editing.** ACL containers are tied to versions of services, which can be cloned and reverted. When you delete an ACL container, only the configuration of the service version you're editing will be affected. We remove the ACL entries inside a container but only for the specific service version you're editing. The ACL entries themselves are not deleted from the ACL in earlier versions of your service's configuration. This allows you to revert your configuration to a previous version in as few steps as possible.

When creating and manipulating ACLs at the edge, keep the following limitations in mind as you develop your service configurations:

- **ACLs created with custom VCL are always versioned.** ACLs created with custom VCL are always tied to a service and require a new service version each time they are updated in any way. This is true for both the ACLs created using custom VCL and for any logic created to interact with those ACLs.
- **ACLs created with custom VCL cannot be manipulated using the API.** If you create an ACL using custom VCL, that ACL must always be manipulated via custom VCL and can never be manipulated using the Fastly API. ACLs created using the API, however, can be manipulated both using the API and custom VCL.

Manually creating access control lists

Last updated: 2019-11-12

https://docs.fastly.com/en/guides/manually-creating-access-control-lists

Varnish allows you to use access control lists (ACLs), a feature that enables fast matching of a client's IP address against a list of defined IP addresses. An ACL looks like this:

```
# Who is allowed access ...
acl local {
  "localhost";
  "192.0.2.0"/24; /* and everyone on the local network */
  ! "192.0.2.1"/32; /* except for the dial-in router */
}
```

Defining an ACL

Using ACLs requires you to create and add custom VCL to Fastly's boilerplate VCL. To define an ACL in your Fastly configuration:

1. Read about how to mix and match custom VCL with Fastly VCL.
2. Create a custom VCL file with your ACL definitions included in the appropriate location. Use the example shown below as a guide. You can reference the ACL in your configuration (`vcl_recv`) using a match operation that can be located above or below `#FASTLY recv`. The placement only matters for the order of operations within Varnish's execution of your configuration.

```
# If you are using the "include" keyword
include "myACL1.vcl";

# And/or if you are using an actual ACL block
acl local {
  "localhost";
  "192.0.2.0"/24; /* and everyone on the local network */
  ! "192.0.2.1"/32; /* except for the dial-in router */
}

sub vcl_recv {
  # block any requests to Admin pages not from local IPs
  if (req.url ~ "^/admin" && req.http.Fastly-Client-IP !~ local) {
    error 403 "Forbidden";
  }
}
```

3. Upload the file in the Varnish Configuration area of your service.

Using the IP block list

Last updated: 2018-11-06

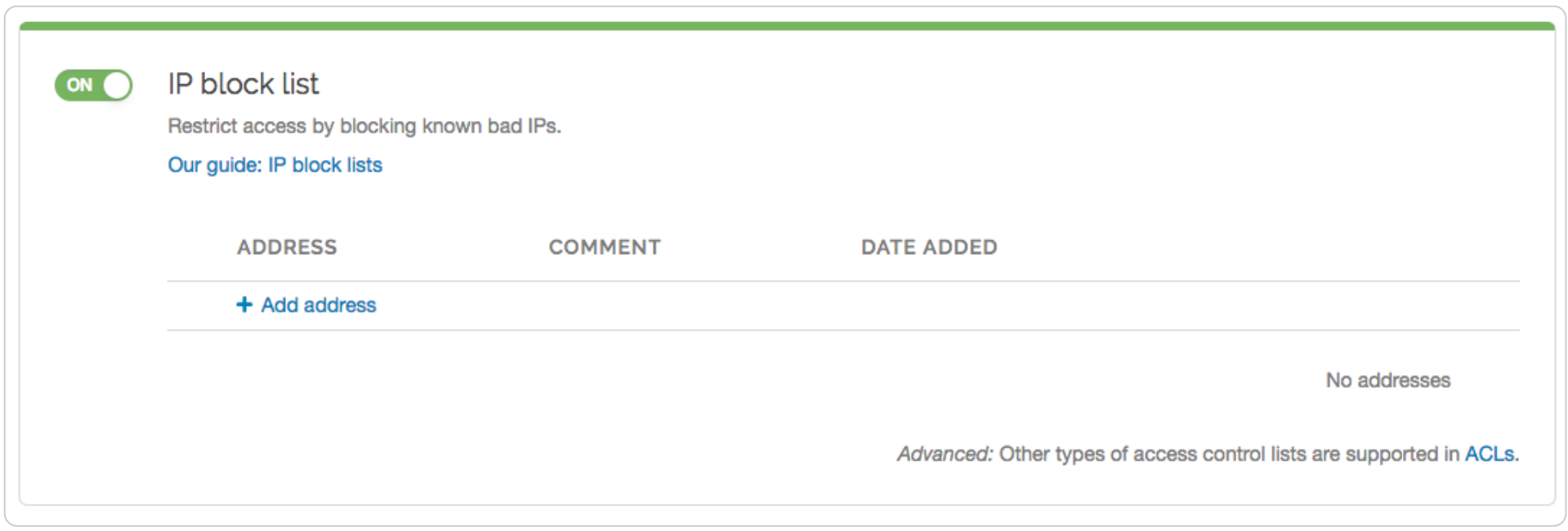
https://docs.fastly.com/en/guides/using-the-ip-block-list

You can prevent specific IP addresses from accessing your service by adding them to a block list. Enabling this feature creates a condition and response that returns a 403 error to anyone trying to access the service from a blocked IP address. You can use this feature to prevent bad actors from interfering with the operation of your web application.

Enabling the IP block list

To enable the IP block list, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **IP block list** switch to **On**.

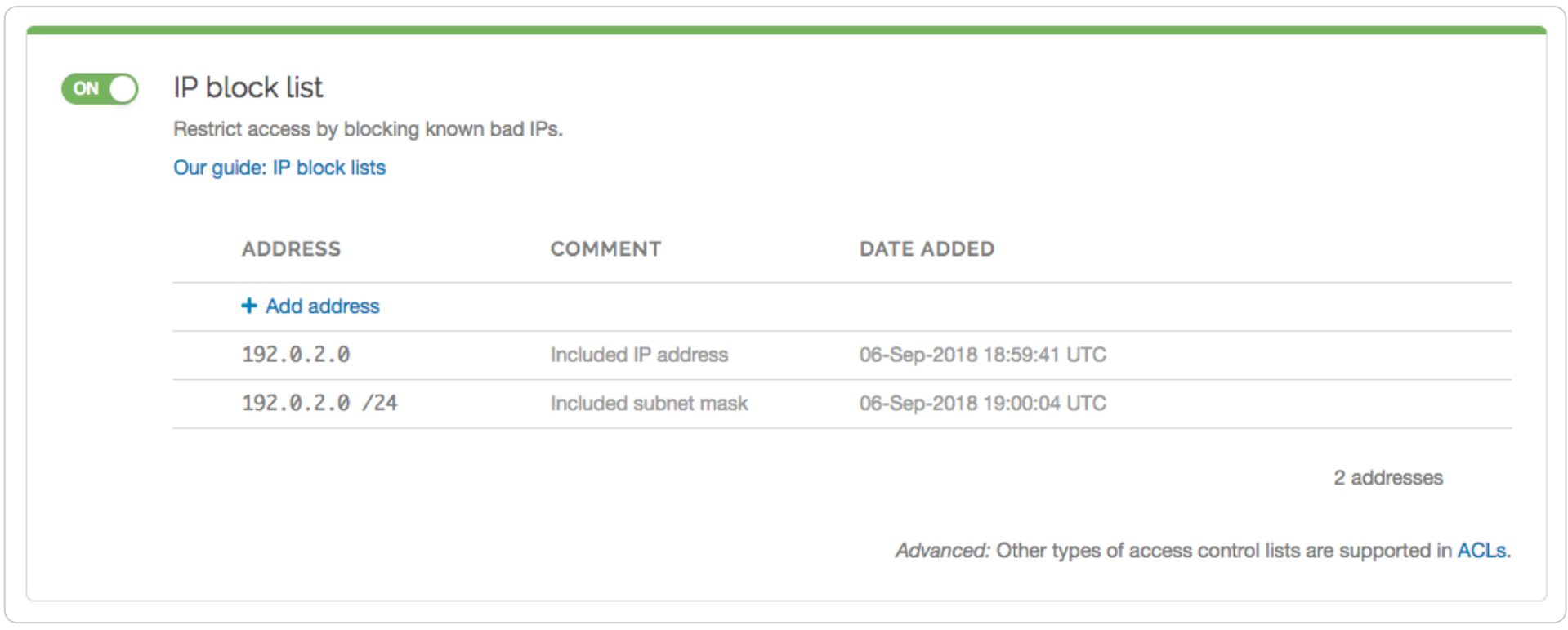


6. Click the **Activate** button to deploy your configuration changes.

Blocking an IP address

To block an IP address, follow the steps below:

1. Click the **Add address** link. The entry fields appear.
2. In the **Address** field, enter an IP address or subnet mask (a range of IP addresses) to block for this service. To add an exception for an IP address, use an exclamation point (for example, use !192.0.2.0 or !192.0.2.0/24).
3. In the **Comment** field, enter an optional comment that describes the IP address or subnet mask.
4. Click the **Add** button. The IP address or subnet mask appears in the list. This addition will become effective immediately.



Editing a blocked IP address

You can edit a blocked IP address or subnet mask at any time. To edit an IP address or a subnet mask, follow the steps below:

1. Find the IP block list associated with your service in which the associated IP addresses or subnet masks appear. Because these entries are versionless, the service version you choose doesn't matter. Choose the one that makes the most sense to you.
2. In the IP block list area, hover your cursor over an entry, then click the pencil icon that appears.
3. Edit the IP address, subnet mask, or comment as necessary.
4. Click the **Save** button. The changes you make will be immediately applied to your configuration. If your IP block list has already been associated with a deployed service version, those changes will happen live.

Deleting an IP block list entry

You can delete individual entries in the IP block list at any time. To delete an IP address or subnet mask that was created via the web interface:

1. Find the IP block list associated with your service in which the associated IP addresses or subnet masks appear. Because these entries are versionless, the service version you choose doesn't matter.
2. In the IP block list area, hover your cursor over an entry, then click the trash can icon that appears.
3. Click the **Confirm and delete** button.

Disabling the IP block list

The IP block list and its associated entries can be disabled in any unlocked service version. To disable the IP block list, follow the steps below:

1. Find the IP block list associated with an unlocked version of your service.
2. Click the **IP block list** switch to **Off**.
3. Click the **Yes** button. This disables the block list and deletes all associated entries.
4. Click the **Activate** button to deploy your configuration changes.

Creating other ACL types

If you need other types of ACLs, you'll need to create them in the Data page of the web interface.

Working with ACLs using the API

Last updated: 2018-07-30

https://docs.fastly.com/en/guides/working-with-acls-using-the-api

Access control lists (ACLs) allow you to store a list of permissions that Varnish will use to grant or restrict access to URLs within your services. You can use the Fastly API to add, remove, and update ACLs programmatically.

Working with ACL containers using the API

Using the Fastly API, you can create view, or delete ACL containers into which ACL entries can be placed.

ACL container attributes

Containers for ACLs at the edge have the following attributes:

- **Service ID:** The unique identifier of the Fastly service the ACL is associated with.
- **Service Version Number:** The service version number the ACL is associated with. Note that the ACL will continue to reside within subsequently cloned counterparts.
- **ACL Name:** The name of the ACL.
- **ACL ID:** The unique identifier of the ACL.

Creating an ACL container

To start using an ACL, you'll need to create an empty container within a version of a service that's unlocked and not yet activated. Make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/version/<service_version_number
>/acl -d name=my_acl
```

The response will look like this:

```
{
    "id": "<service_version_number>",
    "name": "my_acl",
    "service_id": "<service_id>",
    "version": "1",
    "created_at": "2016-04-14 21:23:21",
    "updated_at": "2016-04-14 21:23:21"
}
```

Be sure to activate the new version of the service you associated with the empty ACL container.

Viewing ACL containers

To see information related to a single ACL (in this example, `my_acl`) attached to a particular version of a service, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/version/<service_version_number>/acl/my_acl
```

The response will look like this:

```
{
    "id": "<acl_id>",
    "name": "my_acl",
    "service_id": "<service_id>",
    "version": "<service_version_number>",
    "created_at": "2016-04-14 21:23:21",
    "updated_at": "2016-04-14 21:23:21"
}
```

To view a list of all ACL containers attached to a particular version of a service, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/version/<service_version_number>/acl
```

The response will look like this:

```
[
    {
        "id": "<acl_1_id>",
        "name": "my_new_acl",
        "service_id": "<service_id>",
        "version": "<service_version_number>",
        "created_at": "2016-04-14 21:23:21",
        "updated_at": "2016-04-15 17:23:09"
},
    {
        "id": "<acl_2_id>",
        "name": "my_other_acl",
        "service_id": "<service_id>",
        "version": "<service_version_number>",
        "created_at": "2016-04-14 21:23:21",
        "updated_at": "2016-04-15 17:23:09"
    }
]
```

Deleting an ACL container

Deleting an ACL deletes the ACL and all of its associated entries. To delete an ACL (in this example, `my_new_acl`), make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" -X DELETE https://api.fastly.com/service/<service_id>/version/<service_version_number>/acl/my_new_acl
```

The response will look like this:

```
{
  "status":"ok"
}
```

Working with ACL entries using the API

ACL entry parameters

ACL entries have the following parameters:

- `service_id`: The ID of the Fastly service the ACL is associated with.
- `acl_id`: The ID of the ACL.
- `id`: The ID of the ACL entry.
- `ip`: The IP address contained within the ACL entry.
- `subnet`: Optional. The range of IP addresses within a single ACL entry.
- `negated`: If true, this entry is an exception to the non-negated entries in the list. Negations override non-negated entries regardless of their order. Valid values are `true` and `false`. Defaults to `false`.
- `comment`: Optional. A descriptive comment indicating why you created the ACL entry.

Creating an ACL entry

To add an entry to an existing ACL, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" -X POST https://api.fastly.com/service/<service_id>/acl/<acl_id>/entry -d 'ip=127.0.0.1&subnet=16&negated=0&comment=test'
```

The response will look like this:

```
{
    "acl_id": "<acl_id>",
    "comment": "test",
    "created_at": "2016-04-22T19:14:02+00:00",
    "deleted_at": null,
    "id": "<acl_entry_id>",
    "ip": "127.0.0.1",
    "negated": "0",
    "service_id": "<service_id>",
    "subnet": 16,
    "updated_at": "2016-04-22T19:14:02+00:00"
}
```

Viewing ACL entries

To see information related to a single ACL entry, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/<service_id>/acl/<acl_id>/entry/<acl_entry_id>
```

The response will look like this:

```
{
    "acl_id": "<acl_id>",
    "comment": "",
    "created_at": "2016-04-22T19:18:42+00:00",
    "deleted_at": null,
    "id": "<acl_entry_id>",
    "ip": "127.0.0.5",
    "negated": "0",
    "service_id": "<service_id>",
    "subnet": 16,
    "updated_at": "2016-04-22T19:18:42+00:00"
}
```

To view a list of all ACL entries attached to a particular ACL, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/acl/<acl_id>/entries
```

The response will look like this:

```
[
    {
        "acl_id": "<acl_id>",
        "comment": "",
        "created_at": "2016-04-22T19:13:03+00:00",
        "deleted_at": null,
        "id": "<acl_entry_1_id>",
        "ip": "127.0.0.1",
        "negated": "0",
        "service_id": "<service_id>",
        "subnet": 16,
        "updated_at": "2016-04-22T19:13:03+00:00"
    },
    {
        "acl_id": "<acl_id>",
        "comment": "",
        "created_at": "2016-04-22T19:14:02+00:00",
        "deleted_at": null,
        "id": "<acl_entry_2_id>",
        "ip": "127.0.0.2",
        "negated": "0",
        "service_id": "<service_id>",
        "subnet": 16,
        "updated_at": "2016-04-22T19:14:02+00:00"
    }
]
```

Updating ACL entries

There are two ways to update ACL entries: you can update a single ACL entry, or you can update multiple ACL entries at the same time.

Updating a single ACL entry

To update an existing ACL entry, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" -X PATCH https://api.fastly.com/service/<service_id>/acl/<acl_id>/entry/<acl_entry_id> -d 'ip=127.0.0.2&subnet=32&negated=0&comment=allow'
```

The response will look like this:

```
{
    "acl_id": "<acl_id>",
    "comment": "allow",
    "created_at": "2016-04-22T19:18:42+00:00",
    "deleted_at": null,
    "id": "<acl_entry_id>",
    "ip": "127.0.0.2",
    "negated": "0",
    "service_id": "<service_id>",
    "subnet": 32,
    "updated_at": "2016-04-22T19:18:42+00:00"
}
```

Updating multiple ACL entries

You can also update multiple ACL entries at the same time. Include an `entries` array of changes in the API call and pass an operation (`op`) parameter for every change. Possible `op` values are `create`, `update`, and `delete`.

To update multiple ACL entries at the same time, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" -H "Content-type: application/json" -X PATCH https://api.fastly.com/service/<service_id>/acl/<acl_id>/entries -d '{"entries":[{"op": "create", "ip": "192.168.0.1","subnet": "8"},{"op": "update", "id": "<acl_entry_id>", "ip": "192.168.0.2", "subnet": "16"},{"op": "delete", "id": "<acl_entry_id>"}]}'
```

The response will look like this:

```
{
  "status":"ok"
}
```

Deleting an ACL entry

> **WARNING**
>
> ACL entry deletions are permanent. If you delete an ACL entry, the entry is permanently removed from all service versions and cannot be recovered.

To permanently delete an ACL entry, make the following API call in a terminal application:

```
$ curl -H "Fastly-Key: FASTLY_API_TOKEN" -X DELETE https://api.fastly.com/service/<service_id>/acl/<acl_id>/entry/<acl_entry_id>
```

The response will look like this:

```
{
  "status":"ok"
}
```

Working with ACLs using the web interface

Last updated: 2018-07-30

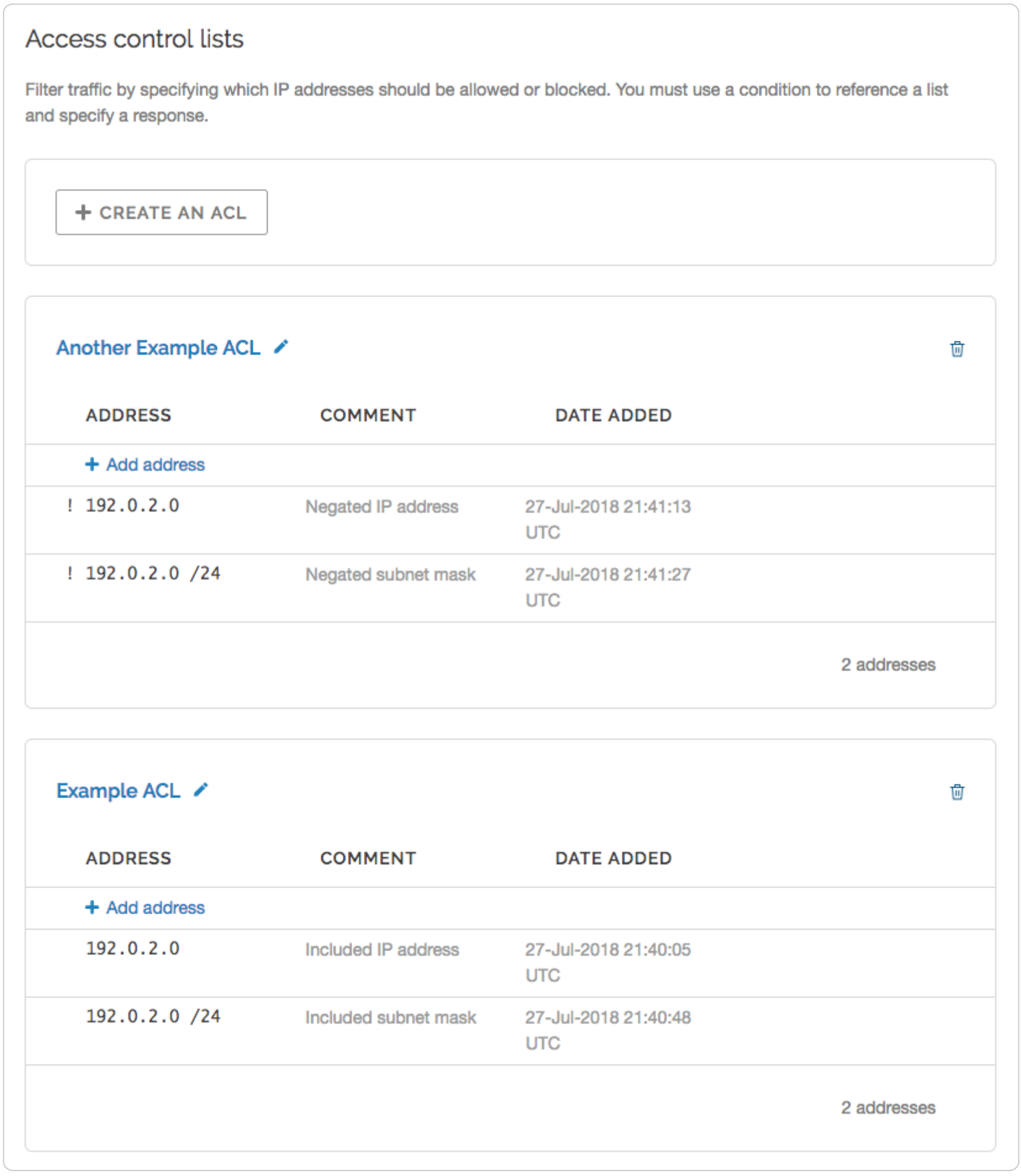
https://docs.fastly.com/en/guides/working-with-acls-using-the-web-interface

Access control lists (ACLs) allow you to store a list of permissions that Varnish will use to grant or restrict access to URLs within a service. You can use the web interface to add, remove, and update ACLs.

Viewing ACLs

To view an ACL, navigate to the ACL management area of your service:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Configuration** button and then select **View Active**.
4. From the service version menu, select an appropriate service version. The Domains page appears.
5. Click the **Data** link. The Data page appears. Existing ACLs, if any, associated with the currently selected service version appear in the Access control lists area.



NOTE

Remember that ACL containers are versioned. If you don't see an ACL attached to your service, check the service version to make sure you're looking at the right one.

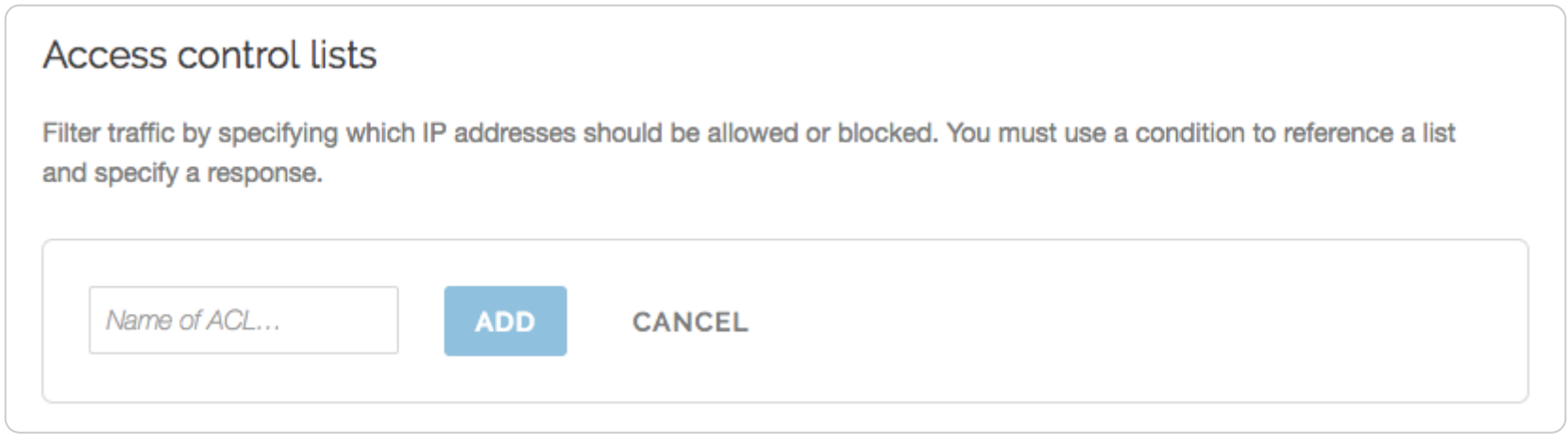
Creating an ACL

ACLs have two parts: an ACL container and the ACL entries within it.

Creating an ACL container

To create an ACL, start by creating an ACL container:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Data** link. The Data page appears.
5. Click **Create an ACL**. The ACL container name field appears.



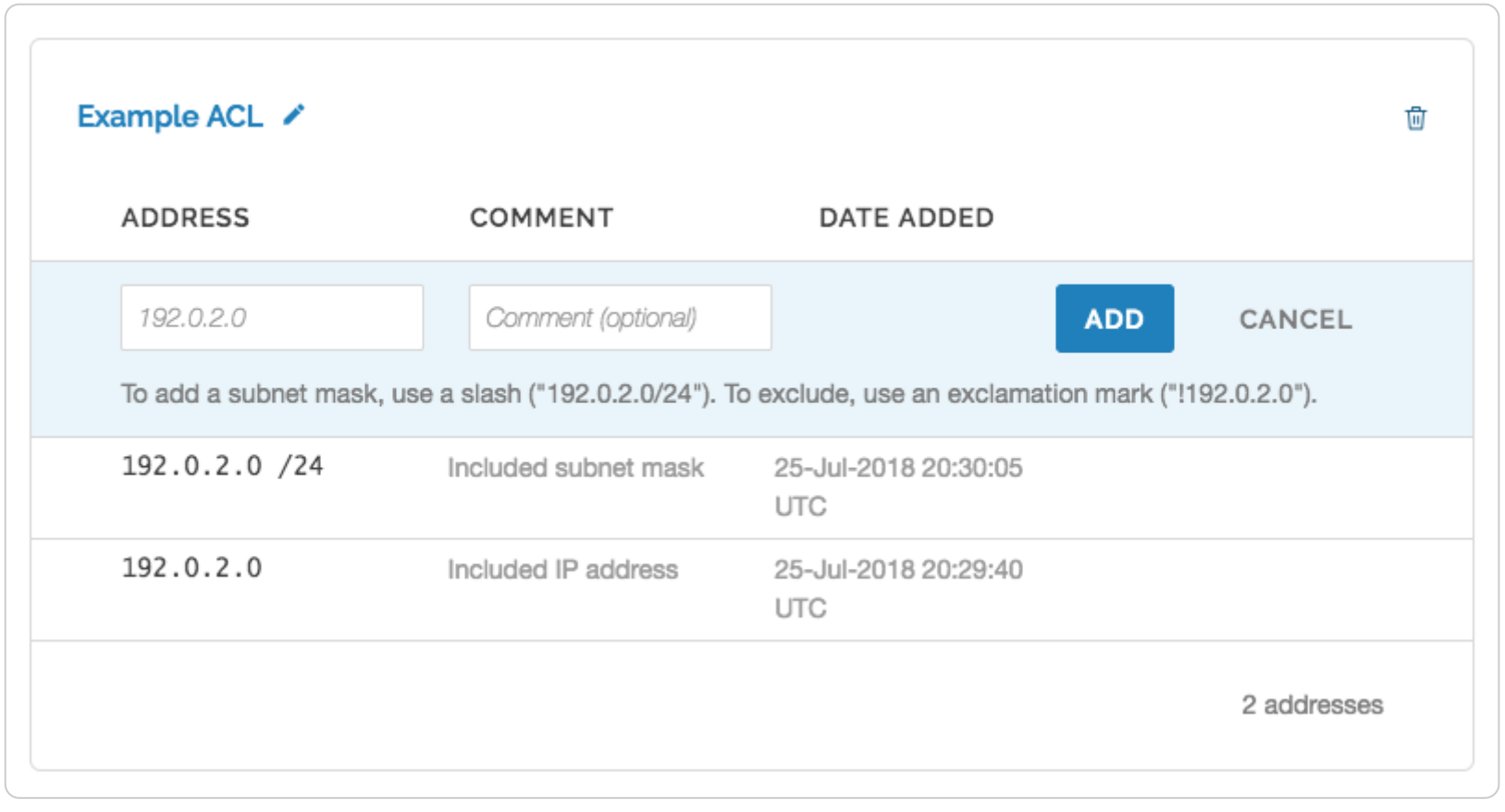
6. In the **Name of ACL** field, enter a descriptive name for the ACL (e.g., `Example ACL`).
7. Click the **Add** button. The empty ACL container you created appears.
8. Click the **Activate** button to deploy your configuration changes to the service version you're editing.

Creating an ACL entry

Once your ACL container is created, add ACL entries into it:

1. Click the **Add address** link. The ACL entry fields appear.

2. In the **Address** field, enter an IP address or subnet mask (a range of IP addresses) to allow or block for this service. To exclude or block an IP address or subnet mask, use an exclamation point (for example, use `!192.0.2.0` or `!192.0.2.0/24`).
3. In the **Comment** field, enter an optional comment that describes the IP address or subnet mask.
4. Click the **Add** button. The IP address or subnet mask appears in the ACL container. This addition will become effective immediately.



Editing an ACL

Keeping in mind their limitations, the containers and entries of ACLs can be edited via the web interface.

Editing an ACL container

You can edit the name of an ACL container that was created via the web interface in any unlocked service version:

1. Find an ACL associated with an unlocked version of your service.
2. Click the pencil icon next to the ACL container name.
3. Change the name, then click the **Save** button.

Editing an ACL entry

You can edit the ACL entries within a container at any time. To edit an IP address or subnet mask included in an ACL container that was created via the web interface:

1. Find any ACL associated with your service in which the associated IP addresses or subnet masks appear. Because ACL entries are versionless, the service version you choose doesn't matter. Choose the one that makes the most sense to you.
2. Hover your cursor over an ACL entry, then click the pencil icon that appears.
3. Edit the IP address, subnet mask, or comment as necessary.
4. Click the **Save** button. The changes you make will be immediately applied to your configuration. If you ACL container has already been associated with a deployed service version, those changes will happen live.

Deleting an ACL

Keeping in mind their limitations, the containers and entries of ACLs can be deleted via the web interface.

Deleting an ACL container

You can delete an ACL container that was created via the web interface in any unlocked service version:

1. Find an ACL associated with an unlocked version of your service.
2. Click the trash can icon in the top right corner of the ACL.
3. Click the **Confirm and delete** button.
4. Click the **Activate** button to deploy your configuration changes to the service version you're editing.

Deleting an ACL entry

You can delete the ACL entries within a container at any time. To delete an IP address or subnet mask included in an ACL container that was created via the web interface:

1. Find any ACL associated with your service in which the associated IP addresses or subnet masks appear. Because ACL entries are versionless, the service version you choose doesn't matter. Choose the one that makes the most sense to you.
2. Hover your cursor over an ACL entry, then click the trash can icon that appears.
3. Click the **Confirm and delete** button.

---

§

These articles describe how to secure communications between Fastly, your origin servers, and your customers.

https://docs.fastly.com/en/guides/security#_securing-communications

---

Accessing Fastly's IP ranges

Last updated: 2021-11-19

https://docs.fastly.com/en/guides/accessing-fastlys-ip-ranges

To help you allowlist Fastly's services through your firewall, we provide access to the list of Fastly's assigned IP ranges at this URL:

https://api.fastly.com/public-ip-list

You can use the API endpoint to programmatically detect when the IP ranges change (for example, by running a script as a cron job).

To make sure you have plenty of time to stay in sync, we post IP address announcements along with other service announcements to our status page, which you can subscribe to.

Support for App Transport Security

Last updated: 2018-08-03

https://docs.fastly.com/en/guides/support-for-app-transport-security

Apple uses App Transport Security (ATS) to improve the security of connections between web services and applications installed on devices using iOS 9 or later, as well as OS X 10.11 (El Capitan) and later. Fastly is fully compliant with all ATS requirements. You shouldn't run into any issues supporting iOS or OS X users while using our service.

Results from the ATS diagnostics tool

We used Apple's ATS diagnostics tool to ensure that Fastly is compliant with all ATS requirements. You can review the output from the diagnostics tool below.

```
$ /usr/bin/nscurl --ats-diagnostics https://www.fastly.com
Starting ATS Diagnostics

Configuring ATS Info.plist keys and displaying the result of HTTPS loads to https://www.fastly.com.
A test will "PASS" if URLSession:task:didCompleteWithError: returns a nil error.

Use '--verbose' to view the ATS dictionaries used and to display the error received in URLSession:task:didCompleteWithE
rror:.
================================================================================

Default ATS Secure Connection
---
ATS Default Connection

Result : PASS
---

================================================================================
Allowing Arbitrary Loads

---
Allow All Loads

Result : PASS
---

================================================================================

Configuring TLS exceptions for www.fastly.com

---
TLSv1.2

Result : PASS
---

---
TLSv1.1

Result : PASS
---

---
TLSv1.0

Result : PASS
---

================================================================================

Configuring PFS exceptions for www.fastly.com

---
Disabling Perfect Forward Secrecy

Result : PASS
---

================================================================================

Configuring PFS exceptions and allowing insecure HTTP for www.fastly.com

---
Disabling Perfect Forward Secrecy and Allowing Insecure HTTP

Result : PASS
---

================================================================================

Configuring TLS exceptions with PFS disabled for www.fastly.com

---
TLSv1.2 with PFS disabled

Result : PASS
---

---
TLSv1.1 with PFS disabled
```

```
Result : PASS
---

---
TLSv1.0 with PFS disabled

Result : PASS
---

================================================================================

Configuring TLS exceptions with PFS disabled and insecure HTTP allowed for www.fastly.com

---
TLSv1.2 with PFS disabled and insecure HTTP allowed

Result : PASS
---

---
TLSv1.1 with PFS disabled and insecure HTTP allowed

Result : PASS
---

---
TLSv1.0 with PFS disabled and insecure HTTP allowed

Result : PASS
---
```

§

These articles provide information about the administrative, physical, and technical safeguards that protect the Fastly CDN service.

https://docs.fastly.com/en/guides/security#_security-measures

Data management

Last updated: 2021-12-14

https://docs.fastly.com/en/guides/data-management

Fastly maintains a “privacy and protection by design” approach that is manifested in Fastly’s data governance program. Fastly is intentional about data processing, collection and access to customer and personal data. Fastly does not collect more data than needed to perform its services. Fastly considers legal, compliance, regulatory, and commercial obligations when working with data. Fastly appropriately protects records and information that are private, confidential, privileged, secret, essential to business continuity, or that otherwise require protection.

Fastly production data

In addition to the Fastly security program, Fastly maintains the following data management practices in production environments.

Data Insights

- *Service management:* Fastly collects and processes data related to the functional performance of Fastly services, anomalous activity, and suspicious behavior detected by the services. Fastly retains and uses this data to monitor, maintain, and improve

its services, business operations, and security and compliance programs.

- *Confidentiality:* Fastly only discloses this data in an anonymized and aggregated form and subject to its confidentiality obligations to customers.

IP addresses

- *Security events:* Fastly may indefinitely retain any non-anonymized, non-aggregated client or customer IP addresses associated with security-related incidents or administrative connections to Fastly's services. Fastly may retain non-anonymized, non-aggregated client or customer IP addresses associated with this anomalous activity or suspicious behavior for a period of up to 30 days.
- *Suspicious activity:* Fastly keeps internal systems logs, including access logs, related to events triggered by anomalous activity or suspicious behavior for at least one year. Fastly may retain IP addresses from Fastly event logs or configurations indefinitely.
- *Fastly application:* Fastly independently collects the IP addresses of users who access services within the Fastly web interface or through the Fastly API.
- *Endpoints:* If a customer defines origin servers or syslog endpoints with IP addresses, Fastly will save those IP addresses as part of the customer's configurations.
- *Client IPs:* Fastly retains client IP addresses in a non-anonymized, non-aggregated fashion for up to two business days, or up to seven days if those addresses are associated with transmission errors.
- *Origin IPs:* Fastly may retain dynamically-resolved origin IP addresses for up to two business days, or up to seven days if associated with transmission errors. The IP addresses are discarded thereafter.

Customer data management

The duration of any data retention will vary based on the type of data and its use.

- *Customer content:* Customer content enters, transits, and departs Fastly's network in response to requests. Generally, customers manage which content is processed, where, and for how long by setting policies that control that content.
- *Customer configurations:* Customer configurations may be stored indefinitely, but can be deleted upon request. Fastly may directly access or modify customer accounts or configurations as necessary to provide services, to prevent or address service or technical issues, as required by law, or as customers expressly permit. Fastly retains encrypted backups of customer configurations, including VCL, and customer provided packages for business continuity purposes.
- *Cached content:* Cached content is retained per customer configuration and use of purge functionality. Customers may control length and type of retention through configuration options to meet requirements for regulatory reasons such as HIPAA or PCI DSS. Fastly deletes cached content according to a customer's use of the purge functionality and as described in documentation.
- *Customer packages deployed to Compute@Edge:* Customer provided compiled code may be stored indefinitely, but can be deleted upon request.

Customer request logs

- *Content request logs:* Customers may stream their content request logs, which may include request headers, including client IP addresses, to a customer-owned and managed endpoint for analysis and use.
- *Request logs retention:* Fastly does not retain customers' request logs except where explicitly stated in the Documentation and related to the functional performance of the services.

Note regarding Signal Sciences data management

The Signal Sciences security measures describe the Signal Sciences data management practices.

Note regarding privacy law

For more information regarding Fastly's compliance with global privacy laws and regulations, refer to the Fastly data processing terms, the list of sub-processors, Fastly's privacy policy, and additional resources on the Fastly Trust page.

Fastly Next-Gen WAF (powered by Signal Sciences) security measures

Last updated: 2022-02-03

https://docs.fastly.com/en/guides/fastly-next-gen-waf-security-measures

Fastly's security measures for the Fastly Next-Gen WAF (powered by Signal Sciences) (Next-Gen WAF) include safeguards that help protect your data as it moves through the Next-Gen WAF. It has three deployment options: Edge, Core, and Cloud WAF. The security measures described on this guide apply to the entirety of the Core and Cloud WAF deployments. For Edge deployments, the security aspects and associated data handling are covered by the terms on this page. The hosted components of Edge deployments are hosted within Fastly's Compute at Edge environment and are subject to our security measures.

> The Fastly Next-Gen WAF now collectively refers to the products that were previously known as the Signal Sciences Cloud WAF and Signal Sciences Next-Gen WAF. The functionality of those products has not changed as part of the new naming convention. Fastly Next-Gen WAF continues to be powered by Signal Sciences technology.

Authentication and authorization

- Our systems and devices enforce user roles or similar measures to control the extent of access we grant individual users.
- We control access to privileged systems using Zero Trust access policies that use client certificates and two-factor authentication.
- Our authentication requirements, such as passwords, are in line with industry standard practices.

Business continuity and operational resilience

- We monitor production operation systems and supporting systems to detect service-related and non-compliance issues on a continuous basis. The systems are monitored 24×7 to ensure constant availability to clients.
- If an update has potential impact to customer uptime, we will determine a timeline for the update and communicate the impact to customers via https://status.signalsciences.net.
- We maintain our services in multiple Availability Zones (AZs) to operate production applications and databases that are more highly available, fault-tolerant, and scalable than would be possible from a single data center.
- We update impacted customers using various communication methods (such as https://status.signalsciences.net), depending on an incident's scope and severity.

Cloud infrastructure data center and physical security

We rely on data center space under the control of Amazon Web Services (AWS) and their physical security controls. As part of our third-party security review process, we confirm that these providers maintain appropriate physical security measures to protect their data center facilities.

Customer and end user data management

- We do not store sensitive customer data processed by Core deployments in the cloud. Customers process this data in local environments under their control with no remote access by our employees.
- We store and retain customer data that is sent to us and that is processed via the security components of Next-Gen WAF for up to 30 days.
- The Next-Gen WAF analyzes requests. We retain and use data about the operation and reliability of our processing of requests to monitor, maintain, and improve our services, our business operations, and our security and compliance programs. Subject to confidentiality obligations to our customers, we only disclose this data in anonymized and aggregated form.

Encryption

We use industry-accepted encryption technologies to encrypt sensitive information. All client data is encrypted in transit using TLS.

Governance

- We have formally assigned information security duties to our personnel. Our Chief Security Officer and Security organization work with other departments to safeguard sensitive information related to our services.
- Our policies and procedures help us maintain security in our systems, processes, and employee practices. Our Security organization formally reviews these policies and procedures at least annually.
- We integrate risk assessment activities with various processes to identify and address information security risk to the company and customer data on our network.
- We perform risk-based evaluations of the security measures of our vendors. We review these security measures before we begin using a vendor, and we ask the vendor to formally acknowledge these measures. We re-evaluate vendor security measures on a recurring basis thereafter.

Human resources security

- Our employees formally agree to safeguard the sensitive information they may view, process, or transmit as part of their job functions.
- We train our people to protect the data and devices they use. Each employee receives security awareness training as part of new hire procedures, and current employees take this training annually.
- We screen new employees as part of the hiring process. Screening activities depend on applicable local regulations and may include criminal background checks and reference checks.

Identity and access management

- We periodically inspect access privileges to make sure our personnel have appropriate access to our systems and data.
- We promptly update or remove an employee's access to our network to match that employee's current job function or employment status.

Logging and monitoring

- We configure thresholds within our monitoring tool to alert when a security policy has been violated. Threshold policies are reviewed on an annual basis for accuracy and appropriateness.
- We restrict, log, and monitor information security management systems activity with anomaly alerting. We aggregate and securely store the activity in a centralized internal log server.

Network and infrastructure security

- We review and validate information systems and network device configurations against established security policies and procedures.
- We regularly perform vulnerability scans and third-party penetration tests on our network. We review and address findings from these activities to help maintain the security of our network.
- To maintain awareness of potential security vulnerabilities, we monitor public and private distribution lists, as well as reports submitted through our responsible disclosure process. We validate and implement security patches for critical vulnerabilities within 24 hours of discovery. For non-critical vulnerabilities and updates, we schedule and deploy vendor-provided patches on a regular basis.
- To protect from known vulnerabilities, we maintain assets at the latest version and patch levels currently supported by vendors. Priority of patch deployment is based on vulnerabilities and risks it poses to the environment.

Security incident management

- We maintain a formal incident response plan with established roles and responsibilities, communication protocols, and response procedures. We review and update this plan periodically to adapt it to evolving threats and risks to our services.
- We will notify affected customers within 48 hours of validating an unauthorized disclosure of customer confidential information.

Penetration testing your service behind Fastly

Last updated: 2018-05-30

https://docs.fastly.com/en/guides/penetration-testing-your-service-behind-fastly

We understand the need for our customers to validate the security of their service behind Fastly.

IMPORTANT

Penetration tests that interfere with or disrupt the integrity or performance of Fastly services violate our acceptable use policy. You must respond immediately to any communication from Fastly regarding your test to help ensure your testing does not adversely affect other customers or the Fastly network.

To perform security testing of your Fastly service configurations, create a Customer Support ticket by contacting Fastly via email at support@fastly.com at least two (2) business days before you begin any security testing. In your ticket, include these details:

- the IDs of the services that will be tested
- the source IP address of the test
- the date of the test
- the start and end time of the test, including the time zone
- the contact information for the individual or third party performing the test, including a phone number and e-mail address
- whether or not the security test is likely to lead to significantly increased traffic volume

The following requirements apply to any security testing you perform:

- Only test Fastly services you own or are authorized by the owner to test. You may not perform tests against other customers without explicit permission or against Fastly-owned resources.
- Do not begin testing until after Fastly has responded affirmatively to your ticket and authorized your request.
- Update the ticket if either the scope or timeframe of your testing changes.
- If you discover vulnerabilities in the Fastly platform during your test, update the ticket with your findings as soon as possible so we can address them.

Fastly maintains programs for security and technology compliance. To perform an independent audit of these programs, contact sales@fastly.com to discuss purchase of Assurance Services.

TIP

We welcome security professionals researching potential vulnerabilities in our network under our guidelines for reporting a security issue.

Security measures

Last updated: 2022-02-07

https://docs.fastly.com/en/guides/security-measures

These articles provide information about the administrative, physical, and technical safeguards that protect the Fastly CDN service.

- Data Management
- Security program
- Fastly Next-Gen WAF security measures

Reporting discovered security issues

We take the security of our network seriously and support the disclosure of security issues related to our service. If you believe you have found a vulnerability, we encourage you to report your discovery to our Support team so we can investigate further. If you plan to do security testing of your service behind Fastly, notify Fastly at least two (2) business days prior to the test. See our penetration testing guidelines for more information.

Related features

- Access control lists
- Configuring user roles and permissions
- Cryptographic VCL features
- Enabling and disabling two-factor authentication
- Miscellaneous VCL features
- Monitoring account activity with the event log
- PerimeterX Bot Defender
- Streaming logs
- Securing communications
- TLS

Security program

Last updated: 2021-12-14

https://docs.fastly.com/en/guides/security-program

Fastly operates a comprehensive information security program that includes administrative, physical, and technical safeguards to protect its infrastructure, data, services, and customers.

Foundation

Fastly's security program is based on the NIST Cybersecurity Framework comprised of annually reviewed security policies, designated roles and responsibilities for its experienced professionals, and formal procedures developed focused on risk.

Security policies

Fastly institutes information security policies that are published internally and reviewed annually. The policies contain principles and point to standards that cover controls and procedures designed to protect Fastly and Fastly's customers.

Experienced professionals

Fastly designates roles and responsibilities for the security of its services. Fastly assigns a Chief Information Security Officer to oversee its security program and retains best in class professionals in the field to apply it.

Risk-based approach

Fastly maintains formal procedures for the identification, assessment, and treatment of information security and availability risks, threats, and vulnerabilities to its services. The procedures include an annual risk assessment, risk analysis and treatment plan, and a risk register.

- *Annual risk assessment:* Fastly conducts an annual risk assessment to measure the state of security risk across the company. The results of this assessment are shared with the senior leadership team to ensure appropriate visibility and treatment.
- *Risk analysis and treatment plan:* Each identified enterprise security risk is evaluated and ultimately managed to acceptable levels by implementing associated controls and mitigation plans commensurate with the risk.

- *Risk register:* Fastly maintains documentation of identified risks, threats, and vulnerabilities related to its services. Assigned personnel help assess and remediate identified items, in line with the related risk and vulnerability management procedures.

Defense-in-depth

Fastly understands that to adequately protect its services, customers, and customer data, multiple safeguards must be applied to all layers of Fastly's business and technology practices. Fastly's process, technology, and physical security controls are designed specifically to provide a defense-in-depth approach and can be categorized as follows:

Identity and access management

Fastly manages access to its production systems using the following:

- *Authentication:* Employees are required to use unique user accounts and multi-factor authentication for remote access to production systems.
- *Authorization:* Employee access to production systems is restricted based on appropriate roles.
- *Audit:* Logs of access attempts (both success and failure) to production systems are kept and monitored.
- *Access grants and revocations:* Employee access to production systems is granted based on the principle of least privilege and manager approval. That access is reviewed at least quarterly and is removed when no longer needed or upon employee separation. Access roles are enforced by Fastly systems and devices.

Data security

Fastly manages data security using the following:

- *Customer credentials management:* Fastly secures customer-provided private keys and credentials throughout their lifecycle and stores private keys and API tokens in encrypted repositories. Customer-provided private keys are encrypted at rest and are re-encrypted on a regular interval. The key encryption keys are stored in a secrets management system and private keys are decrypted in memory at the edge when requested and removed from memory after a short period of time. Customer passwords are salted and hashed at rest and Fastly enables encryption for customer account passwords in transit. Access to private keys is restricted to only those individuals whose role requires it.
- *Authorized access to customer data:* Fastly may directly access or modify customer accounts or configurations as necessary to provide the services, prevent or address service or technical issues, as required by law, or as customers expressly permit. For the same reasons, Fastly may also access or modify equipment, systems, or services that manage customer data.
- *Privacy and protection-by-design approach:* Fastly maintains a "privacy and protection-by-design" approach that is manifested in a data governance program and documented separately in the data management documentation online.

Application security

Fastly manages application security using the following:

- *Secure development practices:* Fastly engineers are trained annually on secure coding concepts, including the OWASP Top 10 and CWE Top 25. Code is peer-reviewed and run through automated testing before deployment to production systems. After review and testing, code is initially deployed to a limited number of locations in the Fastly network for further monitoring. If no problems are encountered, code is gradually deployed across the Fastly network.
- *Application security analysis:* Fastly security engineers and third-party validators conduct periodic analysis and regular penetration testing of Fastly-written code.
- *Automated code analysis:* Fastly deploys technology to automatically identify and report on identified vulnerable dependencies.

System and network security

Fastly manages system and network security using the following:

- *Asset management:* Fastly maintains an inventory of its hardware and services deployed within the Fastly network.
- *Configuration standards:* Fastly maintains secure configuration standards, including restricted ports, protocols, and services, and removal of insecure default settings.

- *Patch management:* Fastly patches its production systems on a regular basis and applies out-of-band patches for newly-identified risks.
- *Endpoint management:* Fastly manages its production systems by verifying appropriate security settings are in place, including logging and monitoring, host-based firewalls, and session management.
- *Audit and monitoring:* Fastly logs relevant security-related events, including authentication successes or failures to production systems and the use of certain commands. Fastly investigates events triggered by anomalous activity or suspicious behavior.
- *Documentation:* Fastly maintains accurate network diagrams and internal documentation of its systems and services.
- *Access Control List (ACL) review:* On at least a semi-annual basis, Fastly conducts a production system ACL review of its endpoint firewall and router rulesets.
- *Intrusion Detection:* Fastly maintains mechanisms designed to detect potential intrusions at the network and host level. Fastly inspects and responds to detected events, as necessary, to address threats.

Physical security

Fastly production systems reside in a combination of Fastly-managed data centers and cloud infrastructure environments. Regardless of the physical location of the infrastructure or its operator, Fastly evaluates and applies the same minimum, mandatory physical security controls.

- *Physical access management:* Fastly uses providers that maintain industry standard physical and environmental protections, including perimeter protection, security guard assignment, access logging and review, and video surveillance.
- *Physical access to production systems:* Physical access is granted only to approved personnel. Requests for access are evaluated by authorized personnel and based on proof of proper credentials, appropriate and documented use-case, and limited to areas specified in their permissions.
- *Environmental security safeguards:* Providers protect their systems with controls including power redundancy, fire suppression, and other environmental controls.
- *Secure hardware destruction:* Providers use industry standard secure destruction of all production hardware prior to disposal.

Human security

Fastly manages human security using the following:

- *Employee background screening:* Fastly conducts background screenings on each of its employees upon hire, with recurring criminal conviction checks periodically thereafter, and maintains a policy requiring employees to report any criminal convictions during the course of employment, each as permitted by applicable local regulations.
- *Confidentiality agreements:* To safeguard sensitive information that employees may view, process, or transmit as part of their job functions, all employees enter into confidentiality agreements with Fastly.
- *Awareness training:* All employees receive security training upon hire and annually thereafter designed to help protect Fastly and its customers. Mandatory annual training includes security awareness that covers application of best security practices in day-to-day work and privacy to ensure each employee understands how to identify sensitive information and comply with regulations.

Continuous monitoring and improvement

To ensure that the controls described above are consistently applied and effective in their intended use, Fastly continuously monitors and improves its security measures. Fastly institutes strict processes and testing procedures as follows.

Change management process

Fastly follows a defined set of procedures to develop and deploy technology changes. These changes include updates to software, configurations, and devices that support Fastly's services.

- *Testing:* Fastly tests changes at various stages of development and confirms the changes operate as expected in a non-production environment before completing a deployment into its services.
- *Change approval and notification:* Fastly prepares, approves, and communicates change notices to maintain awareness among employees who manage the Fastly network and systems. Fastly maintains rollback procedures to address deployment

issues if they arise.

- *Post-implementation review:* Fastly confirms the success of changes after deployment.
- *Change monitoring:* Fastly uses multiple monitoring and alert mechanisms to enhance the visibility of technical changes and help ensure adherence to change management processes.

Vulnerability management

Fastly monitors for vulnerabilities in its production systems using the following measures:

- *Internal and external vulnerability scanning:* On a regular basis, Fastly automatically analyzes its production systems for vulnerabilities.
- *Vulnerability mitigation:* Fastly assesses the risk of identified or reported vulnerabilities, and mitigates vulnerabilities in a timely manner. Mitigations for vulnerabilities deemed highest severity are implemented within twenty-four (24) hours of validation.
- *Distribution lists and vendor notification:* Fastly monitors publicly disclosed and vendor confidential distribution lists and notifications from software vendors for vulnerabilities.

Penetration testing

On a semi-annual basis, Fastly engages a third-party to conduct a penetration test of Fastly production systems. Identified issues are prioritized and handled in order based upon the severity of the evaluated risk they pose.

Compliance and audits

Fastly maintains recurring audits and assessments that confirm its security program meets various industry standards and regulatory requirements.

Fastly vendor management

Fastly uses third-party vendors and service providers to support its services. Fastly evaluates its vendors for security controls and risk to Fastly and its services prior to using vendor services, and regularly thereafter based on vendor risk.

When something goes wrong

Fastly aims to provide a consistently reliable and secure platform. With this in mind, Fastly is always monitoring for threats and systems disruptions so incidents are detected, responded to, and recovered from in a timely manner.

Incident management plan

Fastly maintains a formal incident response plan to address security-related incidents. The plan contains established roles and responsibilities, communication protocols, and response procedures. Fastly reviews and updates the plan periodically to adapt it to evolving threats and risks to its services. Representatives from key departments are assigned to address security-related incidents. These personnel coordinate the full lifecycle of incidents, from detection, through response, and recovery. Included within these processes is communication with external contacts as needed.

Incident notification

Fastly notifies affected customers within forty-eight (48) hours of validating any unauthorized disclosure of customer data. Following any security-related incident, Fastly investigates and takes corrective action in a timely manner according to the incident management plan and provides affected customers with periodic updates.

Business continuity

Fastly manages business continuity using the following:

- *Service failover:* Fastly production systems are designed to be prepared for service failover. Production systems are deployed on infrastructure in multiple regions or zones to provide redundancy in the event of degraded performance or operational issues with a provider. If failure of a service occurs within a single region or zone, Fastly will automatically attempt to use infrastructure in another region or another infrastructure provider.

- *Internet redundancy:* Fastly data centers and cloud infrastructure providers have connections with multiple internet service providers.
- *Service monitoring:* Fastly monitors reporting channels to detect service-related issues. Personnel are available 24×7×365 to confirm and respond to disruptions of its services.
- *Communication and Reporting:* Fastly provides service interruption updates to customers using various communication methods (including status.fastly.com), depending on an incident's scope and severity.
- *Business continuity plan and testing:* Fastly has a business continuity plan for its production systems that is reviewed, approved, and updated annually. Fastly tests its business continuity plan on an annual basis.
- *Data backups:* Fastly conducts regular backups of data, excluding cached customer data, to support the recovery and availability of its services. Data backups are tested on a quarterly basis to validate backup recovery procedures.

---

§

These articles describe how to set up TLS certificates with Fastly services.

https://docs.fastly.com/en/guides/security#_tls

Enabling HSTS through Fastly

Last updated: 2020-11-20

https://docs.fastly.com/en/guides/enabling-hsts-through-fastly

The HTTP Strict Transport Security (HSTS) security enhancement specification provides a way to force modern browsers to communicate only via the Transport Layer Security (TLS) protocol. Once enabled, HSTS will force the browser to redirect (typically with a status code 307) to the HTTPS URL as long as the URL has previously been visited. For example, making a request for **http**://www.example.com would force a redirect to **https**://www.example.com as long as **https**://www.example.com has been visited once before.

NOTE

Because HSTS only takes effect after a site has been visited on a trusted HTTPS connection, we recommend forcing TLS and enabling HSTS. If you'd prefer not to automatically enable HSTS, you can still manually enable it after setting up TLS redirects.

Prerequisites

These instructions assume that you've set up TLS service with Fastly.

Forcing TLS and enabling HSTS

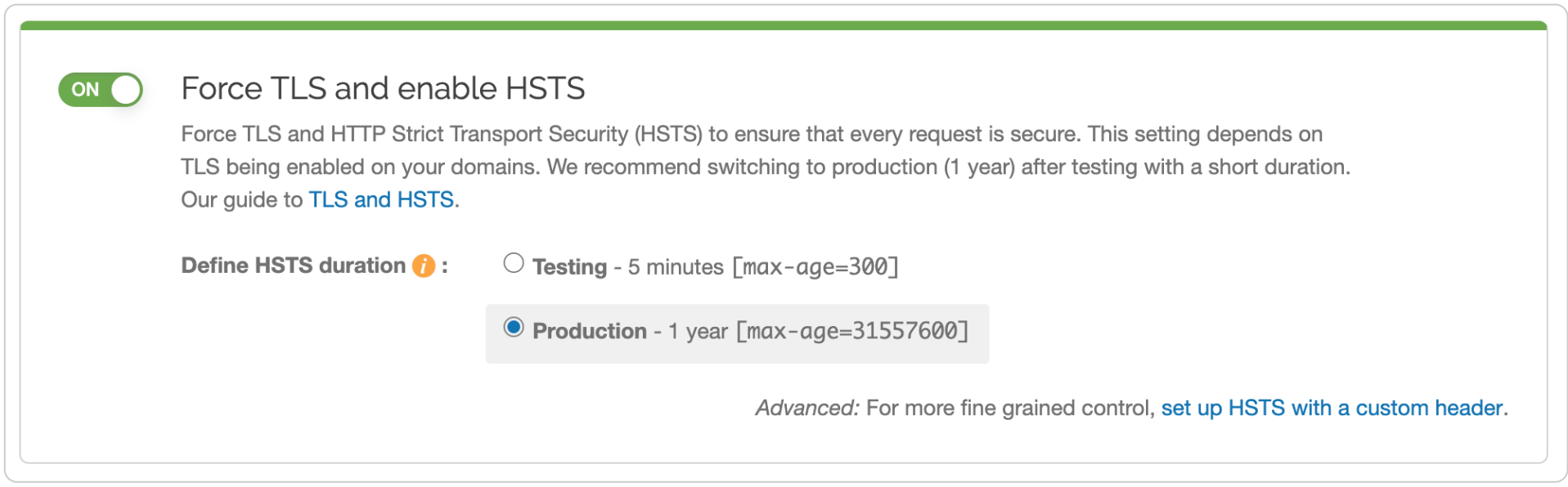
To force TLS and enable HSTS, follow these steps.

NOTE

Services activated using a previous version of the Force TLS controls may temporarily display an additional, older testing duration. Once you select the recommended new testing duration, this older option will disappear.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. If you have **Force TLS** enabled for any request settings without a condition, conflicts in the VCL logic may occur. Delete the existing request setting or modify it to add a condition.

6. Click the **Force TLS and enable HSTS** switch to force TLS and enable HSTS for the service.



The request setting for forcing TLS and the header for enabling HSTS will automatically be created for you.

7. Click the **Activate** button to deploy your configuration changes.

WARNING

You may experience problems if you enable this setting along with the override host setting. Instead of enabling the override host setting, create a new request setting and specify the override host in the advanced options.

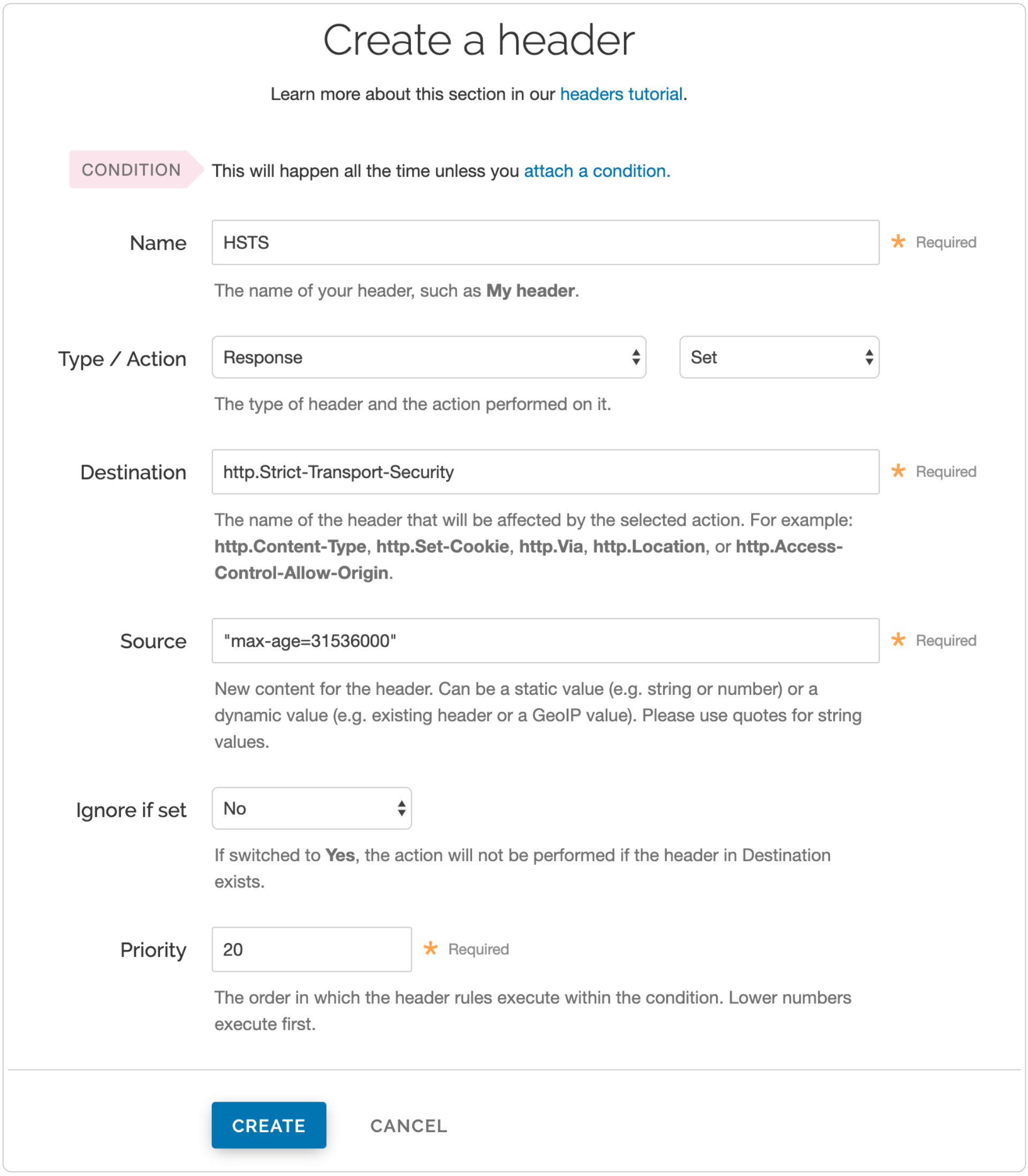
Manually enabling HSTS

If you'd like configure additional HSTS options, you'll need to manually enable HSTS by adding a new header as follows.

NOTE

If you followed the instructions in the previous section, click the **Force TLS and enable HSTS** switch to remove the request setting and header that were automatically created.

1. Follow the instructions in forcing a TLS redirect to force unencrypted requests over to TLS.
2. Click the **Content** link. The Content page appears.
3. Click the **Create header** button to create a new header. The Create a header page appears.



4. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a human-readable name, such as `HSTS`. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Response**, and from the **Action** menu select **Set**.
   - In the **Destination** field, enter `http.Strict-Transport-Security`.
   - In the **Source** field, enter `"max-age=<max age in seconds>"`. For example, `"max-age=31536000"`. As described below, `max-age` is required and two additional HSTS options can be specified.
   - Leave the **Ignore if set** menu and the **Priority** field set to their defaults (or set them as appropriate for your service).
5. Click the **Create** button.
6. Click the **Activate** button to deploy your configuration changes.

HSTS options

If you manually configured the HSTS header, you can specify additional HSTS options.

HSTS requires the max-age directive be set in order to function properly. It specifies how long in seconds to remember that the current domain should only be contacted over HTTPS. The example shown above sets `max-age` to one year (31536000 seconds = 1 year). You may want to experiment using a smaller value than what is shown.

Two additional options can be specified with the HSTS response header:

- `includeSubdomains` - This token applies HSTS to all of your site's subdomains. Before you include it, be certain none of your subdomains require functionality on HTTP in a browser. Ensure your TLS certificate is a wildcard or has coverage for all subdomain possibilities.

  > **IMPORTANT**
  >
  > All subdomains will be unreachable on HTTP by browsers that have seen the HSTS header once `includeSubdomains` is enabled.

- `preload` - This token allows you to submit your domain for inclusion in a preloaded HSTS list that is built into several major browsers. Although the token is not part of the HSTS specification, including it in the header is a prerequisite for submitting to this preloaded list.

  > **WARNING**
  >
  > Don't request browser preload inclusion unless you're sure that you can support HTTPS for the long term. Inclusion in the HSTS Preload List cannot be undone easily. See https://hstspreload.org/ for submission instructions and more information.

Combining all of these options together in the **Source** field would look like this:

```
"Strict-Transport-Security: max-age=<max age in seconds>; includeSubDomains; preload"
```

To disable HSTS for whatever reason, simply set the `max-age` to `0` on an HTTPS connection.

The HSTS Preload List is managed by a third party, not by Fastly. See https://hstspreload.org/ for more information.

Additional reading

- RFC 6797, which describes the HSTS specification
- the Wikipedia description of HSTS, including the currently known limitations and a browser support list
- the OWASP.org explanation of HSTS, including descriptions of the threats it addresses
- the Chromium Projects description of HSTS and preloading HSTS sites

Enabling TLS 1.3 through Fastly

Last updated: 2022-03-29

https://docs.fastly.com/en/guides/enabling-tls-1-3-through-fastly

This guide describes how to use Fastly TLS to enable TLS 1.3 for a domain using a TLS certificate you provide or one that Fastly provides and manages.

About TLS 1.3

To serve secure, encrypted traffic from Fastly using the Hypertext Transfer Protocol Secure (HTTPS) protocol, a website or application must provide a valid TLS certificate that is digitally signed by a trusted certification authority. Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are the protocols that allow clients to form secure communication connections between web browsers or applications and the servers they request information from.

TLS 1.3, the newest version of the TLS protocol, was designed to improve the performance and security of traffic for HTTPS domains. Specifically, this version of the protocol was designed to help speed up encrypted connections to servers by eliminating an entire round trip from its connection establishment handshake. The Zero Round Trip Time (0-RTT) feature can reduce the latency of resumed connections by encrypting requests in the initial ClientHello, a step in the client handshake process that specifies the maximum protocol version the client wishes to support.

In addition, TLS 1.3 allows only cipher suites that offer Perfect Forward Secrecy (PFS) for securing and encrypting traffic. TLS 1.3 also specifically prohibits TLS renegotiation, a process that allows changing the details of a TLS handshake after a connection has already been established with the server. Both restrictions make TLS 1.3 more secure than previous versions of the protocol.

When to use the web interface and when to contact Support

You can only enable TLS 1.3 via the web interface if you have purchased or are using:

- Fastly TLS,
- Concierge TLS, or
- Fastly's Legacy Customer-Provided TLS Certificate Hosting Service

and your domains have not been configured on Dedicated IP addresses that Fastly maintains and manages for you.

If you have purchased or are using Platform TLS or Dedicated IP addresses, you must contact support@fastly.com and have them enable TLS 1.3 for you.

Limitations and key behaviors

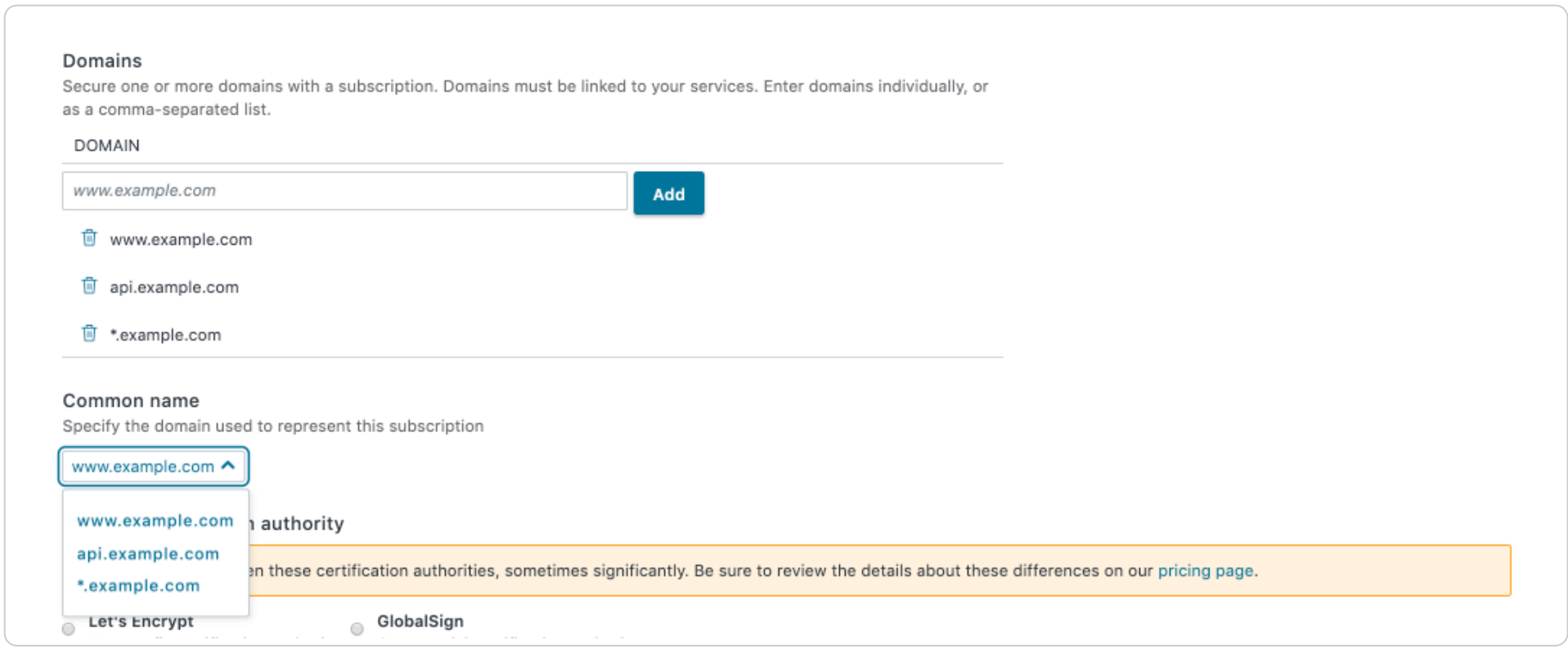
Before enabling or requesting this functionality, keep the following in mind:

- Negotiation of the TLS protocol will only happen if the requesting client also supports TLS 1.3. If a request comes from an older client, Fastly's default behavior is to downgrade to TLS 1.2.
- Fastly currently only supports 0-RTT between Fastly and requesting clients. We do not support 0-RTT between Fastly and your origin servers.
- By default, Fastly only answers idempotent requests (GET and HEAD requests without query parameters) over 0-RTT. This helps protect customer applications from replay attacks.
- Requests issued with 0-RTT will include an `Early-Data:1` header per RFC 8470. This attribute can be queried and logged via VCL using `req.http.early-data`.

Setting up TLS 1.3 for a new domain

Setting up TLS for a domain requires you to secure the domain by registering it with a certification authority. To start this process through Fastly's web interface (instead of programmatically) follow these steps.

1. Log in to the Fastly web interface and click the **Secure** link. The TLS domains page appears, displaying any domains for which you have TLS either enabled or for which TLS can be enabled. If you've not yet started setting up TLS on any of your domains, this page appears empty.
2. Click the **Secure another domain** button.
3. Decide what to do next:
   - If you have your own TLS certificates and private keys, click the **Use certificates you've provided** link and then follow the instructions in the guide to uploading and deploying your own certificates instead of this one.
   - If you want Fastly to procure and manage your TLS certificates and keys, continue with the remaining steps that follow.
4. From the selection menu that appears, select **Use certificates Fastly obtains for you**. The Enter subscription details page appears.
5. In the **Domain** field, enter one or more apex domains (e.g., `example.com`), subdomains (e.g., `www.example.com` or `api.example.com`), or a wildcard domain (e.g., `*.example.com`) and click the **Add** button. Domains you add appear in the Common name area of the page.



If you only have one domain, the common name will be the same as the domain name. If you add more than one domain, they will appear in a menu. By default, the first domain you add will be selected for you. Select another domain from the **Common name** menu if that's not the one you want.

6. From the **Select a certification authority** controls, choose one of the certification authorities to secure your certificate. Prices vary between certification authorities, sometimes significantly. Be sure to review the details about these differences on our pricing page.

7. If you previously enabled TLS in your Fastly account, use the **Select a TLS configuration** menu to select a TLS 1.3 configuration to apply. Your selection will specify both the IP addresses that the certificate will be deployed to and the associated TLS settings that will be applied to them.

   - Select **HTTP/3 & TLS v1.3** to apply the latest version of the protocol, but without 0-RTT.
   - Select **HTTP/3 & TLS v1.3 + 0RTT** to apply the latest version of the protocol with 0-RTT.

   However, if you are enabling TLS in your Fastly account for the first time on or after March 29, 2022, this menu will not appear. Your TLS configuration will use HTTP/3 & TLS v1.3 + 0RTT by default.

8. Click **Submit**. The Subscription details page appears displaying your domains along with detailed steps on how to verify you own them.

9. Click the **View details** button to view information for your domain and use it to update your DNS records with your DNS provider.

Applying TLS 1.3 to an existing domain

To migrate an existing domain to a new TLS 1.3 configuration, follow these steps:

1. Log in to the Fastly web interface and click the **Secure** link. The TLS domains page appears, displaying any domains for which you have TLS either activated or for which TLS can be activated.

2. Find the card for the appropriate domain.

3. Click the **View/Edit Activation** button next to the appropriate certificate.

4. From the list of configurations that appear, click **Activate** next to the TLS 1.3 configuration you want to apply. Your selection will specify both the IP addresses that the certificate will be deployed to and the associated TLS settings that will be applied to them.
   - Activate **HTTP/3 & TLS v1.3** to apply the latest version of the protocol, but without 0-RTT
   - Activate **HTTP/3 & TLS v1.3 + 0RTT** to apply the latest version of the protocol with 0-RTT.

5. Click **Done**.

6. Watch the **TLS status** area for the certificate. Once the configuration is selected, the status for the domain will change to `Deploying`. When the TLS status area changes back to `Activated`, the TLS configuration selected will have been applied to the domain.

7. Click the **View details** button to view information for your domain and use it to update your DNS records with your DNS provider.

8. Confirm the new DNS records have propagated across the internet (this can take up to 48 hours), then delete the old TLS configuration by clicking the trash can icon.

Forcing a TLS redirect

Last updated: 2020-11-20

https://docs.fastly.com/en/guides/forcing-a-tls-redirect

If you want to only allow TLS on your site, we have you covered. We've built a switch into the request settings that will allow you to force unencrypted requests over to TLS. It works by returning a **301 Moved Permanently** response to any unencrypted request, which redirects to the TLS equivalent. For instance, making a request for **http**://www.example.com would redirect to **https**://www.example.com.

NOTE

Because requests can still happen over HTTP first even if you force a TLS redirect using these instructions, we recommend enabling HSTS as well. Fastly provides a different switch that lets you easily force TLS and enable HSTS at the same time. Alternatively, you can follow these instructions to force a TLS redirect and manually enable HSTS later.

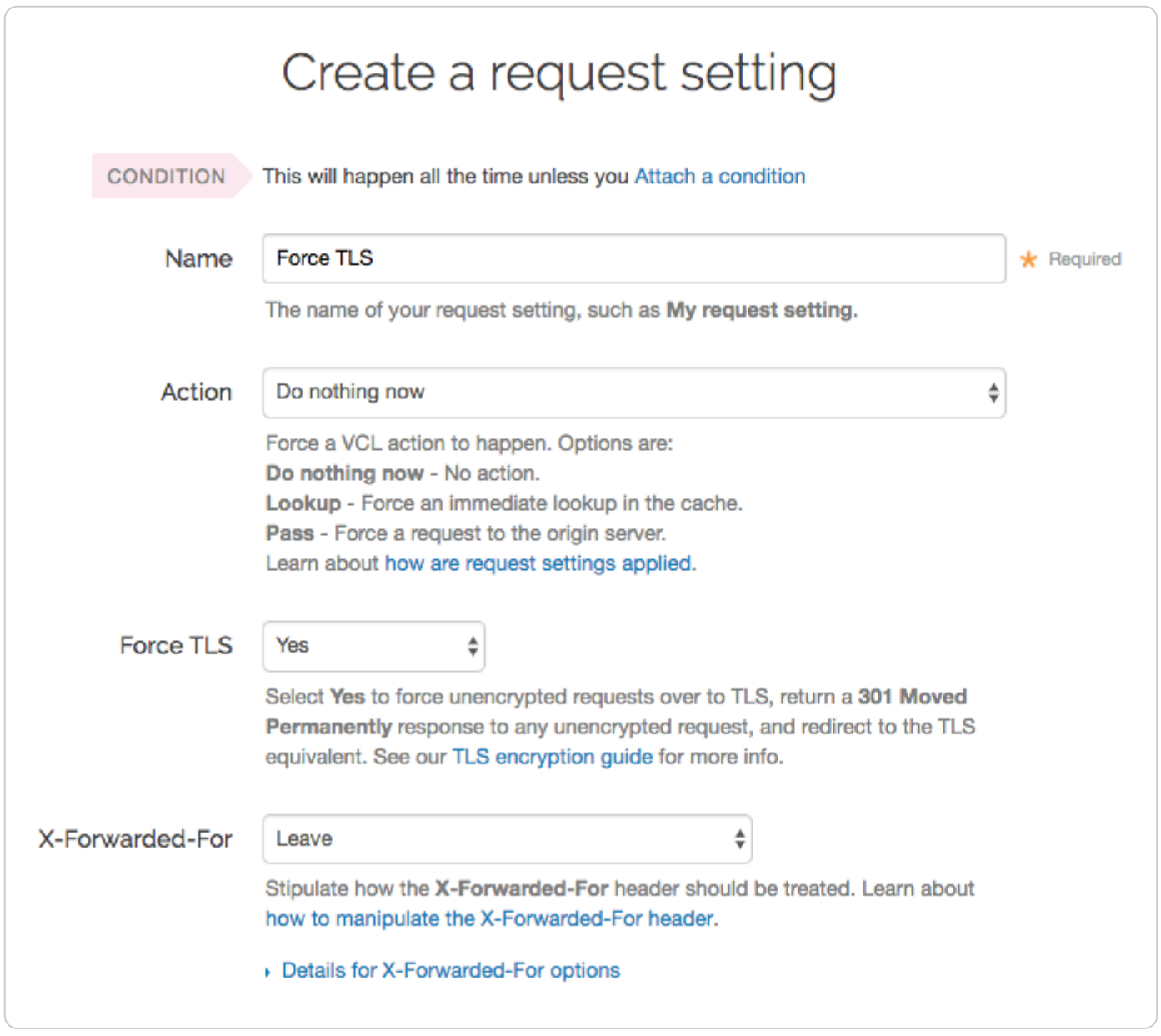
Prerequisites

These instructions assume that you've set up TLS service with Fastly.

Forcing a TLS redirect

To force a TLS redirect, follow these steps:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **Create request setting** button. The Create a request setting page appears.



6. Fill out the **Create a request setting** fields as follows:
   - In the **Name** field, enter a human-readable name for the request setting. This name is displayed in the Fastly web interface.
   - From the **Force TLS** menu, select **Yes**.
7. Click the **Create** button to save your request setting changes.
8. Click the **Activate** button to deploy your configuration changes.

Serving HTTPS traffic using certificates you manage

Last updated: 2022-03-21

https://docs.fastly.com/en/guides/serving-https-traffic-using-certificates-you-manage

This guide describes how to use the Fastly TLS product to upload and deploy your own TLS certificates and private keys using the Fastly web interface.

To serve secure traffic from Fastly using HTTPS, a website or application needs to provide clients with a valid TLS certificate signed by a trusted Certification Authority (CA). TLS (Transport Level Security) and its predecessor SSL (Secure Sockets Layer) are the protocols that allow clients to form secure server connections so traffic can be served over HTTPS.

> **TIP**
>
> Fastly offers an API for uploading and managing your keys and certificates used to activate TLS for your domains on Fastly.

Before you begin

To use Fastly TLS with certificates and keys that you upload and deploy via the web interface, be sure you have the following prerequisites in place:

- a paid Fastly user account (not a developer's trial) assigned the role of superuser or assigned a user role with added TLS management permission

- a valid X.509 TLS certificate from a trusted CA and a matching 2048-bit RSA private key
- permission to modify the DNS records on the relevant domains that appear as Subject Alternative Name (SAN) entries on the TLS certificate
- the relevant domains added to a properly configured Fastly service

The Fastly TLS web interface is compatible with certificates that have been uploaded as part of the Customer-Provided TLS Certificate Hosting Service with the following limitations:

- If you update previously uploaded certificates, you can continue to use the Customer-Provided TLS Certificate Hosting Service with no changes to your bill.
- Removing a previously uploaded certificate from the Customer-Provided TLS Certificate Hosting Service and uploading a new one using Fastly TLS will result in the new certificate being counted in your bill for Fastly TLS. The old certificate will continue to be billed per any contracted term for Customer-Provided TLS Certificate Hosting Service.

For information on migrating certificates from the Customer-Provided TLS Certificate Hosting Service to Fastly TLS, contact support@fastly.com.

In addition to these prerequisites, be sure you understand the following limitations:

- Fastly TLS comes with a 50 certificate limit. To discuss how to raise this product's certificate limit, contact sales@fastly.com.
- Uploaded certificates require clients to support TLS v1.2 and Server Name Indication (SNI) by default. To discuss how you can use settings other than these defaults, contact sales@fastly.com. The ability to use custom settings may require you to use a dedicated Fastly IP address pool, which must be purchased separately.
- Fastly TLS does not support the Triple DES (3des) cipher suite.
- If you're a DigiCert customer, be aware that upon making certificate changes, DigiCert will revoke your original certificate 72 hours after re-issuance. Be sure you upload your new certificate and switch all hostnames as soon as possible.

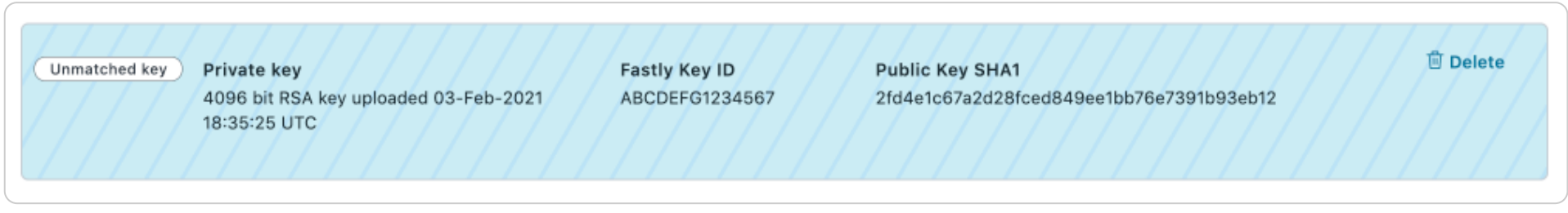
Uploading a private key and certificate

To upload a TLS certificate and the relevant private key (used to initially generate the certificate) follow the steps below.

> **IMPORTANT**
>
> The key file upload tool currently only accepts 2048-bit RSA keys. If you require longer key lengths, contact support@fastly.com.

1. Log in to the Fastly web interface and click the **Secure** link. The TLS domains page appears, displaying any domains for which you have TLS either activated or for which TLS can be activated. If you've not yet started setting up TLS on any of your domains, this page appears empty.
2. Click the **Secure another domain** button. The Enter domain window appears.
3. Click the **I want to bring my own certificate and private key** link. The TLS certificates tab appears displaying the secure upload controls.
4. Drag and drop your private key file into the drag and drop area to upload your private key file. Alternately, click the **Browse for key file** link to navigate to the file on your system using the file picker.

   The private key appears below the upload controls and displays the Unmatched key label prominently on the left.



> **TIP**
>
> Unmatched keys are private keys that have no matching TLS certificate. If you have multiple private keys, you can identify each by the unique SHA1 hash. Private keys can only be deleted if they are in the unmatched state.

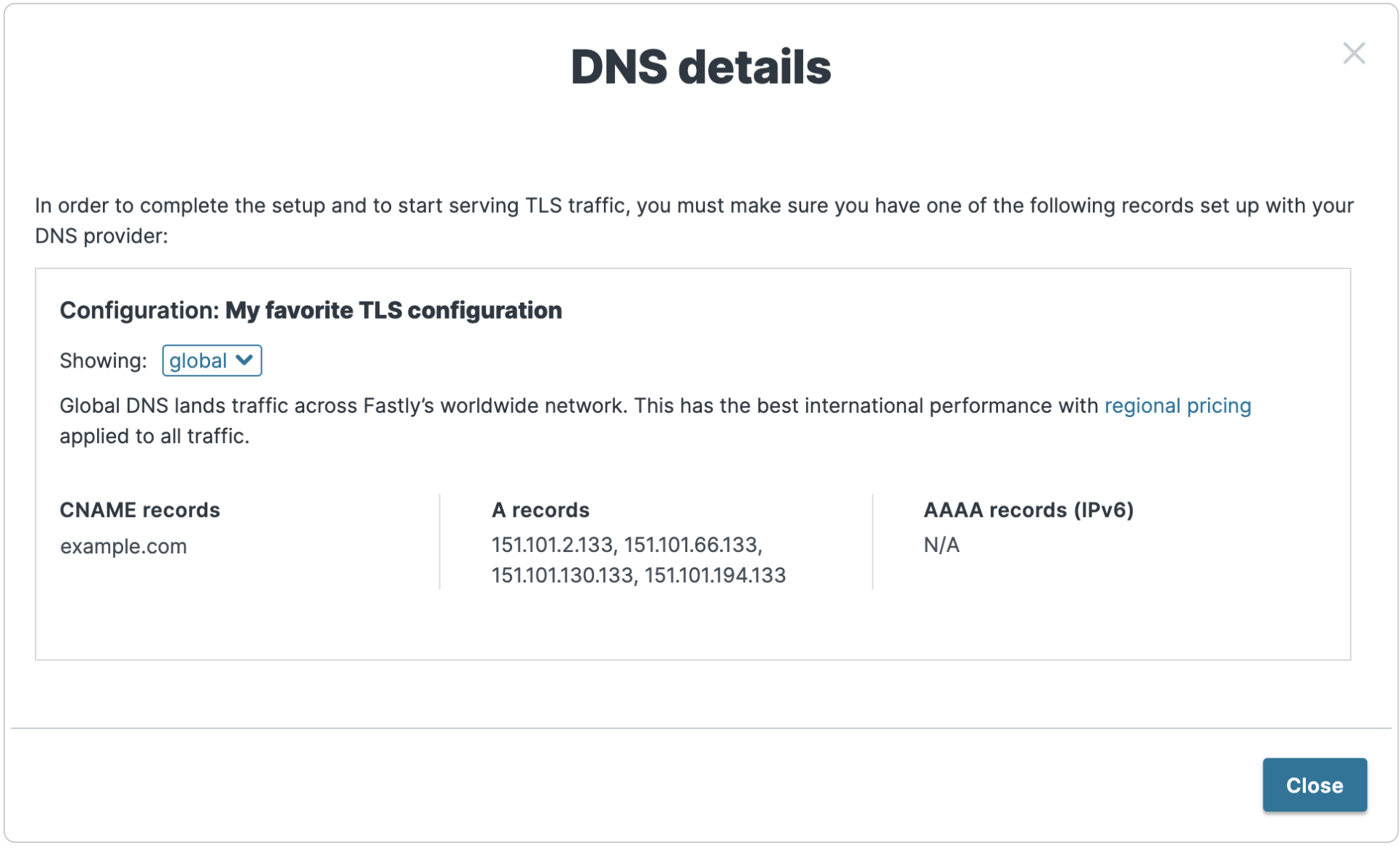
5. Upload a TLS certificate using the same drag-and-drop area or file picker you used to upload your private key. A success message appears.

Activating TLS on a domain

Once a valid certificate and private key have been uploaded, all domains that appear as SAN entries will be listed on the TLS domains page with a status of TLS ready. To serve HTTPS traffic using your certificate, follow the steps below to activate TLS for the domain and point the DNS records to the certificate's location.

1. Click the **TLS domains** tab. The list of all domains that appear as SAN entries appear. Domains in a disabled state will have the status of `Ready to activate`.
2. Find the card for the domain where you want to activate TLS.
3. Click the **Activate this certificate** button to the right of the certificate you want to activate. The DNS details appears.

   Fastly deploys your TLS certificate to the entire Fastly edge network. It may take up to an hour for your certificate to become available throughout the world.
4. Use the DNS details displayed in this section to configure your DNS records so that a TLS connection can be established using your certificate.



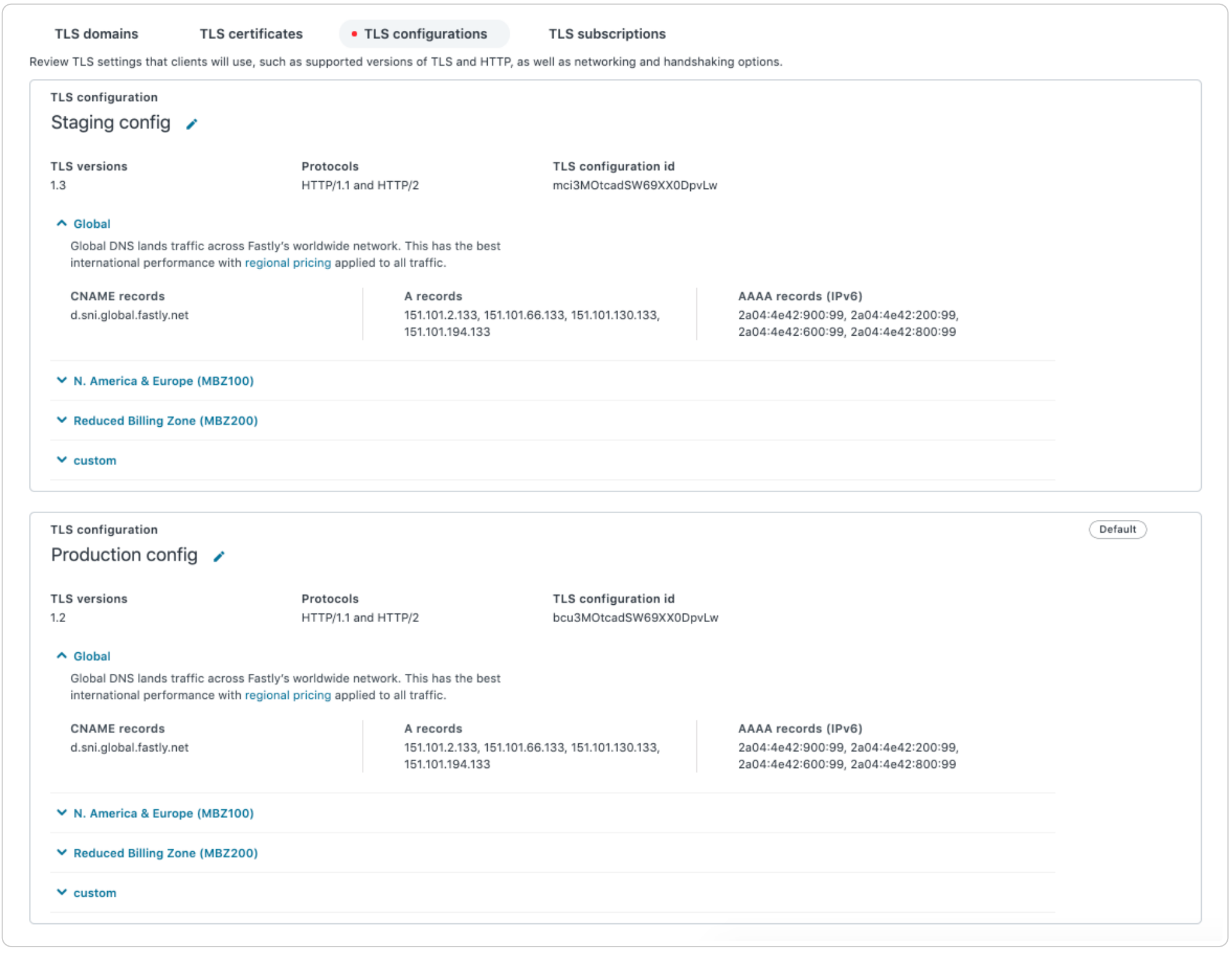
- For TLS on an apex domain (e.g., `example.com`), you'll need to create an A record with your DNS provider.
- For subdomains and wildcard domains (e.g, `www.example.com` or `*.example.com`), you'll need to create a relevant CNAME record.

IMPORTANT

It can take up to 48 hours for new DNS records to propagate across the internet.

Applying a TLS configuration to a domain

TLS configurations are a collection of TLS settings that include the supported versions of TLS and HTTP, along with networking and handshaking options that clients will use. For accounts with more than one TLS configuration, the default configuration has a label in the right corner of the card.



TIP

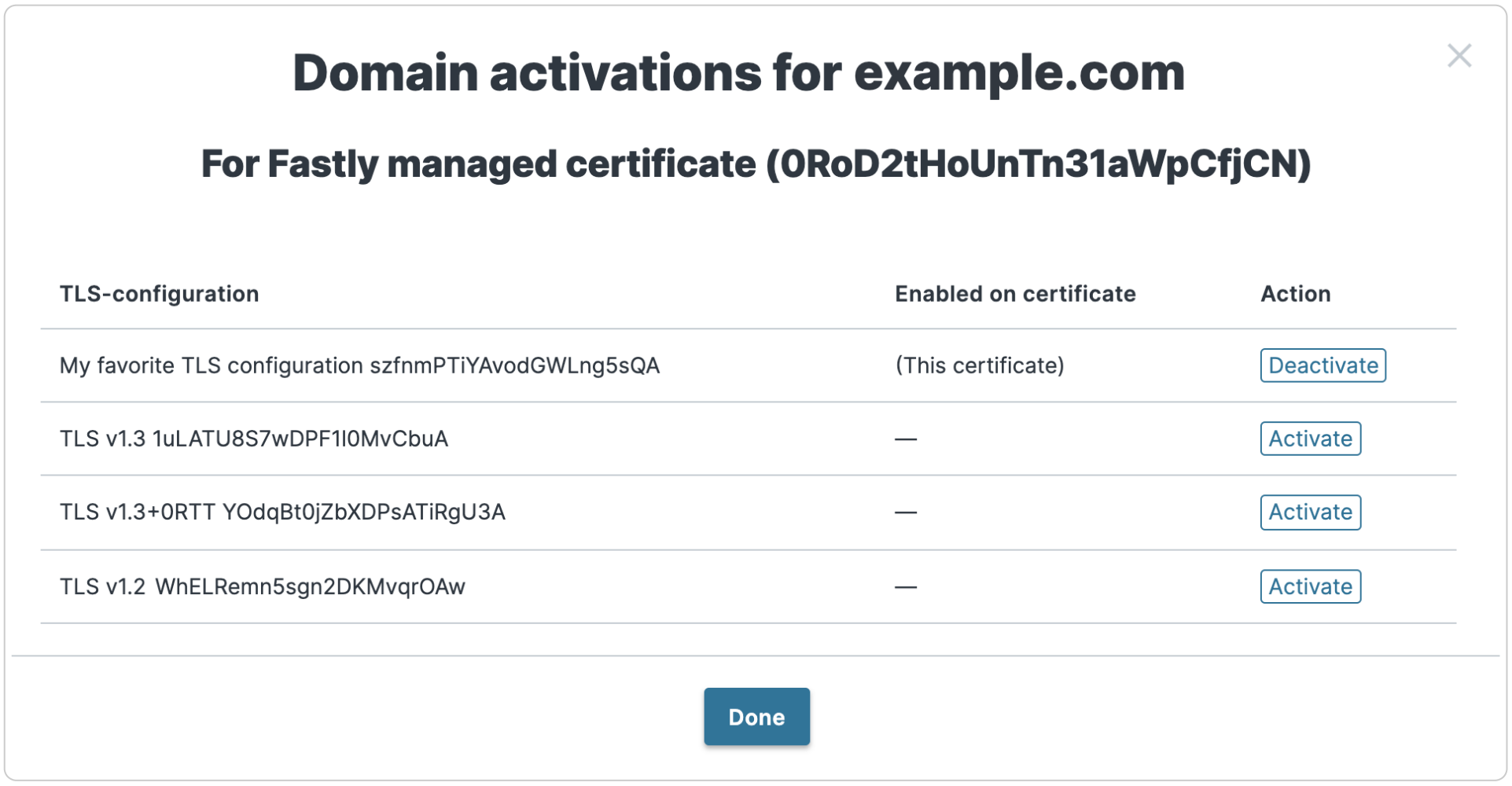
TLS configuration names are editable by clicking the pencil icon next to the name.

To override the default TLS configuration applied to a domain or to migrate a domain to use a different configuration follow these steps.

1. Click the **TLS domains** tab.
2. Find the card for the domain where you want to change the configuration.
3. Click the **View/Edit Activations** button next to the certificate you want to modify. A list of TLS configuration(s) appears.
4. Click **Activate** next to the configuration you want to activate.

NOTE

While you may have multiple certificates on a given domain, only one certificate can be active for a given TLS configuration. If the TLS configuration is already in use by another certificate, a **Switch to this certificate** button appears.



Once the configuration is selected, the TLS configuration is applied to the domain. Each TLS configuration is active and available at their respective IP addresses and DNS records.

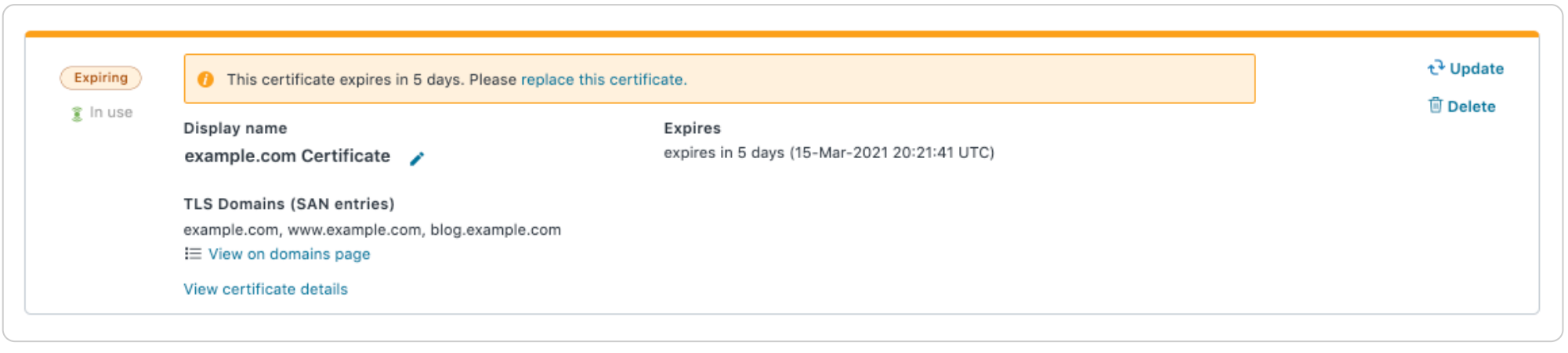
5. Click **See DNS details** and use the information to configure your DNS records so that a TLS connection can be established using your certificate.

6. Confirm the new DNS records have propagated across the internet (this can take up to 48 hours), then delete the old TLS configuration by clicking the trash can icon.

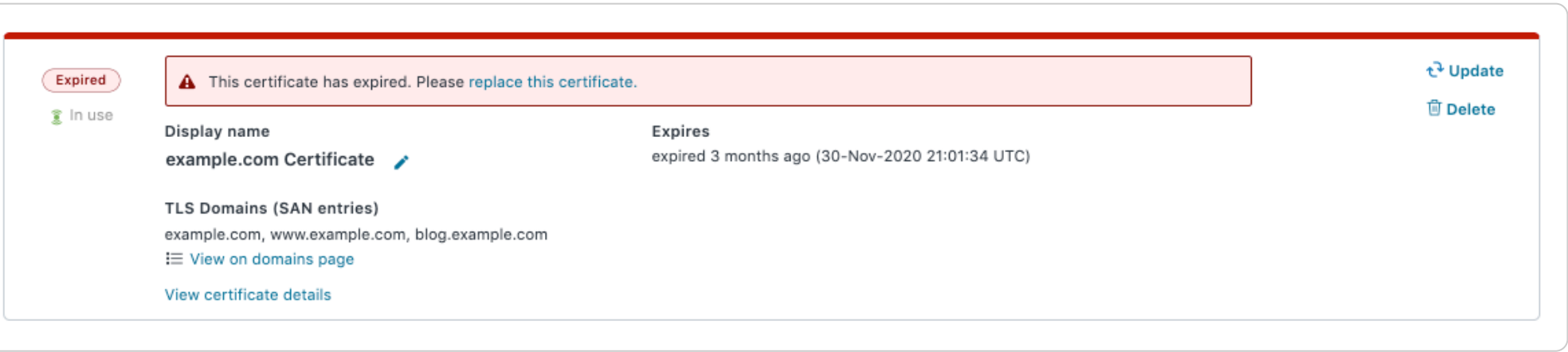
> **NOTE**
>
> For HTTP/1.1, be sure to activate TLS for each of your domains in the web application. If you upload a certificate with multiple SANs, each domain must have TLS explicitly activated if you want to secure these domains on Fastly.
>
> Exceptions may apply in the case of HTTP/2 if your browser coalesces secure connections and has previously received a TLS certificate from an earlier handshake. In this case, some browsers may reuse an existing secure connection to Fastly if its certificate has a matching SAN entry.

Updating a certificate

The TLS certificates page warns you when a certificate is nearing its expiration date:



or when a certificate is past its expiration date:



There may be situations where Fastly identifies certificates that should be updated and replaced. These certificates will be clearly marked with a recommendation:

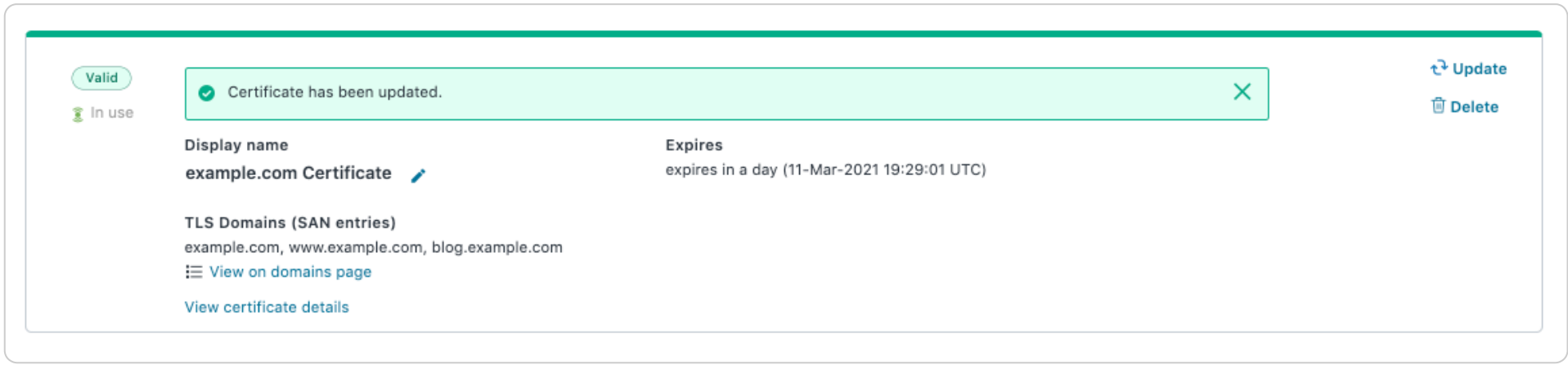
Fastly allows you update a certificate by replacing it with a new one at any time.

Updating certificates when there are no removed domains

To update a TLS certificate by replacing it with one that contains all the domains as the original (either as a superset or a matching list) follow these steps.

1. Generate a new certificate with your preferred Certification Authority.
2. If a new key was generated with the new certificate, click the **Upload a new key or certificate** button to upload it.
3. Find the certificate you're replacing on the **TLS certificates** tab in the web interface.
4. Click the word **Update** in the upper right corner of the certificate's card.

5. Using the file picker that appears, find the new certificate you're replacing the old one with. The certificate you select should be PEM-formatted and the SAN entries of this new certificate must contain all entries in the current certificate (i.e., it must either have an exact matching list or contain a superset).
6. Wait for the certificate update process to complete. A success message appears indicating the certificate has been successfully updated.

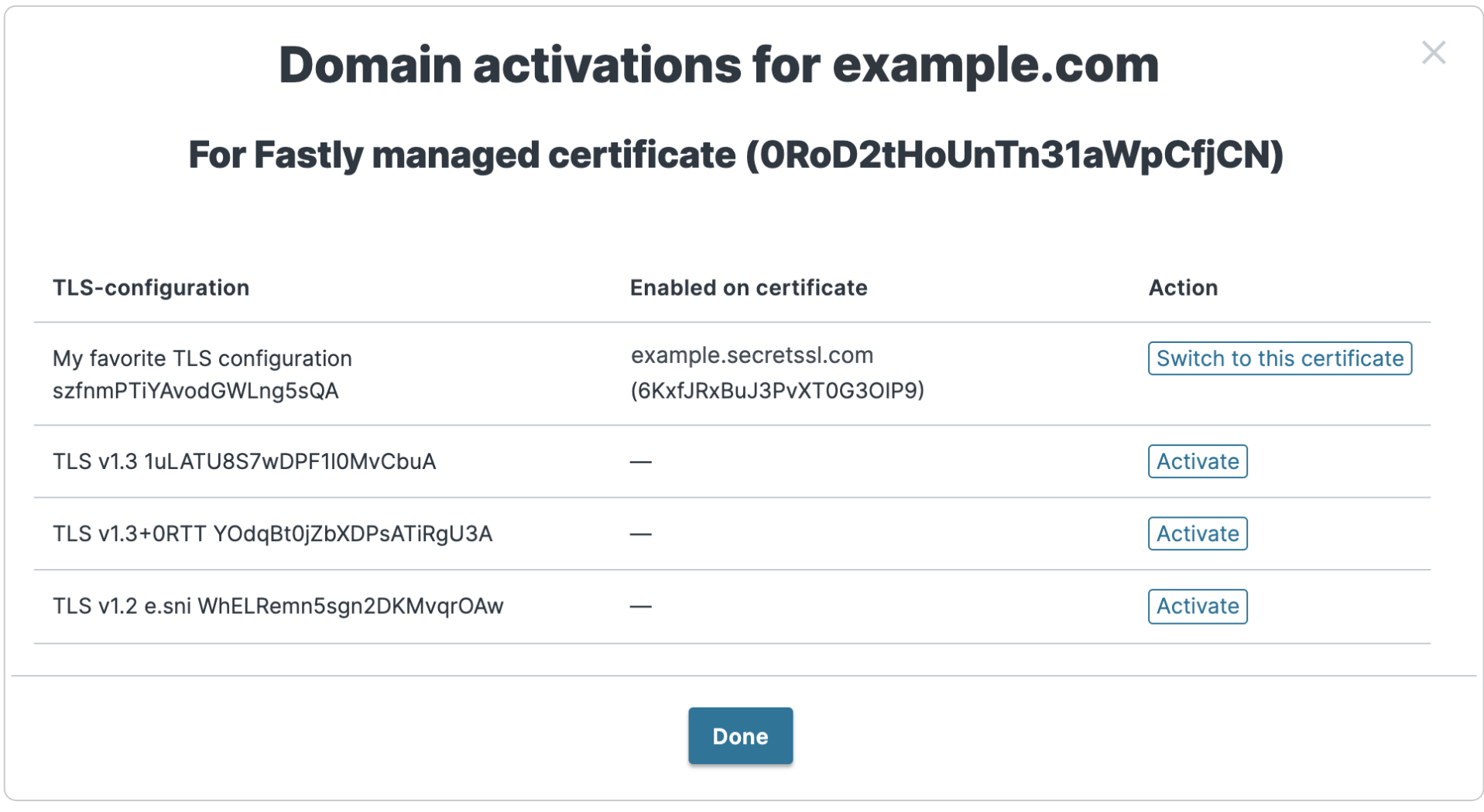


All domains actively serving TLS traffic on the old certificate will be automatically transitioned to the updated certificate within a matter of minutes. Any new domains will need to be manually activated by following the steps for activating TLS on a domain.

Updating certificates when there have been changes to domains

If you want to update one of your certificates that requires removing domains, you will need to procure a new certificate with the updated list of SAN entries. Follow the steps to upload this certificate as a new certificate.

1. Upload the new certificate.
2. Click the **TLS domains** tab and find the previously existing domains that have already been activated with the certificate you're replacing.
3. Click **View/Edit Activations** next to the appropriate certificate on the domain card to see the active TLS configurations.
4. From the list of configurations that appear, click the **Switch to this certificate** button next to the configuration you want to apply.



5. Click **Done**.
6. Delete the old certificate.
7. Manually activate any new domains that have been added to the updated certificate by following the steps for activating TLS on a domain. Certificates for newly activated domains can take between 5 minutes to an hour to fully deploy across Fastly's global network. If the new certificate is not being used to serve TLS traffic within 1 hour, contact support@fastly.com for assistance.

> **TIP**
>
> Certificate display names are editable by clicking the pencil icon next to the name.

Deactivating TLS and deleting certificates and private keys

Once a domain has TLS activated, you have the option to deactivate TLS via the **Deactivate TLS** button listed on the TLS domains page. If a domain has multiple certificates, you can elect to deactivate a specific certificate by clicking **Add/Edit Activations** and clicking the **Deactivate** button next to any active configurations. If all certificates are deactivated, Fastly will no longer serve TLS traffic on the selected domain and it will become disabled.

To delete a certificate from the TLS certificates page, be sure to disable TLS for all domains on that specific certificate. You will also need to delete all certificates before you can delete a matching private key.

Certificate expirations

Thirty days before any certificate is due to expire, the web interface will display warnings on certificates soon to expire. Fastly will also begin to periodically send automated expiration notification emails to all superusers. If the certificate is not replaced or removed, Fastly will continue to email users on the account until the certificate expires. Once expired, Fastly will no longer send

automatic notifications.

Serving HTTPS traffic using Fastly-managed certificates

Last updated: 2022-03-23

https://docs.fastly.com/en/guides/serving-https-traffic-using-fastly-managed-certificates

This guide describes how to use Fastly TLS to enable HTTPS for a domain using a certificate managed by Fastly. Fastly-managed certificates use the ACME protocol to procure and renew TLS certificates from Let's Encrypt, a non-profit certification authority, and GlobalSign, a commercial certification authority.

To serve secure traffic from Fastly using HTTPS, a website or application needs to provide clients with a valid TLS certificate signed by a trusted certification authority. TLS (Transport Level Security) and its predecessor SSL (Secure Sockets Layer) are the protocols that allow clients to form secure server connections so traffic can be served over HTTPS.

TIP

Our TLS subscriptions API allows you to manage Fastly TLS subscriptions programmatically.

Before you begin

To use Fastly TLS with Fastly-managed certificates, be sure you have the following prerequisites in place:

- a paid Fastly user account (not a developer's trial) assigned the role of superuser or assigned a user role with added TLS management permission
- permission to modify the DNS records on the relevant domains that appear as SAN entries on the TLS certificate
- the relevant domains added to a properly configured Fastly service

The Fastly TLS web interface is compatible with certificates that have been uploaded as part of the Customer-Provided TLS Certificate Hosting Service with the following limitations:

- If you replace previously uploaded certificates, you can continue to use the Customer-Provided TLS Certificate Hosting Service with no changes to your bill.
- Removing a previously uploaded certificate from the Customer-Provided TLS Certificate Hosting Service and uploading a new one using Fastly TLS will result in the new certificate being counted in your bill for Fastly TLS. The old certificate will continue to be billed per any contracted term for the Customer-Provided TLS Certificate Hosting Service.

For information on migrating certificates from the Customer-Provided TLS Certificate Hosting Service to Fastly TLS, contact support@fastly.com.

In addition to these prerequisites, be sure you understand the following limitations:

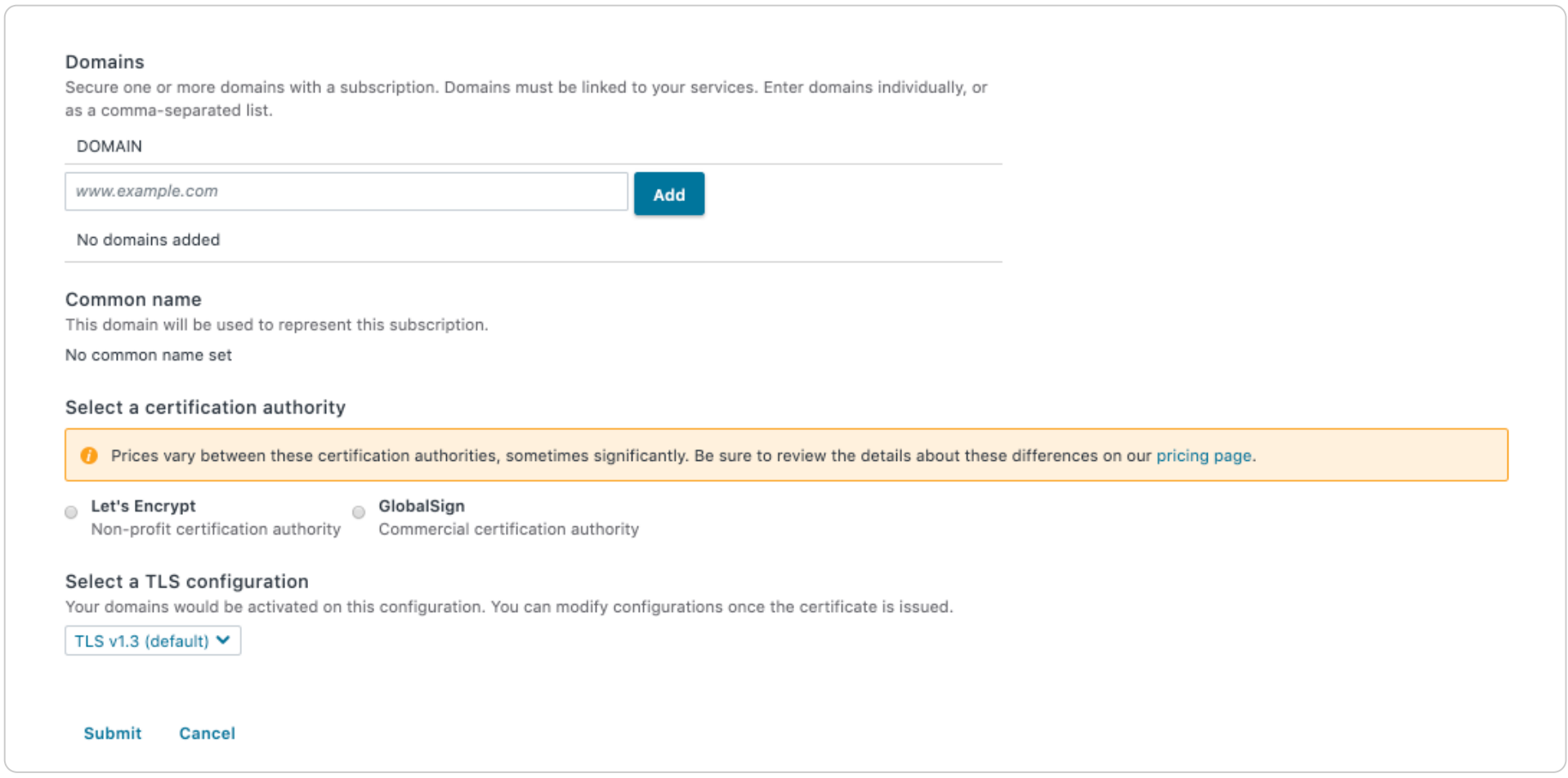
- Fastly TLS comes with a 50 certificate limit. To discuss how to raise this product's certificate limit, contact sales@fastly.com.
- Fastly managed certificates require clients to support TLS v1.2 and Server Name Indication (SNI) by default. To discuss how you can use settings other than these defaults, contact sales@fastly.com. The ability to use custom settings may require you to use a dedicated Fastly IP address pool, which must be purchased separately.
- Fastly TLS does not support the Triple DES (3des) cipher suite.

Setting up TLS for a domain

Setting up TLS for a domain requires you to secure the domain by registering it with a certification authority. To start this process through Fastly's web interface (instead of programmatically) follow these steps.

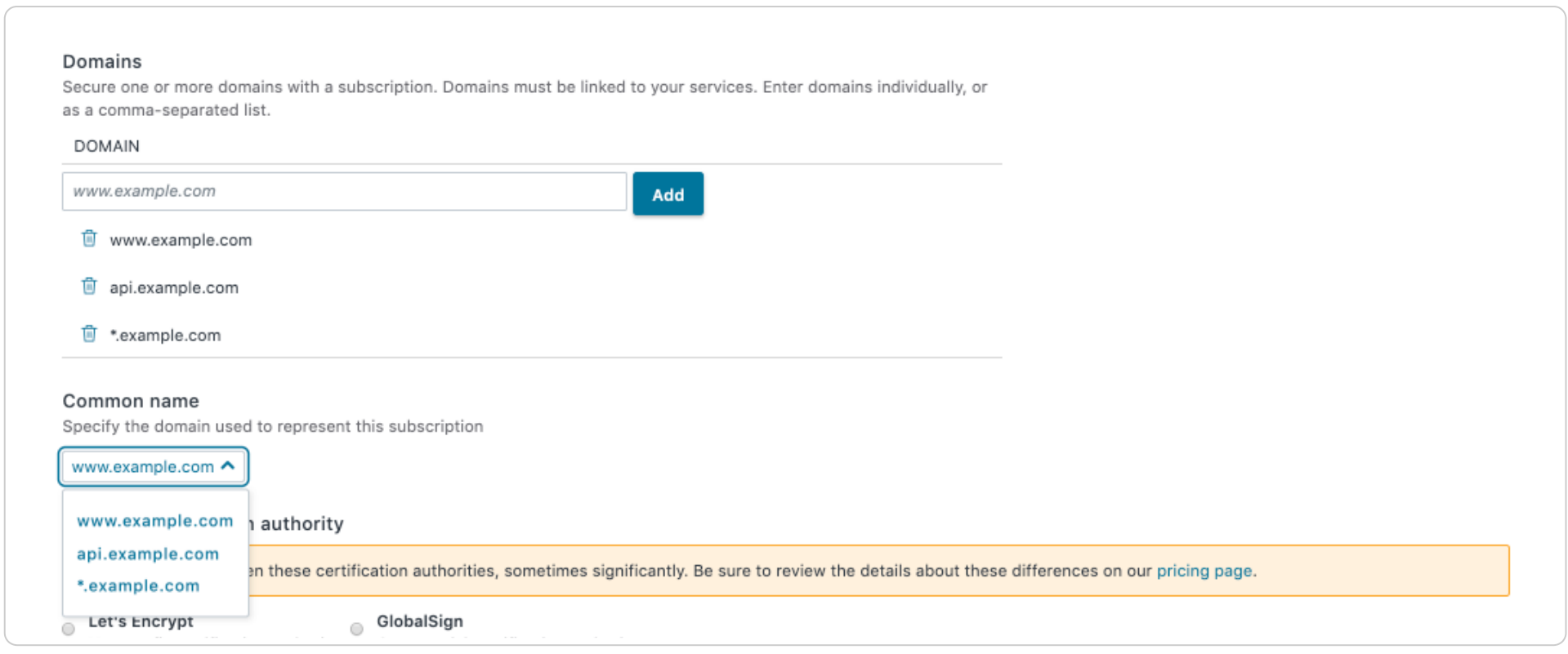
1. Log in to the Fastly web interface and click the **Secure** link. The TLS domains page appears, displaying any domains for which you have TLS either activated or for which TLS can be activated. If you've not yet started setting up TLS on any of your domains, this page appears empty.
2. Click the **Secure another domain** button.

3. From the selection menu that appears, select **Use certificates Fastly obtains for you**. The Enter subscription details page appears.



4. In the **Domain** field, enter one or more apex domains (e.g., `example.com`), subdomains (e.g., `www.example.com` or `api.example.com`), or a wildcard domain (e.g., `*.example.com`) and click the **Add** button. Domains you add appear in the Common name area of the page.

   If you only have one domain, the common name will be the same as the domain name. If you add more than one domain, they will appear in a menu. By default, the first domain you add will be selected for you. Select another domain from the **Common name** menu if that's not the one you want.



5. From the **Select a certification authority** controls, choose one of the certification authorities to secure your certificate. Prices vary between certification authorities, sometimes significantly. Be sure to review the details about these differences on our pricing page.

6. Optionally, if you have access to multiple Fastly IP addresses or have multiple Fastly CNAME records created with different networking or TLS configurations, then from the **Enable on** menu in the **Select a TLS configuration** area, select which TLS configuration to apply. This option defines both the IPs that the certificate will be deployed to and the associated TLS settings that will be applied.

7. Click **Submit**. The Subscription details page appears displaying your domains along with detailed steps on how to verify you own them.

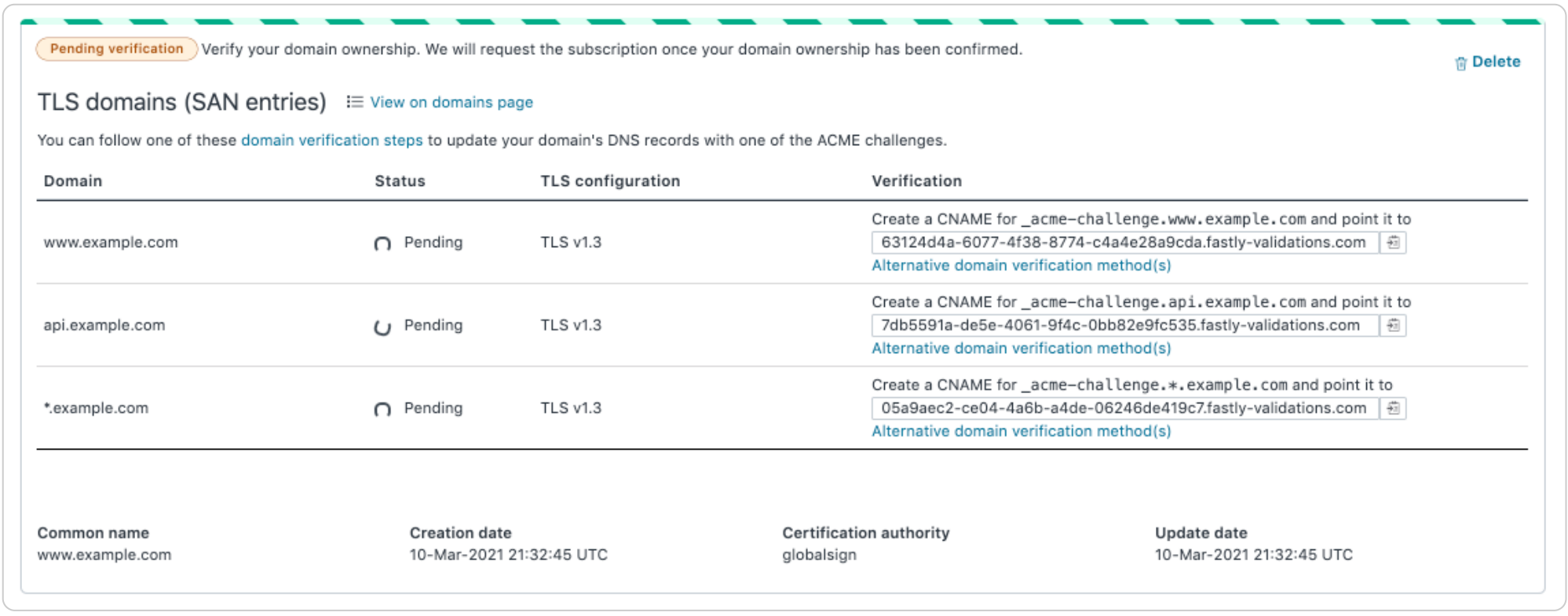
Verifying domain ownership

To begin serving HTTPS traffic, Fastly needs to verify that you control any domain you've added to the web interface. Fastly allows you to verify apex domains and subdomains via the ACME DNS challenge, the ACME HTTP challenge, or via email validation. Each requires you to make specific DNS changes. Wildcard domains require the DNS challenge or email validation challenge type.

> **IMPORTANT**
>
> Fastly may modify the behavior of your services and DNS to complete ACME challenges for domain verification.

Using the ACME DNS challenge to verify domain ownership

The default method for verifying you control a domain being added to a Fastly managed TLS certificate uses the ACME DNS challenge type. It's suitable for all kinds of domains (apex, subdomains, and wildcards). It will only point the `_acme-challenge` subdomain at Fastly, allowing you to set up TLS first, before pointing production traffic at Fastly.



To use this verification method, create a CNAME record with a unique target for your domain. The formats for the record and target appear in the **What's next** notification above your domain's name in the list of TLS domains.

The steps to create the CNAME record will vary depending on your DNS provider's control panel interfaces. Refer to your DNS provider's documentation for exact instructions on how to do this. Your CNAME record must use the format `_acme-challenge.DOMAIN_NAME` (e.g., `_acme-challenge.www.example.com`) and must be pointed to a unique target for your domain (e.g., `domain_token.fastly-validations.com`). Once you've pointed your DNS records at Fastly, we encourage you to keep the `_acme-challenge` subdomain CNAME in place to avoid interruptions in service.

Using the ACME HTTP challenge to verify domain ownership

Another method for verifying you control a domain uses the ACME HTTP challenge. This method is only suitable for apex domains and subdomains (wildcard domains can only be verified using DNS or email challenges). It will point production traffic immediately at Fastly.

> **WARNING**
>
> The ACME HTTP challenge domain verification method can potentially point all end-user traffic to Fastly before you've completed TLS setup. This means that your end users may get an insecure warning in their browser related to your website or be totally unable to access your domain. To avoid this, ensure you have a properly configured Fastly service and have disabled any forced TLS redirection before you use this verification method.

To use this verification method, click the **Alternative domain verification method(s)** link in the **What's next** notification above your domain's name in the list of TLS domains. A verification options window appears asking you to choose the verification alternative that suits your needs for the ACME HTTP challenge:

- for a subdomain, create a CNAME record that points directly to the Fastly hostname
- for an apex domain, create A records for the domain with the noted IP addresses

Once set up using either alternative, production traffic will immediately begin flowing through Fastly.

Using an email challenge to verify domain ownership

Domain control can also be verified via email when you've chosen GlobalSign as your certification authority. (Let's Encrypt does not support email challenges for domain verification.) To use this verification method, follow the steps below.

1. Log in to the Fastly web interface and click the **Secure** link. The TLS domains page appears, displaying any domains for which you have TLS either activated or for which TLS can be activated.
2. Click the **TLS subscriptions** tab.
3. Click the **Alternative domain verification method(s)** link. The verify domain ownership page appears.
4. From the Email validation menu, select the email address you want email verification sent to. When selected as the certification authority, GlobalSign will provide Fastly with a list of acceptable email addresses to which a verification email can be sent. Generally, the list will include email addresses like the following:
    - admin@example.com
    - administrator@example.com
    - hostmaster@example.com
    - postmaster@example.com
    - webmaster@example.com
5. Click the **Get verification email** button.

Fastly will then instruct GlobalSign to send a verification email to the address you specify. It will contain a link that you must click to complete the domain ownership verification process.

What happens next

It should take no more than an hour for the TLS enablement process to progress through all of the TLS statuses shown below:

TLS Status	Description
Checking domain DNS records... Step 1 of 3	Domain validation is in progress. Fastly is checking domain DNS records to verify that you control the domain being added to a certificate. To advance to the next enablement state, you must verify control of the domain by updating that domain's DNS records to complete one of the ACME challenge types.
Certificate requested. Waiting for response from CA... Step 2 of 3	Domain validation has been confirmed. Fastly has verified you control the domain and has requested a TLS certificate for it from the certification authority.
TLS enabled (certificate being deployed globally)	The certification authority has issued a TLS certificate. Newly issued certificates can take between 20 minutes to an hour to fully deploy across Fastly's global network.

Troubleshooting

If more than an hour has passed and TLS enablement appears to be stalled in one of the stages of adding a domain, there is likely an issue.

Domains stuck in the Checking domain DNS records state

If the domain is stuck in the `Checking domain DNS records` state, it is likely that you have not configured your DNS records correctly in order to verify domain ownership. You can check the DNS records yourself using a `dig` command in a command line application as follows:

ACME challenge type	Command to type

ACME challenge type	Command to type
HTTP	`dig www.example.com +short`
DNS	`dig _acme-challenge.www.example.com +short`

Be sure to replace `example.com` with hostname you used when you configured your DNS records.

If you have correctly configured your DNS records, the result from this command will include one of the CNAME or A Records required for verification as defined in the Verifying domain ownership instructions.

If you recently added or modified DNS records, you may need to wait up to 72 hours for your DNS changes to propagate across the internet. If you don't see these addresses within that time period, you may have misconfigured your DNS records.

If you are still having issues, there may be a Certification Authority Authorization (CAA) record on your domain that is blocking the certification authority from issuing certificates. This CAA record is used to specify which certification authorities (CAs) are allowed to issue certificates for a domain. If a CAA record exists, you may need to remove this record in order to use a managed Fastly TLS certificate.

Domains stuck in the Certificate requested state

If the domain is stuck in the `Waiting for a response from CA` state, this is likely a temporary issue with the certification authority. Be sure to allow up to an hour in this state before contacting support@fastly.com for assistance. If this is a new certificate request, you can also try deleting the domain and starting again.

TLS activated but certificate not deployed everywhere

If the domain displays the `Activated` state but the certificate doesn't appear to be available everywhere, the certificate is likely still in the process of being deployed throughout the Fastly network. It can take anywhere from 20 minutes to an hour for certificates to fully deploy. Be sure to allow up to an hour in this state before contacting support@fastly.com for assistance.

Pointing DNS to serve HTTPS traffic

To serve secure traffic via HTTPS once the certificate is deployed, follow these steps.

1. Ensure that the domains you've added via the TLS domains interface have been added to a properly configured Fastly service.

2. Configure your DNS records to point traffic at the newly created certificate's IP addresses. If you used the HTTP challenge method to verify domain ownership, you're already pointing traffic at the certificate. All DNS details (CNAME, A records, and optionally AAAA records) can be found by clicking **See DNS details** to view the TLS configuration associated with the domain.

   For an apex domain (e.g., `example.com`), you'll need to create an A record with your DNS provider. For subdomains and wildcard domains (e.g, `www.example.com` or `*.example.com`), you'll need to create a relevant CNAME record.

Your domains and certificates can be set to use one or more TLS configurations. For more information, refer to the details on managing DNS and TLS configurations.

> **WARNING**
>
> If you point your DNS away from Fastly after the initial setup, we will be unable to terminate TLS on your behalf and we will also be unable to renew your certificate, which will expire after 90 days.

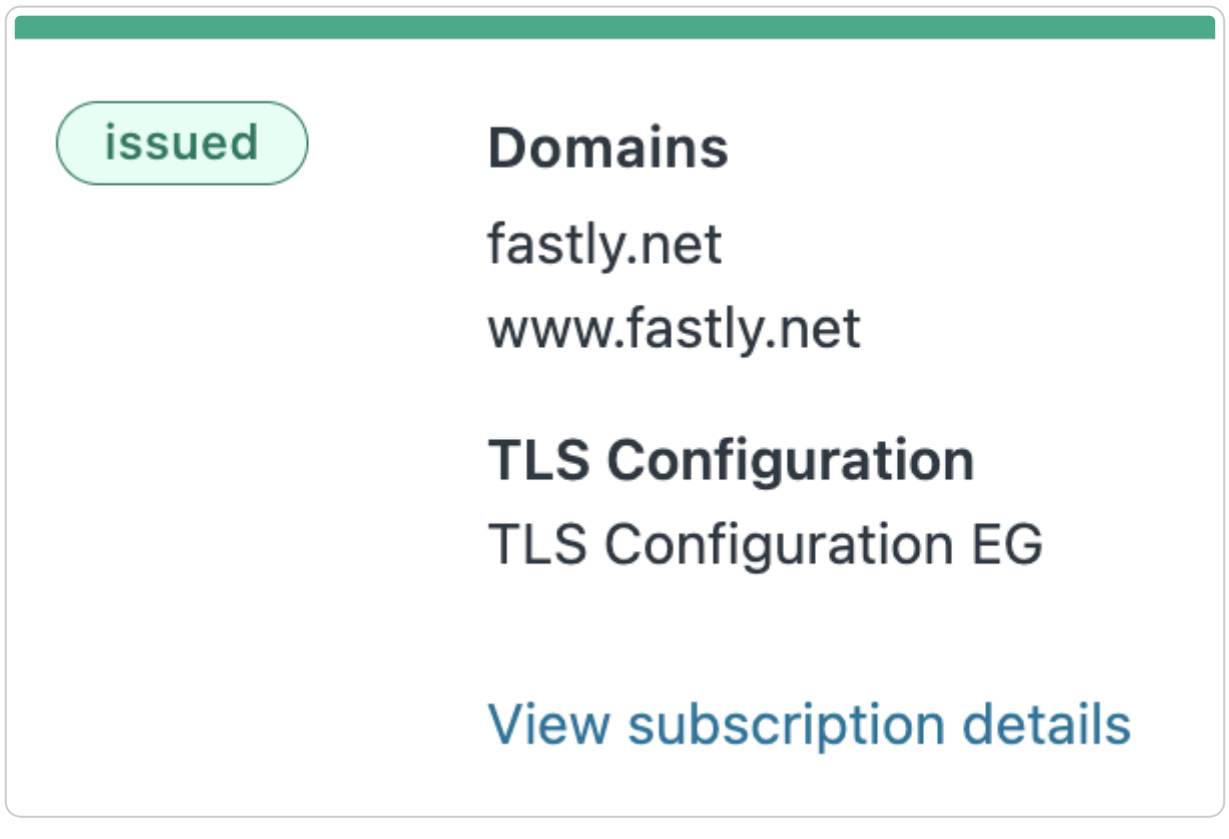
Managing TLS subscriptions

You can use the TLS subscriptions page to add or remove domains on the subscription or change the common name. To manage your TLS subscriptions, follow these steps.

1. Log in to the Fastly web interface and click the **Secure** link. The TLS domains page appears, displaying any domains for which you have TLS activated or for which TLS can be activated.

2. Click the **TLS subscriptions** tab. The TLS subscriptions page appears, displaying your subscriptions and their state.

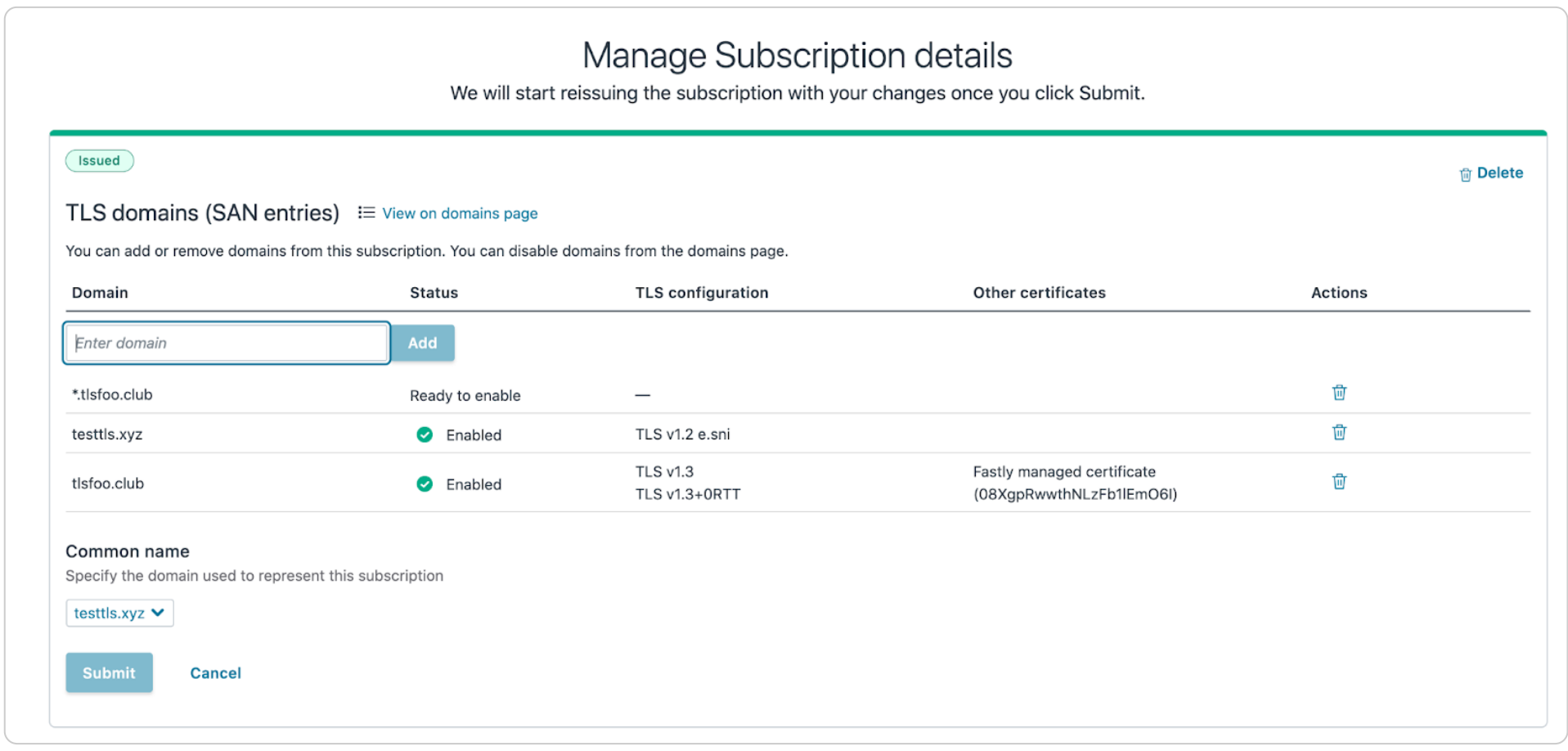
3. Click the **View subscription details** link for the subscription you want to make changes to. The Subscription details page appears.



4. Click the **Manage Subscription** button. The Manage Subscription details page appears.

> **NOTE**
>
> You can only manage a subscription in `Pending`, `Issued`, and `Renewing` states.

5. From the Manage Subscription details page, you can do the following:



- **Add new domains:** In the **Domain** field, enter one or more apex domains (e.g., `example.com`), subdomains (e.g., `www.example.com` or `api.example.com`), or a wildcard domain (e.g., `*.example.com`) and click the **Add** button. Separate multiple domains with a comma.
- **Remove existing domains:** Click the trash can icon in the **Actions** column on the same line as the domain you want to delete. Follow the instructions in the confirmation window to complete the deletion.
- **Change the subscription common name:** From the **Common name** menu, select the domain used to represent this subscription.

6. After making any changes to the subscription, click **Submit**. A message appears asking to confirm your changes.
7. Click **Confirm changes** to submit your changes. To return to the previous screen, click **No, review changes**.

> **NOTE**

If you added a new domain, you must validate domain ownership after confirming the change. See Domain validation for TLS certificates.

Deactivating TLS and deleting a TLS domain

Once a domain has TLS activated, you have the option to deactivate TLS via the **Deactivate TLS** button listed on each domain card on the TLS domains page. If a domain has multiple certificates, you can elect to deactivate a specific certificate by clicking **Add/Edit Activations** and clicking the **Deactivate** button next to any active configurations. If all certificates are deactivated, Fastly will no longer serve TLS traffic on the selected domain and it will become disabled. Fastly will attempt to renew a certificate for a disabled domain. To prevent this renewal process, delete the associated subscription after you disable it. Fastly will not renew certificates for deleted subscriptions.

Certificate management and renewals

Fastly currently works with two certification authorities, each with separate verification and renewal time frames that Fastly follows when managing your certificates:

- **Let's Encrypt renewals.** Let's Encrypt issues certificates that are valid for 90 days. Fastly will attempt to re-verify your domain and renew your certificate after 60 days. However, if DNS records no longer point at Fastly or if a CAA record blocks Let's Encrypt, the certificate will lapse at the end of the 90-day period.
- **GlobalSign renewals.** GlobalSign issues certificates that are valid for 365 days. Fastly will attempt to re-verify your domain and renew your certificate after 335 days. However, if DNS records no longer point at Fastly, or if a CAA record blocks GlobalSign, the certificate will lapse at the end of the 365-day period. Certificates provided by GlobalSign are subject to the terms of GlobalSign's Subscriber Agreement, which can be found at https://www.globalsign.com/repository.

Fastly checks on Let's Encrypt renewals 30 days before certificates are due to expire and GlobalSign renewals 90 to 120 days before certificates are due to expire. If a DNS check indicates that a renewal is failing, Fastly will automatically email all account users with TLS management permissions, notifying them of the upcoming expiration. If the renewal continues to fail, Fastly will continue to email users on the account on a schedule up until the expiry date.

In addition, you must verify domain ownership as part of the management process. If you have the correct DNS records for verifying domain ownership and there is no blocking CAA record, but you are still receiving renewal failure emails, contact support@fastly.com for assistance.

Setting up free TLS

Last updated: 2021-09-30

https://docs.fastly.com/en/guides/setting-up-free-tls

Customers can use our free TLS option to add TLS to a website or application using a shared Fastly domain (e.g., `your-name.global.ssl.fastly.net`).

TIP

Free TLS uses a shared domain name and does not support use with your own domain name (`www.example.com`). If you want to use your own domain, you can upgrade to a paid account and use Fastly TLS to add up to five domains for free on dedicated managed certificates.

Before you begin

Before you begin setting up free TLS, understand the following:

- Free TLS uses a shared domain name and does not support use with your own domain name (`www.example.com`). Customers typically use the free TLS hostname in links directly to assets (e.g., linking to `https://example.global.ssl.fastly.net/example.jpg`) or for testing purposes.
- You cannot DNS alias your own domain to the shared domain. If you do, a TLS name mismatch warning will appear in the browser.
- When using free TLS, all traffic is routed through Fastly's entire global network.

If you want to use your own domain or have the ability to route traffic through specific POPs, use another TLS service option.

Setting up free TLS for the first time

Follow the steps below to set up free TLS:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Create domain** button. The domain creation controls appear.

5. Fill out the domain creation fields as follows:
   - In the **Domain name** field, enter `<name>.global.ssl.fastly.net`, replacing `<name>` with a single word that claims the domain you're creating. You can't use a dot-separated name (e.g., `www.example.org.global.ssl.fastly.net`) because TLS certificates don't support nesting. If the name you choose has already been claimed, you will need to pick a different one.
   - In the **Comment** field, enter a human-readable name for the domain. This name appears in the Fastly web interface.
6. Click the **Create** button to save the domain. The new domain appears in the list of domains.
7. Click the **Activate** button to deploy your configuration changes.

Once you've set up free TLS, you'll be able to access your host domain via the `https://<name>.global.ssl.fastly.net/` URL. You won't need to add CNAME records to use the shared domain certificate.

Support for HTTP/2, IPv6, and TLS 1.2

Your `<name>.global.ssl.fastly.net` domain name currently supports the HTTP/1.x protocols and IPv4 network addresses on Fastly's free shared domain TLS wildcard certificate. TLS 1.0, 1.1, and 1.2 are all supported.

To test HTTP/2, you can use `<name>.freetls.fastly.net`, which is automatically made available for all Fastly free TLS services. For example, if you used `example.global.ssl.fastly.net` during setup, Fastly automatically created `example.freetls.fastly.net` with support for HTTP/2 and HTTP/1.1, as well as support for IPv6 and IPv4 network addresses. Names ending in `.freetls.fastly.net` require TLS 1.2.

> **NOTE**
>
> As noted in the previous section, you can't use a dot-separated name (e.g., `www.example.org.freetls.fastly.net`) because TLS certificates don't support nesting. If you experience problems testing your domain name with `freetls.fastly.net`, verify that `<name>` in `<name>.freetls.fastly.net` is a single word that doesn't contain dots.

TLS termination

Last updated: 2018-10-03

https://docs.fastly.com/en/guides/tls-termination

Identifying TLS terminated requests

To maintain optimal caching performance, Fastly uses a TLS terminator separate from the caching engine. This means, however, that the caching engine doesn't know that it was originally a TLS request. As a result, we set the `Fastly-SSL` header when fetching the content from your servers.

Because we set this header, you can check for its presence on your backend by doing something like:

```
if (req.http.Fastly-SSL) {
  set resp.http.X-Is-SSL = "yes";
}
```

and that should tell you if the request was a TLS request or not.

When using WordPress

If you're using Fastly TLS services with WordPress, you'll want to add a check for the `HTTP_FASTLY_SSL` header so that WordPress can build URLs to your CSS or JS assets correctly. Do this by placing a check in your `wp-config.php` file to override the SSL flag that is checked later:

```
if (!empty( $_SERVER['HTTP_FASTLY_SSL'])) {
  $_SERVER['HTTPS'] = 'on';
}
```

As usual, this must be placed anywhere before the `require_once` line with `wp-settings.php`.

Finding the original IP when using TLS termination

Because Fastly uses a TLS terminator, separate from the caching engine for performance, the engine overwrites the original IP address briefly due to the re-request to your origin servers once decrypted and causes anything that references the original IP to show up as 127.0.0.0/8 IPs. To find the original IP via VCL:

- use `req.http.Fastly-Client-IP` if you're using shielding
- use `client.ip` if you're not using shielding or if you're building an ACL

Fastly also sends along the client IP to the origin in a HTTP header, `Fastly-Client-IP`, which can be used by server software to adjust as needed.

---

§

These articles provide information about the Fastly Web Application Firewall (WAF 2020) security product.

https://docs.fastly.com/en/guides/security#_web-application-firewall

About the Fastly WAF dashboard (2020)

Last updated: 2020-12-07

https://docs.fastly.com/en/guides/about-the-fastly-waf-dashboard

IMPORTANT

As of June 30, 2021, the Fastly WAF (WAF 2020) offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

The Fastly WAF dashboard allows you to monitor the Fastly WAF deployed within your Fastly service. If you've been assigned the role of engineer or superuser, you can use the information in the Fastly WAF dashboard to determine whether or not the WAF is active, see how many requests the WAF is currently processing, review recent changes, and manage your WAF.

The Fastly WAF dashboard consists of the following pages:

- WAF summary
- Manage rules
- WAF audit history
- Settings

Accessing the Fastly WAF dashboard

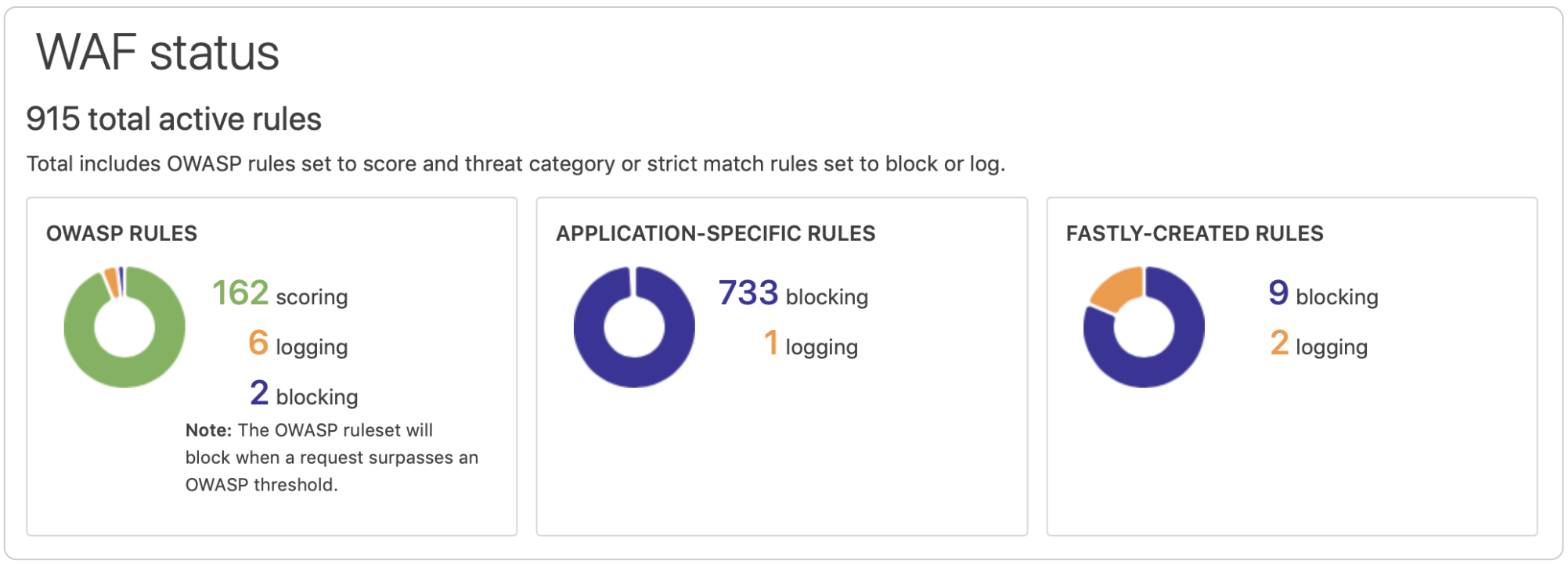
To access the Fastly WAF dashboard, log in to the Fastly web interface and click the WAF link.

> **NOTE**
>
> To access the Fastly WAF dashboard, you must sign up for a Fastly account and purchase the Fastly WAF. Contact our sales team to get started.

About the WAF summary page

The WAF summary page displays the status of your WAF.

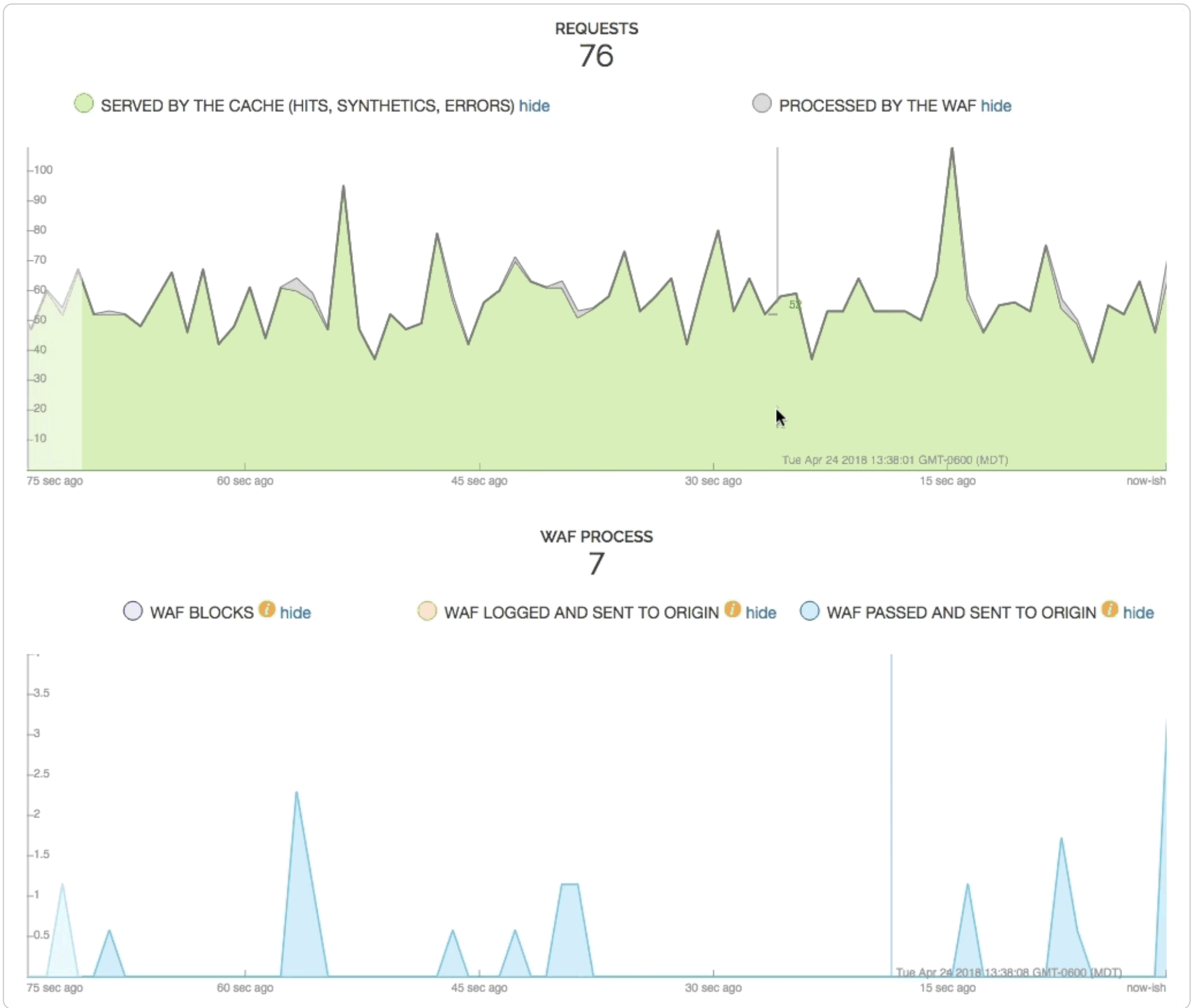
The WAF status section indicates whether the WAF is currently active. To be considered active, the WAF must not be disabled and must have at least one active rule's status set in either logging or blocking. You can see the total number of active rules. This number includes scoring, logging, and blocking rules added to your WAF. The charts show the number of scoring, logging, and blocking OWASP rules, application-specific rules, and Fastly-created rules. Sample charts are shown below.



The **Requests** graph displays how many requests are served from cache and how many requests are processed by the WAF. Of the requests that are processed by the WAF, the **WAF Process** graph displays how many requests were blocked by the WAF, logged by the WAF and sent to the origin server, and were passed (not blocked or logged) and sent to the origin server.

You can exclude certain data from the graphs by clicking the **hide** link next to a data label. Clicking this link will hide that value in the graph's display.

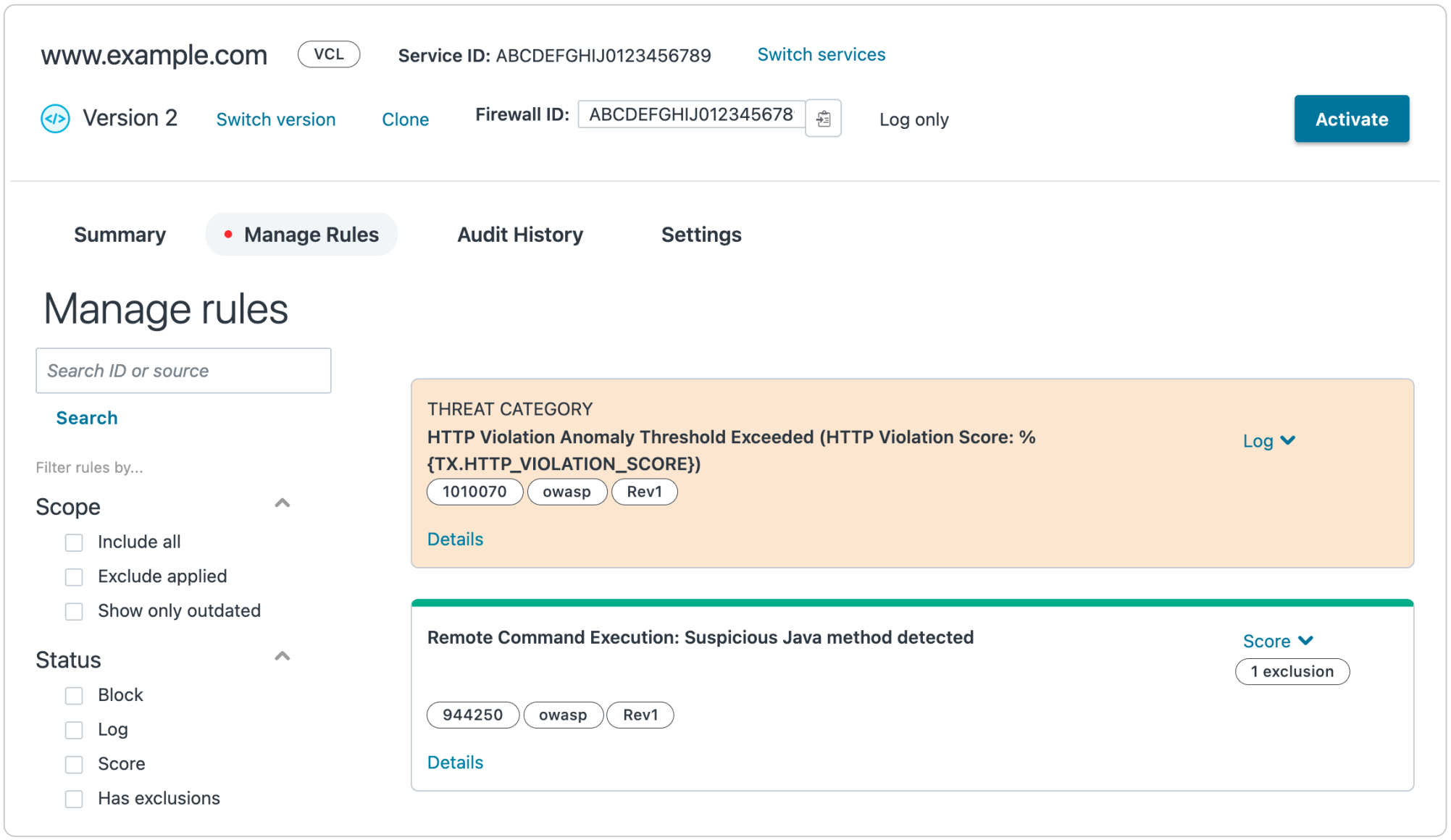
> **TIP**
>
> The Fastly WAF only executes on traffic sent to the origin server.



About the Manage rules page

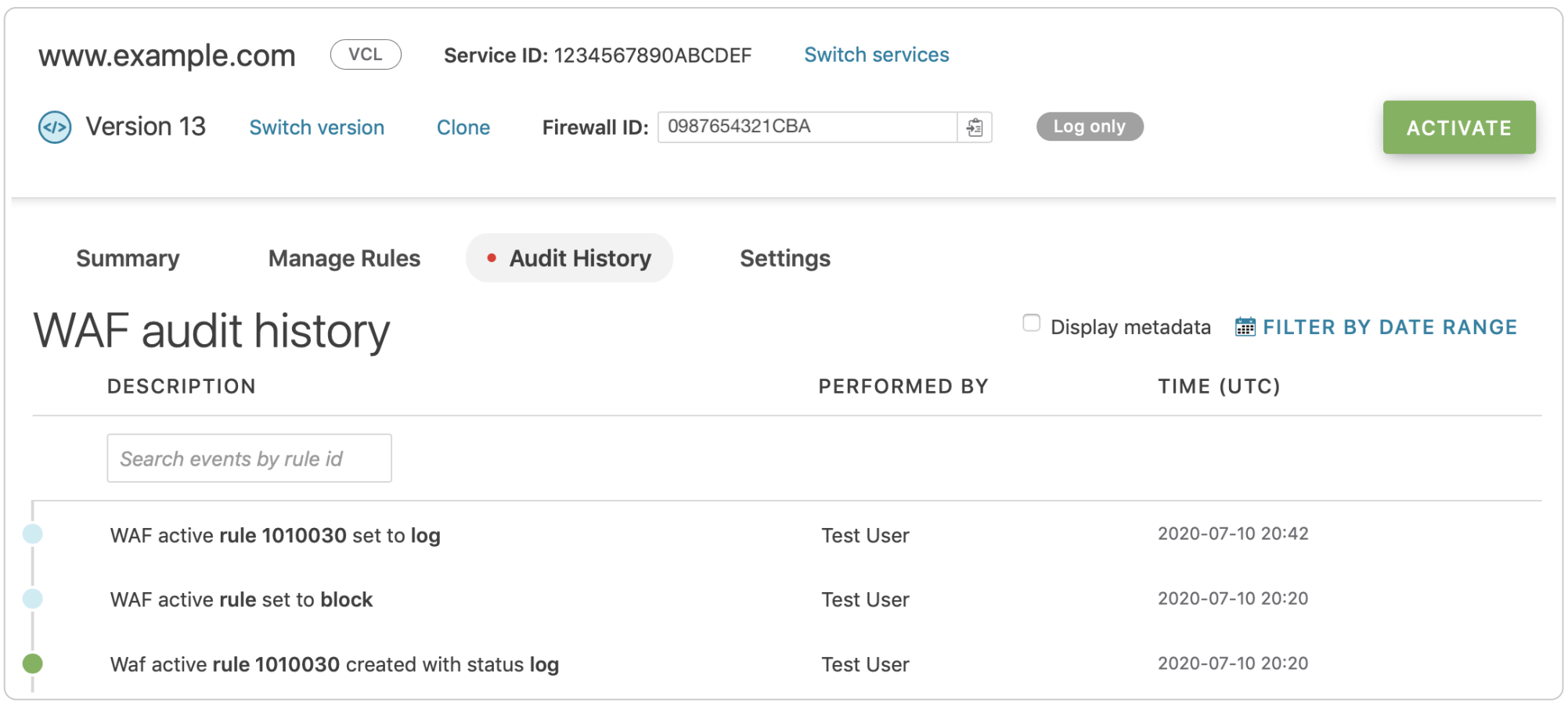
The Manage rules page allows you to manage the rules on your WAF. There you can:

- Add new rules to your WAF
- Change the status of rules already on your WAF, e.g. log ⇒ block
- View details about individual rules, i.e. the ModSec source code and the generated VCL
- View potential rules to add to your WAF
- Remove rules from your WAF



About the WAF audit history page

The WAF audit history page displays all changes made to your WAF. You can use this page to determine who made certain types of changes to the WAF and when the changes were made. The line items indicate when rules were set to log or block, when they were updated, and when firewall versions are cloned or deployed.



TIP

You can use the Fastly WAF active rules API endpoint to view the state of an individual rule.

Some entries contain information about the WAF's OWASP properties. To learn more about the OWASP properties, refer to the OWASP properties section.

```
waf.firewall_version.update critical_anomaly_score: 5
error_anomaly_score: 4
warning_anomaly_score: 3
notice_anomaly_score: 2
http_violation_score_threshold: 27
inbound_anomaly_score_threshold: 25
lfi_score_threshold: 23
php_injection_score_threshold: 999
rce_score_threshold: 999
rfi_score_threshold: 999
session_fixation_score_threshold: 999
sql_injection_score_threshold: 999
xss_score_threshold: 999
paranoia_level: 1
arg_name_length: 100
arg_length: 400
total_arg_length: 6400
max_file_size: 10000000
combined_file_sizes: 10000000
max_num_args: 255
high_risk_country_codes:
allowed_methods: GET HEAD POST OPTIONS PUT PATCH DELETE
restricted_headers: /proxy/ /lock-token/ /content-range/ /translate/ /if/
restricted_extensions: .asa/ .asax/ .ascx/ .axd/ .backup/ .bak/ .bat/ .cdx/ .cer/ .cfg/ .cmd/ .com/
.config/ .conf/ .cs/ .csproj/ .csr/ .dat/ .db/ .dbf/ .dll/ .dos/ .htr/ .htw/ .ida/ .idc/ .idq/ .inc/ .ini/ .key/ .licx/
.lnk/ .log/ .mdb/ .old/ .pass/ .pdb/ .pol/ .printer/ .pwd/ .resources/ .resx/ .sql/ .sys/ .vb/ .vbs/ .vbproj/
.vsdisco/ .webinfo/ .xsd/ .xsx/
allowed_request_content_type: application/x-www-form-urlencoded|multipart/form-
data|text/xml|application/xml|application/soap+xml|application/x-
amf|application/json|application/octet-stream|application/csp-report|application/xss-auditor-
report|text/plain
allowed_request_content_type_charset: utf-8|iso-8859-1|iso-8859-15|windows-1252
allowed_http_versions: HTTP/1.0 HTTP/1.1 HTTP/2
crs_validate_utf8_encoding: false
comment:
waf_id:
waf_firewall_version_id:
```

OWASP properties

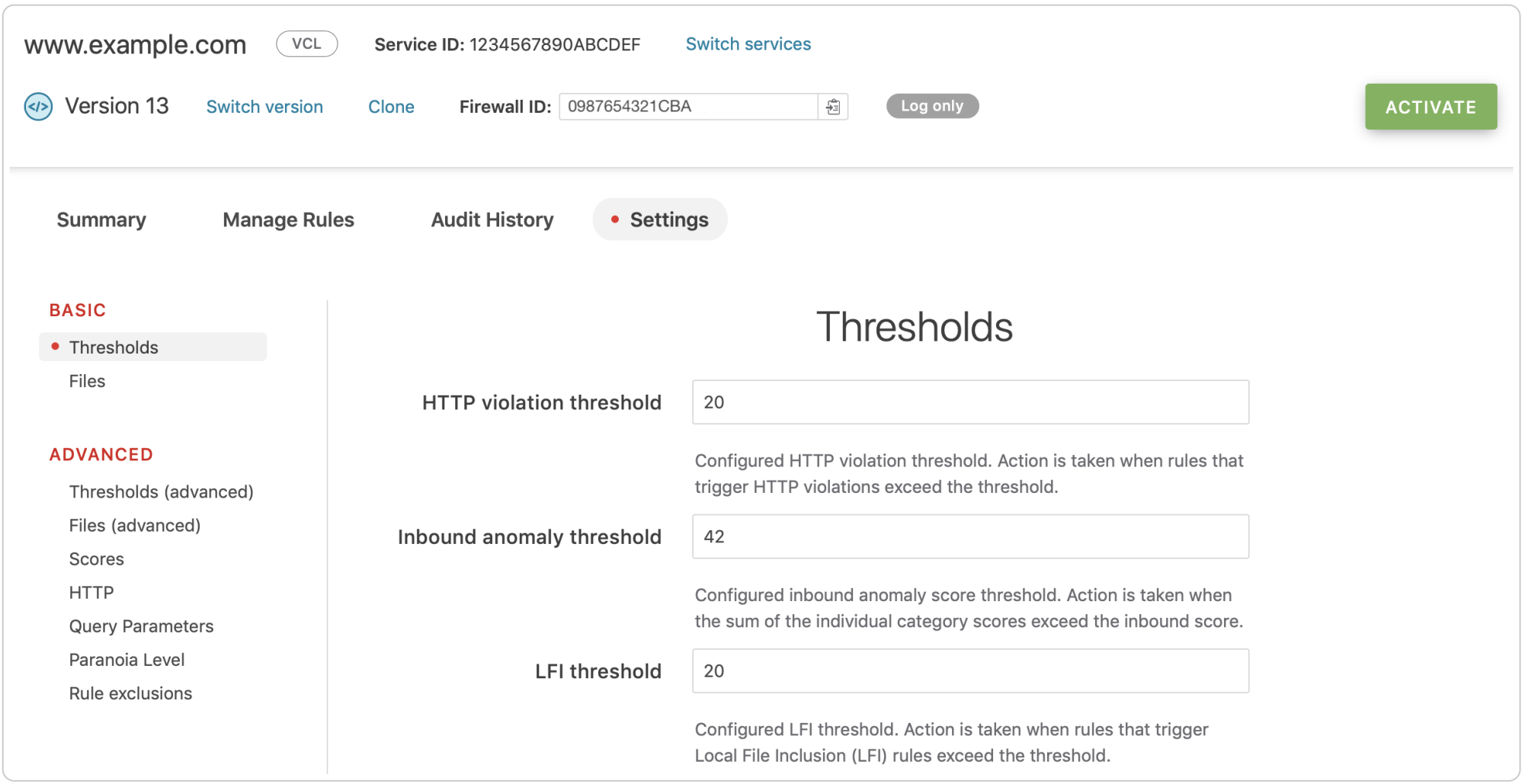
You may see OWASP properties referenced on the WAF audit history page. The table below contains a list of all available properties and their descriptions. The properties shown here reflect changes made by altering the settings in the firewall version.

OWASP property	Description
Allowed HTTP versions	HTTP versions that a client is allowed to use.
Allowed HTTP methods	HTTP methods that a client is allowed to use.
Allowed client content types	HTTP Content-Types that a client is allowed to use.
Maximum length for parameter names	Maximum length of any parameter names passed in the query string and request body.
Maximum length for parameter values	Maximum length of any parameter values passed in the query string and request body.
Combined file sizes	Total size of MIME bodies in the request.
Critical anomaly score	Configured critical anomaly score. Rules using the critical severity will increment scores using this value.

OWASP property	Description
Validate UTF8 encoding	Validates the client request as UTF-8 prior to the execution of WAF rules.
Error anomaly score	Configured error anomaly score. Rules using the error severity will increment scores using this value.
High risk countries	Block clients from high risk countries based on their IP address.
HTTP violation threshold	Configured HTTP violation threshold. Action is taken when rules that trigger HTTP violations exceed the threshold.
Inbound anomaly threshold	Configured inbound anomaly threshold. Action is taken when the sum of the individual category scores exceed the threshold.
LFI threshold	Configured LFI threshold. Action is taken when rules that trigger Local File Inclusion (LFI) rules exceed the threshold.
Maximum file size (bytes)	Maximum size of any MIME body in the request.
Maximum argument count	Maximum number of parameters in the query string and request body.
Notice anomaly score	Configured notice anomaly score. Rules using the notice severity will increment scores using this value.
Paranoia level	The paranoia level setting can be set from 1 through 4 and determines the number of rules to include by default. Higher levels indicate higher levels of security but potentially a larger number of false positives.
PHP injection threshold	Configured PHP injection score threshold. Action is taken when rules that trigger PHP related violations exceed the threshold.
RCE threshold	Configured RCE injection score threshold. Action is taken when rules that trigger Remote Code Execution (RCE) violations exceed the threshold.
Restricted extensions	Control on restricted file extensions in the client request.
Restricted headers	Control on restricted HTTP headers in the client request.
RFI threshold	Configured RFI violation threshold. Action is taken when rules that trigger Remote File Inclusion (RFI) violations exceed the threshold.
Session fixation threshold	Configured Session Fixation violation threshold. Action is taken when rules that trigger Session Fixation violations exceed the threshold.
SQLi threshold	Configured SQLi threshold. Action is taken when rules that trigger SQL Injection (SQLi) violations exceed the threshold.
Total parameter length	Maximum length of all parameters passed in the query string and request body.
Warning anomaly score	Configured warning anomaly score. Rules using the warning severity will increment scores using this value.
XSS threshold	Configured XSS threshold. Action is taken when rules that trigger Cross-Site Scripting (XSS) violations exceed the threshold.

About the Settings page

The Settings page allows you to adjust various settings for your WAF. To understand the behavior of thresholds and scores, see Managing rules.



About the Fastly WAF rule management interface (2020)

Last updated: 2020-12-07

https://docs.fastly.com/en/guides/about-the-fastly-waf-rule-management-interface

IMPORTANT

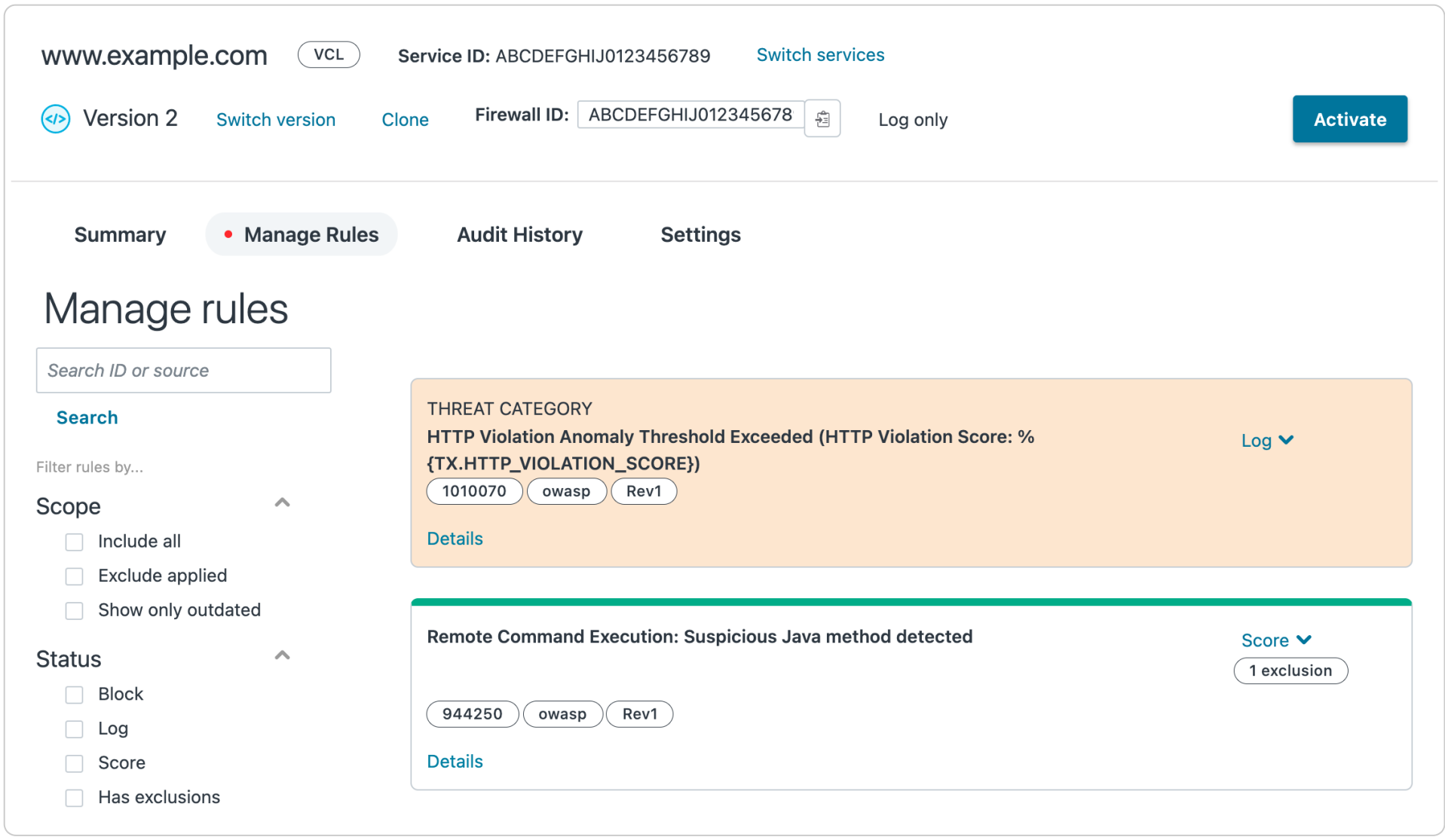
As of June 30, 2021, the Fastly WAF (WAF 2020) offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

The Fastly WAF rule management interface provides visibility and management for rules enabled on a WAF associated with a Fastly service. If you've been assigned the role of engineer or superuser, you can use the rule management interface to inspect the details of WAF rules, search and filter by rule ID or category, manage thresholds and scores, change rule modes, and deploy changes into production.

Accessing the Fastly WAF rule management interface

You can access the Fastly WAF rule management interface from the WAF dashboard. To access the Fastly WAF rule management interface, follow the steps below:

1. Log in to the Fastly web interface. The Home page appears displaying a list of all services associated with your account.
2. Find your Fastly service in the list and then click the **WAF** link. The WAF summary page appears.
3. Click the **Manage Rules** link. The Manage rules page appears.



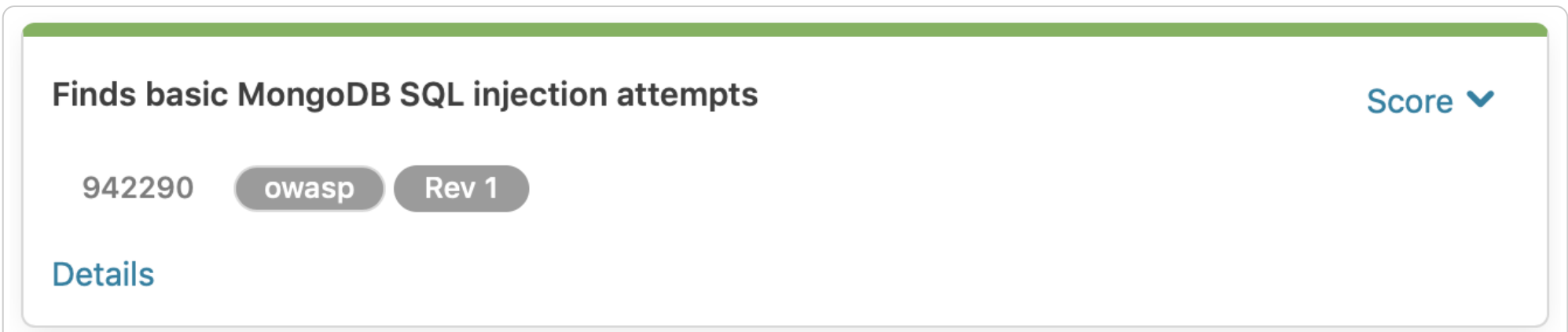
About rules

The Fastly WAF rule management interface displays the rules currently enabled on the WAF associated with the selected Fastly service. If you haven't enabled rules or you don't see any rules on your WAF, contact support@fastly.com.

There are three types of rules: scoring rules, threshold rules, and application-specific rules. The revision indicator is used to indicate whether or not a rule has been revised. A revised rule may provide additional protection over and above the earlier revision.

Scoring rules

Scoring rules increment a score based on anomalies detected in the incoming HTTP request. An example scoring rule is shown below.



Fastly sets default anomaly scores for four categories of rules. When an HTTP request trips a rule in a category, that numeric value is added to the total score for that category. The sum of all the rules' scores is the anomaly score for the HTTP request. For example, if an HTTP request trips a critical rule and an error rule, and the critical anomaly score is set to 6 and the error anomaly score is set to 5, the score for the HTTP request would be 11.

Scoring rules have two possible modes:

- ***Scoring:** This mode means a rule is active and looking for threats. Rules in this mode will increment scores and log their result to your currently configured logging provider based on the setting for the corresponding threshold rule. For more information on threshold rules, see the section on threshold rules.
- **Disabled:** This mode means a rule is inactive. Threats detected by this rule will not count towards your total anomaly score and their result will not be logged.

You can click the **Details** link to see the format of a rule in the Apache ModSecurity format as well as the corresponding generated VCL.

IMPORTANT

The generated VCL shows a general transformation from the Apache ModSecurity language to VCL. The VCL shown here does not show how a rule increments your anomaly score based on your configured values.

Threshold rules

Threshold rules cover a specific category of attack against your web application or API. These rules check the total score (as calculated by scoring rules) against the value configured for the appropriate threshold. An example threshold rule is shown below.

The threshold rule has a category name, revision indicator, tag, rule ID, mode selector, and Details link.

Threshold rules perform an action and either log or block and log client HTTP requests. These rules take action when a score exceeds a given threshold value. The corresponding threshold value for each category is configured on the Settings page.

Lowering thresholds increases the sensitivity of your WAF. Raising thresholds reduces the sensitivity of your WAF across the various threshold categories.

The Fastly WAF includes the following threshold rules organized into categories. Each rule has a corresponding threshold value that controls its sensitivity.

ID	Threshold Name	Rule Action Condition	Corresponding Threshold Value	Action Choice
1010090	Inbound Anomaly Score	Action taken when the inbound anomaly score exceeds the configured inbound anomaly threshold	Inbound anomaly threshold	Log or Block & Log
1010080	Session Fixation	Action taken when the session fixation score exceeds the configured session fixation threshold	Session fixation threshold	Log or Block & Log
1010070	HTTP Violation	Action taken when the HTTP violation score exceeds the configured HTTP violation threshold	HTTP violation threshold	Log or Block & Log
1010060	PHP Injection	Action taken when the PHP injection score exceeds the configured PHP injection threshold	PHP injection threshold	Log or Block & Log
1010050	Remote Command Execution (RCE)	Action taken when the RCE anomaly score exceeds the configured RCE threshold	RCE threshold	Log or Block & Log
1010040	Local File Inclusion (LFI)	Action taken when the LFI score exceeds the configured LFI threshold	LFI threshold	Log or Block & Log
1010030	Remote File Inclusion (RFI)	Action taken when the RFI score exceeds the configured RFI threshold	RFI threshold	Log or Block & Log

ID	Threshold Name	Rule Action Condition	Corresponding Threshold Value	Action Choice
1010020	Cross-site Scripting (XSS)	Action taken when the XSS score exceeds the configured XSS threshold	XSS threshold	Log or Block & Log
1010010	SQL Injection	Action taken when the SQL injection score exceeds the configured SQL injection threshold	SQL injection threshold	Log or Block & Log

IMPORTANT

Disabling a threshold rule removes an entire category of protection from your WAF. Use caution when disabling threshold rules.

Application-specific rules

Application-specific rules look at incoming HTTP requests to find signatures designed to take advantage of specific vulnerabilities within the context of a specific library, framework, or component. They take effect immediately. Application-specific rules have three possible modes:

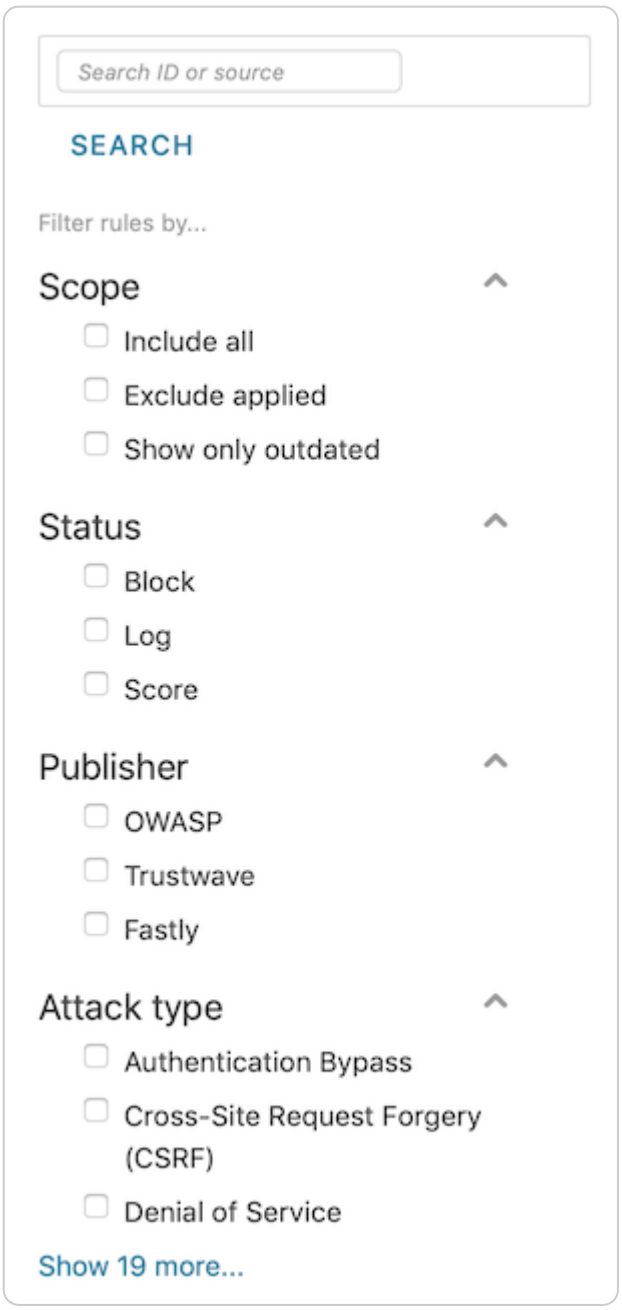
- **Logging Mode:** This mode means a rule is active and looking for threats. Rules in this mode will log their result on an exact match. The result is logged to your currently configured logging provider.
- **Blocking Mode:** This mode means a rule is active and looking for threats. Rules in this mode block the HTTP request and prevent it from going to the origin. Rules in blocking mode also log the result to your currently configured logging provider.
- **Disabled:** This mode means a rule is inactive. HTTP requests that match a rule will flow directly to your origin.

Searching for rules

The rule search box allows you to search for a specific rule using the rule ID or keyword. The result is shown in the rule view.

Keyword search allows you to search for rules that contain a specific query string in the rule source. For example, you can search for rules that contain keywords such as `java`, `php`, `loic`, or `ddos`. Keyword search compares your query string against the rule's mod_security source.

You can control the scope of the search through the use of the Include all and Exclude applied filters. Selecting **Include all** expands the search to the entire rule library as well as the rules currently active on your WAF. Selecting **Exclude applied** searches only the rules in the library that are not currently active on your WAF.



Filtering rules by category

The category filters allow you to view the different types of rules currently configured on your WAF. Filters can be combined.

- **Status filter:** Allows you to filter by rule mode for log, block, or disabled.
- **Publisher filter:** Allows you to filter by rule publisher. Currently supported publishers include OWASP, Trustwave, and Fastly.

  > **TIP**
  >
  > Use the Publisher filter and select OWASP to show all rules in scoring mode.

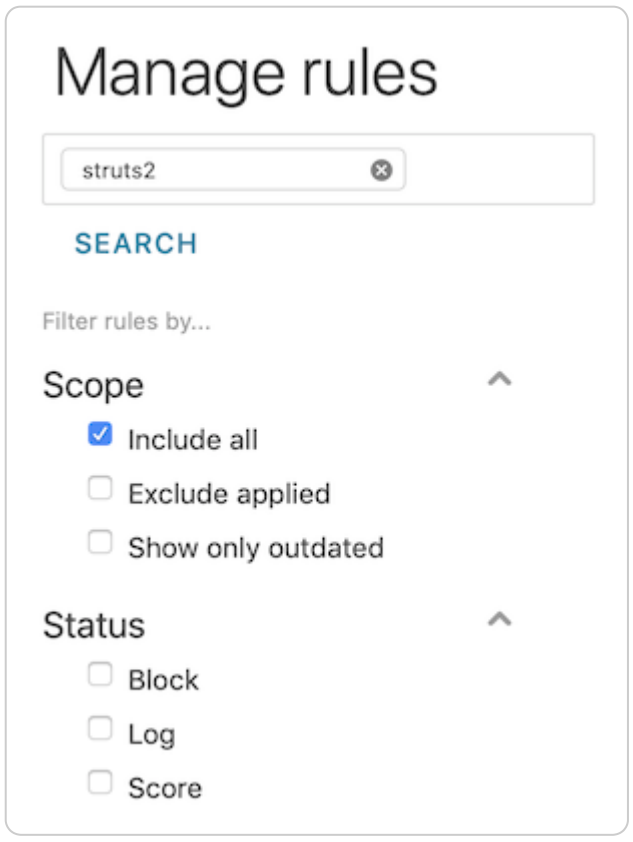
- **Attack type filter:** Allows you to show the rules enabled on your WAF that offer protection for specific categories of attack.

Adding new rules to your WAF

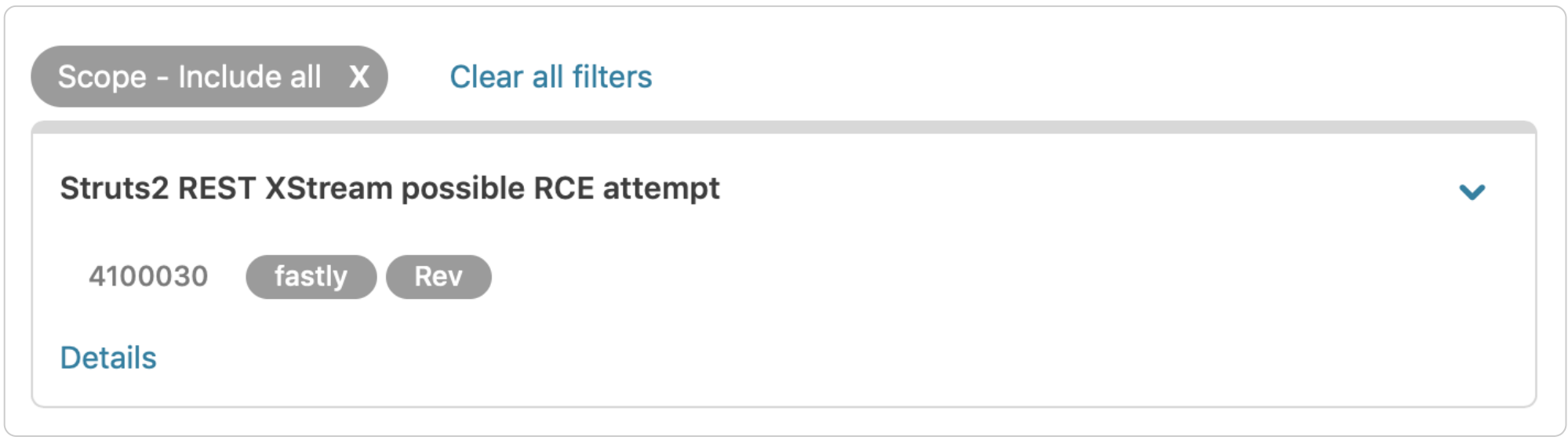
You may want to add new rules to your WAF based on changing attack patterns and risks to your application. You can add new rules by using the scope filters displayed under the search box to browse, search, and select new rules. Selecting the **Include all** filter will display all rules in the current rule library in the rule view.

> **TIP**
>
> By selecting **Include all**, you will see considerably more rules than you have currently active. Pagination allows you to browse through the rule base.

Once the rules are available to browse, you can use the search box to search for a specific rule ID or keyword. For example, if you're interested in protecting against Apache Struts2 vulnerabilities published in CVE-2017-9805, you might search for `struts2` or `RCE`.



The rule view will appear as follows.



You can view the rule source by selecting the **Details** link. This provides you with more information about how the rule executes in VCL.

You can enable a rule by selecting **Options** and changing the mode to either **Log** or **Block**.

> **TIP**
>
> The available options for OWASP rules are Scoring and Disabled. To enable a new OWASP rule, select **Scoring**.

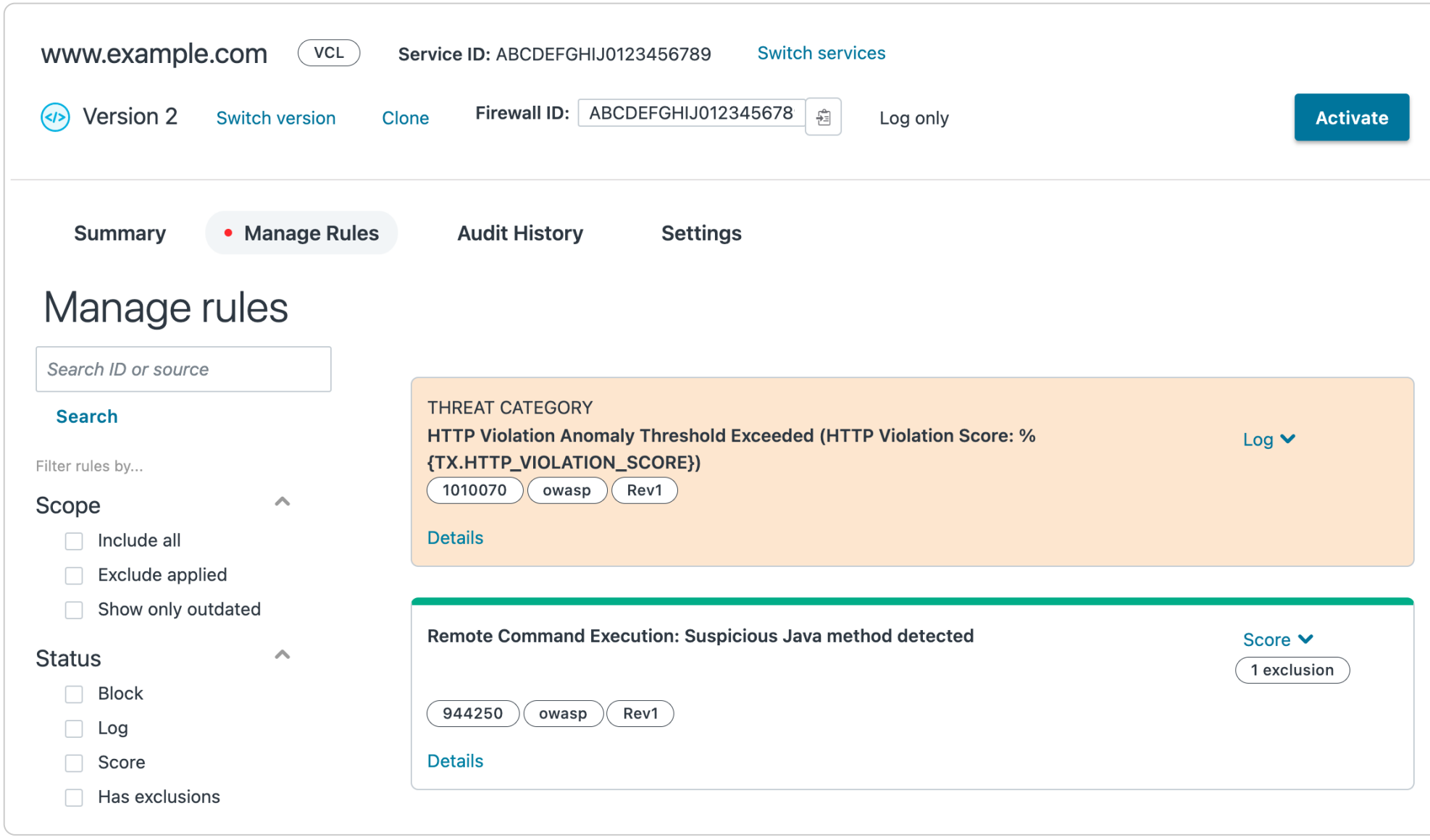
Verifying a rule is active

After you select the rule mode, the rule appears at the top of the rule view. To verify that your rule was added, you should deselect the **Include all** filter and use the **Status**, **Publisher**, or **Attack type** filters and confirm that the rule has been added to your WAF.

After you verify that the rule has been added, follow the instructions in the activating changes section to activate your changes.

Activating changes

The Fastly WAF rule management interface allows you to change rule modes and threshold values that change the way rules behave in the face of potential web-oriented attacks. After changing rule modes, you can click the **Activate** button to put these changes into production.



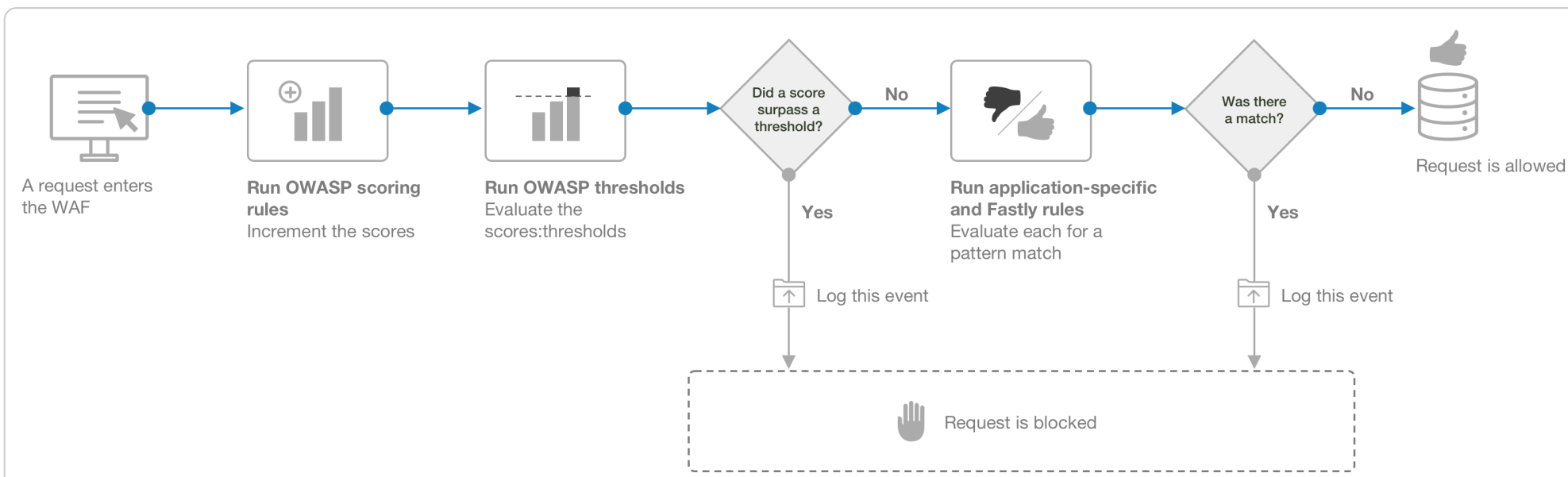
After clicking **Activate**, you'll see a confirmation message asking if you want to activate your changes. Click **Yes** to continue.

> **IMPORTANT**
>
> Changes to your WAF are applied only after you click the Activate button.

WAF policy execution

When the Fastly WAF processes an inbound request, scoring rules execute first followed by threshold rules. Application-specific and Fastly rules are executed last.

If the accumulated score exceeds the configured threshold, the threshold rules take action. For more information on the threshold categories and action condition, please see the table in the threshold rules section.



Creating a custom WAF error page (2020)

Last updated: 2020-07-13

https://docs.fastly.com/en/guides/creating-custom-waf-error-page

> **IMPORTANT**

> As of June 30, 2021, the Fastly WAF (WAF 2020) offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

You can create a custom HTML error page that will be presented to users who are blocked by the Fastly WAF response object. The attributes of the response object include the HTTP status code, the HTTP response text, the content type, and the returned content.

For this example, we'll:

- use a dynamic VCL snippet to create a custom `req.http.x-request-id` HTTP header,
- use that header as a global variable to store the transaction ID of the request so that it can be used in both the request and WAF logs, and
- create a synthetic response to present the user with an HTML response.

The error page will display the transaction ID, something that might be useful if, for example, the user decides to contact your support team.

Creating a dynamic VCL Snippet

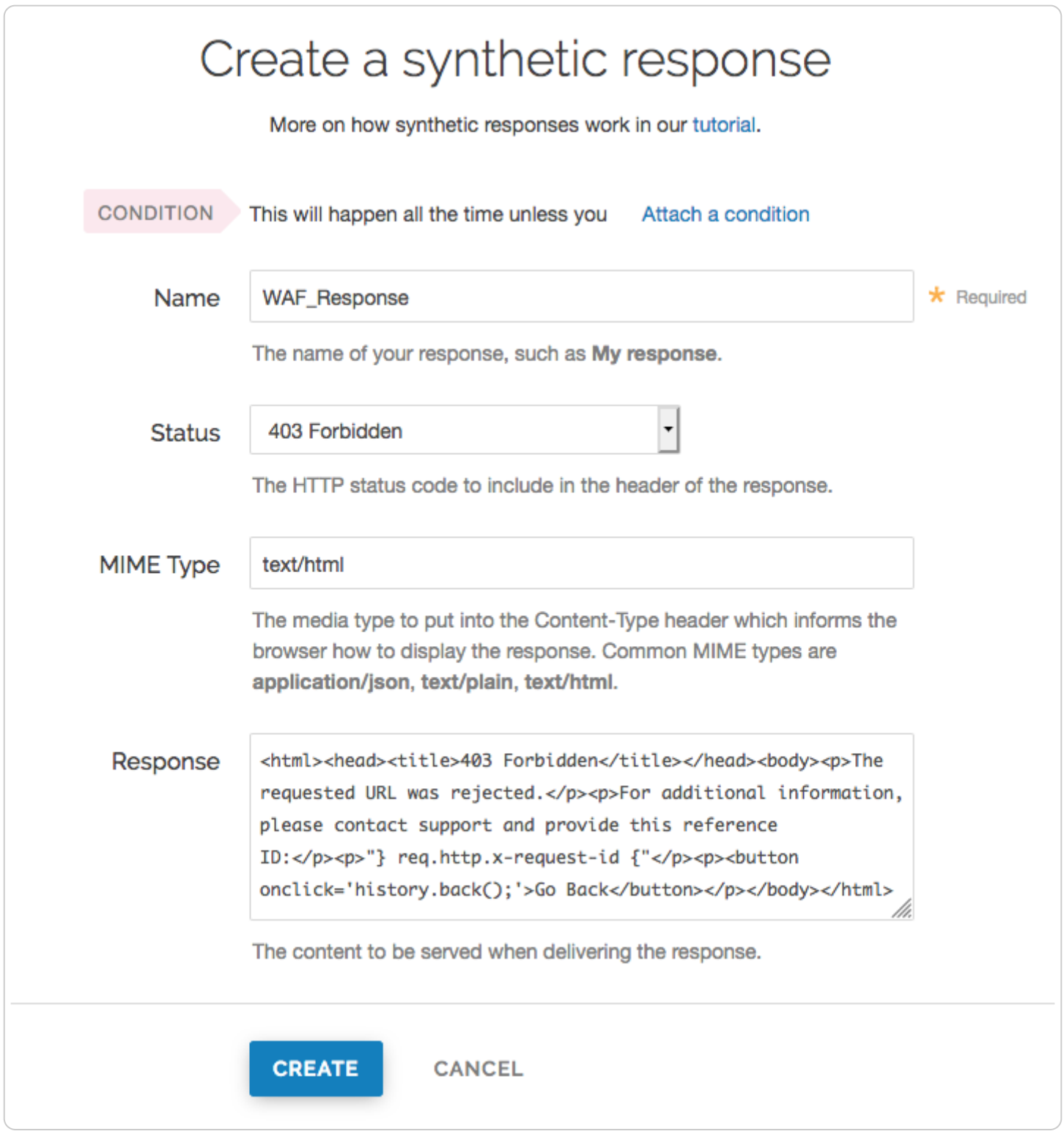
To create a dynamic VCL Snippet for the transaction ID, make the following API call in a terminal application:

```
$ curl -X POST -s https://api.fastly.com/service/<Service ID>/version/<Editable Version Number>/snippet -H "Fastly-Key:FASTLY_API_TOKEN" -H 'Content-Type: application/x-www-form-urlencoded' --data $'name=my_dynamic_snippet_name&type=recv&dynamic=1&content=if (!req.http.x-request-id) {\n set req.http.x-request-id = digest.hash_sha256(now randomstr(64) req.http.host req.url req.http.Fastly-Client-IP server.identity);\n}'
```

Creating a synthetic response

To create a synthetic response for the custom HTML error page, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Set up advanced response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, enter `WAF_Response`.
   - From the **Status** menu, select `403 Forbidden`.
   - In the **MIME Type** field, specify the Content-Type of the response (e.g., `text/html`).
   - In the **Response** field, enter the following HTML. This response will display the value of `req.http.x-request-id`.

```
<html>
  <head>
    <title>403 Forbidden</title>
  </head>
  <body>
    <p>The requested URL was rejected.</p>
    <p>For additional information, please contact support and provide this reference ID:</p>
    <p>"} req.http.x-request-id {"</p>
    <p><button onclick='history.back();'>Go Back</button></p>
  </body>
</html>
```

7. Click the **Create** button. Your new response appears in the list of responses.
8. Click the **Activate** button to deploy your configuration changes.

Additional notes

- You can change the composition of the transaction ID, but care should be taken to minimize the probability that multiple requests within a specified window of time (e.g., per day) have the same transaction ID value.
- A VCL Snippet was used to simplify the example presented and is not explicitly required for a custom WAF error page. As an alternative, you can use custom VCL to create the transaction ID.

- It's useful to include the transaction ID in the request and WAF logging formats to allow multiple messages generated for the same request to be correlated.

Fastly WAF logging (2020)

Last updated: 2019-07-13

https://docs.fastly.com/en/guides/fastly-waf-logging

IMPORTANT

As of June 30, 2021, the Fastly WAF (WAF 2020) offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

Fastly provides a number of WAF-specific logging variables to help you monitor and identify potentially malicious traffic. These variables provide specific details about the actions Fastly WAF performed on a request.

Setting up a logging endpoint

To begin monitoring requests for potential malicious activity, set up remote logging so you can log WAF variables. You can use an existing logging endpoint or add a new endpoint specially for Fastly WAF. You'll use the information provided in the logs to monitor WAF events.

OWASP rules

A single request can trigger multiple OWASP rules. By default, logging occurs in `vcl_deliver` or `vcl_log`. When logs are captured in `vcl_deliver` or `vcl_log`, it will show the last WAF rule triggered and the cumulative anomaly score.

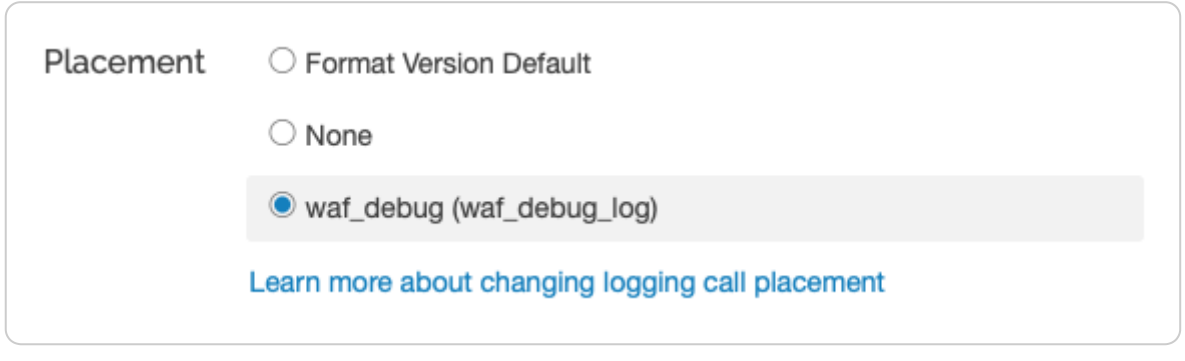
waf_debug_log

The `waf_debug_log` subroutine allows logging of each OWASP rule triggered for a single request. You can use the web interface or the API to update the logging placement parameter to `waf_debug`.

Using the web interface

Follow these instructions to set a logging endpoint's placement parameter to `waf_debug`:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears.
5. Click the name of a logging endpoint to edit it. The Edit this endpoint page appears.
6. Click the **Advanced options** link.
7. In the Placement section, select the **waf_debug (waf_debug_log)** setting.



8. Click the **Update** button.

9. Click the **Activate** button to deploy your configuration changes.

Using the API

You can also update the logging placement parameter to `waf_debug` by running the following curl command in a terminal application:

```
$ curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/<your Fastly service ID>/version/<version_id>/logging/<logging_integration>/<logging_name>' --data-binary '{"placement":"waf_debug"}'
```

- `waf_debug_log` accepts the logging format via the web interface only
- `waf_debug_log` is called in `vcl_miss` and `vcl_pass`. The logging format can include request headers and WAF variables. Response headers will result in an error message.
- `<logging_integration>` can be found listed in our remote logging API.

We recommended creating a `request_id` header to track a single request through multiple OWASP rules:

```
set req.http.x-request-id = digest.hash_sha256(now randomstr(64) req.http.host req.url req.http.Fastly-Client-IP server.identity);
```

Using WAF-specific variables

Fastly provides a number of WAF-specific logging variables to help you monitor and identify potentially malicious traffic. These variables provide specific details about the actions Fastly WAF performed on a request:

- **Whether or not Fastly WAF inspected a request.** Fastly WAF only inspects traffic that is forwarded to your origin server (e.g., MISS or PASS requests for content that is not already cached).
- **Whether or not a rule matched the request.** When Fastly WAF inspects a request, it checks to see if the request matches any of the rules in your rule set.
- **The severity of the rule that matched.** If the request matches a rule, the log indicates the severity of the rule.
- **The action taken, if any.** If the request matches a rule or OWASP threshold, the log indicates whether Fastly WAF simply logged the request or blocked it.

You can use the following variables to examine Fastly WAF log events.

Variable	Description
`waf.executed`	A response header indicating if WAF was executed or not. Appears as `1` (true) when executed or `0` (false) when not.
`waf.blocked`	Set to true when the request matches and the specific rule or OWASP threshold is configured to block. Will appear in log files as `1` (true) when blocked or `0` (false) when not.
`waf.logged`	Set to true when the request matches and the specific rule or OWASP threshold is configured to log. In active (blocking) mode, set to `true` when `waf.blocked` is also `true`. Will show up in the logs as `1` (true) or `0` (false).
`waf.failures`	A request exits the WAF rule set due to a failure to evaluate. Will show up in the logs as `1` (true) or `0` (false).
`waf.logdata`	Why (specifically) this rule matched. Includes the portion of the request that triggered the match, so it may look different depending on the rule.
`waf.message`	A message describing the generic condition this rule matched. For example, `SLR: Arbitrary File Upload in Wordpress Gravity Forms plugin`.
`waf.rule_id`	The rule ID for this rule.

Variable	Description
`waf.severity`	The severity of the rule. `0` is the highest severity and `7` is the lowest severity. `99` indicates that severity is not applicable (e.g., the request did not match any rules).
`waf.anomaly_score`	Cumulative score returned if request triggers OWASP rules. See OWASP category score variables.
`waf.passed`	Indicates if the request doesn't match any rules in the WAF rule set. Will show up in the logs as `1` (true) or `0` (false). `waf.passed` is readable in `vcl_deliver` and `vcl_log`. It is not readable in `waf_debug_log`. The value is determined after the request has gone through the WAF rule set.

OWASP category score variables

As a request goes through the OWASP rules, it can trigger different rule IDs from different attack categories. OWASP category score variables track which categories were triggered and the scoring that contributed to the cumulative score. They can be used to get a sense of minimum, average, and maximum values for a specific attack category and set thresholds individually. When in active (block) mode, if a request exceeds the category threshold, it will be blocked.

- `waf.sql_injection_score`
- `waf.rfi_score`
- `waf.lfi_score`
- `waf.rce_score`
- `waf.php_injection_score`
- `waf.session_fixation_score`
- `waf.http_violation_score`
- `waf.xss_score`

Managing rules on the Fastly WAF (2020)

Last updated: 2020-07-13

https://docs.fastly.com/en/guides/managing-rules-on-the-fastly-waf

IMPORTANT

As of June 30, 2021, the Fastly WAF (WAF 2020) offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

The Fastly WAF uses firewall versions. With firewall versions, you can roll back changes you've made to your WAF by deploying a previous firewall version. This allows you to quickly redeploy a known good firewall version if changes deployed to your WAF start causing false positives.

You can also update rules individually. As Fastly publishes new rules or updates old rules, you can choose to apply those updates on a rule by rule basis at a time that's convenient for you.

Cloning a firewall version

Before you can manage any active rules on your WAF, you'll need to have an editable firewall version. Much like working with service versions, if the current version you're viewing is locked, you can clone the firewall version to edit it.

Using the Fastly WAF dashboard

To clone the firewall version in the Fastly WAF dashboard, click the **clone** link next to the version number. This should display a new firewall version number with a blue icon beside it.

Using the API

Before you can clone a firewall version, you'll need to see what your current active version is and check to see if you already have a firewall version that you're editing. If you've been editing your firewall frequently, you might have a long list of firewall versions, so you can limit the list of returns. First, you can see the full list of firewall versions for your WAF with the following request:

```
$ curl -sg -H "Fastly-Key:$token" \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions?page[size]=5"
```

Each version returned in that list will contain an `active` field and a `locked` field. If `active` is true, that firewall version is the current deployed version. If `locked` is true, that firewall version is no longer editable. For an uneditable firewall version, you will need to clone it to make any edits:

```
$ curl -X PUT -s \
  -H "Fastly-Key:$token" \
  -H "content-type: application/vnd.api+json" \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/clone"
```

Activating a firewall version

Using the Fastly WAF dashboard

After you've made changes to your cloned firewall version, click the **Activate** button to deploy the new firewall version.

You may find you need to roll back to a previous firewall version. Use the version selector to choose the previous version, then click the **Activate** button to reactivate that version.

Using the API

After you've made changes to your firewall version, you can activate it with the following request:

```
$ curl -X PUT -s \
     -H "Fastly-Key:$token" \
     -H "Content-Type: application/vnd.api+json" \
     -H "Accept: application/vnd.api+json" \
     "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/activate"
```

As you make edits to your firewall, you may find that you need to quickly roll back a change that you've made. A previously used firewall version can be re-activated. When re-activated, that firewall version's associated active rules and rule revisions will be deployed.

Firewall version deployment happens asynchronously. To check the status of your firewall version, make a GET request for the version:

```
$ curl -X PUT -s \
     -H "Fastly-Key:$token" \
     -H "Content-Type: application/vnd.api+json" \
     -H "Accept: application/vnd.api+json" \
     "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/activate"
```

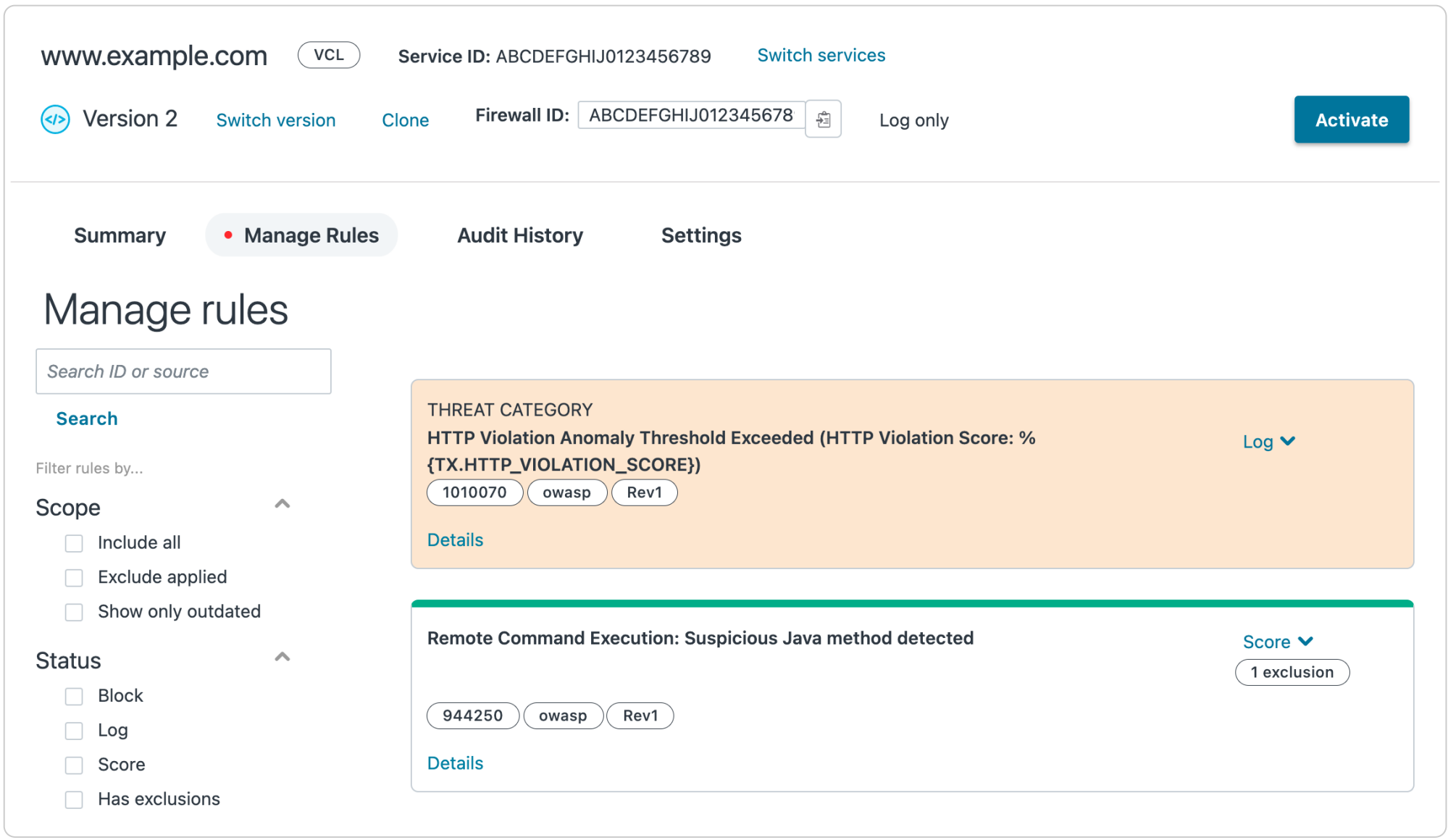
When the firewall version has been fully deployed, the `last_deployment_status` will be `completed` and the `active` status will be `true`. If the firewall version fails to deploy, the `last_deployment_status` will be `failed` and the error will be captured in the `error` attribute.

Finding a rule

Using the Fastly WAF dashboard

Before you can add new rules to your WAF, you have to find the rules you want to add. Once you've logged into the Fastly WAF dashboard summary page for a single WAF, navigate to the Manage Rules page. From here, you can use the sidebar to filter the rules already on your WAF by Status, Publisher, or Attack type. If you'd like to see rules that are not already on your WAF, expand

the **Scope** menu and select **Include all**. Aside from Status, any filters you'd previously applied would continue to filter for the rules not already on your WAF.



Using the API

You can also get a list of rules by making calls directly to our API using an API token of a user with at least Engineer permissions.

Given that this will return a list of several thousand rules, using filters can help you find the rules that you're interested in. For example, you might want to see all rules for a Drupal application:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules?filter[waf_tags][name]=application-drupal"
```

You can get a list of all tags for this filter:

```
$ curl -s -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/tags"
```

You can also filter to exclude rules already added to your firewall:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules?filter[waf_firewall.id][not][match]=:FIREWALL_ID"
```

You might notice that this will again return several thousand rules. You can combine filters to return just the rules you're interested in:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules?filter[waf_firewall.id][not][match]=:FIREWALL_ID&filter[waf_tags][name]=application-drupal"
```

Fastly has three publishers of firewall rules: OWASP, Trustwave, and Fastly itself. You can filter the rule list by publisher:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules?filter[publisher]=owasp"
```

If you'd like to know more about how a rule would potentially block traffic, look at the Rule revision. A rule can have many revisions, which is how Fastly can update the definition of a rule without impacting your service until you're ready to update to the new revision. Rule revisions contain both the ModSec source and the generated VCL. To view a list of revisions for a rule, grab the rule ID and run a request to:

```
$ curl -s -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules/:RULE_ID/revisions"
```

To view the ModSec source and the generated VCL on a specific revision:

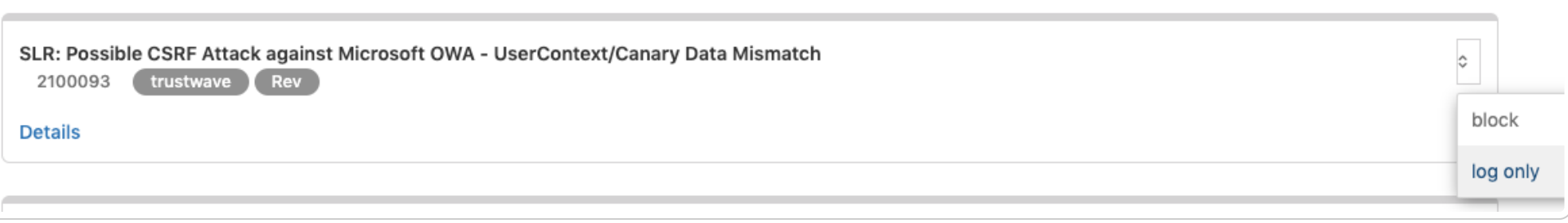
```
$ curl -s -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules/:RULE_ID/revisions/:REVISION_NUMBER?include=source,vcl"
```

Adding rules

Rules that have been added to your WAF are called *active rules*. This is the association between a rule, a particular revision of that rule, and a firewall version of your WAF. Before you can add any active rules to your WAF, you must have an editable firewall version.

Using the Fastly WAF dashboard

When you find a rule you would like to add to your WAF, use the menu on the right to select a status for the active rule. You can use the menu to set the status on any rules you'd like to add to your WAF. When you're done adding rules, click the **Activate** button to deploy the firewall version with all of your newly added rules.



Using the API

You can add an active rule to your WAF via ModSec rule ID. The active rule will be added with the most recent updated revision. If Fastly releases another rule revision in the future, you can update the active rule to the most recent version at your convenience. When you add active rules to your WAF, you can use the `status` attribute to set the active rule to `log` or `block`.

```
$ curl -X POST -s \
  -H "Content-Type: application/vnd.api+json" \
  -H "Fastly-Key:$token" \
  -d @- \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules" \
<<JSON
{
  "data": {
    "type": "waf_active_rule",
    "attributes": {
      "status": "log",
      "modsec_rule_id": MODSEC_RULE_ID
    }
  }
}
JSON
```

You can add active rules from a previous rule revision using the rule revision ID:

```
$ curl -X POST -s \
  -H "Content-Type: application/vnd.api+json" \
  -H "Fastly-Key:$token" \
  -d @- \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules" \
<<JSON
{
  "data": {
    "type": "waf_active_rule",
    "attributes": {
      "status": "log"
    },
    "relationships": {
      "waf_rule_revision": {
        "data": {
          "type": "waf_rule_revision",
          "id": "RULE_REVISION_ID"
        }
      }
    }
  }
}
JSON
```

If you'd like to add many rules at once, create a file to setup the JSON for your request body. You can set the status for each rule and you can use a mix of ModSec rule IDs and rule revision IDs:

```
{
  "data": [
    {
      "type": "waf_active_rule",
      "attributes": {
        "status": "block"
      },
      "relationships": {
        "waf_rule_revision": {
          "data": {
            "type": "waf_rule_revision",
            "id": "RULE_REVISION_ID"
          }
        }
      }
    },
    {
      "type": "waf_active_rule",
      "attributes": {
        "status": "block",
        "modsec_rule_id": MODSEC_RULE_ID,
        "revision": 1
      }
    },
    {
      "type": "waf_active_rule",
      "attributes": {
        "status": "block",
        "modsec_rule_id": MODSEC_RULE_ID
      }
    }
  ]
}
```

The request to add these rules to a firewall version is:

```
$ curl -X POST -s \
  -H "Content-Type: application/vnd.api+json; ext=bulk" \
  -H "Fastly-Key:$token" \
  -H "Accept: application/vnd.api+json; ext=bulk" \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules" \
  -d @/path/to/json-document.json
```

When you're done adding rules to your WAF, be sure to activate your firewall version to see those rules in production.

Removing a rule

If you're seeing an active rule generating false positives, you'll need to remove it from your WAF. You cannot keep the active rule on your WAF in a disabled state. Before you can remove any active rules from your WAF, you must have an editable firewall version.

Using the Fastly WAF dashboard

Clone the firewall version and find the rule you would like to remove. Click on the `Remove` link for that rule, then deploy the new firewall version. You can remove as many rules as needed before you deploy the firewall version.

Using the API

If you want to remove an active rule from your WAF, you can use the active rule ID to delete the rule from an editable firewall version.

```
$ curl -X DELETE -s \
  -H "Content-Type: application/vnd.api+json" \
  -H "Fastly-Key:$token" \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules/:ACTIVE_RULE_ID"
```

If you would like to remove several rules at once, you can make a single bulk request. First, save your rules to a JSON file. You can reference an active rule ID or use ModSec rule IDs to remove the rules from your WAF:

```
{
  "data": [
    {
      "type": "waf_active_rule",
      "attributes": {
        "modsec_rule_id": MODSEC_RULE_ID
      }
    },
    {
      "type": "waf_active_rule",
      "id": "ACTIVE_RULE_ID"
    }
  ]
}
```

You can then use this JSON file as the body on the delete request:

```
$ curl -X DELETE -s \
  -H "Accept: application/vnd.api+json; ext=bulk" \
  -H "Content-Type: application/vnd.api+json; ext=bulk" \
  -H "Fastly-Key:$token" \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules" \
  -d @/path/to/rules-to-delete.json
```

When you've removed any rules you no longer need, activate the firewall version to deploy your changes to production.

Updating a rule status to log or block

Active rules have a status field that can be updated to change your WAF behavior. You can change the status of an active rule from `log` ⇒ `block` or from `block` ⇒ `log`. You cannot change the status of a scoring rule. Before you can change the status of any active rules on your WAF, you must have an editable firewall version.

Using the Fastly WAF dashboard

On an editable firewall version, find a rule with a status you need to update. Use the menu to the right of the rule to choose the status you'd prefer. You can update the status on as many rules as needed. When you're done, activate the firewall version to deploy your changes to production.

Using the API

On an editable firewall version, you can update a single active rule status using the active rule ID:

```
$ curl -X PATCH -s \
  -H "Fastly-Key:$token" \
  -H "content-type: application/vnd.api+json" \
  "https://api.fastly.com.net/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules/:ACTIVE_RULE_ID" \
  -d @- \
<<JSON
  {
    "data": {
      "id": "ACTIVE_RULE_ID",
      "type": "waf_active_rule",
      "attributes": {
        "status": "block"
      }
    }
  }
JSON
```

You can update the status of multiple active rules on your WAF using the rule revision ID or the ModSec ID for each rule. To update multiple rules at once, start by saving the request body in a JSON file:

```
{
  "data": [
    {
      "type": "waf_active_rule",
      "attributes": {
        "status": "block",
        "modsec_rule_id": MODSEC_RULE_ID
      }
    },
    {
      "type": "waf_active_rule",
      "attributes": {
        "status": "block"
      },
      "relationships": {
        "waf_rule_revision": {
          "data": {
            "type": "waf_rule_revision",
            "id": "RULE_REVISION_ID"
          }
        }
      }
    }
  ]
}
```

Then, you can make a request to update the status of all rules listed in your JSON file:

```
$ curl -X POST -s \
  -H "Accept: application/vnd.api+json; ext=bulk" \
  -H "Content-Type: application/vnd.api+json; ext=bulk" \
  -H "Fastly-Key:$token" \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules" \
  -d @/path/to/update-rule-status.json
```

You can move all of the active rules on your firewall version to either `log` or `block` using the bulk update endpoint. Active rules in block status will block traffic that trigger those active rules on your WAF. To prevent false positives, you may choose to tune your WAF first.

```
$ curl -X PATCH -s \
  -H "Accept: application/vnd.api+json" \
  -H "Content-Type: application/vnd.api+json" \
  -H "Fastly-Key:$token" \
  -d @- \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules/bulk"
<<JSON
{
  "data": {
    "type": "waf_active_rule",
    "attributes": {
      "status": "block"
    }
  }
}
JSON
```

Viewing all revisions for a rule (API only)

When you add an active rule to your WAF, the default is to add the most recent rule revision for the referenced rule. You can add an active rule with a previous rule revision once you know the rule revision ID. To view all details and the revisions for a rule:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules/:RULE_ID?include=waf_rule_revisions"
```

In this return, the `state` will tell you if a rule is the `latest` version or if it's `outdated`. To see all rules with rule revisions updated since a specific date, start by getting the list of rules:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules?filter[waf_rule_revisions.created_at][gt]="YYYY-MM-DD""
```

You can further filter this list to only show the rules already on your WAF that have updates since a specific date:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules?filter[waf_rule_revisions.created_at][gt]="YYYY-MM-DD&filter[waf_firewall.id][not][match]=:FIREWALL_ID""
```

Once you have the list of rules, you can query each rule to get the latest rule revision:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules/:RULE_ID/revisions?filter[state]=latest"
```

To view the `VCL` or the `source` of a rule revision, use the rule revision number to make the following request:

```
$ curl -sg -H "Fastly-Key:$token" \
    "https://api.fastly.com/waf/rules/:RULE_ID/revisions/:REVISION_NUMBER?include=source,vcl"
```

Finding outdated rules (API only)

You can find any active rules on your WAF that have updated rule revisions:

```
$ curl -g \
  -H "Fastly-Key:$token" \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules?filter[outdated]=true"
```

Updating to the latest revision (API only)

If you have any outdated active rules that you would like to update to the latest revision, you can update the rules one at a time on an editable firewall version:

```
$ curl -X PATCH \
  -H "Accept: application/vnd.api+json" \
  -H "Content-Type: application/vnd.api+json" \
  -H "Fastly-Key:$token" \
  -d @- \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules/:ACTIVE_RULE_ID" \
<<JSON
{
  "data": {
    "id": "ACTIVE_RULE_ID",
    "type": "waf_active_rule",
    "attributes": {
      "revision": "latest"
    }
  }
}
JSON
```

When you're updating a single rule, you can also specify a specific rule revision number instead of `latest`.

You can use the bulk endpoint to update *all* of the outdated active rules on a firewall version:

```
$ curl -X PATCH -s \
  -H "Accept: application/vnd.api+json" \
  -H "Content-Type: application/vnd.api+json" \
  -H "Fastly-Key:$token" \
  -d @- \
  "https://api.fastly.com/waf/firewalls/:FIREWALL_ID/versions/:VERSION_NUMBER/rules/bulk"
<<JSON
{
  "data": {
    "type": "waf_active_rule",
    "attributes": {
      "revision": "latest"
    }
  }
}
JSON
```

You can only update to the `latest` revision when bulk updating all of the active rules on your firewall version. You cannot select a specific revision.

Understanding the types of rule statuses

Active rules have a status field that tells your VCL how to respond if a request triggers the rule. There are three possible statuses:

- `log`: Active rules set to `log` will report to your logging endpoint if a request triggered the rule. The request will still to proceed to your origin.
- `block`: A request that triggers an active rule set to `block` will be blocked. Fastly will respond back to the client with the response you defined when setting up your WAF. Fastly will also generate a log for the request with any information you have setup on your WAF logging endpoint.
- `score`: Some active rules only allow a status of `score`. These scoring rules are used together to determine if a request should be blocked.

Active rules that have a status of `log` or `block` are considered `strict` rules. If a request triggers these active rules, Fastly will follow the protocol of the status and either log or log and block the request.

Active rules that have a status of `score` are only used to build a total score at the end of their assessment. This score is then used to determine if Fastly should log or log and block the request. See more about this behavior in Understanding scoring active rule behavior.

Understanding scoring active rule behavior

Scoring rules assign a score and categorize requests. These rules do not block requests. When a request is run past your WAF, for each scoring rule that a request triggers, the corresponding score is added to a running total for the category, or categories, the rule belongs to.

The running total of each category is compared to the threshold defined for that category once all the rules are processed. Depending on the status of the threshold rule, your WAF will `log` or `log` and `block` the request if the running score exceeds the value set for the threshold rule. Lowering the values of your thresholds makes your WAF more likely to log or block requests, which might return false positives. Raising the values of the thresholds means that the request must potentially trigger more scoring rules before the request is blocked.

Tuning your WAF

Running live traffic through your WAF is the best way to tune it. As you add active rules and adjust thresholds and scores, you want to avoid false positives blocking legitimate traffic wherever possible.

When you first set up your WAF, one option is to add any active rules you're interested in using in `log` mode. After you've logged traffic for a couple of weeks, you can analyze how requests triggered active rules in order to determine if an active rule looks like it will block legitimate traffic. If the active rule is a threshold rule for scoring rules, you can tweak the threshold value for that active rule. If the active rule is a strict match rule, you can choose to remove the rule from your WAF.

When you check your logs and it looks like Fastly is ready to block malicious traffic and let legitimate traffic through to your origin, then change the status of all of your rules to `block`. At that point, Fastly will start to block any requests that match against what you've identified as malicious.

WAF Rule Exclusions (2020)

Last updated: 2021-09-27

https://docs.fastly.com/en/guides/waf-rule-exclusions

IMPORTANT

As of June 30, 2021, the Fastly WAF (WAF 2020) offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

The WAF Rule Exclusions feature provides the ability to define criteria for allowing requests to proceed to the origin in cases where they would otherwise be blocked by Fastly's Web Application Firewall (WAF) security product. Creating exclusions allows you to reduce the rate of false positives for requests rejected by the WAF's detection logic while still protecting against application-layer attacks. You can use this feature to exclude WAF rules on a per-request basis.

How the WAF Rule Exclusions feature works

You can use the web interface or the Fastly API to create a rule exclusion policy. Every rule exclusion policy has two parts:

- **Conditions:** The URLs, cookie names, and request parameters that won't be processed by the rules associated with the rule exclusion policy.
- **Rules:** One or more rules that will be excluded for the conditions you specified in the rule exclusion policy. You can associate up to 30 rules per rule exclusion policy.

After you specify conditions and associate rules with a rule exclusion policy, you can activate the changes in production.

When a request matches the conditions set in a rule exclusion policy, the associated rules won't be triggered for the request parameters, URLs, and cookie names you specified in the rule exclusion policy. The parts of the request that haven't been excluded will still be processed by the WAF.

When rule exclusions can be useful

The WAF Rule Exclusions feature might be useful for:

- excluding a specific request parameter from being processed by a specific rule
- excluding a specific URL from being processed by a specific rule

- excluding a specific cookie from being processed by a specific rule
- specifying a single set of conditions to exclude many rules
- combining multiple rule exclusion policies using AND or OR logic to exclude a rule

For example, you could set up a rule exclusion that would ignore a rule if it was triggered by the following conditions:

- Does not match the following paths: `/checkout/*`, `/selfhelp/help`
- Matches on cookie name: `ec_bn`, `touch_cookie`
- Matches on request parameter name: `AdditionalInfo`, `data`, `Comment`, `terms`, `notes`
- Matches on JSON parameter name: `links.sending.Example.LimlURL`

Working with WAF exclusions using the web interface

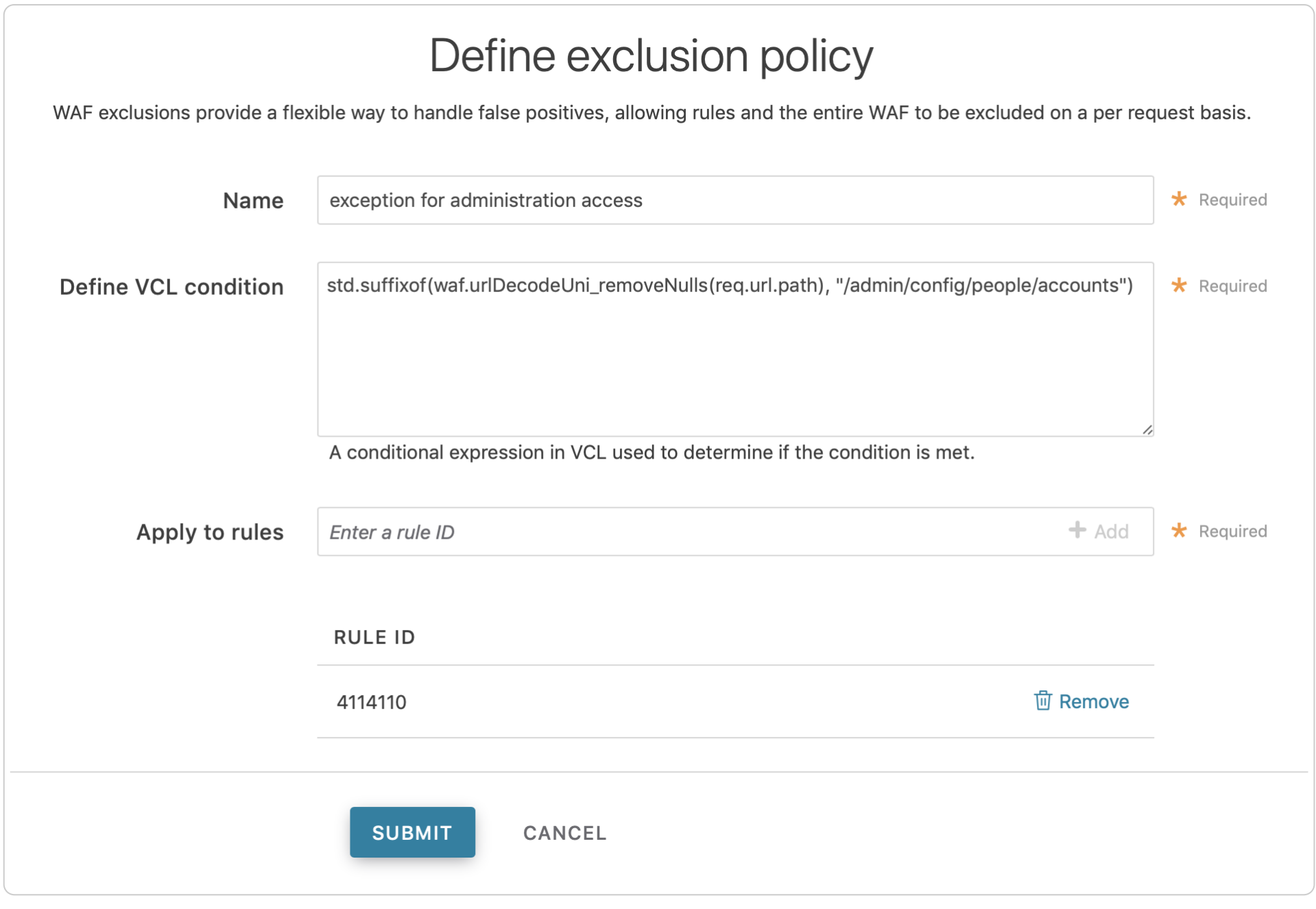
You can use the web interface to add, update, and remove rule exclusion policies.

> **NOTE**
>
> You can't use the web interface to configure variable rule exclusions. Those can be configured via the API.

Creating a rule exclusion policy

To create a rule exclusion policy, follow these instructions:

1. Log in to the Fastly web interface. The Home page appears displaying a list of all services associated with your account.
2. Find your Fastly service in the list and then click the **WAF** link. The WAF summary page appears.
3. Click the **Settings** link. The Settings page appears.
4. Click the **Rule exclusions** link. The Rule exclusions policies page appears.
5. Click the **Create New Exclusion** button. The Define exclusion policy page appears.

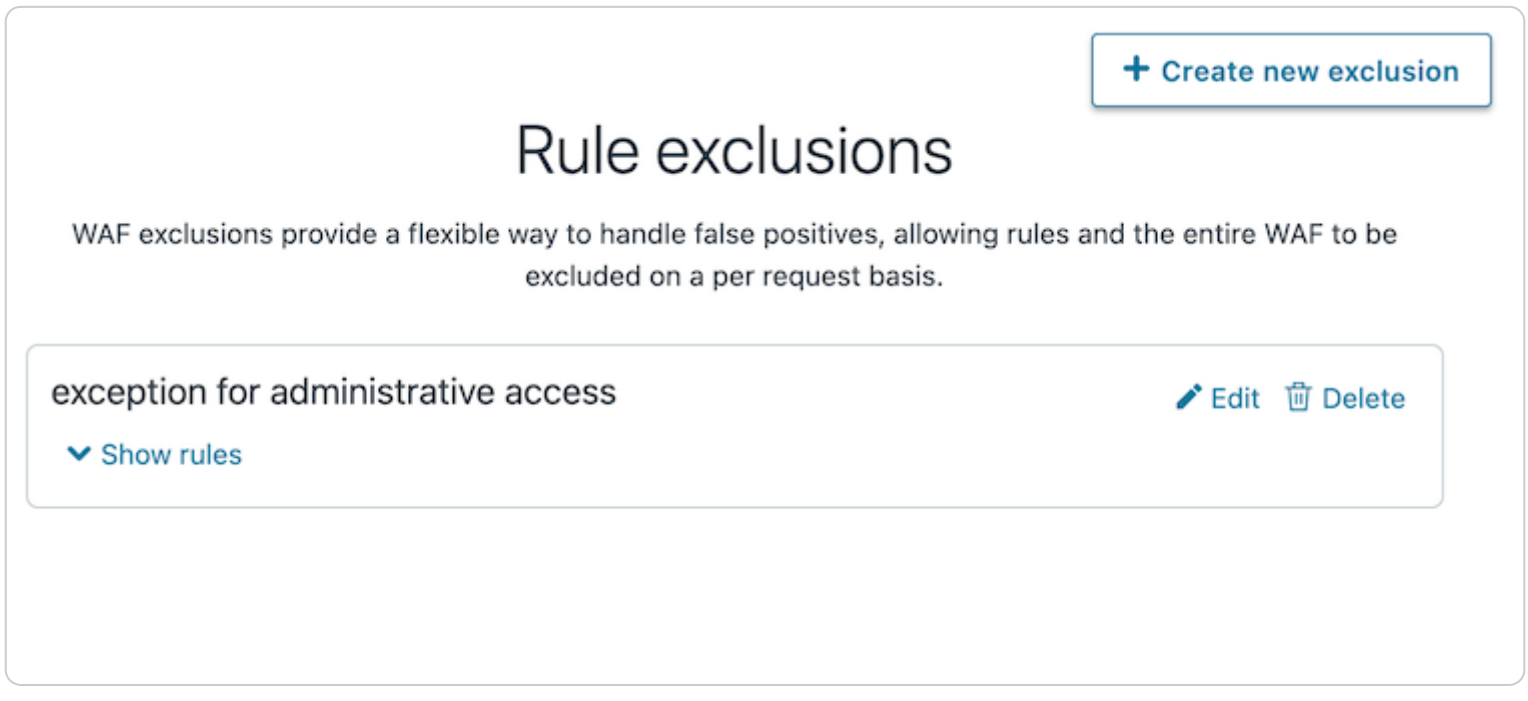


6. In the **Name** field, enter a human-readable name for the rule exclusion policy.
7. In the **Define VCL condition** field, enter the conditional expression in VCL that will be used to determine if the condition is met.
8. In the **Apply to rules** field, enter a WAF rule ID and click **Add**. The rules you add will be excluded for the conditions you specified in the Define VCL condition field.
9. Click the **Submit** button to save the rule exclusion policy.
10. Activate the changes to your WAF.

Updating a rule exclusion policy

To update an existing rule exclusion policy, follow these instructions:

1. Log in to the Fastly web interface. The Home page appears displaying a list of all services associated with your account.
2. Find your Fastly service in the list and then click the **WAF** link. The WAF summary page appears.
3. Click the **Settings** link. The Settings page appears.
4. Click the **Rule exclusions** link. The Rule exclusions policies page appears.

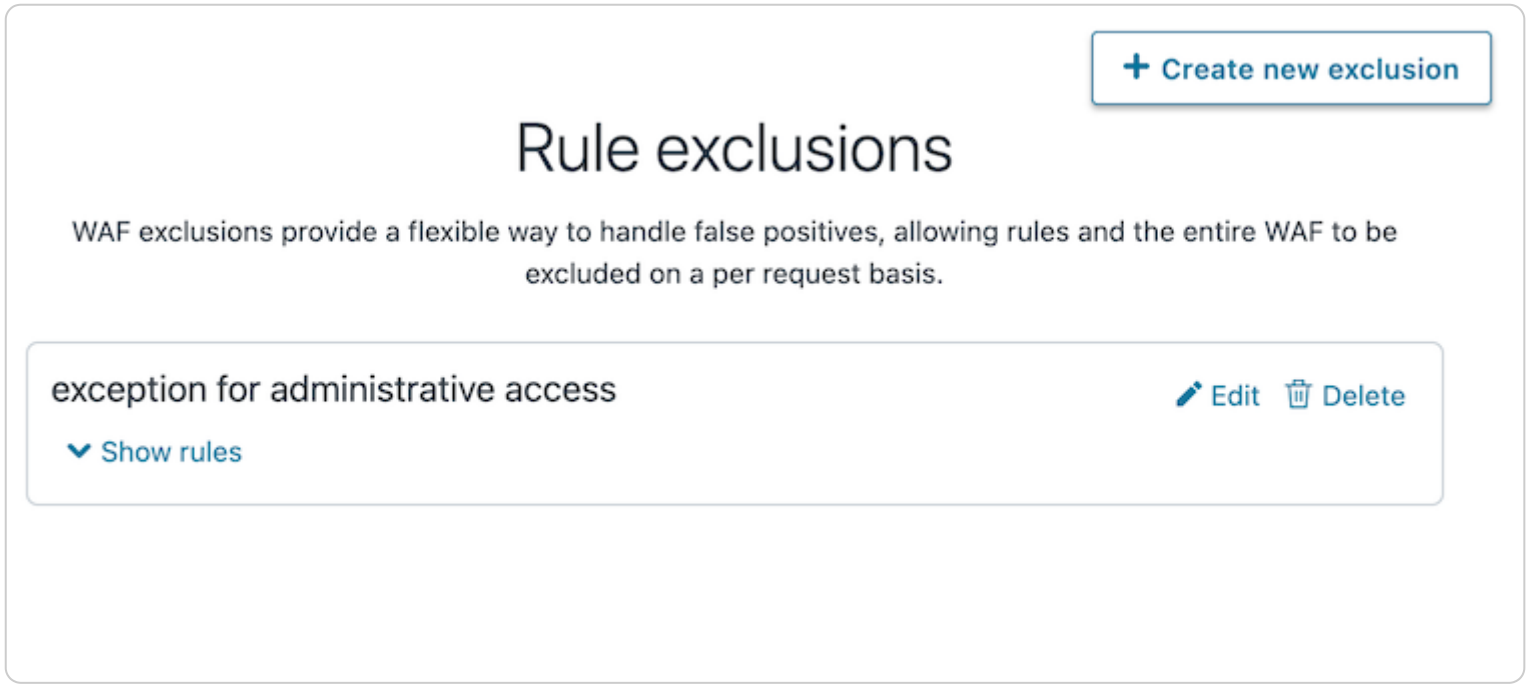


5. Find the rule exclusion policy you want to edit and then click the **Edit** link. The Define exclusion policy page appears.
6. Edit the conditions or rules as necessary.
7. Click the **Submit** button.
8. Activate the changes to your WAF.

Deleting a rule exclusion policy

To delete a rule exclusion policy, follow these instructions:

1. Log in to the Fastly web interface. The Home page appears displaying a list of all services associated with your account.
2. Find your Fastly service in the list and then click the **WAF** link. The WAF summary page appears.
3. Click the **Settings** link. The Settings page appears.
4. Click the **Rule exclusions** link. The Rule exclusions policies page appears.



5. Find the rule exclusion policy you want to edit and then click the **Delete** link.

6. Activate the changes to your WAF.

Creating a policy to exclude all rules

In certain circumstances, you may need to create an exclusion for all WAF rules. You can use a VCL condition to exclude all rules, as shown in the following example.

```
if (req.http.host != "www.example.com") {
    set req.http.rqpass = "1";
   }
```

Working with WAF exclusions using the API

You can use the Fastly API to add, view, update, and remove rule exclusion policies. For documentation and examples, see the WAF Rule Exclusions API documentation.

> **NOTE**
>
> To reduce the number of log entries generated, we recommend using the API to disable logging once the rule exclusion is working as expected.

Limitations

The WAF Rule Exclusions feature currently has the following limitations:

- A firewall can have a maximum of 300 rule exclusion policies.
- A rule exclusion policy can have a maximum of 30 rules associated with it.
- A rule exclusion policy can be only be associated with strict and scoring rules, not threshold rules.

Web Application Firewall (WAF) (2020)

Last updated: 2020-07-14

https://docs.fastly.com/en/guides/web-application-firewall

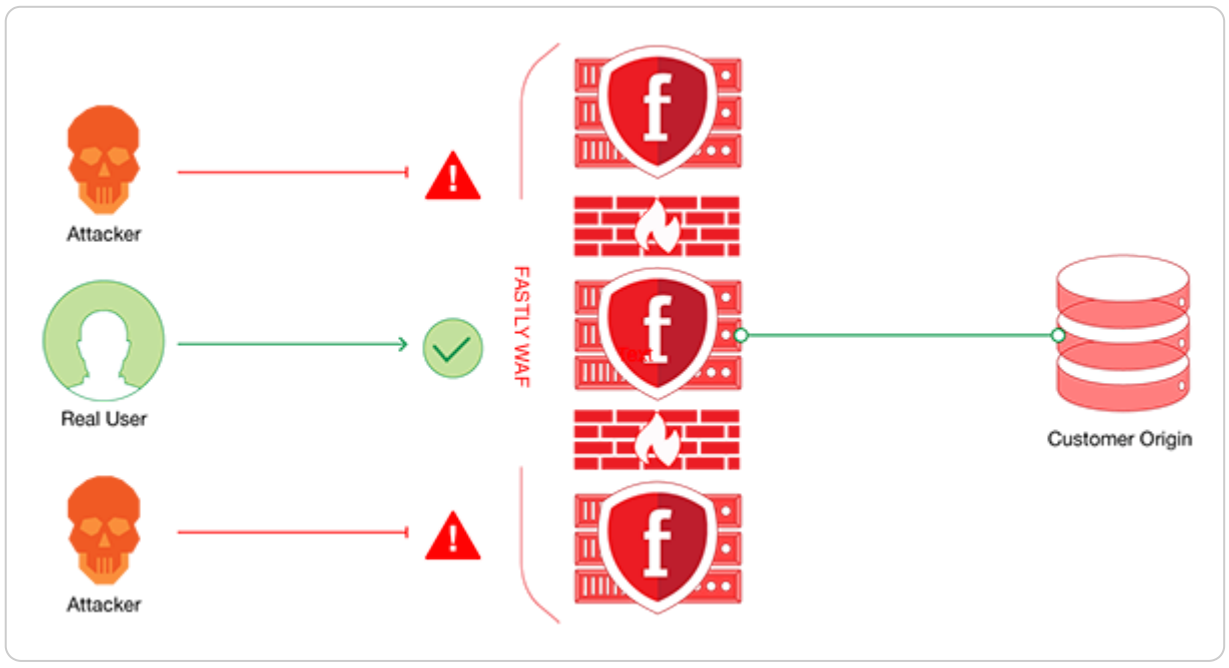
> **IMPORTANT**
>
> As of June 30, 2021, the Fastly WAF (WAF 2020) offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be

controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

Fastly offers a Web Application Firewall (WAF) security product that detects malicious request traffic and can log or log and block that traffic before it reaches your web application. The Fastly WAF provides rules that detect and block potential attacks. The rules are collected into a policy and deployed within your Fastly service at the edge. To get started, email our sales team for product information.

How the Fastly WAF works

The Fastly WAF is designed to protect production web applications running over HTTP or HTTPS against known vulnerabilities and common attacks such as cross-site scripting (XSS) and SQL injection. The Fastly WAF can provide a layer of protection logically positioned at the client edge of your distributed application to detect and block malicious activity from exploiting vulnerabilities in web applications and APIs.



Unlike traditional firewalls that inspect traffic at the network or transport layer, the Fastly WAF works by analyzing web traffic primarily at the HTTP application layer. The Fastly WAF inspects all HTTP and HTTPS headers and post body requests by applying a policy that is selected and tuned for your specific service environment.

Once the Fastly WAF is enabled for a version of your service, you can add or remove rules. The status of a WAF rule can be changed after it is added to your WAF. To update an active rule's status, you must clone your firewall version, make status changes to any active rules you'd like to update, then deploy the new firewall version.

NOTE

The Fastly WAF only works when traffic is directed through it. Make sure that you've signed up for Fastly, created a service, and added a CNAME DNS record for your domain name to direct traffic to Fastly and through the Fastly WAF.

Enabling the Fastly WAF

Enabling Fastly WAF doesn't require modifications to your web application or origin servers.

Refining the default WAF policy once it's enabled

Once you purchase the Fastly WAF, our customer support team will enable it with the default WAF policy for any service you've provided a service ID for. They will then work closely with you on additional configuration refinements, including:

- setting up a logging endpoint,
- adding a prefetch condition,
- customizing the response, and
- selecting rules.

You can then begin monitoring logs to determine which requests to your origin are legitimate and which you should consider blocking.

Setting up a logging endpoint

To begin monitoring requests for potential malicious activity, set up remote logging so you can log WAF variables. You can use an existing logging endpoint or add a new endpoint specifically for Fastly WAF. You'll use the information provided in the logs to monitor WAF events.

Adding a prefetch condition

A prefetch condition is the condition under which Fastly should run a block of code before sending a request to your origin. To avoid running every request against your WAF, add a default prefetch condition, (`req.backend.is_origin`), for the WAF policy. This ensures that the Fastly WAF only inspects traffic to the origin and accounts for whether or not a service has shielding configured.

You can modify the prefetch condition, but keep in mind that you cannot modify a condition's type (`type:request` to `type:prefetch`) after it has been created. If a condition with a `type:prefetch` hasn't been created for your WAF, you can create one using the POST method. For example:

```
$ curl -s -X POST https://api.fastly.com/service/$service_id/version/$version/condition -H "Fastly-Key:$token" -H "Content-Type: application/json" -H "Accept: application/json" -d '{"name": "Waf_Prefetch","priority": "10","statement": "req.backend.is_origin", "type": "prefetch"}'
```

You can modify an existing prefetch statement using the PUT method. For example, you could update the prefetch condition's statement to run the WAF policy on origin traffic and requests from IP addresses that aren't part of an allowlist:

```
$ curl -s -X PUT https://api.fastly.com/service/$service_id/version/$version/condition/Waf_Prefetch -H "Fastly-Key:$token" -H "Content-Type: application/json" -d '{"statement": "req.backend.is_origin && !(client.ip ~ allowlist)"}' -H "Accept: application/json"
```

Customizing the response

Before you can enable your WAF, you must create a custom response and assign an HTTP status code for all requests that Fastly WAF blocks. If you've configured Fastly WAF to block requests, that response will be served directly from the cache when a request matches a rule. If you would like to customize the response, use the web interface to make your changes.

> **TIP**
>
> You can create a custom HTML error page that will be presented to users who are blocked by the Fastly WAF response object. For more information, see our guide on creating a custom WAF error page.

> **WARNING**
>
> If your WAF is created by customer support, do not modify the **Status** or **Description** of the Fastly WAF response.

Selecting rules

The WAF includes rules based on Fastly's own rules, Trustwave ModSecurity Rules, and the OWASP Top Ten. These rules help you monitor web application traffic for a wide range of common attacks. The rules must be explicitly enabled because they are not enabled by default. When you identify rules needed to protect your origin infrastructure, you can add them to your WAF. You can deploy your WAF when you are ready to run traffic past the rules you've added.

Fastly provides rules you can add to your WAF for specific applications or technologies (e.g., WordPress, Drupal, PHP, .Net). Keep in mind that adding additional rules can increase latency for requests being evaluated against the published WAF policy.

Once you've selected the rules for your WAF policy, you can apply the updates Fastly creates at your convenience. If you modify the applications or technologies that are present at the origin, you will need to review the rules available and apply any relevant rules to your WAF.

Monitoring the Fastly WAF

You can use the Fastly WAF dashboard to monitor the Fastly WAF deployed within your Fastly service.

Disabling Fastly WAF for your service

Users with superuser access tokens can disable the WAF on a Fastly service:

```
$ curl -s -X PATCH \
  -H "Fastly-Key:$token" \
  -H "Content-Type: application/vnd.api+json" \
  -d @-
  https://api.fastly.com/service/waf/firewalls/$firewall_id \
<<JSON
{
  "data": {
    "type": "waf_firewall",
    "attributes": {
      "disabled": true
    }
  }
}
JSON
```

§

These articles provide information about the original Fastly Web Application Firewall (WAF) security product.

https://docs.fastly.com/en/guides/security#_legacy-web-application-firewall

About the Fastly WAF dashboard (original)

Last updated: 2020-07-13

https://docs.fastly.com/en/guides/about-the-fastly-waf-dashboard-legacy

IMPORTANT

As of July 13, 2020, Fastly's original WAF offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

The Fastly WAF dashboard allows you to monitor the Fastly WAF deployed within your Fastly service. If you've been assigned the role of engineer or superuser, you can use the information in the Fastly WAF dashboard to determine whether or not the WAF is active, see how many requests the WAF is currently processing, and review recent configuration changes. The Fastly WAF dashboard consists of the WAF summary and WAF audit log pages.

Accessing the Fastly WAF dashboard

To access the Fastly WAF dashboard, follow the steps below:

1. Log in to the Fastly web interface. The Home page appears displaying a list of all services associated with your account.

2. Find your Fastly service in the list, and then click the **WAF** link. The WAF summary page appears.

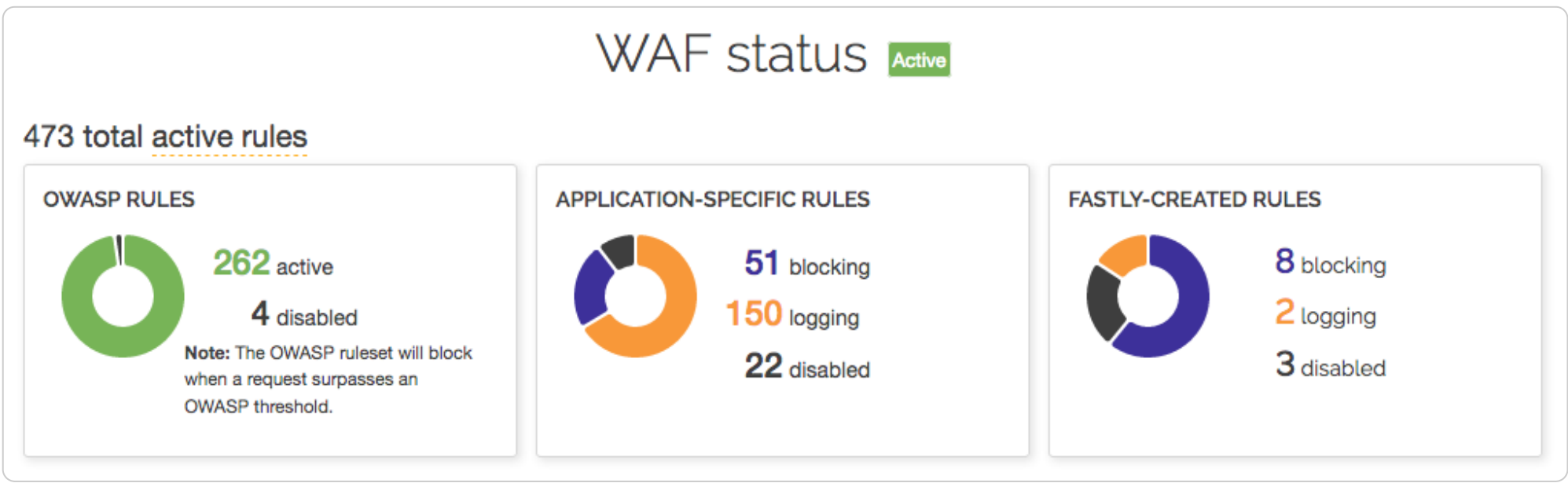
NOTE

To access the Fastly WAF dashboard, you must sign up for a Fastly account and purchase the Fastly WAF. Contact our sales team to get started.

About the WAF summary page

The WAF summary page displays the status of your WAF. The top of the page provides links to the WAF audit log page and Fastly control panel.

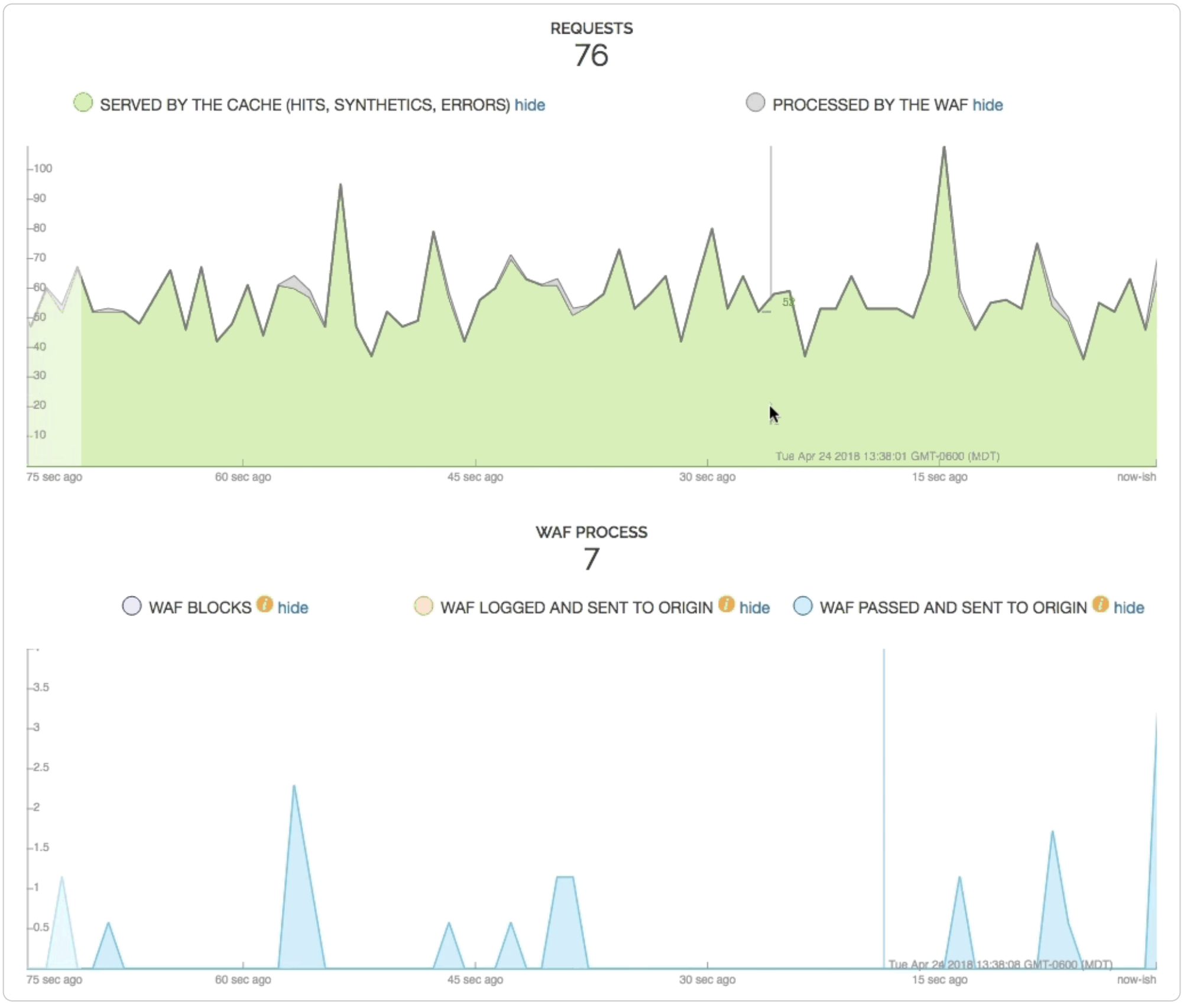
The WAF status section indicates whether the WAF is currently active. To be considered active, the WAF must not be disabled and must have at least one rule status set in either logging or blocking mode. The total number of active rules appears first and represents the combined total of OWASP rules set to active and strict match rules set to blocking or logging. The charts below the total number of active rules separates these out into the number of active and disabled OWASP rules, application-specific rules, and Fastly-created rules. Sample charts are shown below.



The **Requests** graph displays how many requests are served from cache and how many requests are processed by the WAF. Of the requests that are processed by the WAF, the **WAF Process** graph displays how many requests were blocked by the WAF, logged by the WAF and sent to the origin server, and were passed (not blocked or logged) and sent to the origin server.

You can exclude certain data from the graphs by clicking the **hide** link next to a data label. Clicking this link will hide that value in the graph's display.

> **TIP**
>
> The Fastly WAF only executes on traffic sent to the origin server.



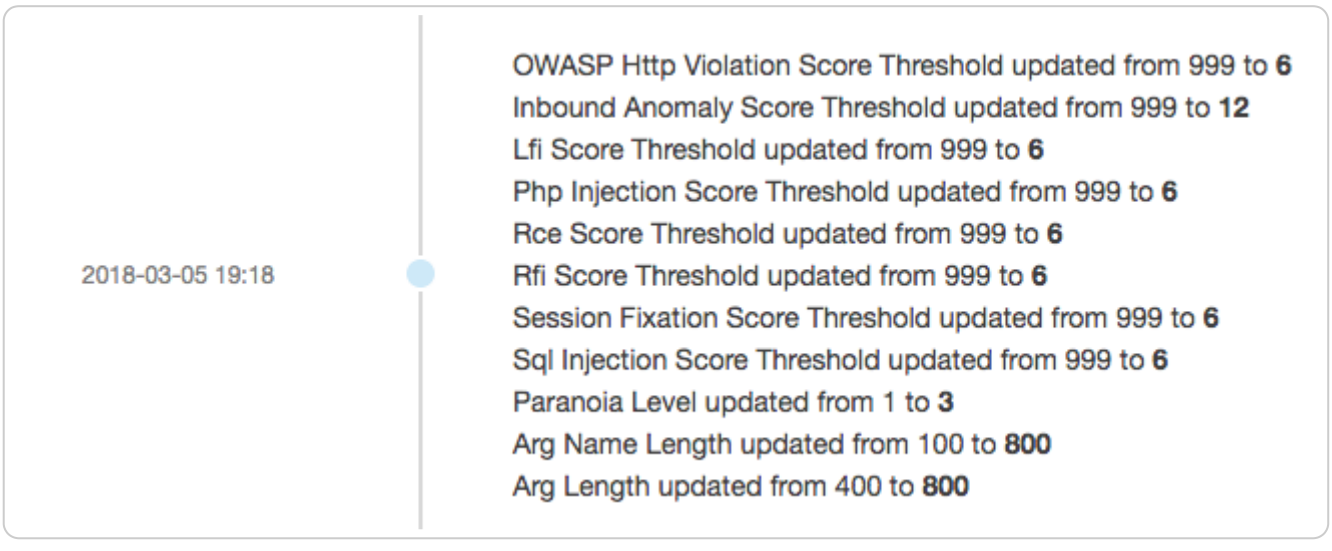
About the WAF audit log page

The WAF audit log page displays all configuration changes made to your WAF. You can use this page to determine who made certain types of configuration changes to the WAF and when the changes were made. The line items indicate when rules were set to log or block, when they were updated, and whether they were disabled.

Some line items include changes for multiple rules. Click **Show rule IDs** to see all of the changes.

> **TIP**
>
> You can use the Fastly WAF rule statuses API endpoint to view the state of an individual rule.

Some entries contain information about the WAF's OWASP properties. To learn more about the OWASP properties, refer to the OWASP properties section.



OWASP properties

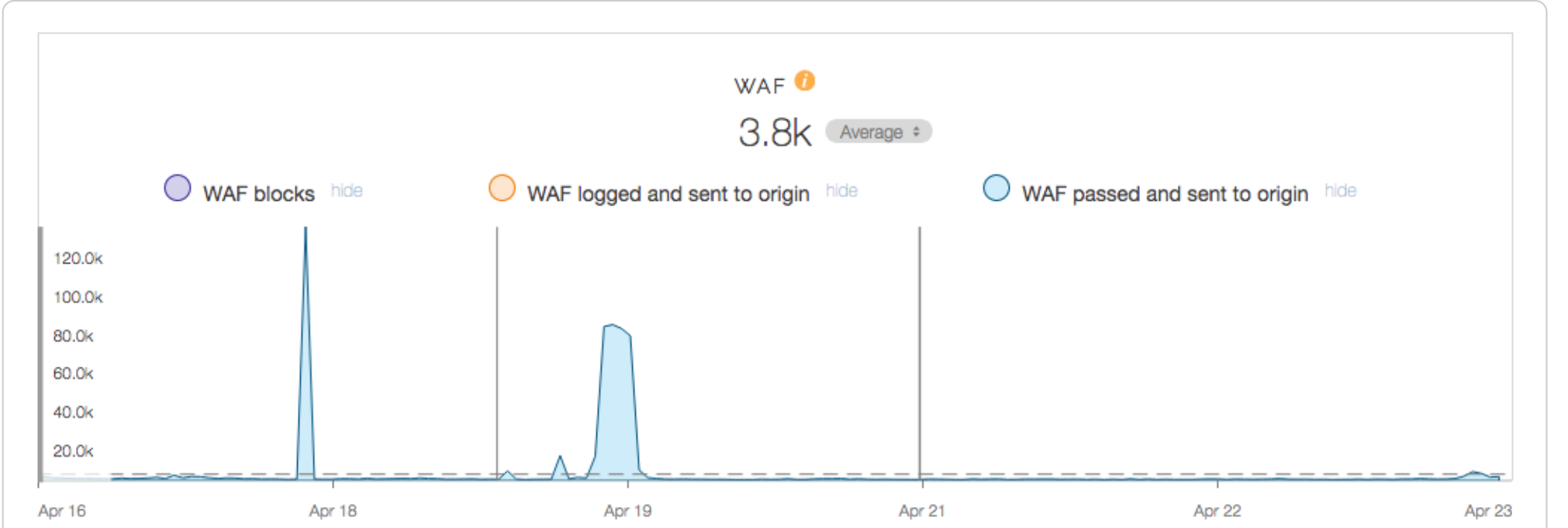
You may see OWASP properties referenced on the WAF audit log page. The table below contains a list of all available properties and their descriptions. The properties shown here reflect changes made by altering the settings in the OWASP object.

OWASP property	Description
Allowed HTTP versions	HTTP versions that a client is allowed to use.
Allowed HTTP methods	HTTP methods that a client is allowed to use.
Allowed client content types	HTTP Content-Types that a client is allowed to use.
Maximum length for parameter names	Maximum length of any parameter names passed in the query string and request body.
Maximum length for parameter values	Maximum length of any parameter values passed in the query string and request body.
Combined file sizes	Total size of MIME bodies in the request.
Critical anomaly score	Configured critical anomaly score. Rules using the critical severity will increment scores using this value.
Validate UTF8 encoding	Validates the client request as UTF-8 prior to the execution of WAF rules.
Error anomaly score	Configured error anomaly score. Rules using the error severity will increment scores using this value.
High risk countries	Block clients from high risk countries based on their IP address.
HTTP violation threshold	Configured HTTP violation threshold. Action is taken when rules that trigger HTTP violations exceed the threshold.
Inbound anomaly threshold	Configured inbound anomaly threshold. Action is taken when the sum of the individual category scores exceed the threshold.
LFI threshold	Configured LFI threshold. Action is taken when rules that trigger Local File Inclusion (LFI) rules exceed the threshold.
Maximum file size (bytes)	Maximum size of any MIME body in the request.
Maximum argument count	Maximum number of parameters in the query string and request body.
Notice anomaly score	Configured notice anomaly score. Rules using the notice severity will increment scores using this value.
Paranoia level	The paranoia level setting can be set from 1 through 4 and determines the number of rules to include by default. Higher levels indicate higher levels of security but potentially a larger number of false positives.
PHP injection threshold	Configured PHP injection score threshold. Action is taken when rules that trigger PHP related violations exceed the threshold.
RCE threshold	Configured RCE injection score threshold. Action is taken when rules that trigger Remote Code Execution (RCE) violations exceed the threshold.
Restricted extensions	Control on restricted file extensions in the client request.

OWASP property	Description
Restricted headers	Control on restricted HTTP headers in the client request.
RFI threshold	Configured RFI violation threshold. Action is taken when rules that trigger Remote File Inclusion (RFI) violations exceed the threshold.
Session fixation threshold	Configured Session Fixation violation threshold. Action is taken when rules that trigger Session Fixation violations exceed the threshold.
SQLi threshold	Configured SQLi threshold. Action is taken when rules that trigger SQL Injection (SQLi) violations exceed the threshold.
Total parameter length	Maximum length of all parameters passed in the query string and request body.
Warning anomaly score	Configured warning anomaly score. Rules using the warning severity will increment scores using this value.
XSS threshold	Configured XSS threshold. Action is taken when rules that trigger Cross-Site Scripting (XSS) violations exceed the threshold.

About the WAF stats

The WAF stats graph appears on the Stats page. For the selected service, this graph shows three things: the blocked traffic that was stopped by the WAF based on rules, the logged traffic that triggered rules but was sent to the origin, and the passed traffic that didn't trigger rules and was sent to the origin.



About the Fastly WAF rule management interface (original)

Last updated: 2020-07-14

https://docs.fastly.com/en/guides/about-the-fastly-waf-rule-management-interface-legacy

IMPORTANT

As of July 13, 2020, Fastly's original WAF offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

The Fastly WAF rule management interface provides visibility and management for rules enabled on a WAF associated with a Fastly service. If you've been assigned the role of engineer or superuser, you can use the rule management interface to inspect the details of WAF rules, search and filter by rule ID or category, manage thresholds and scores, change rule modes, and deploy changes into production.

Limitations

The Fastly WAF rule management interface currently has the following limitations:

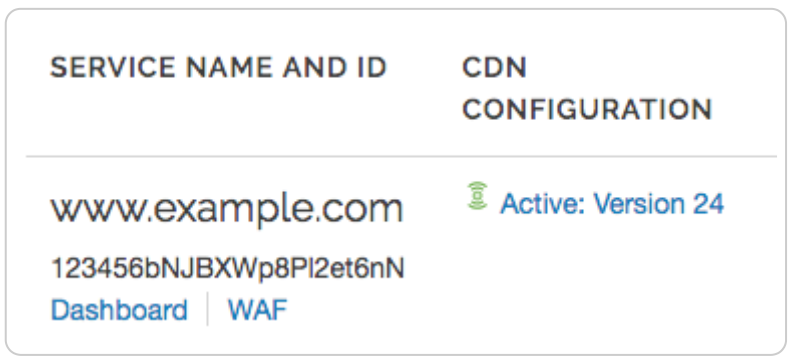
- You can disable threshold rules, which is not recommended. We don't recommend disabling these because it removes categories of protection from your WAF.
- You can't roll back or undo individual rule mode changes. If you change a rule mode inadvertently, you have to change the specific rule back to its original mode. For more information on making production changes to your WAF, see the section on deploying changes.

> **IMPORTANT**
>
> This interface allows you to modify the rule modes on your production service. Lowering threshold scores or changing rules from log to block may cause unexpected false positives.

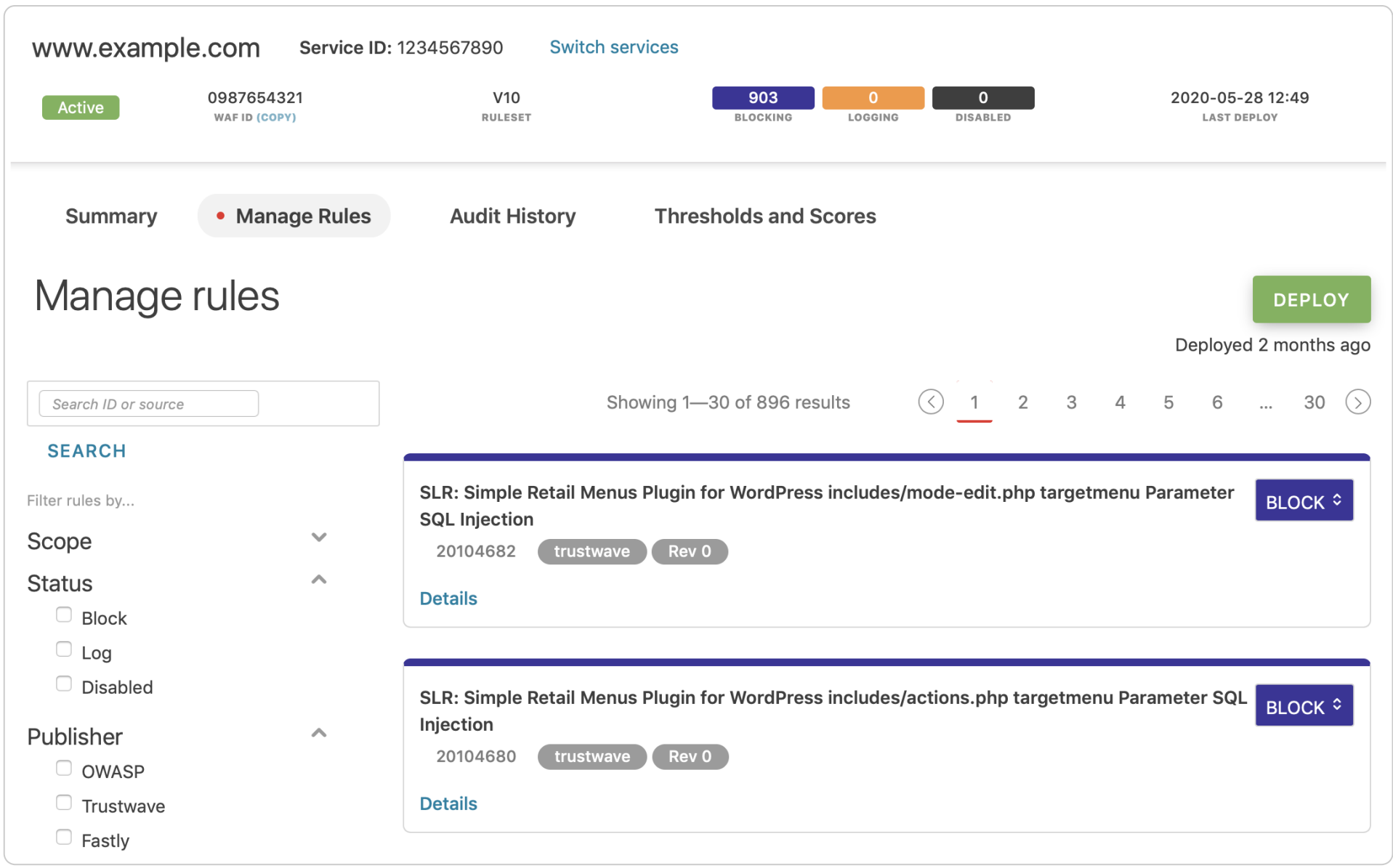
Accessing the Fastly WAF rule management interface

You can access the Fastly WAF rule management interface from the WAF dashboard. To access the Fastly WAF rule management interface, follow the steps below:

1. Log in to the Fastly web interface. The Home page appears displaying a list of all services associated with your account.



2. Find your Fastly service in the list, and then click the **WAF** link. The WAF summary page appears.
3. Click the **Manage Rules** link. The Manage rules page appears.



Using the Fastly WAF rule management interface

The Fastly WAF rule management interface displays the rules currently enabled on the WAF associated with the selected Fastly service. If you haven't enabled rules or you don't see any rules on your WAF, contact support@fastly.com.

The Fastly WAF rule management interface consists of the following main sections:

- WAF Status Bar
- Rule View
- Rule Search
- Category Filters
- Thresholds and Scores

WAF status bar

The status bar summarizes the status of your WAF.

- **Status Indicator:** Displays Active (green) when the WAF is active on your service and protecting your origin.
- **WAF ID:** Displays the WAF ID with a feature that allows you to copy the ID to the clipboard. The WAF ID may be used when accessing the Fastly WAF API.
- **Abbreviated Rule Summary:** Shows your active rules and their associated mode.
- **Deployment Date:** Shows the date and time (UTC) of the last successful service deployment that includes the VCL generated from the WAF rules enabled on this service.

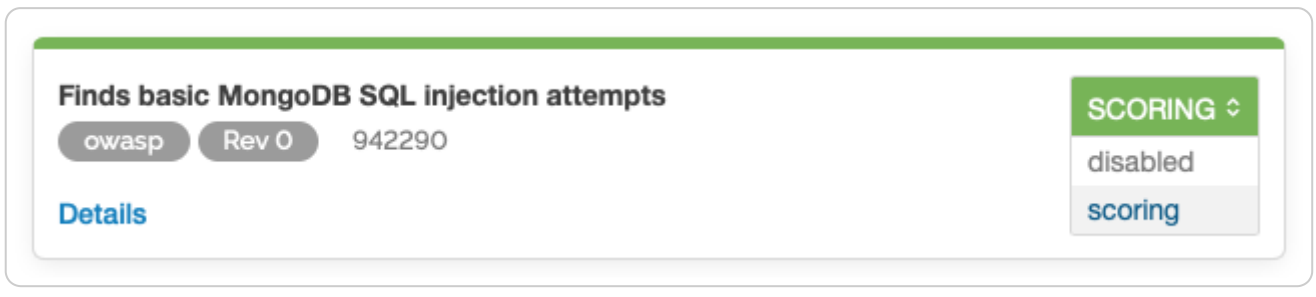
Rule view

The rule view shows a list of all of rules currently enabled on your WAF and their associated mode. There are three types of rules: scoring rules, threshold rules (also called threat categories), and application-specific rules.

Rules have a rule name, tags, revision indicator, rule ID, mode selector, and Details link. The revision indicator is used to indicate whether or not a rule has been revised. A revised rule may provide additional protection over and above the earlier revision.

Scoring rules

Scoring rules increment a score based on anomalies detected in the incoming HTTP request, and threshold rules check that total against the value configured for the appropriate threshold. An example scoring rule is shown below.



TIP

Fastly sets default anomaly scores for four categories of rules. When an HTTP request trips a rule in a category, that numeric value is added to the total score for that category. The sum of all the rules' scores is the anomaly score for the HTTP request. For example, if an HTTP request trips a critical rule and an error rule, and the critical anomaly score is set to 6 and the error anomaly score is set to 5, the score for the HTTP request would be 11.

We don't recommend changing the values of the anomaly scores. Instead, we recommend changing the values of the thresholds on the Thresholds and Scores page.

Scoring rules have two possible modes:

- **Scoring:** This mode means a rule is active and looking for threats. Rules in this mode will increment scores and log their result to your currently configured logging provider based on the setting for the corresponding threshold rule. For more information

on threshold rules, see the section on threshold rules.

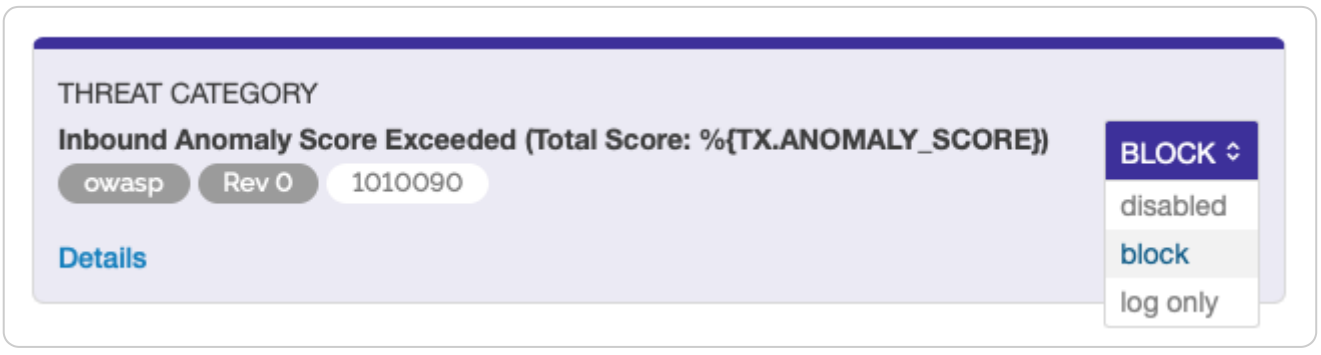
- **Disabled:** This mode means a rule is inactive. Threats detected by this rule will NOT count towards your total anomaly score and their result will not be logged.

You can click the **Details** link to see the format of a rule in the Apache ModSecurity format as well as the corresponding generated VCL.

> **IMPORTANT**
>
> The generated VCL shows a general transformation from the Apache ModSecurity language to VCL. The VCL shown here does not show how a rule increments your anomaly score based on your configured values.

Threshold rules

Threshold rules cover a specific category of attack against your web application or API. An example threshold rule is shown below.



The threshold rule has a category name, revision indicator, tag, rule ID, mode selector, and Details link.

Threshold rules perform an action and either log or block and log client HTTP requests. These rules take action when a score exceeds a given threshold value. The corresponding threshold value for each category is configured on the Thresholds and Scores page.

Lowering thresholds increases the sensitivity of your WAF. Raising thresholds reduces the sensitivity of your WAF across the various threshold categories.

The Fastly WAF includes the following threshold rules organized into categories. Each rule has a corresponding threshold value that controls its sensitivity.

ID	Threshold Name	Rule Action Condition	Corresponding Threshold Value	Action Choice
1010090	Inbound Anomaly Score	Action taken when the inbound anomaly score exceeds the configured inbound anomaly threshold	Inbound anomaly threshold	Log or Block & Log
1010080	Session Fixation	Action taken when the session fixation score exceeds the configured session fixation threshold	Session fixation threshold	Log or Block & Log
1010070	HTTP Violation	Action taken when the HTTP violation score exceeds the configured HTTP violation threshold	HTTP violation threshold	Log or Block & Log
1010060	PHP Injection	Action taken when the PHP injection score exceeds the configured PHP injection threshold	PHP injection threshold	Log or Block & Log
1010050	Remote Command Execution (RCE)	Action taken when the RCE anomaly score exceeds the configured RCE threshold	RCE threshold	Log or Block & Log
1010040	Local File Inclusion (LFI)	Action taken when the LFI score exceeds the configured LFI threshold	LFI threshold	Log or Block & Log
1010030	Remote File Inclusion (RFI)	Action taken when the RFI score exceeds the configured RFI threshold	RFI threshold	Log or Block & Log

ID	Threshold Name	Rule Action Condition	Corresponding Threshold Value	Action Choice
1010020	Cross-site Scripting (XSS)	Action taken when the XSS score exceeds the configured XSS threshold	XSS threshold	Log or Block & Log
1010010	SQL Injection	Action taken when the SQL injection score exceeds the configured SQL injection threshold	SQL injection threshold	Log or Block & Log

IMPORTANT

Disabling a threshold rule removes an entire category of protection from your WAF. Use caution when disabling threshold rules.

Application-specific rules

Application-specific rules look at incoming HTTP requests to find signatures designed to take advantage of specific vulnerabilities within the context of a specific library, framework, or component. They take effect immediately. Application-specific rules have three possible modes:

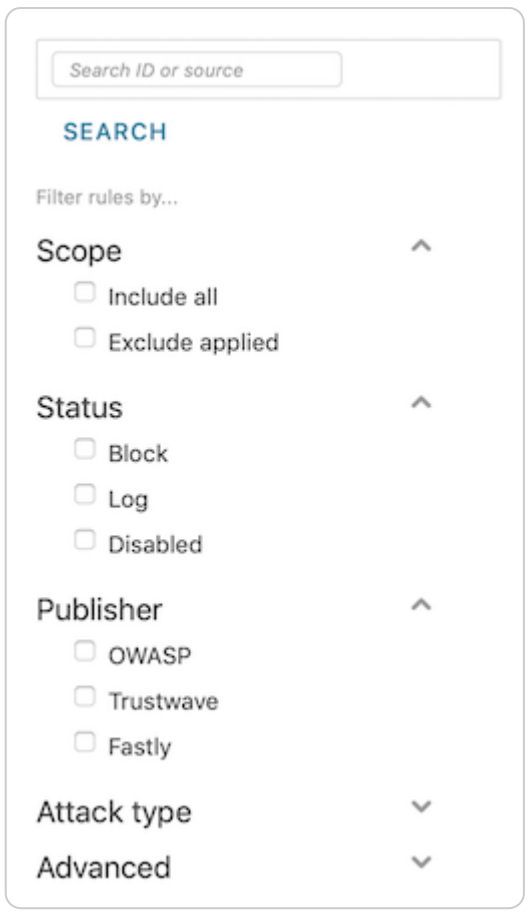
- **Logging Mode:** This mode means a rule is active and looking for threats. Rules in this mode will log their result on an exact match. The result is logged to your currently configured logging provider.
- **Blocking Mode:** This mode means a rule is active and looking for threats. Rules in this mode block the HTTP request and prevent it from going to the origin. Rules in blocking mode also log the result to your currently configured logging provider.
- **Disabled:** This mode means a rule is inactive. HTTP requests that match a rule will flow directly to your origin.

Rule search

The rule search box allows you to search for a specific rule using the rule ID or keyword. The result is shown in the Rule View.

Keyword search allows you to search for rules that contain a specific query string in the rule source. For example, you can search for rules that contain keywords such as `java`, `php`, `loic`, or `ddos`. Keyword search compares your query string against the rule's mod_security source.

You can control the scope of the search through the use of the Include all and Exclude applied filters. Selecting **Include all** expands the search to the entire rule library as well as the rules currently active on your WAF. Selecting **Exclude applied** searches only the rules in the library that are not currently active on your WAF.



Category filters

The category filters allow you to view the different types of rules currently configured on your WAF. Filters can be combined.

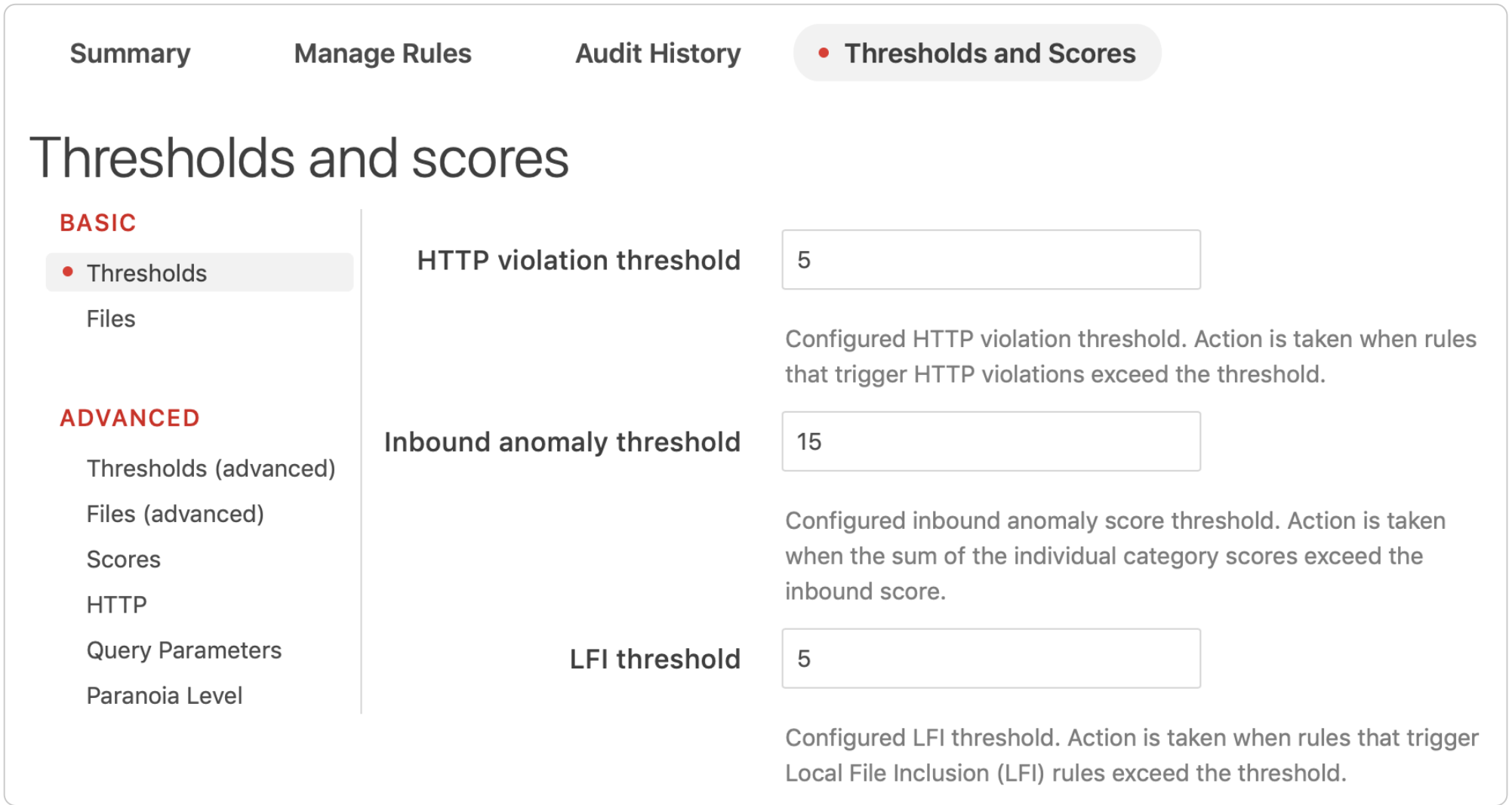
- **Status filter:** Allows you to filter by rule mode for log, block, or disabled.
- **Publisher filter:** Allows you to filter by rule publisher. Currently supported publishers include OWASP, Trustwave, and Fastly.

> **TIP**
>
> Use the Publisher filter and select OWASP to show all rules in scoring mode.

- **Attack type filter:** Allows you to show the rules enabled on your WAF that offer protection for specific categories of attack.

Thresholds and scores

The Thresholds and scores page allows you to configure thresholds and other OWASP security policy settings. You can access these settings by clicking the **Thresholds and Scores** link.



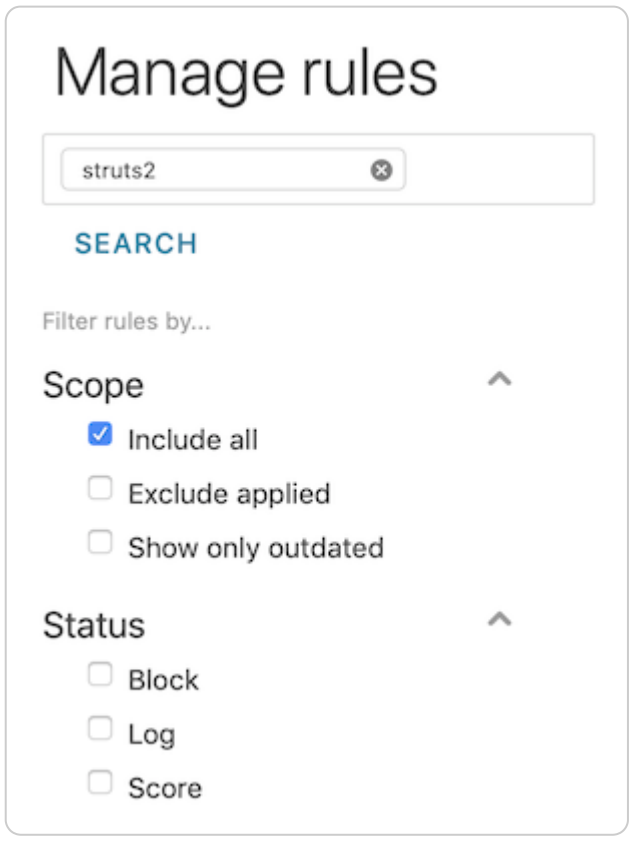
This page can be used to tune the sensitivity of your WAF with respect to thresholds and scores.

Adding new rules to your WAF

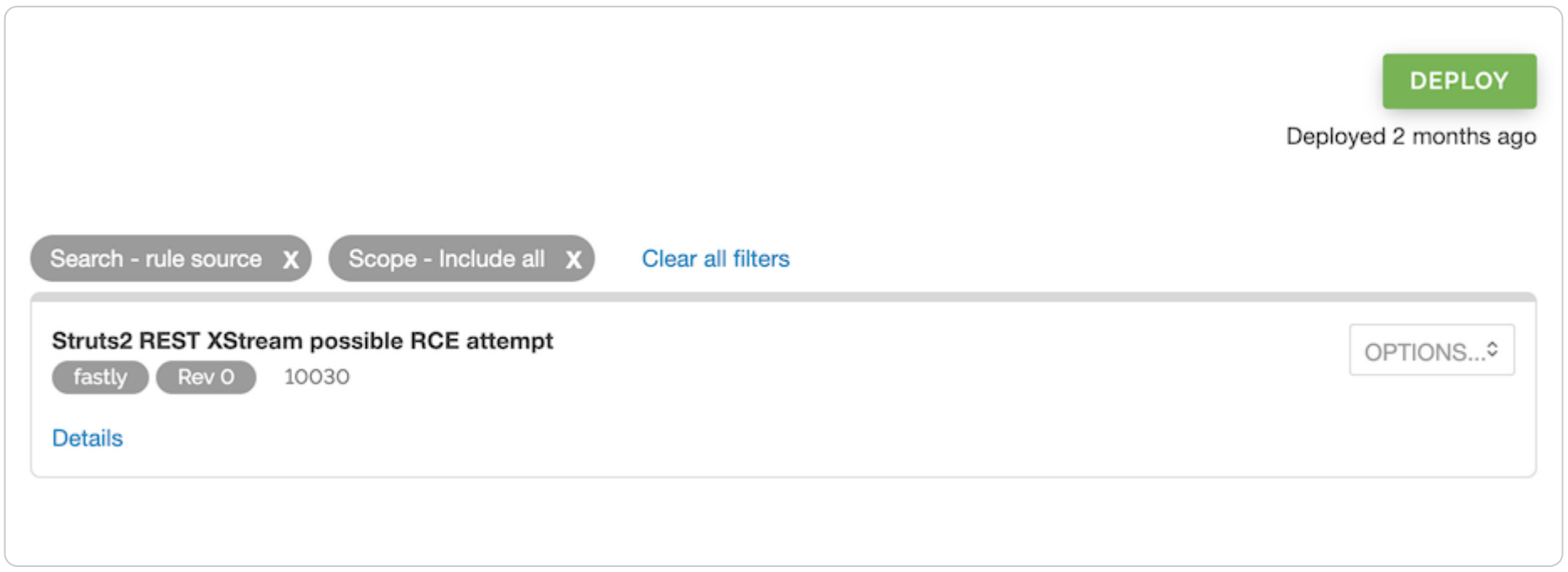
You may want to add new rules to your WAF based on changing attack patterns and risks to your application. You can add new rules by using the scope filters displayed under the search box to browse, search, and select new rules. Selecting the **Include all** filter will display all rules in the current rule library in the Rule View.

> **TIP**
>
> By selecting **Include all**, you will see considerably more rules than you have currently active. Pagination allows you to browse through the rule base.

Once the rules are available to browse, you can use the search box to search for a specific rule ID or keyword. For example, if you're interested in protecting against Apache Struts2 vulnerabilities published in CVE-2017-9805, you might search for `struts2` or `RCE`.

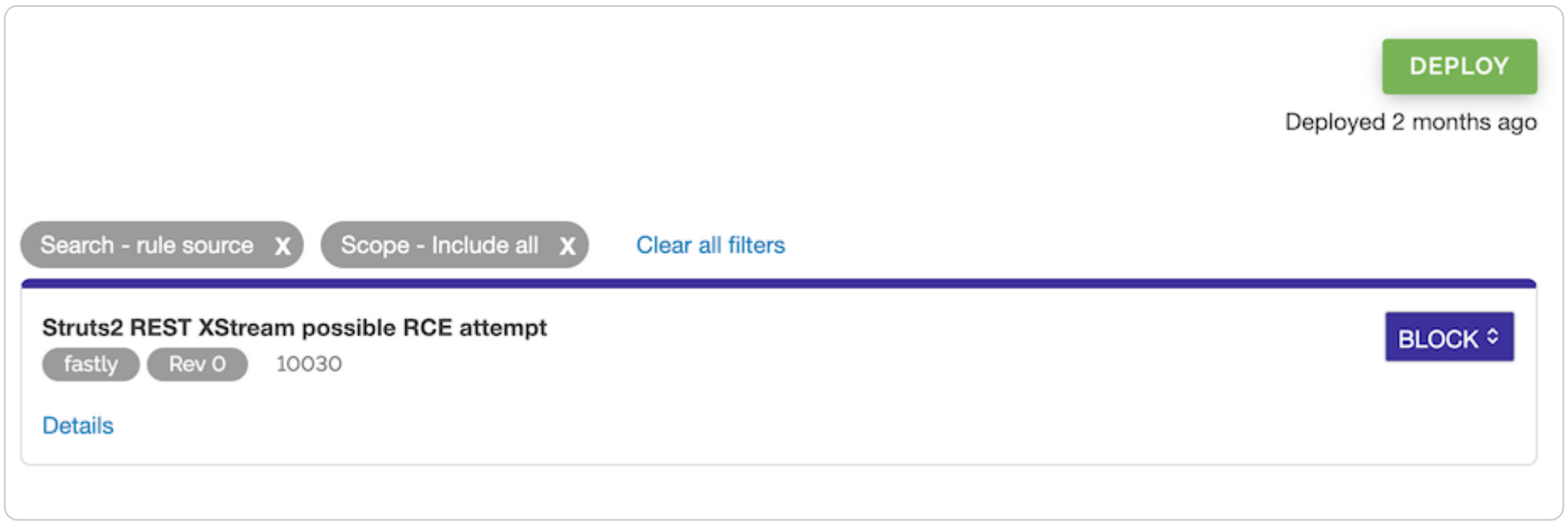


The Rule View will appear as follows.



You can view the rule source by selecting the **Details** link. This provides you with more information about how the rule executes in VCL.

You can enable a rule by selecting **Options** and changing the mode to either **Log only** or **Block**.



> **TIP**
>
> The available options for OWASP rules are Scoring and Disabled. To enable a new OWASP rule, select **Scoring**.

Verifying a rule is active

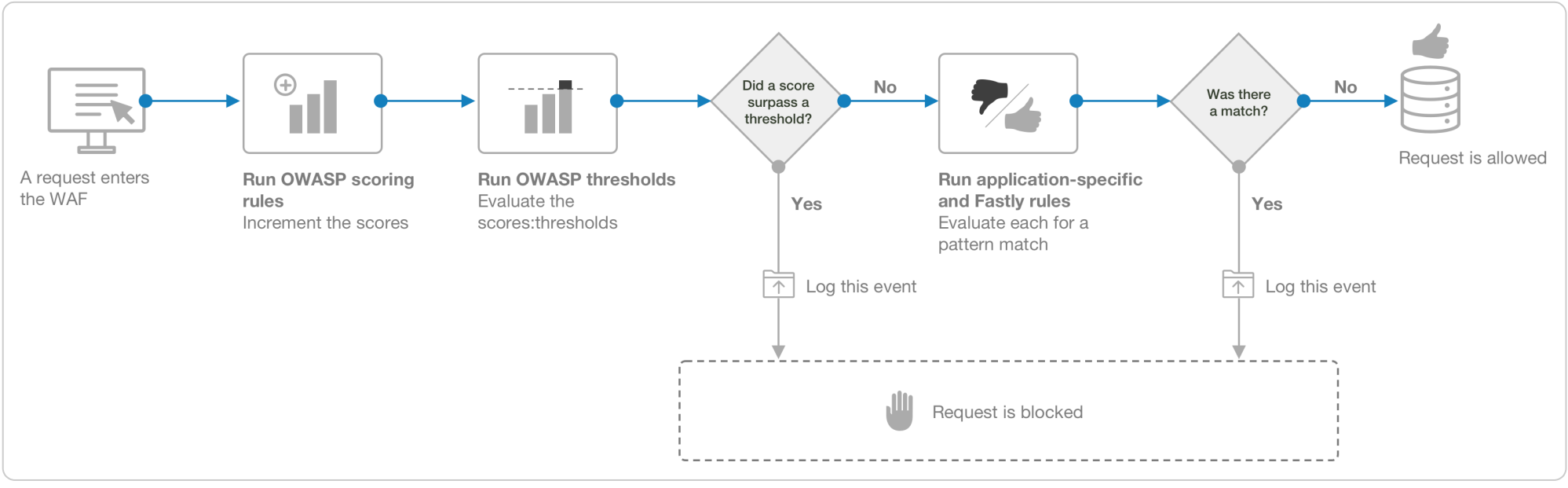
After you select the rule mode, the rule appears at the top of the rule view. To verify that your rule was added, you should deselect the **Include all** filter and use the **Status**, **Publisher**, or **Attack type** filters and confirm that the rule has been added to your WAF.

After you verify that the rule has been added, follow the instructions in the deploying changes section to deploy your changes.

WAF policy execution

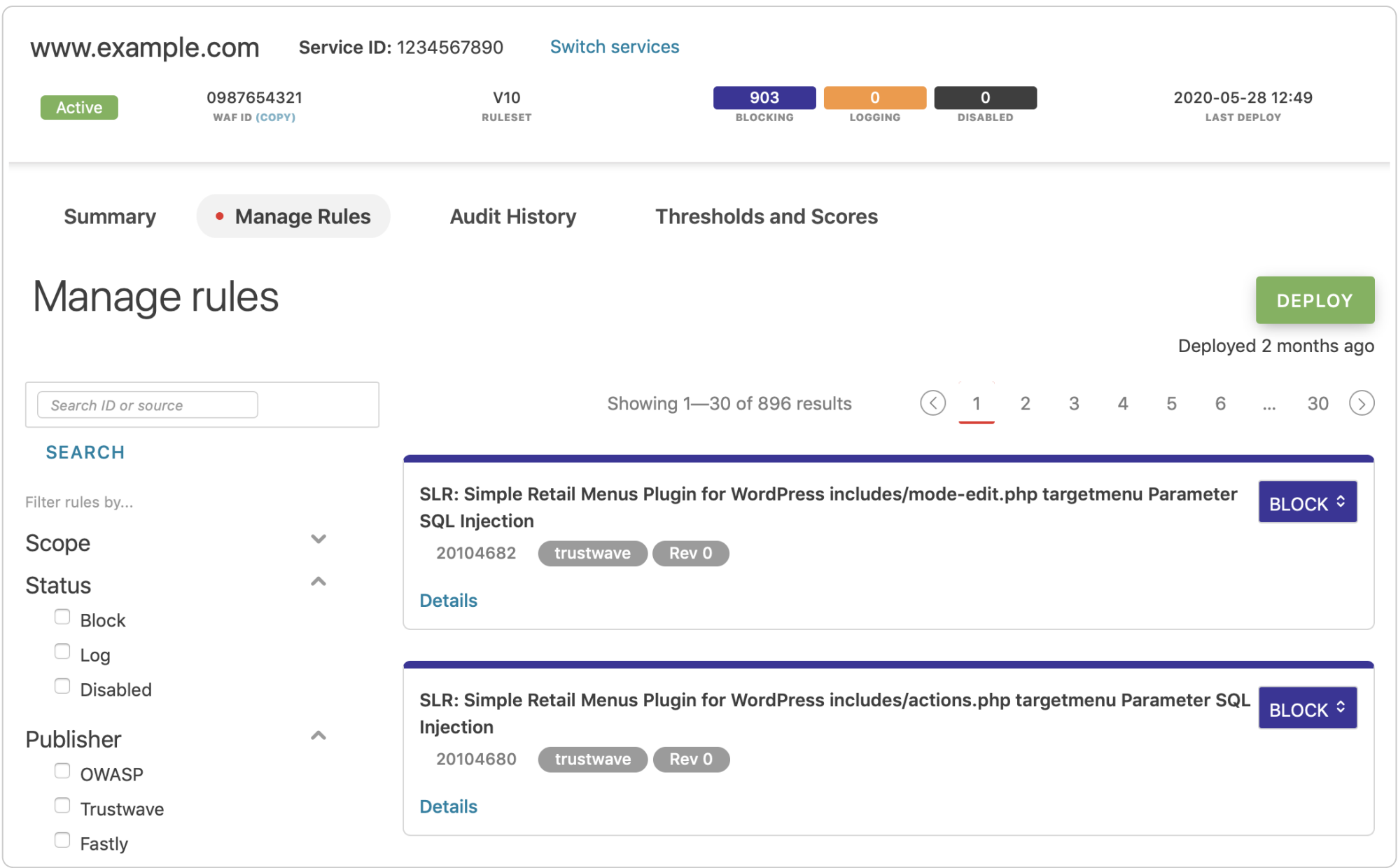
When the Fastly WAF processes an inbound request, scoring rules execute first followed by threshold rules. Application-specific and Fastly rules are executed last.

If the accumulated score exceeds the configured threshold, the threshold rules take action. For more information on the threshold categories and action condition, please see the table in the threshold rules section.

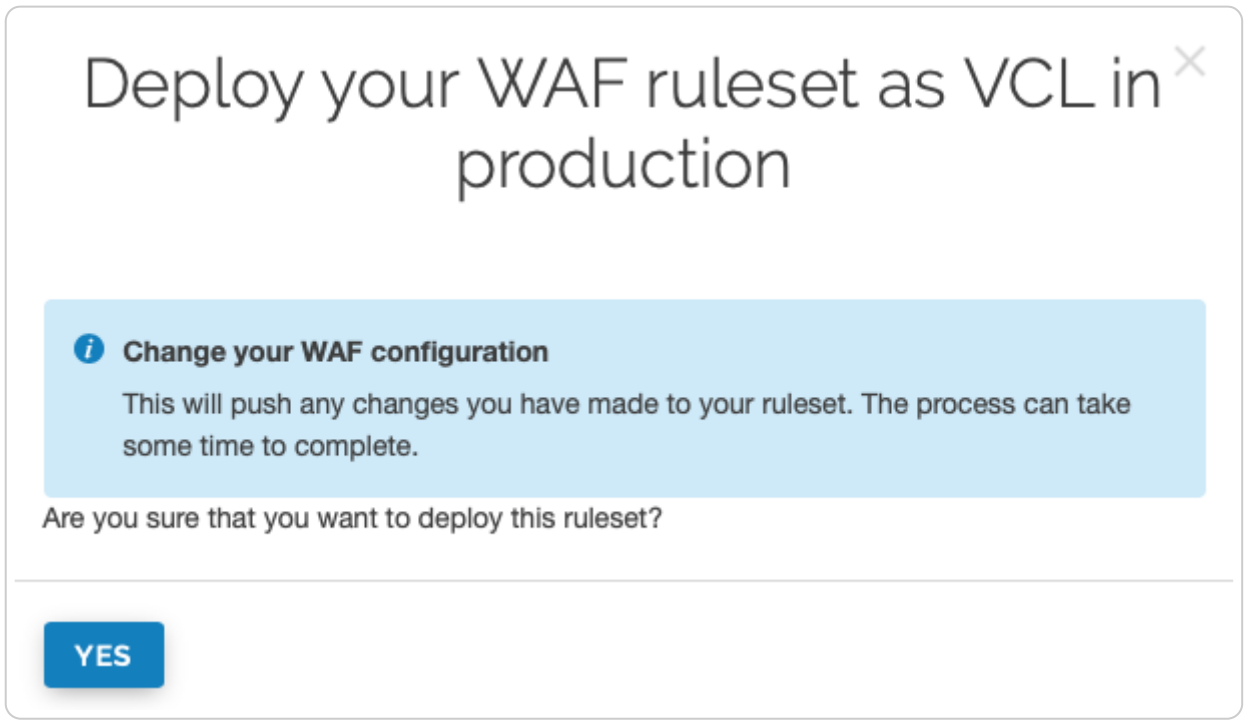


Deploying changes

The Fastly WAF rule management interface allows you to change rule modes and threshold values that change the way rules behave in the face of potential web-oriented attacks. After changing rule modes, you can click the **Deploy** button to put these changes into production.



After clicking **Deploy**, you'll see a confirmation message asking if you want to deploy your changes. Click **Yes** to continue.



Once the deployment is complete, you can verify that the changes were deployed by looking at the date below the deploy button.

IMPORTANT

Changes to your WAF only occur after you select the Deploy button.

Creating a custom WAF error page (original)

Last updated: 2018-11-01

https://docs.fastly.com/en/guides/creating-custom-waf-error-page-legacy

IMPORTANT

As of July 13, 2020, Fastly's original WAF offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

You can create a custom HTML error page that will be presented to users who are blocked by the Fastly WAF response object. The attributes of the response object include the HTTP status code, the HTTP response text, the content type, and the returned content.

For this example, we'll:

- use a dynamic VCL snippet to create a custom `req.http.x-request-id` HTTP header,
- use that header as a global variable to store the transaction ID of the request so that it can be used in both the request and WAF logs, and
- create a synthetic response to present the user with an HTML response.

The error page will display the transaction ID, something that might be useful if, for example, the user decides to contact your support team.

Creating a dynamic VCL Snippet

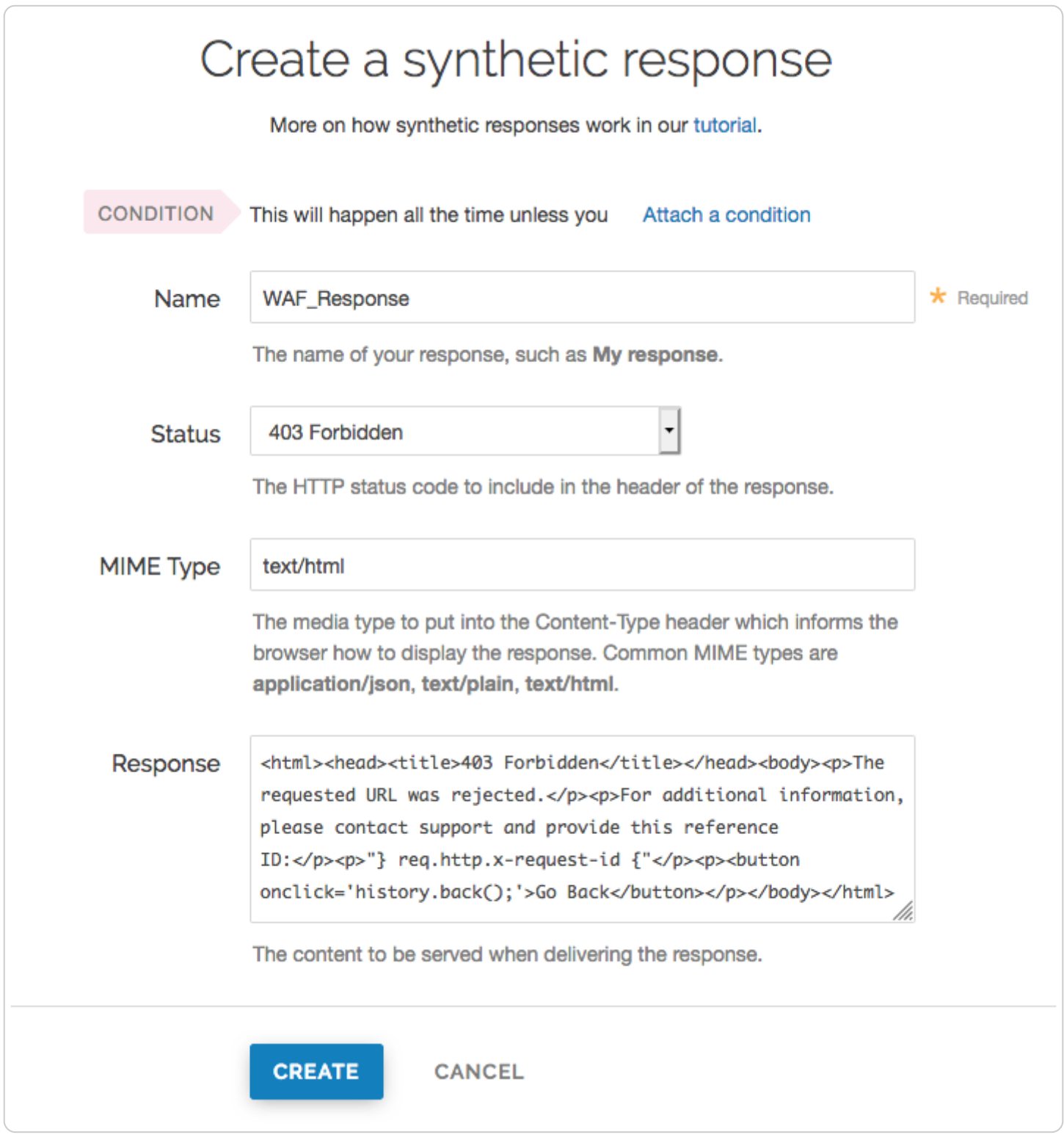
To create a dynamic VCL Snippet for the transaction ID, make the following API call in a terminal application:

```
$ curl -X POST -s https://api.fastly.com/service/<Service ID>/version/<Editable Version Number>/snippet -H "Fastly-Key:FASTLY_API_TOKEN" -H 'Content-Type: application/x-www-form-urlencoded' --data $'name=my_dynamic_snippet_name&type=recv&dynamic=1&content=if (!req.http.x-request-id) {\n set req.http.x-request-id = digest.hash_sha256(now randomstr(64) req.http.host req.url req.http.Fastly-Client-IP server.identity);\n}'
```

Creating a synthetic response

To create a synthetic response for the custom HTML error page, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Set up advanced response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, enter `WAF_Response`.
   - From the **Status** menu, select `403 Forbidden`.
   - In the **MIME Type** field, specify the Content-Type of the response (e.g., `text/html`).
   - In the **Response** field, enter the following HTML. This response will display the value of `req.http.x-request-id`.

```
<html>
  <head>
    <title>403 Forbidden</title>
  </head>
  <body>
    <p>The requested URL was rejected.</p>
    <p>For additional information, please contact support and provide this reference ID:</p>
    <p>"} req.http.x-request-id {"</p>
    <p><button onclick='history.back();'>Go Back</button></p>
  </body>
</html>
```

7. Click the **Create** button. Your new response appears in the list of responses.

8. Click the **Activate** button to deploy your configuration changes.

Additional notes

- You can change the composition of the transaction ID, but care should be taken to minimize the probability that multiple requests within a specified window of time (e.g., per day) have the same transaction ID value.

- A VCL Snippet was used to simplify the example presented and is not explicitly required for a custom WAF error page. As an alternative, you can use custom VCL to create the transaction ID.

- It's useful to include the transaction ID in the request and WAF logging formats to allow multiple messages generated for the same request to be correlated.

Fastly WAF logging (original)

Last updated: 2019-05-30

https://docs.fastly.com/en/guides/fastly-waf-logging-legacy

IMPORTANT

As of July 13, 2020, Fastly's original WAF offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

Fastly provides a number of WAF-specific logging variables to help you monitor and identify potentially malicious traffic. These variables provide specific details about the actions Fastly WAF performed on a request.

Setting up a logging endpoint

To begin monitoring requests for potential malicious activity, set up remote logging so you can log WAF variables. You can use an existing logging endpoint or add a new endpoint specially for Fastly WAF. You'll use the information provided in the logs to monitor WAF events.

OWASP rules

A single request can trigger multiple OWASP rules. By default, logging occurs in `vcl_deliver` or `vcl_log`. When logs are captured in `vcl_deliver` or `vcl_log`, it will show the last WAF rule triggered and the cumulative anomaly score.

waf_debug_log

The `waf_debug_log` subroutine allows logging of each OWASP rule triggered for a single request. You can use the web interface or the API to update the logging placement parameter to `waf_debug`.

Using the web interface

Follow these instructions to set a logging endpoint's placement parameter to `waf_debug`:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears.
5. Click the name of a logging endpoint to edit it. The Edit this endpoint page appears.
6. Click the **Advanced options** link.
7. In the Placement section, select the **waf_debug (waf_debug_log)** setting.

8. Click the **Update** button.
9. Click the **Activate** button to deploy your configuration changes.

Using the API

You can also update the logging placement parameter to `waf_debug` by running the following curl command in a terminal application:

```
$ curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/<your Fastly service ID>/version/<version_id>/logging/<logging_integration>/<logging_name>' --data-binary '{"placement":"waf_debug"}'
```

- `waf_debug_log` accepts the logging format via the web interface only
- `waf_debug_log` is called in `vcl_miss` and `vcl_pass`. The logging format can include request headers and WAF variables. Response headers will result in an error message.
- `<logging_integration>` can be found listed in our remote logging API.

We recommended creating a `request_id` header to track a single request through multiple OWASP rules:

```
set req.http.x-request-id = digest.hash_sha256(now randomstr(64) req.http.host req.url req.http.Fastly-Client-IP server.identity);
```

Using WAF-specific variables

Fastly provides a number of WAF-specific logging variables to help you monitor and identify potentially malicious traffic. These variables provide specific details about the actions Fastly WAF performed on a request:

- **Whether or not Fastly WAF inspected a request.** Fastly WAF only inspects traffic that is forwarded to your origin server (e.g., MISS or PASS requests for content that is not already cached).
- **Whether or not a rule matched the request.** When Fastly WAF inspects a request, it checks to see if the request matches any of the rules in your rule set.
- **The severity of the rule that matched.** If the request matches a rule, the log indicates the severity of the rule.
- **The action taken, if any.** If the request matches a rule or OWASP threshold, the log indicates whether Fastly WAF simply logged the request or blocked it.

You can use the following variables to examine Fastly WAF log events.

Variable	Description
`waf.executed`	A response header indicating if WAF was executed or not. Appears as `1` (true) when executed or `0` (false) when not.
`waf.blocked`	Set to true when the request matches and the specific rule or OWASP threshold is configured to block. Will appear in log files as `1` (true) when blocked or `0` (false) when not.
`waf.logged`	Set to true when the request matches and the specific rule or OWASP threshold is configured to log. In active (blocking) mode, set to `true` when `waf.blocked` is also `true`. Will show up in the logs as `1` (true) or `0` (false).
`waf.failures`	A request exits the WAF rule set due to a failure to evaluate. Will show up in the logs as `1` (true) or `0` (false).
`waf.logdata`	Why (specifically) this rule matched. Includes the portion of the request that triggered the match, so it may look different depending on the rule.
`waf.message`	A message describing the generic condition this rule matched. For example, `SLR: Arbitrary File Upload in Wordpress Gravity Forms plugin`.
`waf.rule_id`	The rule ID for this rule.
`waf.severity`	The severity of the rule. `0` is the highest severity and `7` is the lowest severity. `99` indicates that severity is not applicable (e.g., the request did not match any rules).
`waf.anomaly_score`	Cumulative score returned if request triggers OWASP rules. See OWASP category score variables.
`waf.passed`	Indicates if the request doesn't match any rules in the WAF rule set. Will show up in the logs as `1` (true) or `0` (false). `waf.passed` is readable in `vcl_deliver` and `vcl_log`. It is not readable in `waf_debug_log`. The value is determined after the request has gone through the WAF rule set.

OWASP category score variables

As a request goes through the OWASP rules, it can trigger different rule IDs from different attack categories. OWASP category score variables track which categories were triggered and the scoring that contributed to the cumulative score. They can be used to get a sense of minimum, average, and maximum values for a specific attack category and set thresholds individually. When in active (block) mode, if a request exceeds the category threshold, it will be blocked.

- `waf.sql_injection_score`
- `waf.rfi_score`
- `waf.lfi_score`
- `waf.rce_score`
- `waf.php_injection_score`
- `waf.session_fixation_score`
- `waf.http_violation_score`
- `waf.xss_score`

Fastly WAF rule set updates and maintenance (original)

Last updated: 2021-03-08

https://docs.fastly.com/en/guides/fastly-waf-rule-set-updates-maintenance-legacy

IMPORTANT

As of July 13, 2020, Fastly's original WAF offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the

web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

Fastly provides rule set updates to the Fastly WAF in a prompt manner to help protect customers against attacks.

For OWASP and Trustwave rules changes we use the following process:

1. We regularly review the rule changes as they happen in both the OWASP Core Rule Set and the Trustwave Rule Set.
2. We translate the rules into Varnish Configuration Language (VCL) to run inside our cache nodes.
3. We test the rules in our platform to ensure they perform adequately. We try to maximize performance and rule efficacy while reducing false positives.
4. We correct bugs, if any are found.
5. We propagate the rule set changes to our platform worldwide.
6. Finally, we will provide customers with a notification and instructions on how to make rule updates.

Rule set maintenance

The following links provide information about the updates and changes to the provided rule sets:

ID	Version/Date	Type of Change	Affected Rule Sets
4fsI2JzgtoAXfwpZ89Fehx	v13 2021-03-08	• The OWASP Core Rule Set (CRS) was updated with 10 new rules and 74 updated rules. • Trustwave rules were updated with 213 new rules and 6 updated rules. • Trustwave rule 2500040 was removed. • Fastly rules were updated with 6 new rules and 1 updated rule.	• OWASP • Fastly Rules • Trustwave

6wvihQHbaCG7NBPTfm20S9	v12 2019-08-29	• The OWASP Core Rule Set (CRS) was updated with 19 new rules that mitigate SQL injection, Content-Type anomalies, client side code injection, PHP injection, and remote code execution. In addition, 95 rules were updated in the OWASP CRS to enhance their effectiveness or reduce incidents of false positives. • The following rules were removed from the OWASP CRS: 920130, 920280, 920290, 921100, 941200, 941310, 941350, and 944220. Rules 941310, 941350, and 941200 specifically were removed due to performance issues that may impact your WAF. • Fastly Rules 4112012 and 4112031 have been updated to reduce incidents of false positives. Fastly Rule 4112030 was removed due to excessive false positives. • The Trustwave rules have been updated with 197 new rules, of which 44 are for WordPress and 94 for Joomla. These rules include better protections for customers using these platforms to publish web content. • Trustwave rules 217055, 2066577, and 2100097 were removed. • Some Fastly and Trustwave rules have been renumbered. Renumbering is handled transparently so there should be no impact to your production WAF objects.	• OWASP • Fastly Rules • Trustwave
1PD2HFpi6qwkAsePake7pw	v11 2019-03-25	• Introduced new Fastly rule 4170010, which detects CVE-2019-6340 (Drupal 8 core Highly critical RCE) • Introduced new Fastly rule 4170020, which detects the Magento Magestore Store Locator extension vulnerability • Updated Fastly rule 4112031 to include additional user agents • Updated Fastly rules 4113001, 4120010, and 4120011 to show correct match data • Removed OWASP rules 905100 and 905110, which would never match • Updated OWASP rules 932100 and 932110 to avoid false positives for Windows and Unix command injection	• OWASP • Fastly Rules

		• Introduced new OWASP rule 932190, which mitigates RCE (OS File Access Attempt) on low paranoia level WAF • Introduced new OWASP rule 941110, which mitigates XSS using script tag vector • Introduced new OWASP rule 944100, which mitigates RCE via Java deserialization vulnerabilities (CVE-2017-9805, CVE-2017-10271) • Introduced new OWASP rule 944110, which mitigates RCE via Java process spawn vulnerability (CVE-2017-9805) • Introduced new OWASP rule 944120, which mitigates RCE via Java serialization (CVE-2015-5842) • Introduced new OWASP rule 944240, which mitigates RCE via Java serialization (CVE-2015-5842) • Introduced new OWASP rule 944130, which detects suspicious Java classes • Introduced new OWASP rule 944250, which detects RCE via Java method • Introduced new OWASP rule 944200, which detects magic bytes being used that signal Java serialization • Introduced new OWASP rule 944210, which detects magic bytes being Base64 encoded that signal Java serialization • Introduced new OWASP rule 944220, which detects vulnerable Java class in use • Introduced new OWASP rule 944300, which detects Base64 encoded string that matched suspicious keyword • Introduced new Fastly internal rule 4134010, which mitigates CVE-2018-11776 Apache Struts v2 vulnerability • Introduced new Fastly internal rule 4113010, which detects suspicious X-Rewrite-URL header • Introduced new Fastly internal rule 4113020, which detects suspicious X-Original-URL header • Introduced new Fastly internal rule 4113030, which detects ESI directives in request • Introduced new Fastly internal rule 4113050, which detects ESI directives in body • Removed Trustwave rule 2200000, IP blocklist • Removed Trustwave rule 2200002, TOR Exit Nodes blocklist	
3vnl3cwPda9Q3WYCDRuGW	v10 2018-09-05		• OWASP • Fastly Rules • Trustwave	

67LUkBwzFzESzumlU2L0T8	v9 2018-08-01	• Introduced new Fastly internal rule 4134010, which mitigates common XXE attacks • Introduced new Fastly internal rule 4112019, which mitigates CtrlFunc Botnet Attack • Introduced new Fastly internal rule 4113001, which mitigates suspicious X-Forwarded-Host headers • Introduced new Fastly internal rule 4113002, which mitigates X-Forwarded-Host and Host headers that do not match • Introduced new Fastly internal rule 4120010, which detects illegal characters found in the client X-Forwarded-Host header • Introduced new Fastly internal rule 4120011, which detects illegal characters found in the client X-Forwarded-For header • Updated OWASP rule 930130 to include additional restricted files	• OWASP • Fastly Rules
552NEtnDyzucKd3vTjLgFC	v8 2018-05-11	• Added logdata fields to OWASP rules 920230, 920260, 920270, 920271, 920272, 920273, 920274, 920360 • Introduce new Fastly internal rule 4170001, which mitigates Drupal sa-core-2018-004 attack • Adjust threshold rule 1010090 message	• OWASP • Fastly Rules
6LG4xlelDKWLblCJczGpi9	v7 2018-03-28	• Introduce new Fastly internal rule 4170000, which mitigates Drupal sa-core-2018-002 attack • Updated Fastly internal 4112060 Wordpress PingBack rule • Updated Fastly internal rules that protect against DDoS bots (Rule IDs: 4112013 and 4112016)	• Fastly Rules
1D0OPmXjm6ZMOe9rMGAeQj	v6 2018-01-25	• Update Trustwave rules to latest available • Introduce new Fastly internal rules to protect against DDoS bots (Rule IDs: 4112010-4112018, 4112030, 4112031, and 4112060) • Introduce new Fastly internal rule 10041 (which complements existing rule 10040) to block any HTTP POST body greater than 2 kibibytes in size that uses chunked encoding	• Trustwave • Fastly Rules
2YXlqZJQxMkWyAjM4kggR3	v5 2017-11-13	• Global update to OWASP 3.0.2 CRS release • Update Trustwave rules to latest available • Introduce new Fastly internal rule 10040 to block any HTTP POST body greater than 2 kibibytes in size.	• OWASP • Trustwave • Fastly Rules

2vyJNHO7fngQYJXU8UGUY6	v4 2017-10-06	• Updates to rule 932140 to account for SAML false positives in Windows • Reintroduction of missing transforms on some OWASP rules • Introduction of Fastly internal rule to protect against CVE-2017-9805	• OWASP • Fastly Rules
4Z09wgjp7do8NrOIzIckFS	v3 2017-08-14	• Reintroduction of individual threshold variables: `http_violation_score_threshold`, `lfi_score_threshold`, `php_injection_score_threshold`, `rce_score_threshold`, `rfi_score_threshold`, `session_fixation_score_threshold`, `sql_injection_score_threshold`, `xss_score_threshold` • Removal of unused threshold variables: `brute_force_counter_threshold`, `dos_counter_threshold`, `outbound_anomaly_score_threshold`, `trojan_score_threshold` • Additional bug fixes in OWASP rule set	• OWASP • Trustwave
39EE4tZnEM9Q8hxFJMHYU5	v2 2017-04-26	• Global update to the OWASP CRS 3.0 rule set • New Fastly rule for the February 2017 Wordpress Code Injection • New Fastly rule for the March 2017 Apache Struts RCE exploit • Updated Trustwave content inspection rules	• OWASP • Trustwave • Fastly Rules

RSS and JSON feeds

You can keep tabs on new rule sets by following our RSS and JSON feeds.

Updating to the newest rule set

Follow these instructions to update a WAF to use the newest rule set.

Reviewing the current rule set

Before updating your WAF to a new rule set, we recommend that you record the value of your WAF's currently active rule set. You can use this information to revert your WAF to its previous state.

Run the following curl command in a terminal application to find the currently active rule set:

```
$ curl -s -H Fastly-Key:<your Fastly API token> -H Accept:application/vnd.api+json \
      https://api.fastly.com/service/<your Fastly service ID>/version/<your service version number>/wafs/<your WAF ID>
```

> **TIP**
>
> You can use this API endpoint to find your WAF's ID.

The output from the curl command is shown below. In the `relationships` object, notice that this WAF is using `<ID of your active configuration set>`. Remember the ID.

```
{
    "data": {
        "attributes": {
            "last_push": null,
            "prefetch_condition": null,
            "response": null,
            "version": "1"
        },
        "id": "<your WAF ID>",
        "relationships": {
            "configuration_set": {
                "data": {
                    "id": "<ID of your active configuration set>",
                    "type": "configuration_set"
                }
            }
        },
        "type": "waf"
    }
}
```

Changing the rule set version

Follow these instructions to change the rule set version for a WAF:

1. Find the ID of the new rule set version you want to use in the rule set maintenance section.
2. On your computer, create a new file called `updated_relationship.json`.
3. Copy and paste the following JSON into the file, replacing `<your rules ID>` with the ID of the rule set version you want to use:

```
{
    "data": {
        "id": "<your WAF ID>",
        "relationships": {
            "configuration_set": {
                "data": {
                    "id": "<your rules ID>",
                    "type": "configuration_set"
                }
            }
        },
        "type": "waf"
    }
}
```

4. Save the changes to the `updated_relationship.json` file.
5. In the directory you saved the file, run the following curl command in a terminal application to change the rule set version for a WAF:

```
$ curl -s -X PATCH -H Fastly-Key:<your Fastly API token> -H Accept:application/vnd.api+json \
  -H Content-Type:application/vnd.api+json -d @updated_relationship.json \
  https://api.fastly.com/service/<your Fastly service ID>/version/<your service version number>/wafs/<your WAF ID>
```

6. Changing the rule set version for a WAF can take some time. Run the following curl command in a terminal application to monitor the status of the process:

```
$ curl -s -H Fastly-Key:<your Fastly API token> -H Accept:application/vnd.api+json \
      https://api.fastly.com/service/<your Fastly service ID>/version/<your service version number>/wafs/<your WAF ID>
```

The process is complete when the output displays the ID of the new rule set version.

Updating to the latest rules

After you've verified that the rule set for the WAF has successfully been changed, follow these rules to update your WAF with the latest rules:

1. Run the following curl command in a terminal application to update the rule set:

```
$ curl -s -X PATCH -H Fastly-Key:<your Fastly API token> -H Accept:application/vnd.api+json \
  -H Content-Type:application/vnd.api+json -d '{"data":{"id":"<your WAF ID>","type":"ruleset"}}' \
  https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/ruleset
```

The response will look like this:

```
{
    "data": {
        "id": "WAF_ID",
        "type": "ruleset"
    },
    "links": {
        "related": {
            "href": "https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/update_statuses/<update status ID>"
        }
    }
}
```

2. Updating the WAF with the latest rules can take some time. Using the URL in the response in the previous step, run the following curl command in a terminal application to monitor the status of the process:

```
$ curl -s -H Fastly-Key: FASTLY_API_TOKEN -H Accept:application/vnd.api+json \
https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/update_statuses/<update status ID>
```

The response for the `waf_update_status` will have a `status` of `complete` when the process is complete.

```
{
    "data": {
        "attributes": {
            "completed_at": "2017-04-05 18:47:28 UTC",
            "created_at": "2017-04-05 18:47:27 UTC",
            "message": null,
            "status": "complete",
            "updated_at": "2017-04-05 18:47:28 UTC"
        },
        "id": "<update status ID>",
        "type": "waf_update_status"
    }
}
```

Reverting to a previous rule set version

If a WAF rule set update doesn't go as planned, you can revert to the previous rule set version. Using the previous rule set ID you recorded in the reviewing the current rule set section, follow the instructions in changing the rule set version and updating to the latest rules.

Managing the Fastly WAF (original)

Last updated: 2018-04-24

https://docs.fastly.com/en/guides/managing-fastly-waf-legacy

IMPORTANT

As of July 13, 2020, Fastly's original WAF offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

The Fastly WAF provides rules that detect and block potential attacks. The rules are collected into a policy and deployed within your Fastly service at the edge.

Inspecting the Fastly WAF rule set

You can inspect your Fastly WAF rule set at any time. By making an API call, you can download all of the data associated with your Fastly WAF rules. To inspect your Fastly WAF rule set, run the following curl command in a terminal application:

```
$ curl -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/service/<your Fastly service ID>/wafs/<your WAF ID>/ruleset
 | perl -pe 's/\\n/\n/g'
```

> **NOTE**
>
> The `| perl -pe 's/\\n/\n/g'` is optional and can assist with formatting.

Inspecting the VCL of a WAF rule

To inspect the VCL of a specific Fastly WAF rule, run the following curl command in a terminal application:

```
$ curl -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/wafs/<your WAF ID> /rules/<rule_id>/vcl
```

See the API documentation for more information.

Blocking requests

When you start using Fastly WAF for the first time, all rules are set to `log` status to minimize false positives. We recommend you monitor the logs for a minimum of two weeks to make sure that the rules will not block legitimate requests to your web application. Requests will not be blocked until you switch one or more rules from `log` to `block` status.

Changing the status of rules

To change a rule from `log` status to `disabled` or `block` status, inspect your rule set or review your logs to find the `waf.rule_id` variable. Then, run the following curl command in a terminal application for each rule:

```
$ curl -H 'Fastly-Key: FASTLY_API_TOKEN' -X PATCH -d '{"data": {"id": "<your WAF ID>-<WAF rule ID>", "type": "rule_status",
 "attributes":{ "status": "block"}}}' -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/<your Fastl
y service ID>/wafs/<your WAF ID>/rules/<WAF rule ID>/rule_status
```

To change the status of a group of rules, use a filter-tag (e.g., `application-WordPress`, `language-html`, or `OWASP`) by running the following curl command in a terminal application:

```
$ curl -H 'Fastly-Key: FASTLY_API_TOKEN' -X POST -d '{"data": {"id": "<your WAF ID>", "type": "rule_status", "attributes":
 {"name": <tag>, "status": "block"}}}' -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/<your Fast
ly service ID>/wafs/<your WAF ID>/rule_statuses
```

> **NOTE**
>
> When changing rule statuses for a group of rules using a filter-tag, the above API call will preserve the status of any disabled rules updated individually. If all rules under the filter-tag should be forced to have a `log` or `block` state, add the parameter `force:true` under attributes in the request body.

See the API documentation for more information. When you've finished setting rules to `block` status, you'll need to activate the changes.

> **NOTE**
>
> If you need to enable more than 1,000 rules, contact our customer support team at support@fastly.com.

OWASP Configuration

OWASP blocking is dependent on the following:

- All OWASP rules (excluding rules changed from `log` to `disabled` mode) set to `block` mode.
- Threshold limits set for the cumulative score and attack categories.

If a request triggers OWASP rules, it returns attack category scores and a cumulative score. If any of the final scores exceed the threshold limit and the OWASP rules are in block mode, Fastly sends the custom error response to the user.

Viewing OWASP settings

To view your OWASP settings, run following curl command in a terminal application:

```
$ curl -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/service/<service_id>/wafs/<your WAF ID>/owasp
```

The cumulative anomaly score is displayed in the `inbound_anomaly_score_threshold` field.

Changing OWASP settings

To change any OWASP settings object, run the following OWASP update command in a terminal application:

```
$ curl -X PATCH -v -H "Content-Type: application/vnd.api+json" -H "Accept: application/vnd.api+json" -H "Fastly-Key: FASTLY_
API_TOKEN" https://api.fastly.com/service/<service_id>/wafs/<waf_id>/owasp -d '{"data": {"attributes":{"inbound_anomaly_scor
e_threshold":"50"}, "id":"<owasp_id>", "type":"owasp"}}'
```

When you've finished modifying OWASP settings, you'll need to activate the changes.

Activating changes

After you modify the status of one or more rules, you must activate the changes by running the following curl command in a terminal application:

```
$ curl -H 'Fastly-Key: FASTLY_API_TOKEN' -X PATCH -d '{"data": {"id": "<your WAF ID>", "type": "ruleset"}}' -H 'Content-Typ
e: application/vnd.api+json' https://api.fastly.com/service/ID/wafs/ID/ruleset
```

See the API documentation for more information.

Rules are versionless. Any changes to the rules will become effective after you run the command shown above. You won't need to activate a new version of your service to have the changes take effect.

Web Application Firewall (WAF) (original)

Last updated: 2020-07-14

https://docs.fastly.com/en/guides/web-application-firewall-legacy

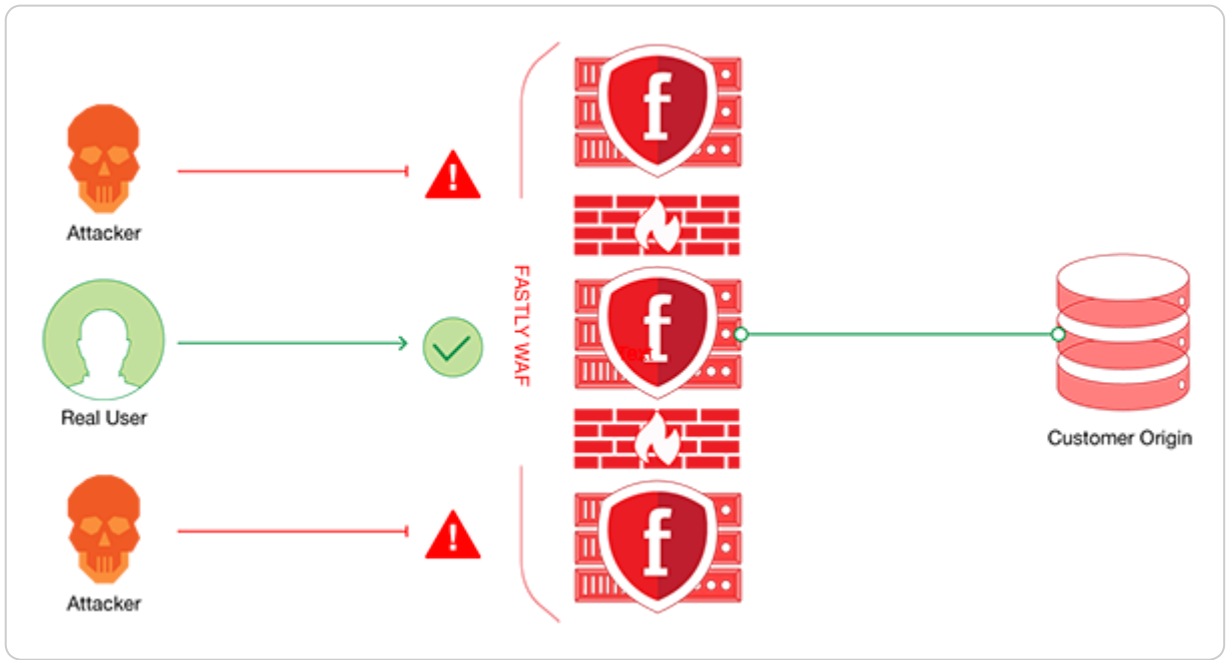
IMPORTANT

As of July 13, 2020, Fastly's original WAF offering became a legacy product. It will continue to be supported for all existing users. As an alternative, Fastly Next-Gen WAF (powered by Signal Sciences) offers proactive monitoring of and protection against suspicious and anomalous web traffic directed at your applications and origin servers. It can be controlled via the web interface dashboard or application programming interface (API). Contact sales@fastly.com or your Fastly account team to evaluate or move to the Fastly Next-Gen WAF option.

Fastly offers a Web Application Firewall (WAF) security product that allows you to detect malicious request traffic and log or log and block that traffic before it reaches your web application. The Fastly WAF provides rules that detect and block potential attacks. The rules are collected into a policy and deployed within your Fastly service at the edge. To get started, email our sales team for product information.

How the Fastly WAF works

The Fastly WAF is designed to protect production web applications running over HTTP or HTTPS against known vulnerabilities and common attacks such as cross-site scripting (XSS) and SQL injection. The Fastly WAF can provide a layer of protection logically positioned at the client edge of your distributed application to detect and block malicious activity from exploiting vulnerabilities in web applications and APIs.



Like traditional network firewall appliances, Fastly WAF uses predetermined security rules to monitor and control incoming traffic to your web application. A network firewall works at the IP level and often blocks IP addresses from untrusted networks, preventing them from gaining access to a private network. Unlike firewalls at the network or transport layer level, the Fastly WAF works by analyzing web traffic primarily at the HTTP application layer. It reads all HTTP(S) headers and the post body of the HTTP(S) requests that it inspects and runs them through a rule set selected for your service environment.

Fastly provides a default WAF rule set to which you can add additional rule sets to help protect against application-specific attacks. Once the Fastly WAF is enabled for a version of your service, you can change the status of any individual rule to logging, blocking, or disabled mode. Rule changes are versionless and become effective immediately.

> **NOTE**
>
> The Fastly WAF only works when traffic is directed through it. Make sure that you've signed up for Fastly, created a service, and added a CNAME DNS record for your domain name to direct traffic to Fastly and through the Fastly WAF.

Enabling the Fastly WAF

Enabling Fastly WAF doesn't require modifications to your web application or origin servers.

Refining the default WAF policy once it's enabled

Once you purchase the Fastly WAF, our customer support team will enable it with the default WAF policy for any service you've provided a service ID for. They will then work closely with you on additional configuration refinements, including:

- setting up a logging endpoint,
- selecting rule sets and a prefetch condition, and
- optionally customizing the request responses.

You can then begin monitoring logs to determine which requests to your origin are legitimate and which you should consider blocking to protect your origin.

Setting up a logging endpoint

To begin monitoring requests for potential malicious activity, set up remote logging so you can log WAF variables. You can use an existing logging endpoint or add a new endpoint for the Fastly WAF. You'll use the information provided in the logs to monitor WAF events.

Selecting rule sets

Fastly provides a default WAF rule set based on Trustwave ModSecurity Rules and the OWASP Top Ten. The default rule set is designed to help you monitor web application traffic for a wide range of common attacks.

Fastly adds a default prefetch condition (`req.backend.is_origin`) for the WAF policy. This ensures that the Fastly WAF inspects traffic to the origin and accounts for whether or not a service has shielding configured.

You can modify the prefetch condition, but keep in mind that you cannot modify a condition's type (`type:request` to `type:prefetch`) after it has been created. If a condition with a `type:prefetch` hasn't been created for your WAF, you must create one using the POST method. For example:

```
$ curl -s -X POST https://api.fastly.com/service/$service_id/version/$version/condition -H "Fastly-Key:$token" -H "Content-Type: application/json" -H "Accept: application/json" -d '{"name": "Waf_Prefetch","priority": "10","statement": "req.backend.is_origin", "type": "prefetch"}'
```

Also, you can attach a prefetch condition to an existing WAF using the PATCH method. For example:

```
$ curl -s -X PATCH https://api.fastly.com/service/$service_id/version/$version/wafs/$waf_id -H "Fastly-Key:$token" -H "Content-Type: application/vnd.api+json" -H "Accept: application/json" -d '{"data": {"attributes": {"prefetch_condition": "WAF_Prefetch"}, "type": "waf", "id": "$WAF_ID" }}'
```

You can modify an existing prefetch statement using the PUT method. For example, you could update the prefetch statement to run the WAF rule set on origin traffic and requests from IP addresses that aren't allowlisted:

```
$ curl -v -X PUT https://api.fastly.com/service/$service_id/version/$version/condition/Waf_Prefetch -H "Fastly-Key:$token" -H "Content-Type: application/json" -d '{"statement": "req.backend.is_origin && !(client.ip ~ allowlist)"}' -H "Accept: application/json"
```

Fastly can add additional rule sets for specific applications or technologies (e.g., WordPress, Drupal, PHP, .Net). Keep in mind that adding additional rule sets can increase latency for requests being evaluated against the published WAF policy.

Once you've selected rule sets, Fastly will maintain rules sourced by Fastly to keep them current. However, you'll need to notify us if you modify the applications or technologies that are present at the origin.

Customizing the response

Fastly's customer support team creates a custom response and assigns an HTTP status code for all requests that Fastly WAF blocks. If you've configured Fastly WAF to block requests, that response will be served directly from the cache when a request matches a rule. If you would like to customize the response, use the web interface to change the following:

- **MIME Type:** The content type of the response.
- **Response:** The content served when delivering the response.

> **TIP**
>
> You can create a custom HTML error page that will be presented to users who are blocked by the Fastly WAF response object. For more information, see our guide on creating a custom WAF error page.

> **WARNING**
>
> Do not modify the **Status** or **Description** of the Fastly WAF response that customer support creates for you.

Monitoring the Fastly WAF

You can use the Fastly WAF dashboard to monitor the Fastly WAF deployed within your Fastly service.

Disabling Fastly WAF for your service

Contact our customer support team at support@fastly.com to disable the Fastly WAF for your service.

Working with rate limiting policies

Last updated: 2022-02-02

https://docs.fastly.com/en/guides/working-with-rate-limiting-policies

The Fastly rate limiting web interface is designed to help you control the rate of requests sent to your origin servers. The feature allows you to count client requests and optionally penalize clients for exceeding rate limits you set.

When you create a rate limiting policy, you define the criteria to track requests counts and their rates over time. Accumulated counts are converted to an estimated rate computed over the time window you specify: either 1s, 10s or 60s. Rates are always measured in requests per second (RPS). If the rate limit is exceeded, Fastly logs the event and can also block additional requests for a time period you specify.

You can also access rate limiting functionality in VCL.

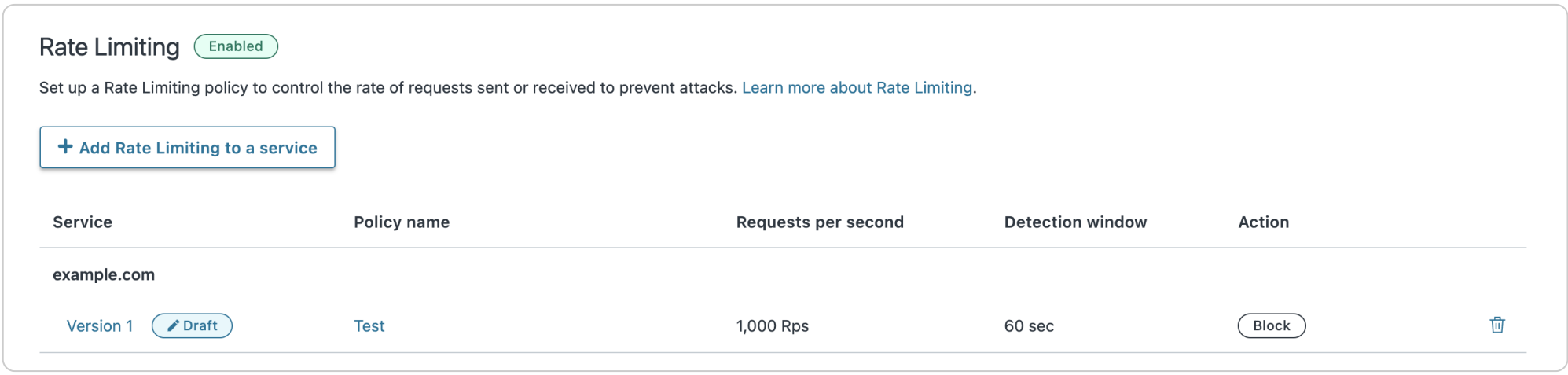
> **IMPORTANT**
>
> This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

Prerequisites

To use this feature you must purchase a Premier Platform subscription for the Fastly Next-Gen WAF (powered by Signal Sciences) and have a paid account for full-site delivery.

About the rate limiting dashboard

To access the Rate Limiting dashboard, log in to the Fastly web interface and click the Secure link. The Rate Limiting dashboard displays a summary of all rate limiting policies currently in effect across your services. Each summary includes the following details:

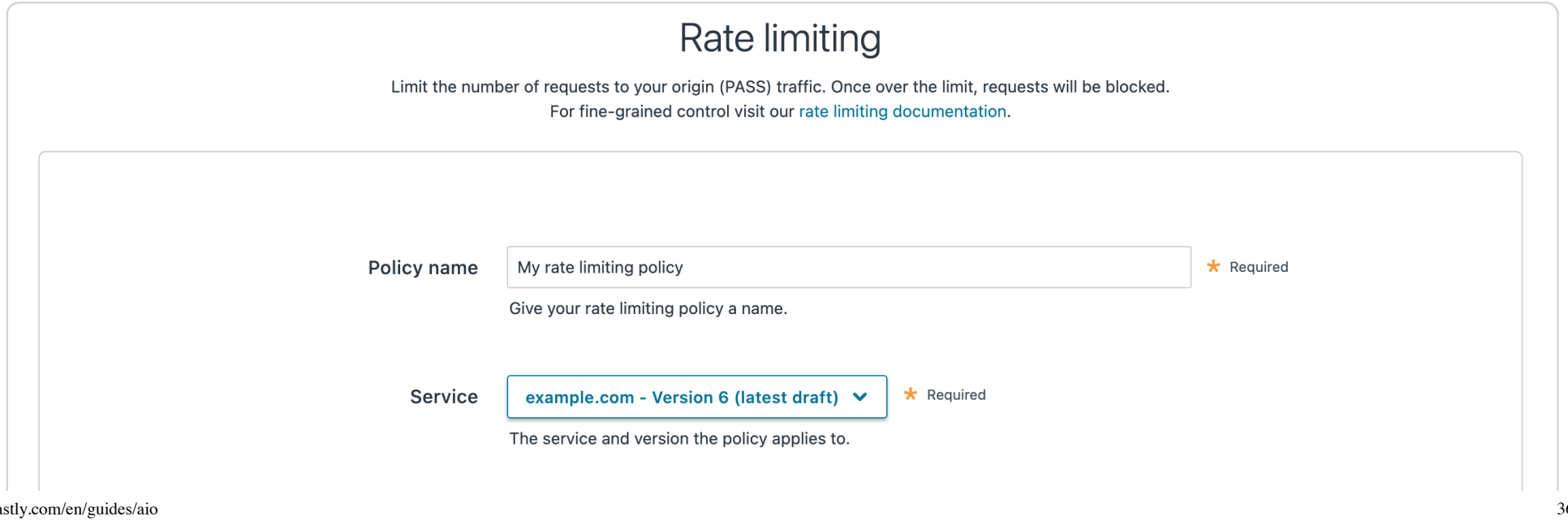


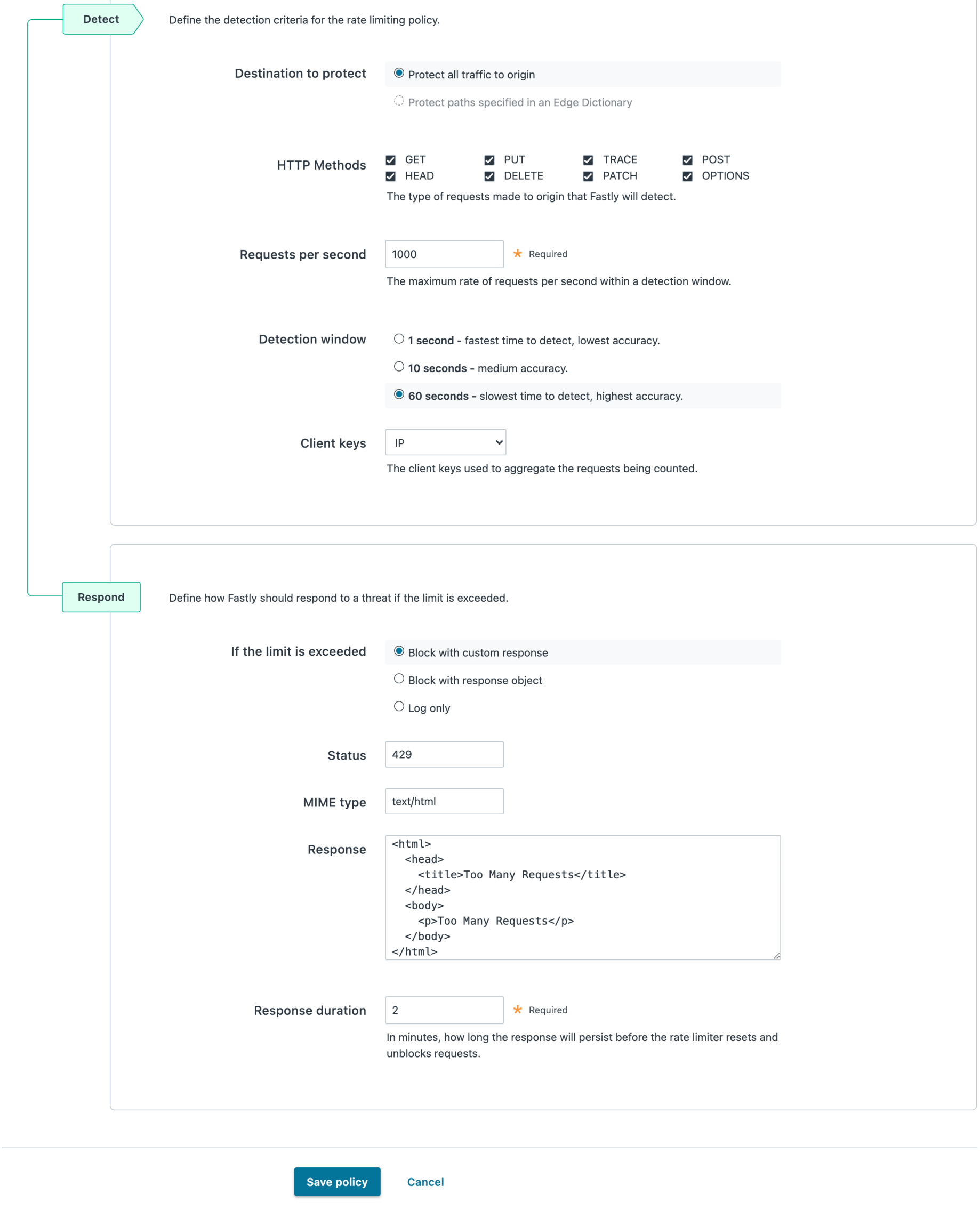
- **Service:** the name of the service
- **Policy name:** the name of the rate limiting policy
- **Requests per second:** the maximum number of requests per second within the detection window counted before enacting the rate limiting policy
- **Detection window:** the duration of the rate limiting window
- **Action:** the action taken once the rate limit is exceeded, either `Block` or `Log only`

Creating a rate limiting policy

Create a rate limiting policy by following these steps:

1. Log in to the Fastly web interface and click the **Secure** link. The Rate Limiting dashboard appears.
2. Click **Add Rate Limiting to a service**. The Rate limiting page appears.





3. In the **Policy name** field, enter a human-readable name for your policy. This name is displayed on the rate limiting dashboard.
4. From the **Service** menu, select the service and version you want to apply the policy to. Use the search box to search by ID or name.
5. In the **Detect** section, fill out the fields to define the detection criteria for the rate limiting policy:
   - In the **Destination to protect** field, select whether to protect all traffic to origin or specific requests via an edge dictionary. If the service you selected doesn't have an edge dictionary, this option is disabled.
   - In the **HTTP methods** field, select the check box next to the types of request to detect.

- In the **Requests per second** field, enter the maximum number of requests per second to count within the detection window before enacting the rate limiting policy. The lowest rate limit supported by this feature to demonstrate effective behavior is 100 RPS. Using a limit below 100 RPS may result in unpredictable accuracy and detection time.
- In the **Detection window** field, select the duration of the rate limiting window. The window size helps determine the effective time to detection (TTD) for excessive traffic to your origin. You can use a shorter window to improve the detection time for attacks at some expense of accuracy. For more information, see Limitations and caveats.
- From the **Client keys** menu, select either **IP**, **User-Agent**, or **IP and User-Agent**.

6. In the **Respond** section, select how Fastly should respond to your rate limiting policy being exceeded, then fill out the additional fields that appear.
   - **Block with custom response:** block requests once the rate limit is exceeded and display a custom response in the browser. See Block with custom response.
   - **Block with response object:** block requests once the rate limit is exceeded and display a response object in the browser. Note that you must have a response object created to use this option. See Block with response object.
   - **Log only:** log when the rate limit is exceeded but still allow requests. See Log only.
7. Click **Save policy**.

Using edge dictionaries for more specific protection

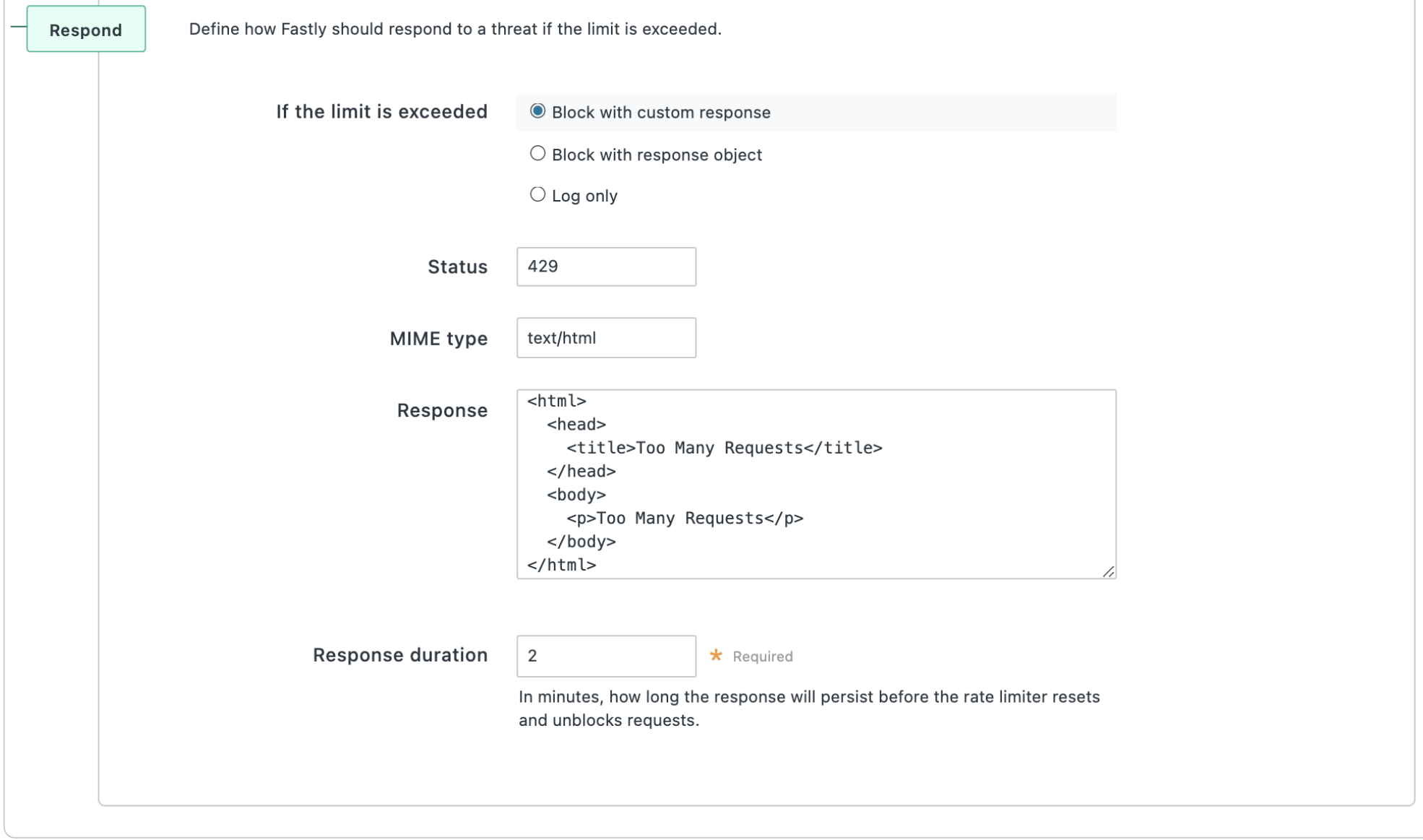
Edge Rate Limiting allows you to specify paths to protect using edge dictionaries. You must specify a path to protect as a key in the key-value pair within a dictionary. For example, to protect a specific API, such as `/checkout`, you would create a key-value pair where the key is `/checkout` and the value is `Checkout`.

Note the following:

- Keys must be specified using relative paths and without using a trailing `/`.
- Query strings are excluded.
- Keys using regex are not supported.

Block with custom response

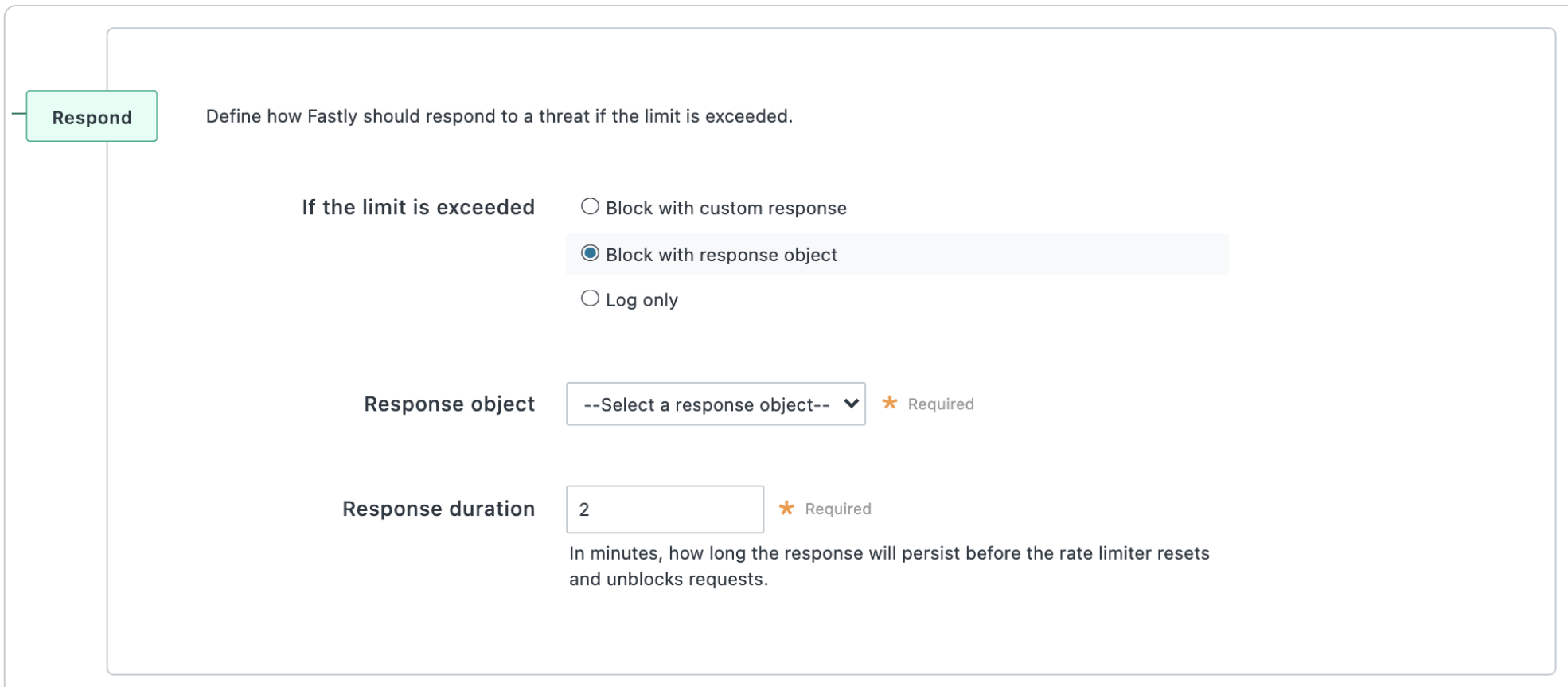
Select the **Block with custom response** option to block requests once the rate limit is exceeded and display a custom response. If you select this option, the following fields become available:



- In the **Status** field, enter an HTTP status code to display in the browser.
- In the **MIME type** field, enter a media type identifier. Any MIME type can be specified as long as it's compatible with the text entered in the `Response` field.
- In the **Response** field, enter the custom response that appears in the browser when the rate limit is exceeded.
- In the **Response duration** field, enter how long, in minutes, to display the custom response before the rate limiter resets and unblocks requests.

Block with response object

Select the **Block with response object** option to block requests once the rate limit is exceeded and display a response object. Note that you must have a response object created to use this option. If you select this option, the following fields become available:



- From the **Response object** menu, select the response object that will appear in the browser when the rate limit is exceeded.

> **NOTE**
>
> Make sure the response object selected has a condition attached or else the response will be returned for all requests.

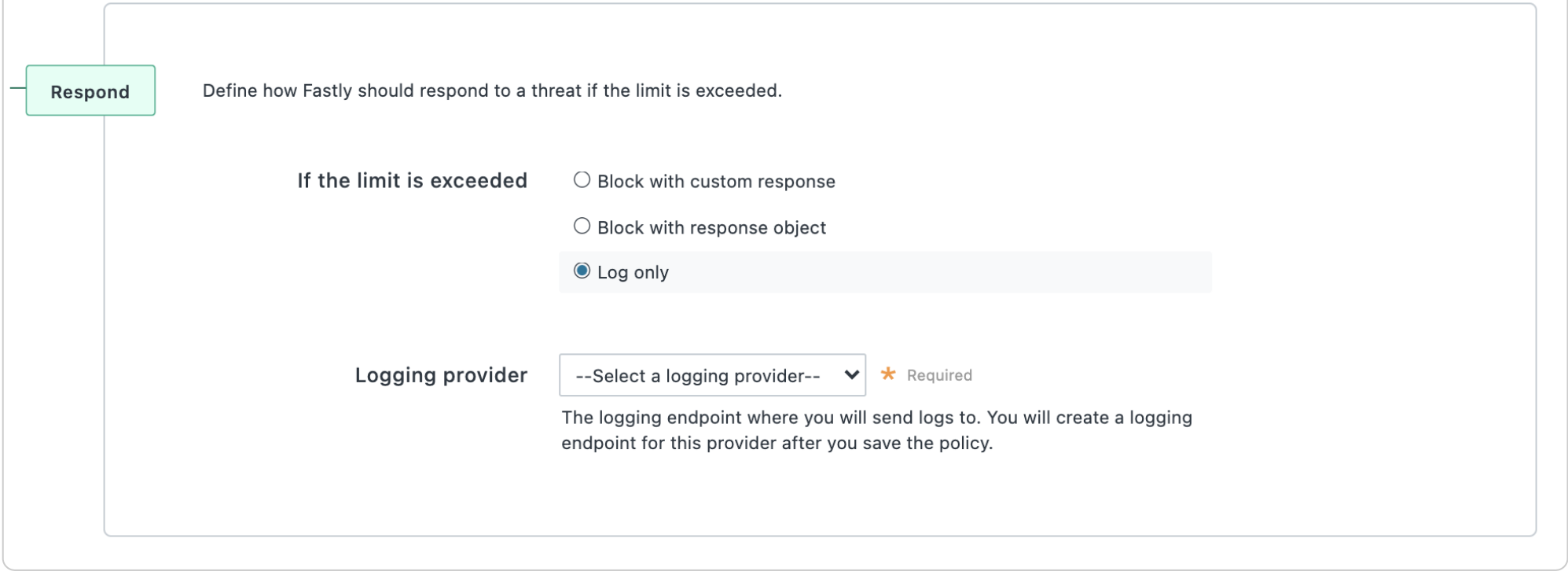
- In the **Response duration** field, enter the number of minutes you want the custom response to appear before the rate limiter resets and unblocks requests.

Log only

Select the **Log only** option to log when the rate limit is exceeded but still allow requests. By default, the following fields are logged: `timestamp`, `policy_name`, `url`, `limit`, `window`, `entry`, and `rate`.

> **NOTE**
>
> In place of the `timestamp` field, Datadog logs `time_start` and Honeycomb logs `time`.

If you select this option, the **Logging provider** menu becomes available. Select the logging endpoint where you want to send logs. Once you click **Save policy**, an abbreviated form appears to create a logging endpoint for the provider you specified. Complete the fields and then click **Save**.



> **NOTE**
>
> You can only have one logging endpoint for all rate limiting policies. By default, the logging endpoint is named `ratelimit-debug`. You can also create this endpoint via the web interface or API.

Log formats

The following JSON log formats are used when you choose the Log only option. These schemas can be useful to understand how data is written to different logging providers.

Unless otherwise specified, the default format is used to log information about particular policies that exceed the rate limit.

Default

```
{
  "timestamp": “%{strftime({"%Y-%m-%dT%H:%M:%S"}, time.start)}V”,
  "policy_name": "%{json.escape(“<rate limiter name>”)}V”,
  "url": “%{json.escape(req.url)}V”,
  "limit": <limit>,
  "window": <window>,
  "entry": "<entry>",
  "rate":  “<rate>"
}
```

Datadog

```
{
  "time_start": “%{strftime({"%Y-%m-%dT%H:%M:%S%Z"}, time.start)}V”,
  "ddsource": "fastly",
  "service": “%{req.service_id}V",
  "policy_name": "%{json.escape(“<rate limiter name>”)}V”,
  "url": “%{json.escape(req.url)}V”,
  "limit": <limit>,
  "window": <window>,
  "entry": "<entry>",
  "rate":  “<rate>"
}
```

Honeycomb

```
{
  "time": “%{strftime({"%Y-%m-%dT%H:%M:%SZ"}, time.start)}V”,
  "data":{
    "service_id": “%{req.service_id}V",
    "policy_name": "%{json.escape(“<rate limiter name>”)}V”,
    "url": “%{json.escape(req.url)}V”,
    "limit": <limit>,
    "window": <window>,
    "entry": "<entry>",
    "rate":  “<rate>"
  }
}
```

Limitations and caveats

Rate limits set by a rate limiting policy happen per Fastly POP and counts are not shared across Fastly POP locations.

The Edge Rate Limiting feature is compatible with Fastly’s origin shield feature and both can be used together. If you have shielding enabled, rate limits will be counted twice, once at the edge and once at the origin shield. This has different implications for where protection is occurring and how the client is identified.

The Edge Rate Limiting feature is not intended to compute rates with high precision. The accuracy you can expect depends on the selected time window over which rates are calculated. Estimated percentage error boundaries under nominal conditions are as follows:

- (+/-) ~50% for the 1 second time window
- (+/-) ~25% for the 10 second time window
- (+/-) ~10% for the 60 second time window

For example, if you are using a 10 second time window and a rate limit of 100 RPS, you may see up to 25% more RPS (125 requests per second) to your origin before the attack is detected by rate limiting. Similarly, rate limiting may report that a rate limit has tripped when the actual rate is 75% of the intended rate (75 requests per second).

Security products note

No security product, such as a WAF or DDoS mitigation product, including those security services offered by Fastly, will detect or prevent all possible attacks or threats. As a subscriber, you should maintain appropriate security controls on all web applications and origins. The use of Fastly's security products do not relieve you of this obligation. As a subscriber, you should test and validate the effectiveness of Fastly's security services to the extent possible prior to deploying these services in production, continuously monitor their performance, and adjust these services as appropriate to address changes in your web applications, origin services, and configurations of the other aspects of your Fastly services.

Integrations

These articles describe how non-Fastly services interoperate with Fastly.

https://docs.fastly.com/en/guides/integrations

§

These articles describe Fastly's support for protocols that allow you to stream logs to a variety of locations, including third-party services, for storage and analysis.

https://docs.fastly.com/en/guides/integrations#_logging-endpoints

Log streaming: Amazon Kinesis Data Streams

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-amazon-kinesis-data-streams

Fastly's Real-Time Log Streaming feature can send log files to Amazon Kinesis Data Streams. Amazon Kinesis Data Streams (KDS) is a real-time data streaming service that can continuously capture data from a variety of sources.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

How Amazon Kinesis Data Streams work with Fastly log streaming

Amazon KDS sends data records to a *stream*. Each stream comprises one or more shards. A shard represents a fixed amount of processing capacity and the total processing capacity of a stream is determined by the number of shards. The number of shards may be increased or decreased over the lifetime of a stream. This is important because the Fastly Kinesis logging endpoint monitors the number of shards and attempts to uniformly distribute the log data records across the available shards. When the number of shards for a stream changes, the Fastly Kinesis logging endpoint automatically adjusts in response. The goal is to make the best use of the throughput capability of the stream while minimizing the configuration overhead required for our customers.

If the log volume exceeds the throughput capacity of the stream, Amazon KDS will return errors to our system that indicate that the stream is being throttled and that may prevent some logs from being delivered. AWS CloudWatch provides a metric for Kinesis Data Streams, `WriteProvisionedThroughputExceeded`, that can be used to monitor this so that adjustments to the stream capacity can be made as necessary.

TIP

For more information about working with Amazon KDS and understanding the capacity limits, refer to the Kinesis Developer Guide.

Prerequisites

Before adding Amazon KDS as a logging endpoint for Fastly services, we recommend creating Identity and Access Management (IAM) credentials in your AWS account specifically for Fastly. Our recommended way for doing this is by creating an AWS IAM role, which lets you grant temporary credentials. For more information, see Creating an AWS IAM Role for Fastly Logging. Alternatively, create an IAM user and grant the user `kinesis:PutRecords` and `kinesis:ListShards` permissions for the logging stream. For more information, see Amazon's guidance on understanding and getting your AWS credentials.

Adding Amazon Kinesis as a logging endpoint

After you've registered for an AWS account and created an IAM user in Amazon Kinesis, follow these instructions to add Amazon KDS as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Amazon Kinesis Data Streams **Create endpoint** button. The Create an Amazon Kinesis Data Streams endpoint page appears.
3. Fill out the **Create an Amazon Kinesis Data Streams endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.

- In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
- In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
- In the **Access method** field, select either **User Credentials** or **IAM Role**.
- If you select **User Credentials**, enter the access key and secret key associated with the IAM user you created in your AWS account specifically for Fastly. See Amazon's documentation on security credentials for more information.

> **NOTE**
>
> Password management software may mistakenly treat the **Secret Key** field as a password field because of the way your web browser works. As such, that software may try to auto-fill this field with your Fastly account password. If this happens to you, the AWS integration with Fastly services won't work and you will need to enter **Secret Key** manually instead.

- If you select **IAM Role**, enter the Amazon Resource Name (ARN) for the IAM role granting Fastly access to KDS. For more information, see Creating an AWS IAM Role for Fastly Logging.
- In the **Stream name** field, enter the name of the Kinesis stream to which log data will be sent.
- From the **Region** menu, select the region to stream logs to. This must match the region where you created your Kinesis stream.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Amazon KDS. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Amazon S3

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-amazon-s3

Fastly's Real-Time Log Streaming feature can send log files to Amazon Simple Storage Service (Amazon S3). Amazon S3 is a static file storage service used by developers and IT teams. You can also use the instructions in this guide to configure log streaming to another S3-compatible service.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Amazon S3 as a logging endpoint for Fastly services, we recommend creating Identity and Access Management (IAM) credentials in your AWS account specifically for Fastly. Our recommended way for doing this is by creating an AWS IAM role, which lets you grant temporary credentials. For more information, see Creating an AWS IAM Role for Fastly Logging. Alternatively, create an IAM user and grant the user `s3:PutObject` permissions for the logging stream. For more information, see Amazon's guidance on understanding and getting your AWS credentials.

Adding Amazon S3 as a logging endpoint

After you've registered for an Amazon S3 account and created an IAM user in Amazon S3, follow these instructions to add Amazon S3 as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

2. Click the Amazon Web Services S3 **Create endpoint** button. The Create an Amazon S3 endpoint page appears.

3. Fill out the **Create an Amazon S3 endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.

   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.

   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.

   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.

   - In the **Bucket name** field, enter the name of the Amazon S3 bucket in which to store the logs.

   - In the **Access method** field, select either **User Credentials** or **IAM Role**.

   - If you select **User Credentials**, enter the access key and secret key associated with the IAM user you created in your AWS account specifically for Fastly. See Amazon's documentation on security credentials for more information.

     > **NOTE**
     >
     > Password management software may mistakenly treat the **Secret Key** field as a password field because of the way your web browser works. As such, that software may try to auto-fill this field with your Fastly account password. If this happens to you, the AWS integration with Fastly services won't work and you will need to enter **Secret Key** manually instead.

   - If you select **IAM Role**, enter the Amazon Resource Name (ARN) for the IAM role granting Fastly access to S3. For more information, see Creating an AWS IAM Role for Fastly Logging.

   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a new S3 endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create an Amazon S3 endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.

   - In the **Domain** field, optionally enter the domain of the Amazon S3 endpoint. If your Amazon S3 bucket was not created in the US Standard region, you must set the domain to match the appropriate endpoint URL. Use the table in the S3 section of the Regions and Endpoints Amazon S3 documentation page. To use an S3-compatible storage system (such as DreamHost's DreamObjects), set the domain to match the domain name for that service (for example, in the case of DreamObjects, the domain name would be `objects.dreamhost.com`).

   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.

- In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
- In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
- From the **Redundancy level** menu, select a setting. This value defaults to **Standard**. Amazon's Using Reduced Redundancy Storage Guide provides more information on using reduced redundancy storage.
- From the **ACL** menu, optionally select an access control header. See Amazon's Access Control List (ACL) Specific Request Headers for more information.
- In the **Server side encryption** area, optionally select an encryption method to protect files that Fastly writes to your Amazon S3 bucket. Valid values are **None**, **AES-256**, and **AWS Key Management Service**. If you select **AWS Key Management Service**, you'll have to provide an AWS KMS Key ID. See Amazon's guide on protecting data using server-side encryption for more information.

6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> Although Fastly continuously streams logs into Amazon S3, the Amazon S3 website and API do not make files available for access until after their upload is complete.

Example format

The following is an example format string for sending data to Amazon S3. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Microsoft Azure Blob Storage

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-azure-blob-storage

Fastly's Real-Time Log Streaming feature can send log files to Microsoft Azure Blob Storage (Blob Storage). Blob Storage is a static file storage service used to control arbitrarily large amounts of unstructured data and serve them to users over HTTP and HTTPS.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Blob Storage as a logging endpoint for Fastly services, create an Azure storage account in the Azure portal. For help creating the account, see Microsoft's account creation documentation.

We recommend creating a Shared Access Signature (SAS) user specifically for Fastly. For more information, see Microsoft's shared access signatures (SAS) documentation, paying specific attention to the Account SAS URI examples.

Here is an example of a SAS token that provides write permissions to a blob:

```
sv=2018-04-05&ss=b&st=2018-04-29T22%3A18%3A26Z&sr=b&se=2020-04-30T02%3A23%3A26Z&sp=w&sig=Z%2FRHIX5Xcg0Mq2rqI3OlWTjEg2tYkboXr1P9ZUXDtkk%3D
```

The table breaks down each part of the token to understand how it contributes to the SAS:

Element	Example	Description
`sv`	`sv=2018-04-05`	Storage services version.
`ss`	`ss=b`	The `signedservice` field. This is required and should be `b` for "blob storage."
`st`	`st=2018-04-29T22%3A18%3A26Z`	The start time of the token, specified in UTC.
`sr`	`sr=b`	Store resources for which this token has access. We require blob (`b`).
`se`	`se=2020-04-30T02%3A23%3A26Z`	The expiry time of the token, specified in UTC. Ensure you update your token before it expires or the logging functionality will not work.
`sp`	`sp=w`	The permissions granted by the SAS token. We require write (`w`).
`sig`	`sig=Z%2FRHIX5Xcg0Mq2...`	The signature to authorize access to the blob.

Adding Blob Storage as a logging endpoint

After you've registered for an Azure account and created a SAS token, follow these instructions to add Blob Storage as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Azure Blob Storage **Create endpoint** button. The Create a Microsoft Azure Blob Storage endpoint page appears.
3. Fill out the **Create a Microsoft Azure Blob Storage endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, enter a string formatted as a comma-separated value (CSV) to use for log formatting. See Ingesting data for Azure Data Explorer for more information.
   - In the **Storage account name** field, enter the unique Azure namespace in which your data objects will be stored.
   - In the **Container** field, enter the name of the Blob Storage container to store logs in. See Microsoft's Blob storage page for more information.
   - In the **SAS token** field, enter the token associated with the container.

     > **TIP**
     >
     > Ensure you update your token before it expires otherwise the logging functionality will not work.

   - In the **Maximum bytes** field, optionally enter the maximum file size in bytes.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.

4. Click the **Advanced options** link of the **Create a Microsoft Azure Blob Storage endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create a Microsoft Azure Blob Storage endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> Although Fastly continuously streams logs into Azure Blob Storage, the storage portal and API do not make files available for access until after their upload is complete.

Ingesting data for Azure Data Explorer

Azure Data Explorer is a data exploration service for log and telemetry data. To ingest your data correctly, Data Explorer requires your logs to be formatted as comma-separated values (CSVs). When creating your logging endpoint:

- Set the **Log format** to a CSV string (`%H,%{time.start.sec}V,%{regsub(req.http.User-Agent, \{"""\}, \{"""\})}V`).
- Specify **blank** when you **Select a log line format** in the **Advanced options**.

Our guide on changing log line formats provides more information.

Log streaming: Cloud Files

Last updated: 2021-11-01

https://docs.fastly.com/en/guides/log-streaming-cloudfiles

Fastly's Real-Time Log Streaming feature can send log file to Cloud Files. Operated by Rackspace, Cloud Files is a file storage service used by developers and IT teams.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

If you don't already have a Rackspace Cloud account, you'll need to register for one. Follow the instructions on Rackspace's website.

Creating a Cloud Files user and container

Start by creating a Cloud Files user with restricted permissions via Rackspace's cloud control panel.

1. Log in to Rackspace's cloud control panel.
2. From the user account menu, select **User Management**.

3. Click **Create User** and fill in all appropriate details.
4. In the **Product Access** section, set **User Role** to **Custom**.
5. Review the **Product Access** list. For all items in the **Product** column, set **Role** to **No Access** except the **Files** item.
6. Set the **Files** item **Role** to **Admin**. This will allow you to create the files to store the logs in, but not access any other services.

Next, find the API key for your Cloud Files account. You'll use this later to authenticate using the Cloud Files API.

1. From the user account menu, select **Account Settings**.
2. Show the API key in the **Login details** and make a note of it.

Now that you've created the Cloud Files user and found the API key, you can set up a Cloud Files container.

1. From the **Storage** menu, select **Files**.
2. Click **Create Container**.
3. Assign the container a meaningful name like `Fastly logs - my service`.
4. Choose a region to keep the files in and make sure the container is private.
5. Click **Create Container**.

Adding a Cloud Files logging endpoint

Once you have created the Cloud Files user and container, follow these instructions to add Cloud Files as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Rackspace Cloud Files **Create endpoint** button. The Create a Cloud Files endpoint page appears.
3. Fill out the **Create a Cloud Files endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Bucket name** field, enter the name of the Cloud Files container in which to store the logs.
   - In the **User** field, enter the username of the Cloud Files user you created above.
   - In the **Access key** field, enter the API key of your Cloud Files account.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
   - From the **Region** menu, select the region to stream logs to.
4. Click the **Advanced options** link of the **Create a Cloud Files endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Cloud Files endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.

- In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Cloud Files. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Coralogix

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-coralogix

Fastly's Real-Time Log Streaming feature can send log files to Coralogix. Coralogix provides an analytics platform that allows you to detect abnormal behavior via dynamic alerts, ratio alerts, flow anomaly detection, and threat discovery.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

If you don't already have a Coralogix account, you'll need to register for one by following the signup instructions on the Coralogix website. Once you've signed up, navigate to the **Send Your Logs** area in the **Settings** section of your Coralogix dashboard and make note of your unique private key. Coralogix uses this to associate data you send them with your account. You'll need it when you set up your endpoint with Fastly.

If you're adding the Coralogix endpoint via the command line, instead of the web interface, you should also have your Fastly API token, the Fastly service ID, and version number of the Fastly service for which you'll be enabling Coralogix logging.

TIP

Consider reading Coralogix's documentation on integrating with Fastly.

Adding Coralogix as a logging endpoint

Follow these instructions to add Coralogix as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

2. Click the HTTPS **Create endpoint** button. The Create an HTTPS endpoint page appears.

3. Fill out the **Create an HTTPS endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, replace the placeholder log format and make the appropriate changes as shown in our log format and recommendations section below.
   - In the **URL** field, enter `https://api.coralogix.com/logs/rest/singles`.
   - In the **Maximum logs** field, leave as `0` (the default).
   - In the **Maximum bytes** field, enter `2000000`.
4. Click the **Advanced options** link of the **Create an HTTPS endpoint** page. The Advanced options appear.
5. Fill out the **Advanced options** of the **Create an HTTPS endpoint** page as follows:
   - In the **Content type** field, enter `application/json`.
   - In the **Custom header name** field, enter `private_key`.
   - In the **Custom header value** field, enter your Coralogix private key.
   - From the **Method** controls, select **POST**.
   - From the **JSON log entry format** controls, select **Array of JSON**.
   - Leave the **Select a log line format** controls set to the defaults.
   - In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported.
   - In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority.
   - In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection.
   - In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log format and field setting recommendations

Use the following log format:

```
{
  "timestamp":%{time.start.msec}V,
  "applicationName":"fastly",
  "subsystemName":"%{req.service_id}V",
  "severity": 3,
  "json": {
    "time": {
        "start":"%{begin:%Y-%m-%dT%H:%M:%S%Z}t",
        "end":"%{end:%Y-%m-%dT%H:%M:%S%Z}t",
        "elapsed":%D
    },
    "cdn_server": {
        "ip_ipaddr":"%A",
        "code":"%{server.datacenter}V",
        "hostname":"%{server.hostname}V",
        "region_code":"%{server.region}V",
        "is_cacheable":%{if(fastly_info.state ~"^(HIT|MISS)$", "true", "false")}V,
        "cache_status":"%{regsub(fastly_info.state, "^(HIT-(SYNTH)|(HITPASS|HIT|MISS|PASS|ERROR|PIPE)).*", "\\2\\3")}V",
        "is_h2":%{if(fastly_info.is_h2, "true", "false")}V,
        "is_h2_push":%{if(fastly_info.h2.is_push, "true", "false")}V,
        "h2_stream_id":"%{fastly_info.h2.stream_id}V"
    },
    "client": {
        "city_name":"%{client.geo.city.utf8}V",
        "country_code":"%{client.geo.country_code}V",
        "country_name":"%{client.geo.country_name}V",
        "continent_code":"%{client.geo.continent_code}V",
        "region":"%{client.geo.region}V",
        "ip_ipaddr":"%h",
        "name":"%{client.as.name}V",
        "number":"%{client.as.number}V",
        "connection_speed":"%{client.geo.conn_speed}V",
        "location_geopoint": {
            "lat":%{client.geo.latitude}V,
            "lon":%{client.geo.longitude}V
        }
    },
    "response": {
        "status":%>s,
        "content_type":"%{Content-Type}o",
        "age":"%{Age}o",
        "cache_control":"%{Cache-Control}o",
        "expires":"%{Expires}o",
        "last_modified":"%{Last-Modified}o",
        "tsv":"%{TSV}o",
        "header_size":%{resp.header_bytes_written}V,
        "body_size":%B
    },
    "request": {
        "host":"%{req.http.host}V",
        "is_ipv6":%{if(req.is_ipv6, "true", "false")}V,
        "backend":"%{req.backend}V",
        "service_id":"%{req.service_id}V",
        "url":"%{cstr_escape(req.url)}V",
        "url_ext":"%{req.url.ext}V",
        "header_size":%{req.header_bytes_read}V,
        "body_size":%{req.body_bytes_read}V,
        "method":"%m",
        "protocol":"%H",
        "referer":"%{Referer}i",
        "user_agent":"%{User-Agent}i",
        "accept_content":"%{Accept}i",
        "accept_language":"%{Accept-Language}i",
        "accept_encoding":"%{Accept-Encoding}i",
        "accept_charset":"%{Accept-Charset}i",
        "connection":"%{Connection}i",
        "dnt":"%{DNT}i",
        "forwarded":"%{Forwarded}i",
        "via":"%{Via}i",
        "cache_control":"%{Cache-Control}i",
        "x_requested_with":"%{X-Requested-With}i",
        "x_att_device_id":"%{X-ATT-Device-Id}i",
        "x_forwarded_for":"%{X-Forwarded-For}i"
    },
    "socket": {
        "cwnd":%{client.socket.cwnd}V,
        "pace":%{client.socket.pace}V,
        "nexthop":"%{client.socket.nexthop}V",
        "tcpi_rcv_mss":%{client.socket.tcpi_rcv_mss}V,
        "tcpi_snd_mss":%{client.socket.tcpi_snd_mss}V,
```

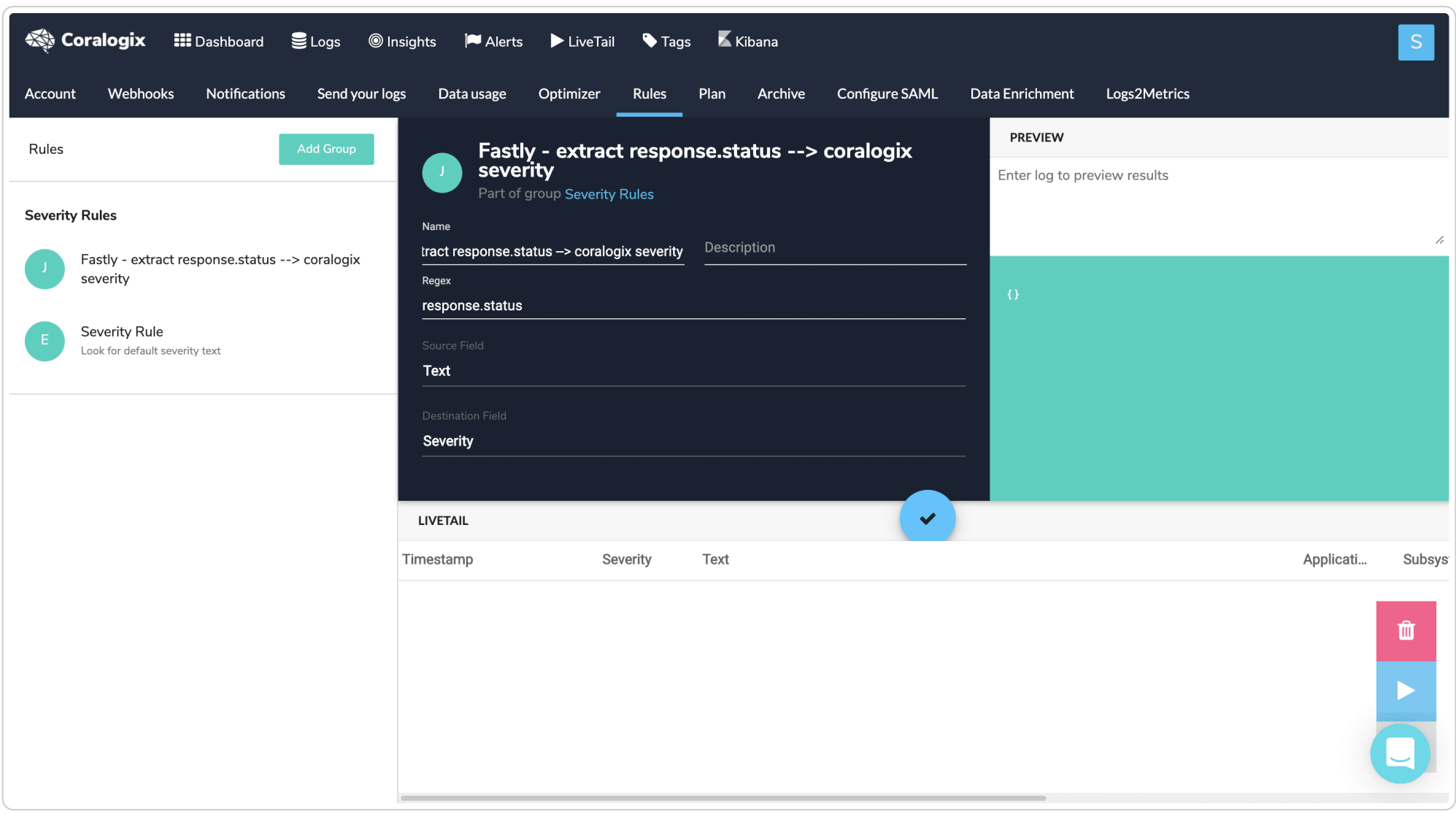
```
        "tcpi_rtt":%{client.socket.tcpi_rtt}V,
        "tcpi_rttvar":%{client.socket.tcpi_rttvar}V,
        "tcpi_rcv_rtt":%{client.socket.tcpi_rcv_rtt}V,
        "tcpi_rcv_space":%{client.socket.tcpi_rcv_space}V,
        "tcpi_last_data_sent":%{client.socket.tcpi_last_data_sent}V,
        "tcpi_total_retrans":%{client.socket.tcpi_total_retrans}V,
        "tcpi_delta_retrans":%{client.socket.tcpi_delta_retrans}V,
        "ploss":%{client.socket.ploss}V
    }
  }
}
```

The first five fields of the recommended format are required:

- `Timestamp` – Leave the format of this field unchanged.
- `applicationName` – Enter the name of the application in this field.
- `subsystemName` – Enter the name of the subsystem in this field. This is used to separate components. We use `req.service_id` in the example, which isn't particularly human readable. Use whatever subsystem name makes sense that helps you identify the subsystem.
- `Severity` – Specify the severity and apply it to all logs using the following choices: 1 (debug), 2 (verbose), 3 (info), 4 (warning), 5 (error), 6 (critical). This can be changed later using an extract rule as described below.
- `JSON (object)` – Add or remove fields as necessary. Static fields can be added. Nested JSON formats are supported including any fields described in the Fastly VCL reference.

The `response.status` field sends the request status. This is a recommended field. Using the Coralogix parsing rules, you can set a JSON Extract rule to extract the status code value from the request into a Coralogix severity, allowing you to define the severity to automatically determine the importance of the type of log. Specifically, you can automatically map HTTP status codes into a severity tag as appropriate. For example, status code `2xx` will set the Coralogix severity as **"INFO"** and status code `4xx` will set Coralogix severity as **"ERROR"**.

In the Coralogix web interface, it will look like this:



Configuring Coralogix dashboards and alerting

Coralogix provides tutorials for integrating their service with Fastly via dashboards and alerting. This includes examples of data dashboards created using Fastly data, including one for a general service overview, a visitor breakdown, and quality of service.

Their tutorials also describe how to set up user-defined alerts for situations like no logs being received from Fastly, outages at your origin, elevated error ratios and cache misses, unusual or suspicious requests of various types, as well as potential website defacement attempts.

Log streaming: Datadog

Last updated: 2021-06-09

https://docs.fastly.com/en/guides/log-streaming-datadog

Fastly's Real-Time Log Streaming feature can be configured to send logs in a format readable by Datadog. Datadog is a cloud-based monitoring and analytics solution that allows you to see inside applications within your stack and aggregate the results.

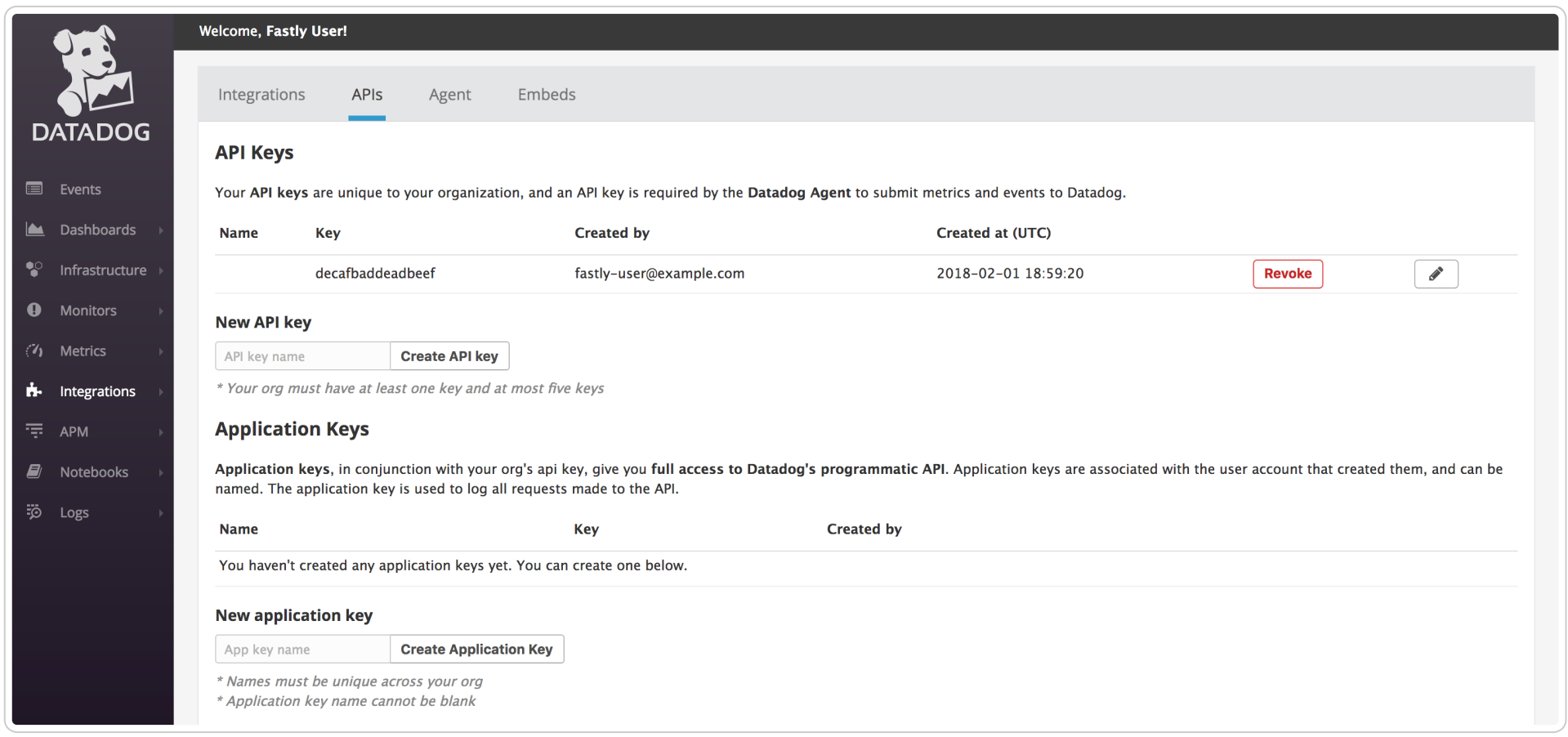
NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Datadog as a logging endpoint for Fastly services, you will need to:

- **Register for a Datadog account.** You can sign up for a Datadog account on their site. A free plan exists that has some restrictions or you can upgrade for more features. Where you register your Datadog setup, either in the United States (US) or the European Union (EU), will affect which commands you use during logging endpoint setup at Fastly.
- **Get your Datadog API key from your settings page on Datadog.** In the Datadog interface, navigate to Integrations → APIs where you'll be able to create or retrieve an API key.



This example displays the key `decafbaddeadbeef`. Your API key will be different. Make a note of this key somewhere.

Adding Datadog as a logging endpoint

After you've created a Datadog account and noted your Datadog API key, follow the steps below to add Datadog as a logging endpoint for Fastly services.

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Datadog **Create endpoint** button. The Create a Datadog endpoint page appears.
3. Fill out the **Create a Datadog endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.

- In the **Log format** field, enter the data to send to Datadog. We've described the use of this format below with additional suggestions.
- From the **Region** menu, select the region to stream logs to.
- In the **API key** field, enter the API key of your Datadog account.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Logs should begin appearing in your Datadog account a few seconds after you've created the endpoint and deployed your service changes. These logs can then be accessed via the Datadog Log Explorer on your Datadog account.

Using the JSON logging format

Datadog automatically parses log files created in JSON format, making this format the easiest way to get data into Datadog because no customized logging rules are required. In addition, Datadog recognizes several reserved fields, such as `service` and `date`.

> **NOTE**
>
> The JSON in this example is formatted for ease of reading. For proper parsing, it must be added as a single line in the **Log format** field, removing all line breaks and indentation whitespace first.

For example, in the JSON below we've set `service` to the ID of the Fastly service that sent the log but you could also use a human-readable name or you could group all logs under a common name such as `fastly`.

```
{
  "ddsource": "fastly",
  "service": "%{req.service_id}V",
  "date": "%{begin:%Y-%m-%dT%H:%M:%S%z}t",
  "time_start": "%{begin:%Y-%m-%dT%H:%M:%S%Z}t",
  "time_end": "%{end:%Y-%m-%dT%H:%M:%S%Z}t",
  "http": {
    "request_time_ms": %D,
    "method": "%m",
    "url": "%{json.escape(req.url)}V",
    "useragent": "%{User-Agent}i",
    "referer": "%{Referer}i",
    "protocol": "%H",
    "request_x_forwarded_for": "%{X-Forwarded-For}i",
    "status_code": "%s"
  },
  "network": {
    "client": {
      "ip": "%h",
      "name": "%{client.as.name}V",
      "number": "%{client.as.number}V",
      "connection_speed": "%{client.geo.conn_speed}V"
    },
    "destination": {
      "ip": "%A"
    },
  "geoip": {
  "geo_city": "%{client.geo.city.utf8}V",
  "geo_country_code": "%{client.geo.country_code}V",
  "geo_continent_code": "%{client.geo.continent_code}V",
  "geo_region": "%{client.geo.region}V"
  },
  "bytes_written": %B,
  "bytes_read": %{req.body_bytes_read}V
  },
  "host":"%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "origin_host": "%v",
  "is_ipv6": %{if(req.is_ipv6, "true", "false")}V,
  "is_tls": %{if(req.is_ssl, "true", "false")}V,
  "tls_client_protocol": "%{json.escape(tls.client.protocol)}V",
  "tls_client_servername": "%{json.escape(tls.client.servername)}V",
  "tls_client_cipher": "%{json.escape(tls.client.cipher)}V",
  "tls_client_cipher_sha": "%{json.escape(tls.client.ciphers_sha)}V",
  "tls_client_tlsexts_sha": "%{json.escape(tls.client.tlsexts_sha)}V",
  "is_h2": %{if(fastly_info.is_h2, "true", "false")}V,
  "is_h2_push": %{if(fastly_info.h2.is_push, "true", "false")}V,
  "h2_stream_id": "%{fastly_info.h2.stream_id}V",
  "request_accept_content": "%{Accept}i",
  "request_accept_language": "%{Accept-Language}i",
  "request_accept_encoding": "%{Accept-Encoding}i",
  "request_accept_charset": "%{Accept-Charset}i",
  "request_connection": "%{Connection}i",
  "request_dnt": "%{DNT}i",
  "request_forwarded": "%{Forwarded}i",
  "request_via": "%{Via}i",
  "request_cache_control": "%{Cache-Control}i",
  "request_x_requested_with": "%{X-Requested-With}i",
  "request_x_att_device_id": "%{X-ATT-Device-Id}i",
  "content_type": "%{Content-Type}o",
  "is_cacheable": %{if(fastly_info.state~"^(HIT|MISS)$", "true","false")}V,
  "response_age": "%{Age}o",
  "response_cache_control": "%{Cache-Control}o",
  "response_expires": "%{Expires}o",
  "response_last_modified": "%{Last-Modified}o",
  "response_tsv": "%{TSV}o",
  "server_datacenter": "%{server.datacenter}V",
  "req_header_size": %{req.header_bytes_read}V,
  "resp_header_size": %{resp.header_bytes_written}V,
  "socket_cwnd": %{client.socket.cwnd}V,
  "socket_nexthop": "%{client.socket.nexthop}V",
  "socket_tcpi_rcv_mss": %{client.socket.tcpi_rcv_mss}V,
  "socket_tcpi_snd_mss": %{client.socket.tcpi_snd_mss}V,
  "socket_tcpi_rtt": %{client.socket.tcpi_rtt}V,
  "socket_tcpi_rttvar": %{client.socket.tcpi_rttvar}V,
  "socket_tcpi_rcv_rtt": %{client.socket.tcpi_rcv_rtt}V,
  "socket_tcpi_rcv_space": %{client.socket.tcpi_rcv_space}V,
  "socket_tcpi_last_data_sent": %{client.socket.tcpi_last_data_sent}V,
  "socket_tcpi_total_retrans": %{client.socket.tcpi_total_retrans}V,
  "socket_tcpi_delta_retrans": %{client.socket.tcpi_delta_retrans}V,
```

```
80    "socket_ploss": %{client.socket.ploss}V
81  }
```

Log streaming: DigitalOcean Spaces

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-digitalocean-spaces

Fastly's Real-Time Log Streaming feature can send log files to DigitalOcean Spaces. DigitalOcean Spaces is an Amazon S3-compatible static file storage service used by developers and IT teams.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding DigitalOcean Spaces as a logging endpoint for Fastly services, you'll need to create a DigitalOcean account if you don't already have one. Then you'll need to create a space with private access permissions on DigitalOcean's website, generate a secret key and an access key, and make a note of the endpoint.

Adding DigitalOcean Spaces as a logging endpoint

After you've created a DigitalOcean Space, follow these instructions to add DigitalOcean Spaces as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Spaces by DigitalOcean **Create endpoint** button. The Create a DigitalOcean endpoint page appears.
3. Fill out the **Create a DigitalOcean endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Space name** field, enter the name of the DigitalOcean Space in which to store the logs.
   - In the **Access key** field, enter the access key associated with the DigitalOcean Space. See the DigitalOcean Spaces Authentication Guide for more information.
   - In the **Secret key** field, enter the secret key associated with the DigitalOcean Space.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
4. Click the **Advanced options** link of the **Create a DigitalOcean endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a DigitalOcean endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **Domain** field, enter the region-specific endpoint for your domain. In most cases, this should be `nyc3.digitaloceanspaces.com`. If the DigitalOcean Space was not created in the `nyc3` region, refer to DigitalOcean's documentation to find the correct domain.

- In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
- In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
- In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.

6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to DigitalOcean. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Elasticsearch

Last updated: 2022-03-18

https://docs.fastly.com/en/guides/log-streaming-elasticsearch

Fastly's Real-Time Log Streaming feature can send log files to Elasticsearch. Elasticsearch is a distributed, RESTful search and analytics engine.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Elasticsearch as a logging endpoint for Fastly services, ensure Elasticsearch is running on a remote server. You'll need to know the endpoint URL that includes a port to which logs should be sent (make sure it can receive traffic from Fastly) and also the name of the index to send logs to. For more information on setting up Elasticsearch, see the Elasticsearch setup documentation.

This logging endpoint works with all actively supported versions of Elasticsearch as well as some versions that have already reached their end-of-life. We also work with OpenSearch server integration. Other distributions that are API-compatible with Elasticsearch may also work but have not been explicitly tested and are not guaranteed.

Required privileges

We send data using the Bulk API via the `index` action. When using basic authentication, ensure that the required index privileges to use the `index` action are granted to the user role.

We also require access to the root path API of the Elasticsearch server. This API returns metadata about the server, such as the version number, that allows our integration to make the best choice about which bulk data API to use for each customer's server. Access to this API allows us to properly work with the wide range of Elasticsearch versions used by our customers as well as other Elasticsearch-compatible distributions.

Adding Elasticsearch as a logging endpoint

Follow these instructions to add Elasticsearch as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Elasticsearch **Create endpoint** button. The Create an Elasticsearch endpoint page appears.
3. Fill out the **Create an Elasticsearch endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, enter the data to send to Elasticsearch. See the example format section for details.
   - In the **URL** field, enter the Elasticsearch endpoint URL that includes a port to which logs should be sent. The URL must be sent using HTTPS on a port that can receive incoming TCP traffic from Fastly.
   - In the **Index** field, enter the name of the Elasticsearch index to send logs to. The index must follow the Elasticsearch index format rules. We support strftime interpolated variables inside braces prefixed with a pound symbol. For example, `#{%F}` will interpolate as YYYY-MM-DD with today's date.
   - In the **Pipeline** field, optionally enter the ID of the Elasticsearch ingest pipeline to apply pre-process transformations to before indexing (for example, `my_pipeline_id`).
   - In the **Maximum logs** field, optionally enter the maximum number of logs to append to a batch, if non-zero.
   - In the **Maximum bytes** field, optionally enter the maximum size of the log batch.
   - In the **BasicAuth user** field, optionally enter your basic authentication username.
   - In the **BasicAuth password** field, optionally enter your basic authentication password.
   - In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported.
   - In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority.
   - In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection.
   - In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

Data sent to Elasticsearch must be serialized as a JSON object. Here's an example format string for sending data to Elasticsearch:

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: FTP

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-ftp

Fastly's Real-Time Log Streaming feature can send log files to password-protected and anonymous FTP servers.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding FTP as a logging endpoint

Follow these instructions to add FTP as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the FTP **Create endpoint** button. The Create a File Transfer Protocol (FTP) endpoint page appears.
3. Fill out the **Create a File Transfer Protocol (FTP) endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Address** field, enter the hostname or IP address of the FTP server. In the port field, enter the port number you're using for FTP (the default is `21`).
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **User** field, enter the username used to authenticate to the FTP server. For anonymous access, use the username `anonymous`.
   - In the **Password** field, enter the password used to authenticate to the FTP server. For anonymous access, use an email address as the password.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.

- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a File Transfer Protocol (FTP) endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create a File Transfer Protocol (FTP) endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to an FTP logging endpoint. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Google BigQuery

Last updated: 2021-10-04

https://docs.fastly.com/en/guides/log-streaming-google-bigquery

Fastly's Real-Time Log Streaming feature can send log files to BigQuery, Google's managed enterprise data warehouse.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

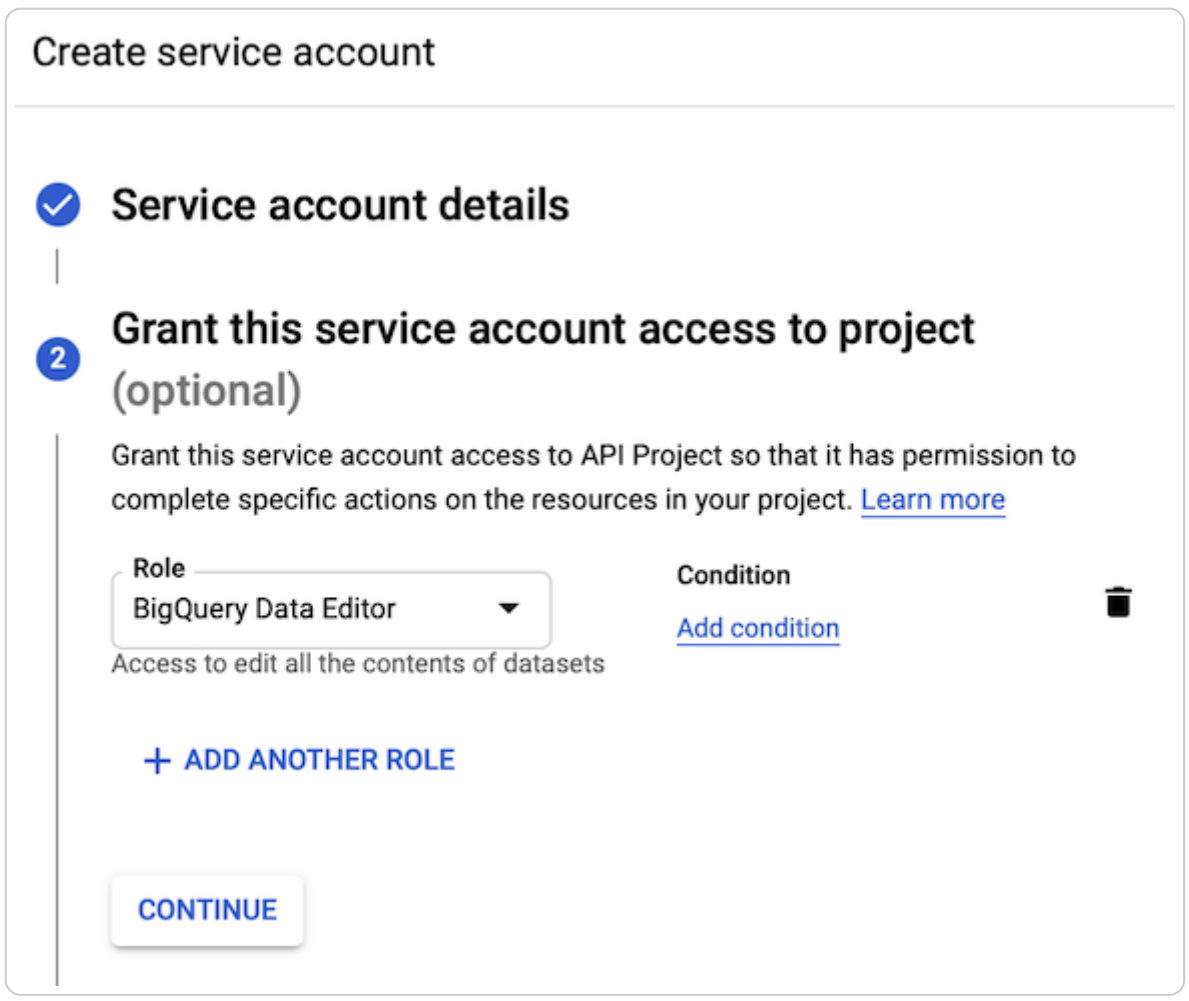
Before adding BigQuery as a logging endpoint for Fastly services, you will need to:

- Register for a Google Cloud Platform (GCP) account.
- Create a service account on Google's website.
- Obtain the `private_key` and `client_email` from the JSON file associated with the service account.
- Enable the BigQuery API.
- Create a BigQuery dataset.
- Add a BigQuery table.

Creating a service account

BigQuery uses service accounts for third-party application authentication. To create a new service account, follow the instructions in the Google Cloud documentation. Keep the following in mind when creating the service account:

- The service account must be assigned the **Big Query Data Editor** role to write to the table you use for Fastly logging. See BigQuery Roles for details about the default permissions assigned to the Big Query Data Editor role.



- Set the key type to **JSON** when creating the service's private key pair.

Obtaining the private key and client email

When you create the BigQuery service account, a JSON file automatically downloads to your computer. This file contains the credentials for your BigQuery service account. Open the file and make a note of the values of the `private_key` and `client_email` fields.

Enabling the BigQuery API

To send your Fastly logs to your BigQuery table, you'll need to enable the BigQuery API in the Google Cloud Platform API Manager.

Creating the BigQuery dataset

After you've enabled the BigQuery API, follow these instructions to create a BigQuery dataset:

1. Open the BigQuery page in the Cloud Console.
2. In the **Explorer** panel, select the project where you want to create the dataset.
3. In the details panel, click **Create dataset**.
4. In the **Dataset ID** field, enter a name for the dataset (e.g., `fastly_bigquery`).
5. Click the **Create dataset** button.

Adding a BigQuery table

After you've created the BigQuery dataset, you'll need to add a BigQuery table. There are four ways of creating the schema for the table:

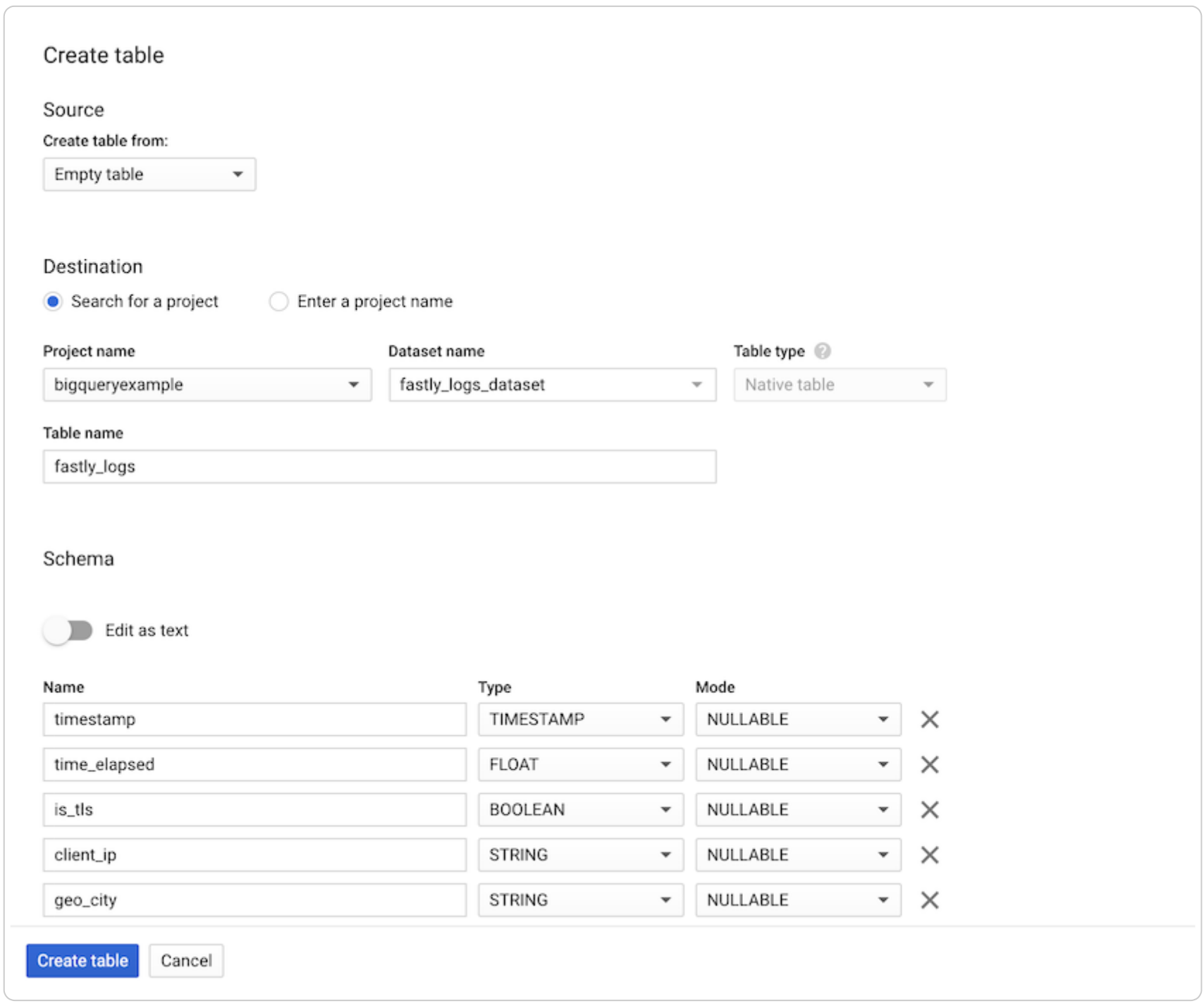
- Edit the schema using the BigQuery web interface.
- Edit the schema using the text field in the BigQuery web interface.

- Use an existing table.
- Set the table to [automatically detect the schema](#).

> **NOTE**
>
> Setting the table to automatically detect the schema may give unpredictable results.

Follow these instructions to add a BigQuery table:

1. Open the [BigQuery page](#) in the Cloud Console.
2. In the Explorer panel, expand your project and select the BigQuery dataset you created [previously](#).
3. In the **Source** section, select **Empty Table** from the **Create table from:** menu. The **Create table** dialog appears.



4. In the **Table name** field, enter a name for the table (e.g., `logs`).
5. In the **Schema** section of the BigQuery website, use the interface to add fields and complete the schema. See the [example schema section](#) for details.
6. Click the **Create Table** button.

Adding BigQuery as a logging endpoint

Follow these instructions to add BigQuery as a logging endpoint:

1. Review the information in our [Setting Up Remote Log Streaming](#) guide.
2. Click the Google BigQuery **Create endpoint** button. The Create a BigQuery endpoint page appears.
3. Fill out the **Create a BigQuery endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.

- In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
- In the **Log format** field, enter the data to send to BigQuery. See the example format section for details.
- In the **Email** field, enter the `client_email` address associated with the BigQuery service account.
- In the **Secret key** field, enter the value of the `private_key` associated with your BigQuery service account.
- In the **Project ID** field, enter the ID of your Google Cloud Platform project.
- In the **Dataset** field, enter the name of your BigQuery dataset.
- In the **Table** field, enter the name of your BigQuery table.
- In the **Template** field, optionally enter an `strftime` compatible string to use as the template suffix for your table.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

Data sent to BigQuery must be serialized as a JSON object, and every field in the JSON object must map to a string in your table's schema. The JSON can have nested data in it (e.g., the value of a key in your object can be another object). Here's an example format string for sending data to BigQuery:

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Example schema

The BigQuery schema for the example format shown above would look something like this:

```
timestamp:TIMESTAMP,client_ip:STRING,geo_country:STRING,geo_city:STRING,host:STRING,url:STRING,request_method:STRING,request_protocol:STRING,request_referer:STRING,request_user_agent:STRING,response_state:STRING,response_status:STRING,response_reason:STRING,response_body_size:STRING,fastly_server:STRING,fastly_is_edge:BOOLEAN
```

Log streaming: Google Cloud Pub/Sub

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-google-cloud-pubsub

Fastly's Real-Time Log Streaming feature can send log files to Cloud Pub/Sub, Google's global messaging and event data ingestion product.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Cloud Pub/Sub as a logging endpoint for Fastly services, you will need to register for a Google Cloud Platform (GCP) account and then:

- Create a service account on Google's website.
- Navigate to the Pub/Sub section of the Google Cloud console. Follow the prompts to enable the API.
- Create a Pub/Sub topic.
- Obtain the private key from the JSON file associated with the service account configured for your Pub/Sub topic.

> **NOTE**
>
> Read more about Cloud Pub/Sub in Google's documentation.

Adding Cloud Pub/Sub as a logging endpoint

Follow these instructions to add Cloud Pub/Sub as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Google Cloud Pub/Sub **Create endpoint** button. The Create a Google Cloud Pub/Sub endpoint page appears.
3. Fill out the **Create a Google Cloud Pub/Sub endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, enter the data to send to Google Cloud Pub/Sub. See the example format section for details.
   - In the **Project ID** field, enter the ID of your Google Cloud Platform project.
   - In the **Email** field, enter the email address of the service account configured for your Pub/Sub topic.
   - In the **Topic** field, enter the Pub/Sub topic to which logs should be sent.
   - In the **Secret Key** field, enter the exact value of the private key associated with the service account configured for your Pub/Sub topic.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

Data sent to Cloud Pub/Sub must be serialized as a JSON object, and every field in the JSON object must map to a string in your table's schema. The JSON can have nested data in it (e.g., the value of a key in your object can be another object). Here's an example format string for sending data:

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Google Cloud Storage

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-google-cloud-storage

Fastly's Real-Time Log Streaming feature can send log files to Google Cloud Storage (GCS). GCS is an online file storage service used for storing and accessing data on Google's infrastructure. One advantage of using GCS is that you can use Google BigQuery to analyze the log files.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding GCS as a logging endpoint for Fastly services, you will need to:

- Register for a GCS account.
- Create a bucket and service account on Google's website.
- Obtain the `private_key` and `client_email` from the JSON file associated with the service account.
- Enable the Google Cloud Storage JSON API.

Creating a GCS bucket

You can create a new GCS bucket to hold the logs, or you can use an existing bucket. Be sure to note the name of the bucket as you will need it later. To learn how to create a GCS bucket, see Google's guide on creating a bucket.

Creating a service account

GCS uses service accounts for third-party application authentication. You will need to create a new service account on Google's website with the role of `Storage Object Creator` and make sure you've added it as a member of the GCS bucket you created. To learn how to create a service account, see Google's guide on generating a service account credential. When you create the service account, be sure to set the **Key Type** to `JSON`.

Obtaining the private key and client email

After you create the service account, a JSON file will be downloaded to your computer. This file contains the credentials for the GCS service account you just created. Open the file with a text editor and make a note of the `private_key` and `client_email`.

Enabling the Google Cloud Storage JSON API

To ensure the Fastly logs are sent to your GCS bucket, you need to enable the Google Cloud Storage JSON API. For more information, see Google's instructions for activating the API.

Adding GCS as a logging endpoint

Follow these instructions to add GCS as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Google Cloud Services **Create endpoint** button. The Create a Google Cloud Storage (GCS) endpoint page appears.
3. Fill out the **Create a Google Cloud Storage (GCS) endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.

- In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
- In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
- In the **Email** field, enter the `client_email` address listed in the JSON file associated with the service account you created on Google's website.
- In the **Bucket name** field, enter the name of the GCS bucket in which to store the logs.
- In the **Secret key** field, enter the `private_key` value listed in the JSON file associated with the service account you created on Google's website. We strip out the JSON newline escape characters for you so don't worry about removing them.
- In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a Google Cloud Storage (GCS) endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Google Cloud Storage (GCS) endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to GCS. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Honeycomb

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-honeycomb

Fastly's Real-Time Log Streaming feature can send logs in JSON format to Honeycomb. Honeycomb is a tool that allows developers to explore the operations of complex systems, microservices, and databases.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Honeycomb as a logging endpoint for Fastly services, you'll need to perform the following steps:

- Sign up for a Honeycomb account if you don't already have one.
- Obtain the Write Key for your team on the Honeycomb Account page.
- Choose a Dataset name. If you plan to collect data from multiple environments (like production, development, staging), Honeycomb recommends creating a Dataset for each environment and naming your Datasets accordingly (e.g., `prod.queries`, `dev.queries`, and `staging.queries`). If a Dataset doesn't exist, Honeycomb will create one automatically.

Adding Honeycomb as a logging endpoint

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Honeycomb **Create endpoint** button. The Create a Honeycomb endpoint page appears.
3. Fill out the **Create a Honeycomb endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, enter the data to send to Honeycomb. See the example format section for details.
   - In the **Write Key** field, enter the write key for your Honeycomb team. This is available on the Honeycomb Account page.
   - In the **Dataset** field, enter the name of the Honeycomb Dataset (e.g., `myDataset`).
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

Data sent to Honeycomb must be serialized as a JSON object. Here's an example format string for sending data to Honeycomb:

```
{
  "time":"%{begin:%Y-%m-%dT%H:%M:%SZ}t",
  "data":  {
    "service_id":"%{req.service_id}V",
    "time_elapsed":%D,
    "request":"%m",
    "host":"%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
    "url":"%{cstr_escape(req.url)}V",
    "protocol":"%H",
    "is_ipv6":%{if(req.is_ipv6, "true", "false")}V,
    "is_tls":%{if(req.is_ssl, "true", "false")}V,
    "is_h2":%{if(fastly_info.is_h2, "true", "false")}V,
    "client_ip":"%h",
    "geo_city":"%{client.geo.city.utf8}V",
    "geo_country_code":"%{client.geo.country_code}V",
    "server_datacenter":"%{server.datacenter}V",
    "request_referer":"%{Referer}i",
    "request_user_agent":"%{User-Agent}i",
    "request_accept_content":"%{Accept}i",
    "request_accept_language":"%{Accept-Language}i",
    "request_accept_charset":"%{Accept-Charset}i",
    "cache_status":"%{regsub(fastly_info.state, "^(HIT-(SYNTH)|(HITPASS|HIT|MISS|PASS|ERROR|PIPE)).*", "\2\3") }V",
    "status":"%s",
    "content_type":"%{Content-Type}o",
    "req_header_size":%{req.header_bytes_read}V,
    "req_body_size":%{req.body_bytes_read}V,
    "resp_header_size":%{resp.header_bytes_written}V,
    "resp_body_size":%{resp.body_bytes_written}V
  }
}
```

Log streaming: HTTPS

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-https

Fastly's Real-Time Log Streaming feature can send log files to an HTTPS endpoint.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

When sending logs to a HTTPS endpoint, Fastly requires proof that you control the domain name specified in the **URL** field by using a HTTP challenge on a well-known path. If, for example, your **URL** field is `foo.example.com/some/log/path`, then the following challenge path must send a 200 response:

`foo.example.com/.well-known/fastly/logging/challenge`

Responses must include the hex representation of the SHA-256 of your Fastly service ID and it must appear on its own line in the response. For example:

```
$ sha256sum <SERVICEID>

ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c
```

If multiple service IDs are used, multiple hex(sha256) lines can be added to that challenge body. In addition, an asterisk (`*`) can be used on a line to allow any service to post to the HTTP endpoint. For example:

```
ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c
06ae6402e02a9dad74edc71aa69c77c5747e553b0840bfc56feb7e65b23f0f61
*
```

Adding HTTPS as a logging endpoint

Follow these instructions to add HTTPS as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

2. Click the HTTPS **Create endpoint** button. The Create an HTTPS endpoint page appears.

3. Fill out the **Create an HTTPS endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **URL** field, enter the URL to which log data will be sent (e.g., `https://logs.example.com/`).
   - In the **Maximum logs** field, optionally enter the maximum number of logs to send as a batch.
   - In the **Maximum bytes** field, optionally enter the maximum size of a log batch.

4. Click the **Advanced options** link of the **Create an HTTPS endpoint** page. The Advanced options appear.

5. Fill out the **Advanced options** of the **Create an HTTPS endpoint** page as follows:
   - In the **Content type** field, optionally enter the content type to use when sending logs (e.g., `application/json`).
   - In the **Custom header name** field, optionally enter a custom header to use when sending logs (e.g., `Authorization`).
   - In the **Custom header value** field, optionally enter a custom header value to use when sending logs (e.g., `Bearer <token>`).
   - In the **Method** area, optionally select the appropriate HTTP method to use.
   - In the **JSON log entry format** area, select the appropriate log entry format to use. The JSON log entry format enforces valid JSON formatting. Selecting **Array of JSON** wraps JSON log batches in an array. Selecting **Newline delimited** places each JSON log entry onto a new line in a batch.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.

6. Fill out the **Using your own certificate authority (CA)** section of the **Advanced options** area as follows:
   - In the **TLS Hostname** field, optionally enter the hostname used to verify the server's certificate. This can be either the Common Name (CN) or Subject Alternative Name (SAN). If the hostname is not specified, the hostname of the first broker in the Brokers field will be used. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.

7. Click the **Create** button to create the new logging endpoint.

8. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to an HTTPS logging endpoint. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Firewall considerations

Your HTTPS endpoint may have limited security features. For this reason, it's best to create a firewall for your HTTP endpoint server and only accept TCP traffic on your configured port from our address blocks. Our list of IP address blocks is dynamic, so we recommend programmatically obtaining the list whenever possible.

Log streaming: Hydrolix

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-hydrolix

Fastly's Real-Time Log Streaming feature can send log files to Hydrolix, a cloud-based time-series data platform. Hydrolix provides a native integration for Fastly log storage and analysis through Fastly's HTTPS logging endpoint. Hydrolix lets you ingest and query those logs in real-time.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

If you don't already have a Hydrolix account, you'll need to sign up on the Hydrolix website. You'll also need to know the following about the target Hydrolix environment:

- Instance name
- Project name
- Table name

TIP

Consider reading Hydrolix's documentation on integrating with Fastly.

Adding Hydrolix as a logging endpoint

Follow these instructions to add Hydrolix as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the HTTPS **Create endpoint** button. The Create an HTTPS endpoint page appears.
3. Fill out the **Create an HTTPS endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.

- In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
- In the **Log format** field, replace the placeholder log format and make the appropriate changes as shown in our log format and recommendations section below.
- In the **URL** field, enter `https://<hydrolix-instance>.hydrolix.live/ingest/event`, replacing `<hydrolix-instance>` with the name of your Hydrolix instance.
- In the **Maximum logs** field, leave as `0` (the default).
- In the **Maximum bytes** field, enter `0`.

4. Click the **Advanced options** link of the **Create an HTTPS endpoint** page. The Advanced options appear.
5. Fill out the **Advanced options** of the **Create an HTTPS endpoint** page as follows:
   - In the **Content type** field, enter `application/json`.
   - In the **Custom header name** field, enter `x-hdx-table`.
   - In the **Custom header value** field, enter `<hydrolix-project-name>.<hydrolix-table-name>`, substituting the values for your Hydrolix project and table names.
   - From the **Method** controls, select **POST**.
   - From the **JSON log entry format** controls, select **Newline delimited**.
   - Leave the **Select a log line format** and **Placement** controls set to the defaults.
   - In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported.
   - Leave **TLS CA certificate** field, **TLS client certificate** field and **TLS client key** field all empty.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log format recommendations

For this example we use the log format shown below. You can customize this format with any values you want as long as you also modify your Transform and View configurations.

```
{
  "timestamp":"%{begin:%Y-%m-%dT%H:%M:%S}t",
  "time_elapsed":%{time.elapsed.usec}V,
  "is_tls":%{if(req.is_ssl, "true", "false")}V,
  "client_ip":"%{req.http.Fastly-Client-IP}V",
  "geo_city":"%{client.geo.city}V",
  "geo_country_code":"%{client.geo.country_code}V",
  "request":"%{req.request}V",
  "host":"%{req.http.Fastly-Orig-Host}V",
  "url":"%{json.escape(req.url)}V",
  "request_referer":"%{json.escape(req.http.Referer)}V",
  "request_user_agent":"%{json.escape(req.http.User-Agent)}V",
  "request_accept_language":"%{json.escape(req.http.Accept-Language)}V",
  "request_accept_charset":"%{json.escape(req.http.Accept-Charset)}V",
  "cache_status":"%{regsub(fastly_info.state, "^(HIT-(SYNTH)|(HITPASS|HIT|MISS|PASS|ERROR|PIPE)).*", "\\2\\3") }V"
}
```

Configuring Hydrolix Streaming Intake

Once you have your `project` and `table` setup and Fastly is configured to send logs to your Hydrolix instance, you can focus on setting up the Fastly Log streaming ingest pipeline by defining an ingest transform schema.

Creating a transform schema

Below is the suggested transform schema to use with the recommended log format described above. Be sure to replace `<table uuid>` with the UUID of your target Hydrolix table. You need to have this transform setup as the default, so `"is_default"` is set to `true`.

```
{
    "name": "fastly_transform",
    "type": "json",
    "table": "<table uuid>",
    "description": "fastly https logs",
    "settings": {
        "is_default": true,
        "output_columns": [
                            {
                                "position": 0,
                                "name": "timestamp",
                                "type": "datetime",
                                "format": "2006-01-02T15:04:05",
                                "treatment": "primary"
                            },
                            {
                                "position": 1,
                                "name": "time_elapsed",
                                "type": "uint64",
                                "treatment": "tag"
                            },
                            {
                                "position": 2,
                                "name": "is_tls",
                                "type": "bool",
                                "treatment": "tag"
                            },
                            {
                                "position": 3,
                                "name": "client_ip",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 4,
                                "name": "geo_city",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 5,
                                "name": "geo_country_code",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 6,
                                "name": "request",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 7,
                                "name": "host",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 8,
                                "name": "url",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 9,
                                "name": "request_referer",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 10,
                                "name": "request_user_agent",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 11,
                                "name": "request_accept_language",
                                "type": "string",
                                "treatment": "tag"
```

```
                            },
                            {
                                "position": 12,
                                "name": "request_accept_charset",
                                "type": "string",
                                "treatment": "tag"
                            },
                            {
                                "position": 13,
                                "name": "cache_status",
                                "type": "string",
                                "treatment": "tag"
                            }
                        ]
        }
    }
```

Once you define the transform schema, Hydrolix is configured to accept the incoming Fastly log data.

Leveraging views

Hydrolix supports having many different query formats for a single dataset. The query data structure, or view, for a given dataset allows you to customized the queried data and restrict a user's access to a specific set of columns.

Hydrolix automatically generates a default view upon transform creation that can be used to immediately query the dataset - no additional configuration is required. However, you are encouraged to familiarize yourself with the view concept and the benefits that the feature can provide. More detailed information can be found on the Hydrolix site.

Further reading

Hydrolix provides a tutorial for querying and analyzing Fastly data from within their system.

Log streaming: Kafka

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-kafka

Fastly's Real-Time Log Streaming feature can send log files to Apache Kafka. Kafka is an open-source, high-throughput, low-latency platform for handling real-time data feeds.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Apache Kafka as a logging endpoint for Fastly services, ensure Kafka is running on a remote server. You'll need to know the hostname or IP address of one or more servers (Brokers) and the category or feed name to which messages will be stored (Topic). For more information on setting up Kafka see the Apache Kafka Quickstart guide.

Adding Kafka as a logging endpoint

Follow these instructions to add Kafka as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Apache Kafka **Create endpoint** button. The Create an Apache Kafka endpoint page appears.
3. Fill out the **Create an Apache Kafka endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.

- In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
- In the **Brokers** field, enter the hostname or IP address of one or more servers (Kafka brokers). Specify multiple servers using a comma-separated string.
- In the **Topic** field, enter the name of the topic to send logs to.
- In the **Maximum bytes** field, optionally enter the maximum size of a log batch.
- From the **Parse key-values** controls, optionally select whether or not to parse any key-value pairs within the log format.
- In the **Write acknowledgement** area, optionally select the appropriate write acknowledgement a leader must receive before a write is considered successful.
- In the **Compression codec** area, optionally select the appropriate codec to use for compression of your logs.
- From the **Use SASL** controls, optionally select whether or not to enable SASL authentication. SASL authentication can be enabled concurrently with TLS encryption. When you select Yes, additional SASL authentication fields appear.
- From the **SASL authentication mechanism** menu, select the appropriate challenge-response mechanism to use for authenticating the SASL client authentication username and password.
- In the **User** field, enter the SASL client authentication username.
- In the **Password** field, enter the SASL client authentication password.
- From the **Use TLS** controls, optionally select whether or not to enable TLS encryption for the Kafka endpoint. TLS encryption can be enabled concurrently with SASL authentication. When you select Yes, additional TLS fields appear.
- In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported. If the hostname is not specified, the hostname of the first broker in the Brokers field will be used. This field only appears when you select Yes from the Use TLS menu.
- In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority. This field only appears when you select Yes from the Use TLS menu.
- In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.
- In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Apache Kafka. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Log Shuttle

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-log-shuttle

Fastly's Real-Time Log Streaming feature can send log files to Log Shuttle. Log Shuttle is an open source application designed to provide simpler encrypted and authenticated log delivery.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding Log Shuttle as a logging endpoint

Follow these instructions to add Log Shuttle as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Log Shuttle **Create endpoint** button. The Create a Log Shuttle endpoint page appears.
3. Fill out the **Create a Log Shuttle endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Token** field, enter the data authentication token. This is required for some endpoints like Heroku's Log Integration.
   - In the **URL** field, enter the URL to which log data will be sent (e.g., `https://logs.example.com/`).
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Log Shuttle. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: LogDNA

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-logdna

Fastly's Real-Time Log Streaming feature can be configured to send logs in a format that is readable by LogDNA. LogDNA is a cloud-based log management system that aggregates system and application logs into a single location.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding LogDNA as a logging endpoint for Fastly services, you'll need to perform the following steps:

- Sign up for a LogDNA account if you don't already have one. You can sign up for a free (but restricted plan) or upgrade a LogDNA plan to include more features.
- Set up a new LogDNA syslog source via the LogDNA web application by following their account-tailored log source instructions. Be sure to make note of the port number displayed at the end of the syslog URL when you complete set up. This is the port number you'll enter when setting up LogDNA as a logging endpoint for Fastly.

Adding LogDNA as a logging endpoint

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the LogDNA (via Syslog) **Create endpoint** button. The Create a Syslog endpoint page appears.
3. Fill out the **Create a Syslog endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Syslog address** field, enter `syslog-a.logdna.com`. In the port field after the colon, enter the LogDNA port number you noted during your LogDNA account setup.
   - From the **TLS** menu, select **Yes** to enable encryption for the syslog endpoint. The TLS Hostname and TLS CA Certificate fields will both appear.
   - In the **TLS Hostname** field, enter `syslog-a.logdna.com`. This is the hostname Fastly will use to verify the syslog server's certificate.

4. Click the **Advanced options** link of the **Create a Syslog endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create a Syslog endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

Logs should begin appearing in your LogDNA account a few seconds after you've created the endpoint and deployed your service.

Example format

The following is an example format string for sending data to LogDNA. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Logentries

Last updated: 2021-11-01

https://docs.fastly.com/en/guides/log-streaming-logentries

Fastly's Real-Time Log Streaming feature can send log files to Logentries. Logentries is a real-time log management and analytics system that you can use to monitor your Fastly logs.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

One-click Logentries account setup

Fastly has partnered with Logentries to offer you a method for automatically creating a Logentries account and configuring a logging endpoint. By using the Logentries one-click integration, you can create a 30 day trial Logentries account with unlimited data. After 30 days, if you don't upgrade to one of the Logentries premium plans, your account will be capped at 5GB per month.

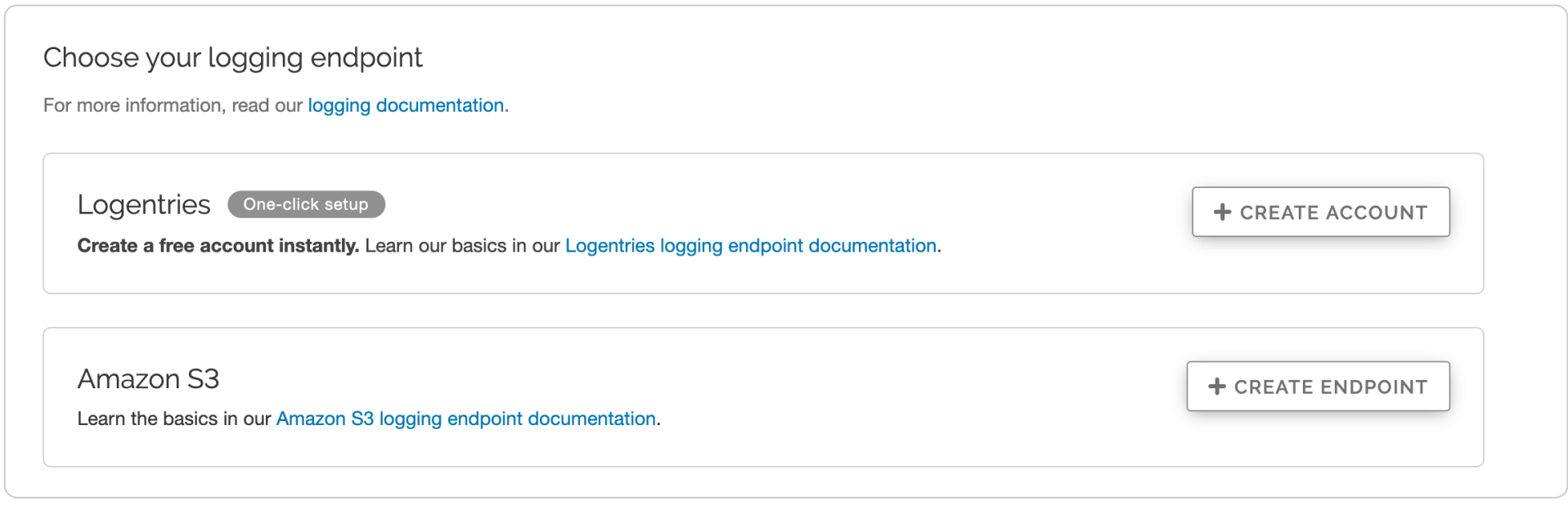
Follow these instructions to create a Logentries logging endpoint and configure the logging endpoint:

1. Log in to the Fastly web interface.

2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.

3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.

4. Click the **Logging** link. The Logging endpoints page appears. If you have an existing logging endpoint, click the **Create endpoint** button.

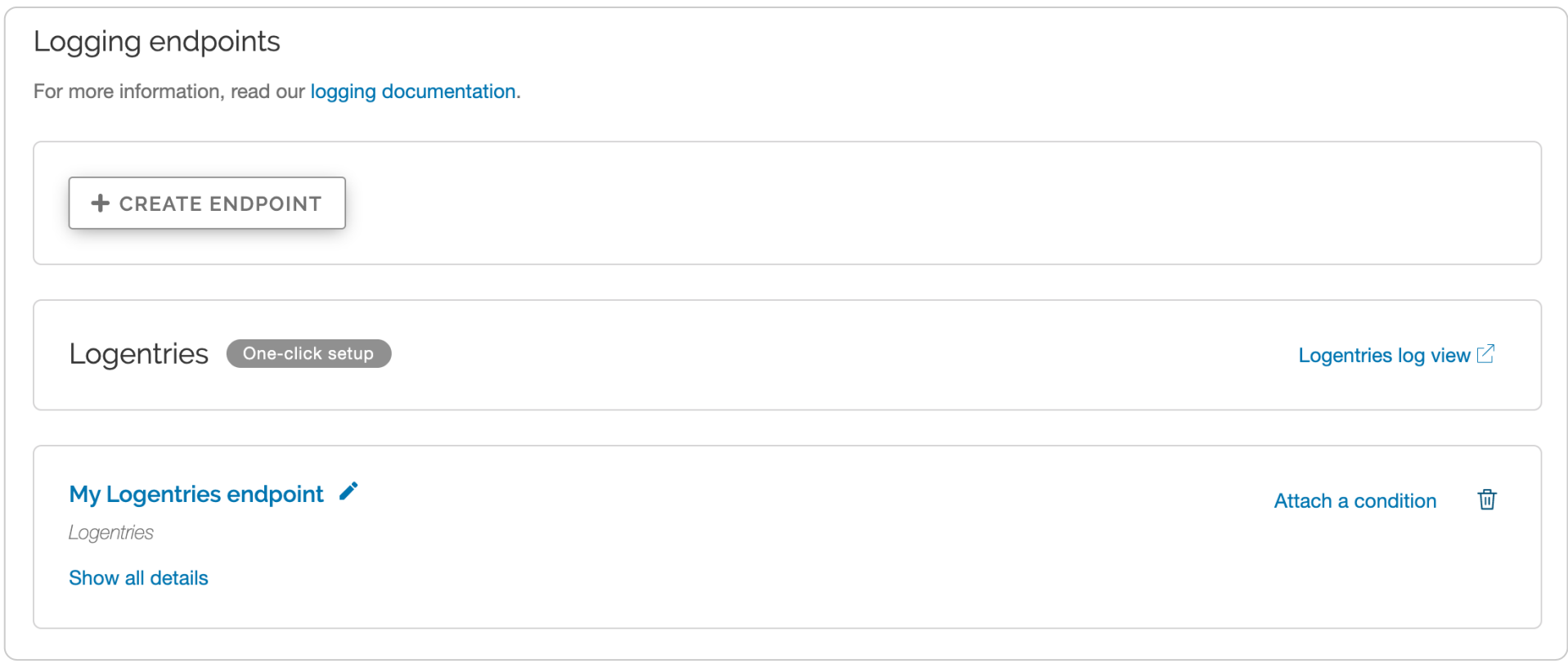


5. In the Logentries One-click setup box, click the **Create Account** button. The Logentries log is automatically created.
6. Click the **Activate** button to deploy your configuration changes.

Accessing your Logentries account

If you created a Logentries account using the one-click integration, you must access your Logentries account from the Fastly web application. Follow these instructions to log in to Logentries:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the Logging link. The Logging endpoints page appears.



5. Click the **Logentries log view** link to access your Logentries account dashboard.

Manually adding Logentries as a logging endpoint

If you already have a Logentries account, or if you'd prefer to sign up for a Logentries account on the Logentries website, you can manually add Logentries as a logging endpoint in the Fastly web interface.

Prerequisites

1. Register for a Logentries account.
2. Create a new log in the Logentries application by following the instructions on the Logentries website.
3. During new log creation, select **Manual Configuration** and **Token TCP**.

4. Make a note of the token provided in the Logentries configuration panel. We recommend you use this token when you create the Logentries logging endpoint for Fastly services.

Creating the logging endpoint in the web interface

After you've created a new log in Logentries and found the token, follow these instructions to add Logentries as a logging endpoint for Fastly services:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Logentries by Rapid7 **Create endpoint** button. The Create a Logentries endpoint page appears.
3. Fill out the **Create a Logentries endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Token** field, enter the token provided in the Logentries configuration panel.
   - From the **Region** menu, select the region to stream logs to. For older Logentries accounts, where the log view URL starts with `https://logentries.com/`, select `EU`. For InsightOps accounts, select the region based on which data storage region you chose on signup for your Rapid7 account. For example, if your log view URL is `https://us2.ops.insight.rapid7.com/`, then your selected region would be `US-2`.
4. Click the **Advanced options** link of the **Create a Logentries endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Logentries endpoint** page as follows:
   - From the **TLS** menu, optionally select **Yes**.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Logentries. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Next steps

Logentries maintains the Fastly Community Pack that leverages custom VCL to provide advanced User-Agent statistics, regional statistics, error tracking, and more.

Log streaming: Loggly

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-loggly

Fastly's Real-Time Log Streaming feature can send log files to Loggly. Loggly is an agent-less log collection and management tool.

NOTE

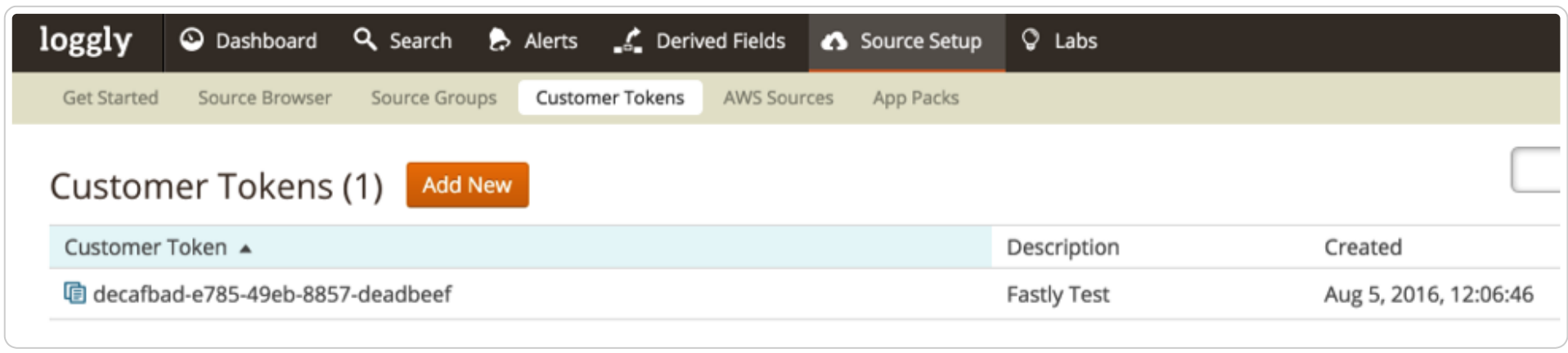
Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

If you don't already have a Loggly account, you'll need to register for one. Follow the signup instructions on the Loggly website.

Follow the steps below to find your Loggly customer token:

1. Navigate to the Customer Tokens area in the **Source Setup** on your Loggly dashboard.



2. Make note of your Loggly customer token. Loggly uses this to associate data you send them with your account.

Adding Loggly as a logging endpoint

After you've created a Loggly account and obtained your customer token, follow these instructions to add Loggly as a logging endpoint for Fastly services:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Loggly **Create endpoint** button. The Create a Loggly endpoint page appears.
3. Fill out the **Create a Loggly endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Token** field, enter your Loggly customer token.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Loggly. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Heroku's Logplex

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-logplex

As part of our Real-Time Log Streaming feature, if you use our Heroku add-on, you can send log files directly to Heroku's Logplex system. Logplex is Heroku's distributed syslog router that collates and distributes log entries from a variety of sources into a single channel.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before continuing, you will need the token from your Heroku Logplex account. If you don't have a Heroku Logplex account, now is the time to set one up by signing up for Heroku.

Once enabled, your Fastly logs will be available in exactly the same way as your regular app and hosted service logs. You can view them using the Heroku command line log viewer or send them to a logging add-on.

Adding Heroku Logplex as a logging endpoint

Follow these instructions to add Heroku Logplex as a logging endpoint for Fastly services:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Heroku Logplex **Create endpoint** button. The Create a Heroku Logplex endpoint page appears.
3. Fill out the **Create a Heroku Logplex endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Token** field, enter your Heroku Logplex token.
   - In the **URL** field, enter `https://1.us.logplex.io/logs` unless otherwise instructed by our support staff.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Logplex. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: New Relic Logs

Last updated: 2021-11-01

https://docs.fastly.com/en/guides/log-streaming-newrelic-logs

Fastly's Real-Time Log Streaming feature can send log files to New Relic Logs.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding New Relic Logs as a logging endpoint for Fastly services, you will need to:

- Register for a New Relic account.
- Obtain your license key or optionally create an Insert API key.

Adding New Relic Logs as a logging endpoint

Follow these instructions to add New Relic Logs as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the New Relic Logs **Create endpoint** button. The Create a New Relic Logs endpoint page appears.
3. Fill out the **Create a New Relic Logs endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **License key / Insert key** field, enter your New Relic license key or Insert API key.
   - From the **Region** controls, select the region to stream logs to.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

Data sent to New Relic Logs must be serialized as a JSON object. Here's an example format string for sending data to New Relic Logs:

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Logging a timestamp

If a timestamp field is present in the Fastly log, it must be specified as millisecond or seconds since Epoch to override the New Relic timestamp. If not included, Fastly will generate a timestamp.

Using New Relic Instant Observability's prebuilt Fastly dashboard

New Relic I/O is an open source ecosystem of community-contributed quickstarts for hundreds of tools and technologies. We've worked with New Relic to develop a prebuilt dashboard that highlights certain key metrics we think are important and useful. Setting up the dashboard is easy and the code is open source in case you want to customize it.

Configure your service to use the recommended expanded logging format

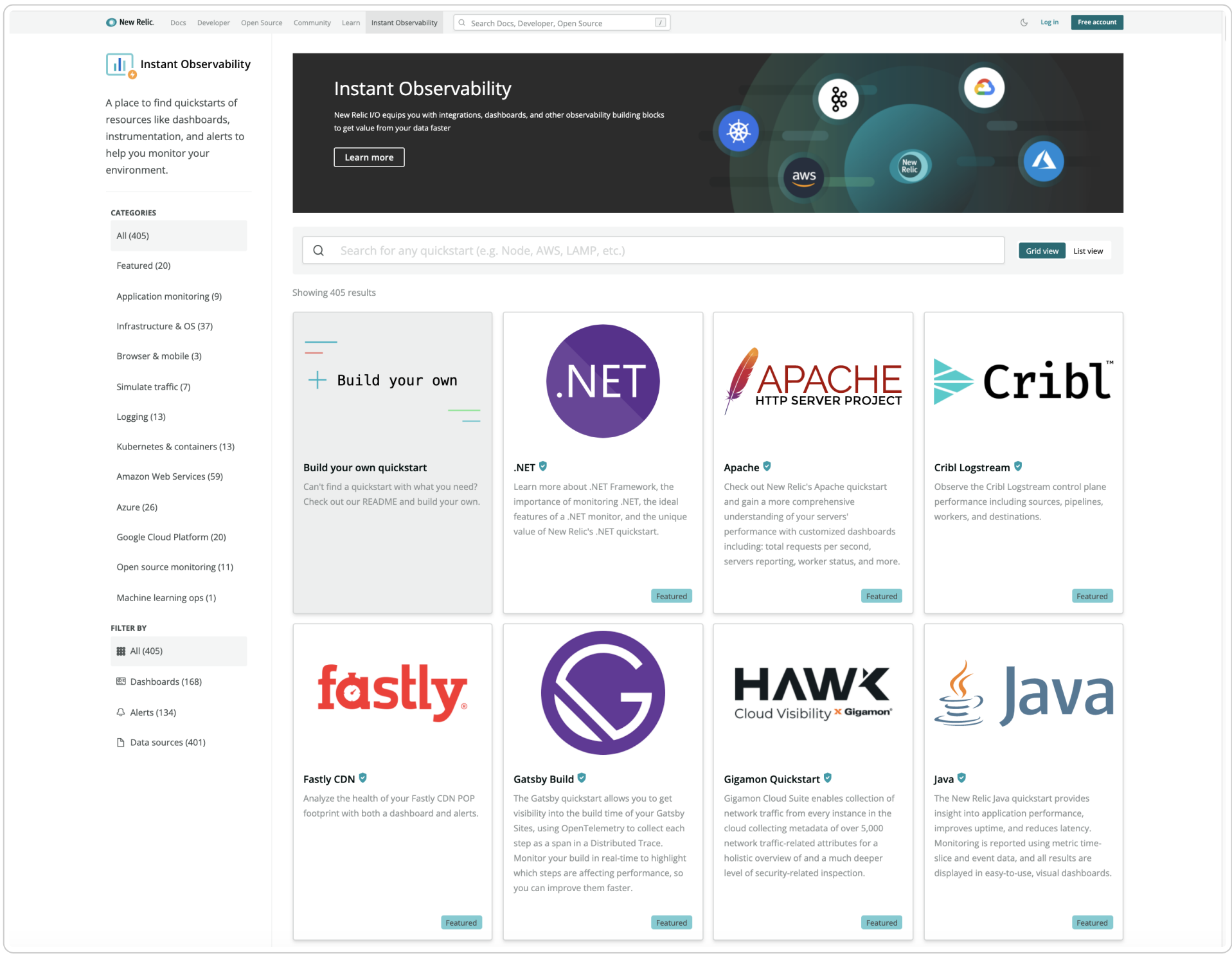
Since the prebuilt dashboard expects certain fields to be present, we recommend using the following logging format. That said, there is nothing to stop you from adding fields for your own purposes or to maintain backward compatibility with existing dashboards you've built. The dashboard won't break if you don't send some fields, but certain charts won't have data.

```
{
  "timestamp": %{time.start.msec}V,
  "logtype": "accesslogs",
  "cache_status":"%{regsub(fastly_info.state, "^(HIT-(SYNTH)|(HITPASS|HIT|MISS|PASS|ERROR|PIPE)).*", "\\2\\3") }V",
  "client_ip":"%h",
  "client_device_type":"%{client.platform.hwtype}V",
  "client_os_name":"%{client.os.name}V",
  "client_os_version":"%{client.os.version}V",
  "client_browser_name":"%{client.browser.name}V",
  "client_browser_version":"%{client.browser.version}V",
  "client_as_name":"%{client.as.name}V",
  "client_as_number":"%{client.as.number}V",
  "client_connection_speed": "%{client.geo.conn_speed}V",
  "client_port": %{client.port}V,
  "client_rate_bps":%{client.socket.tcpi_delivery_rate}V,
  "client_recv_bytes":%{client.socket.tcpi_bytes_received}V,
  "client_requests_count":%{client.requests}V,
  "client_resp_body_size_write": %{resp.body_bytes_written}V,
  "client_resp_header_size_write": %{resp.header_bytes_written}V,
  "client_resp_ttfb": %{time.to_first_byte}V,
  "client_rtt_us":%{client.socket.tcpi_rtt}V,
  "content_type":"%{Content-Type}o",
  "domain": "%{Fastly-Orig-Host}i",
  "fastly_datacenter": "%{server.datacenter}V",
  "fastly_host": "%{server.hostname}V",
  "fastly_region": "%{server.region}V",
  "host": "%v",
  "origin_host":"%v",
  "origin_name":"%{req.backend.name}V",
  "request":"%{req.request}V",
  "request_method":"%m",
  "request_accept_charset":"%{json.escape(req.http.Accept-Charset)}V",
  "request_accept_language":"%{json.escape(req.http.Accept-Language)}V",
  "request_referer":"%{json.escape(req.http.Referer)}V",
  "request_user_agent":"%{json.escape(req.http.User-Agent)}V",
  "resp_status":"%s",
  "response": "%{resp.response}V",
  "service_id":"%{req.service_id}V",
  "service_version": "%{req.vcl.version}V",
  "status":"%s",
  "time_start":"%{begin:%Y-%m-%dT%H:%M:%SZ}t",
  "time_end":"%{end:%Y-%m-%dT%H:%M:%SZ}t",
  "time_elapsed":%D,
  "timestamp": %{time.start.msec}V,
  "tls_cipher":"%{json.escape(tls.client.cipher)}V",
  "tls_version":"%{json.escape(tls.client.protocol)}V",
  "url":"%{json.escape(req.url)}V",
  "user_agent":"%{json.escape(req.http.User-Agent)}V",
  "user_city":"%{client.geo.city.utf8}V",
  "user_country_code":"%{client.geo.country_code}V",
  "user_continent_code":"%{client.geo.continent_code}V",
  "user_region":"%{client.geo.region}V"
}
```

Install the Fastly dashboard

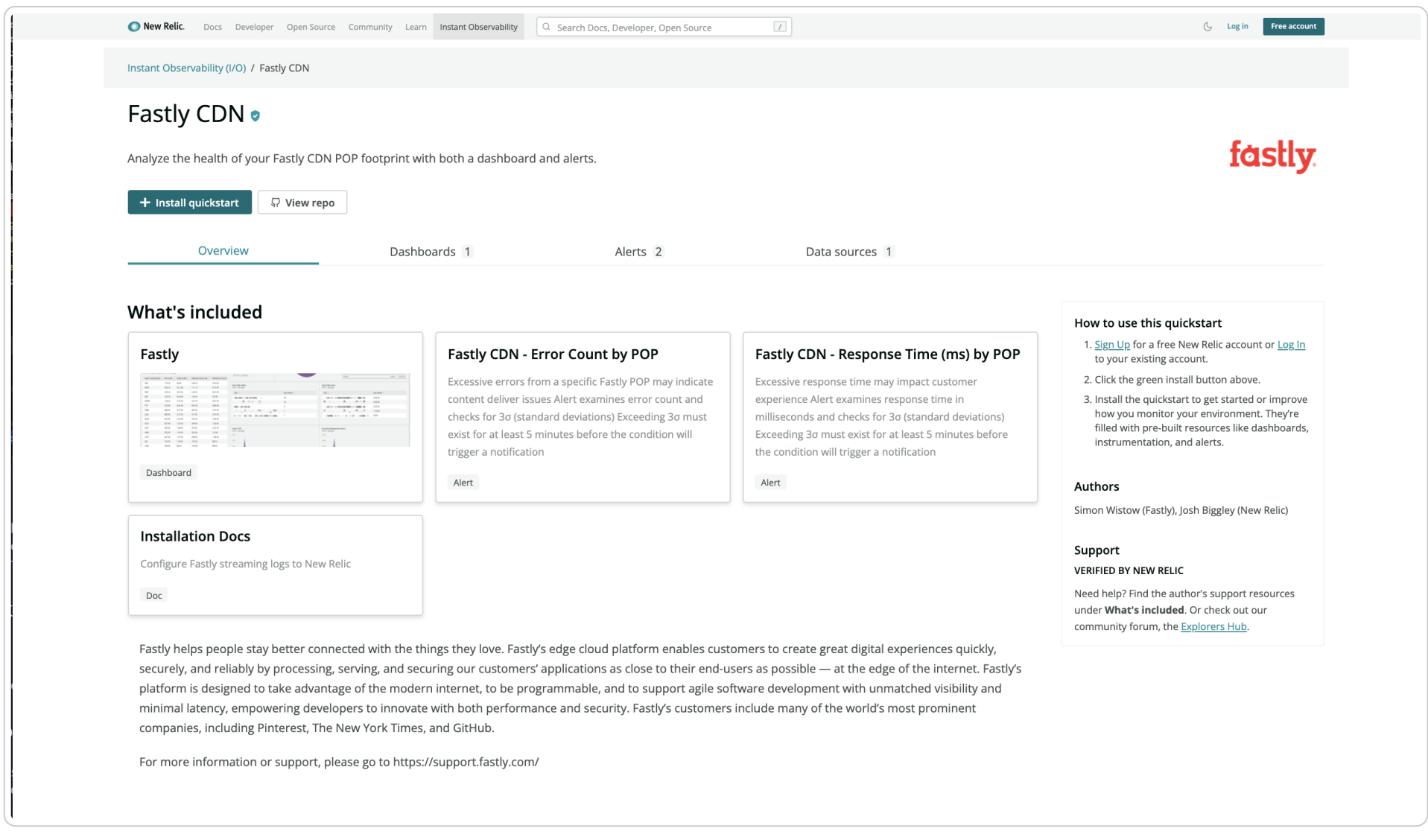
Follow these instructions to install the Fastly dashboard quickstart:

1. Select the Fastly dashboard quickstart from the New Relic marketplace.

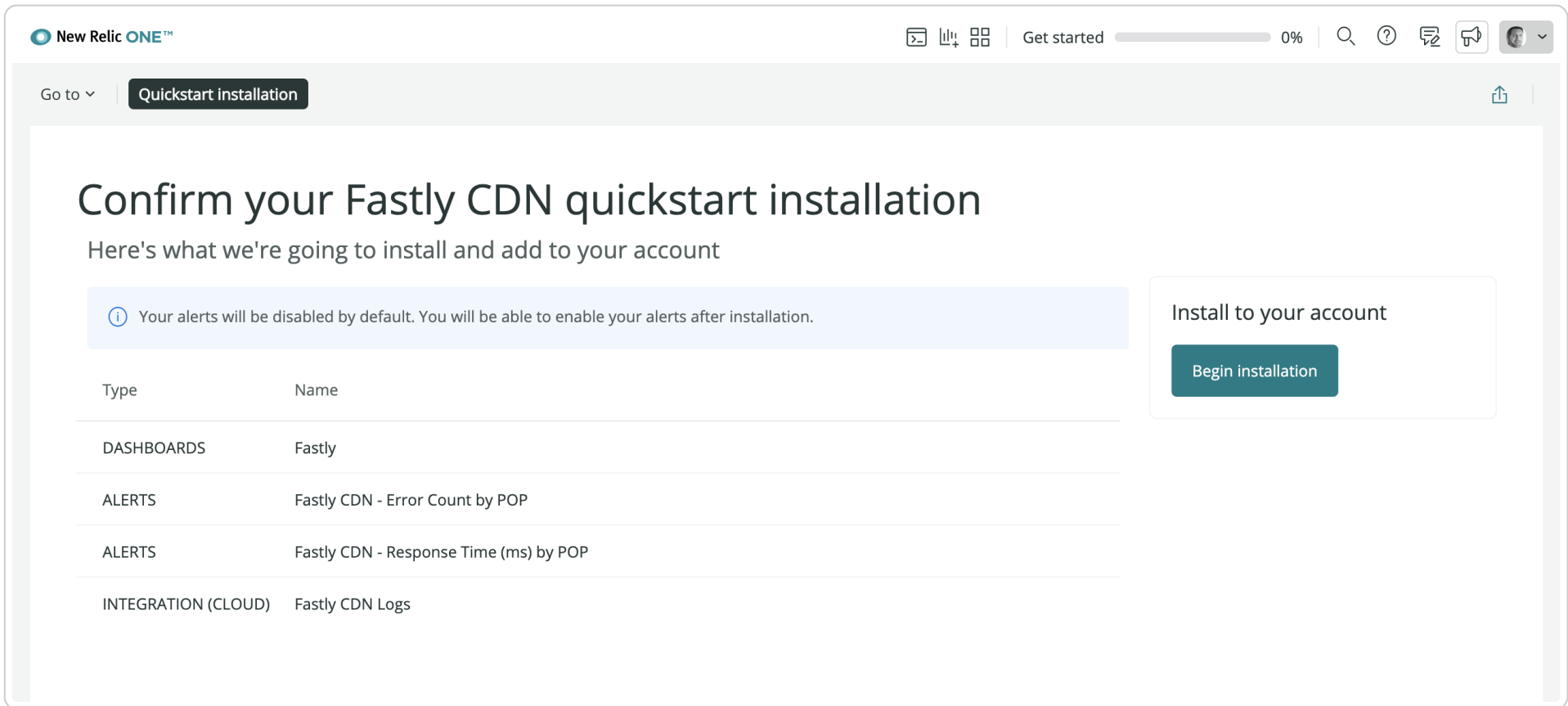


Or navigate directly to the [Fastly dashboard page](#).

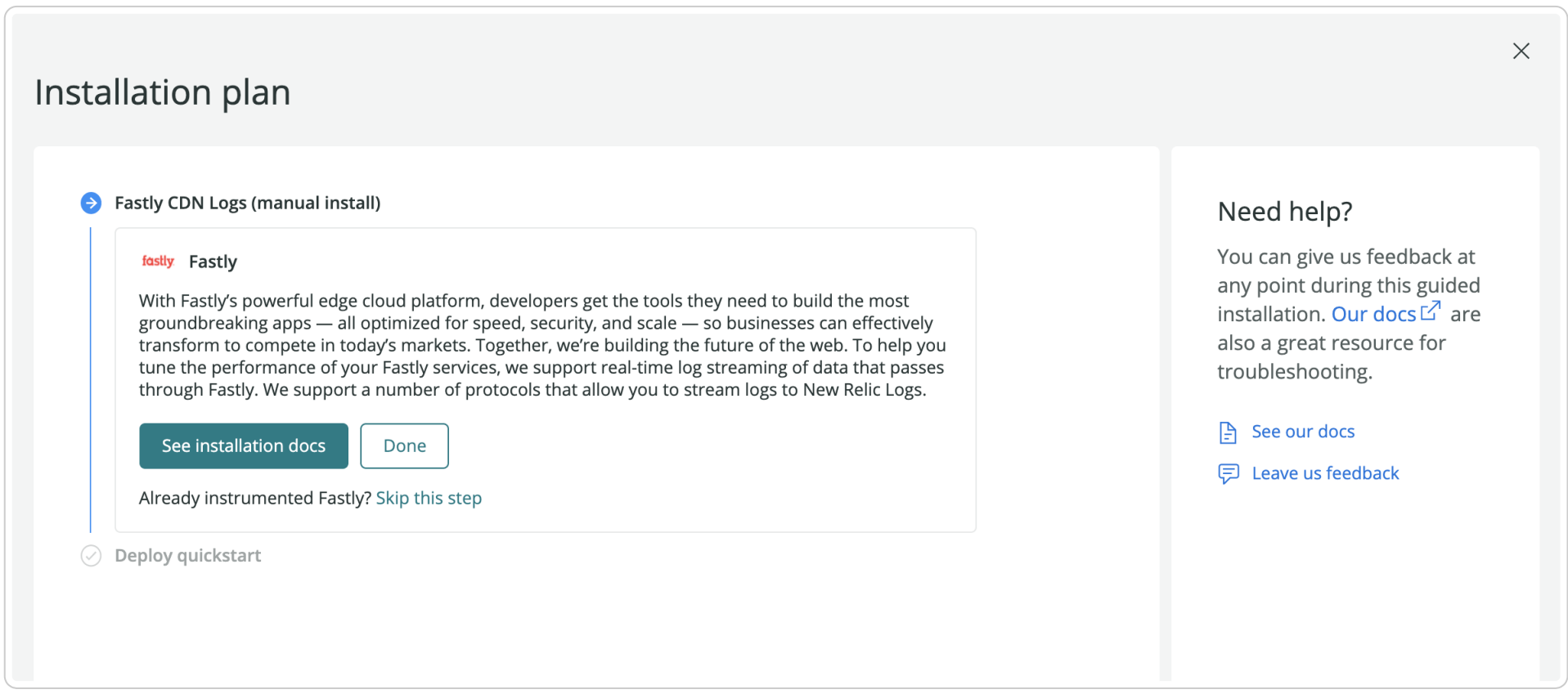
2. Click the **Install quickstart** button.



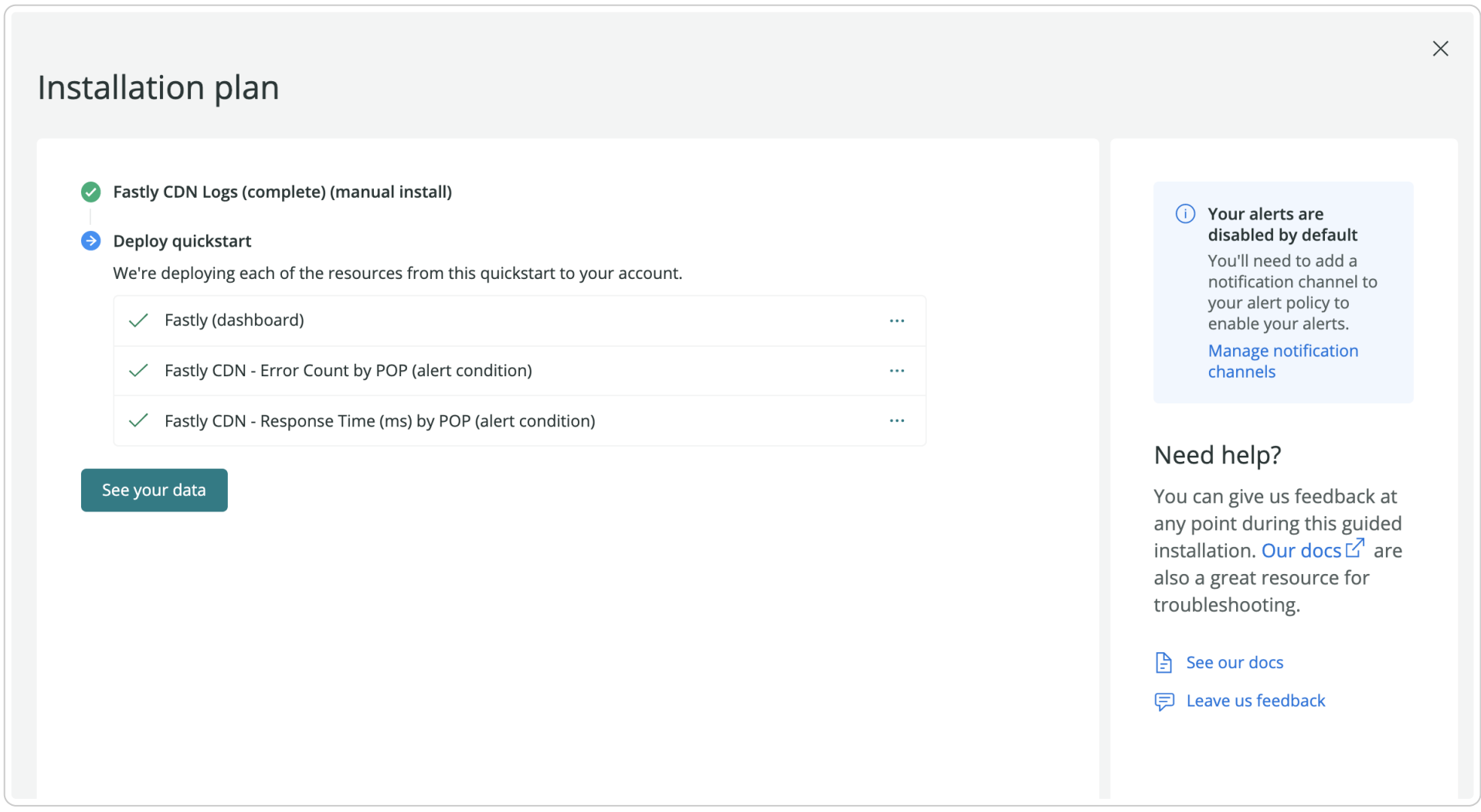
3. Click the **Begin installation** button.



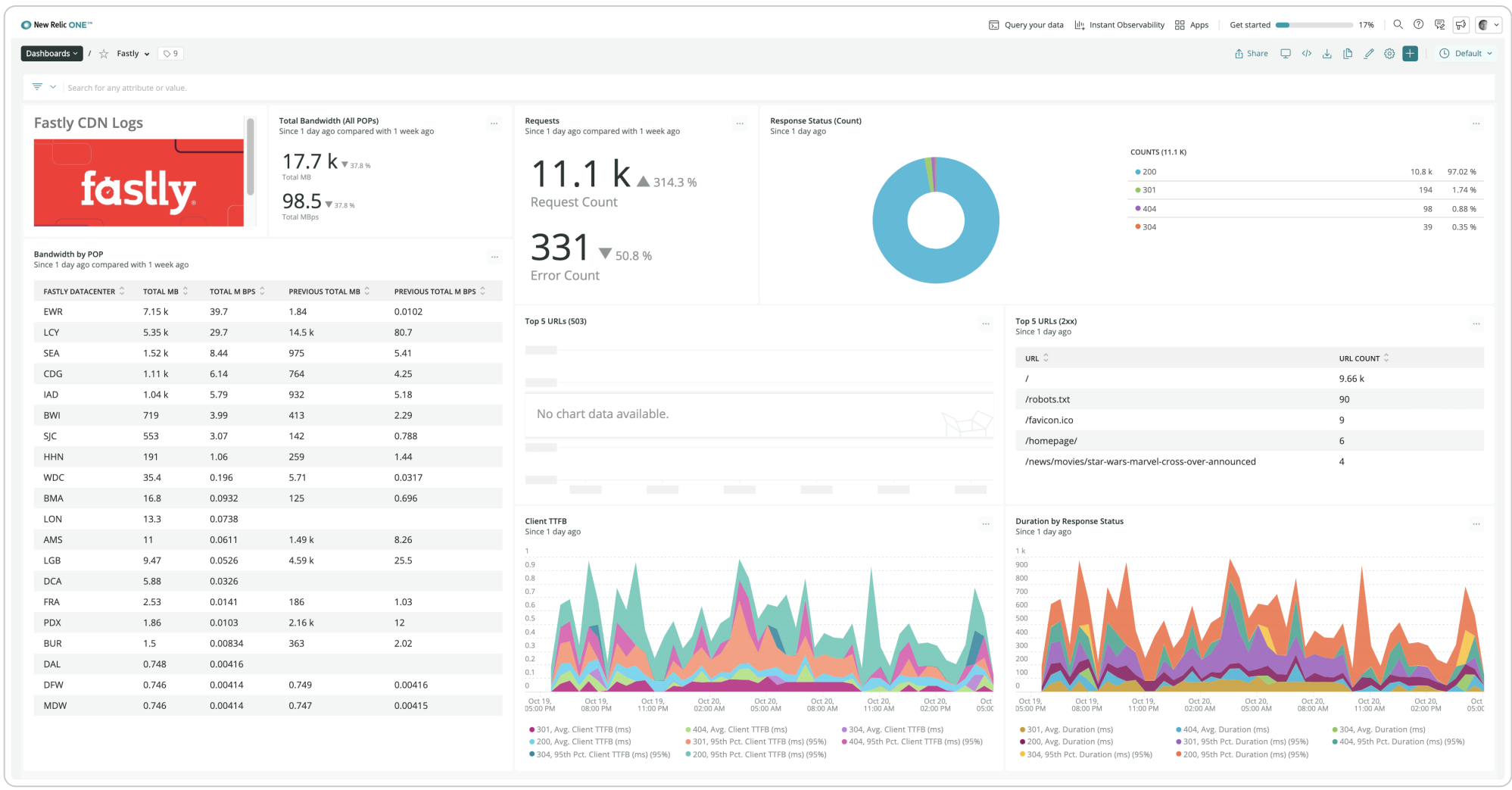
4. Click the **Done** button or the **Skip this step** link.



5. Click the **See your data** button.



The Fastly dashboard will appear.



Log streaming: OpenStack

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-openstack

Fastly's Real-Time Log Streaming feature can send log files to OpenStack. OpenStack is an open-source platform for cloud-computing that many companies deploy as an infrastructure-as-a-service.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding OpenStack as a logging endpoint

Follow these instructions to add OpenStack as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the OpenStack **Create endpoint** button. The Create an OpenStack endpoint page appears.
3. Fill out the **Create an OpenStack endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Auth URL** field, enter the URL used for OpenStack authentication (e.g., `https://auth.api.rackspacecloud.com/v1.0`).
   - In the **Bucket name** field, enter the name of the OpenStack bucket in which to store the logs.
   - In the **User** field, enter your OpenStack username.
   - In the **Access Key** field, enter your OpenStack access key.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
4. Click the **Advanced options** link of the **Create a new OpenStack endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create an OpenStack endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to OpenStack. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Oracle Cloud Storage

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-oracle-cloud-storage

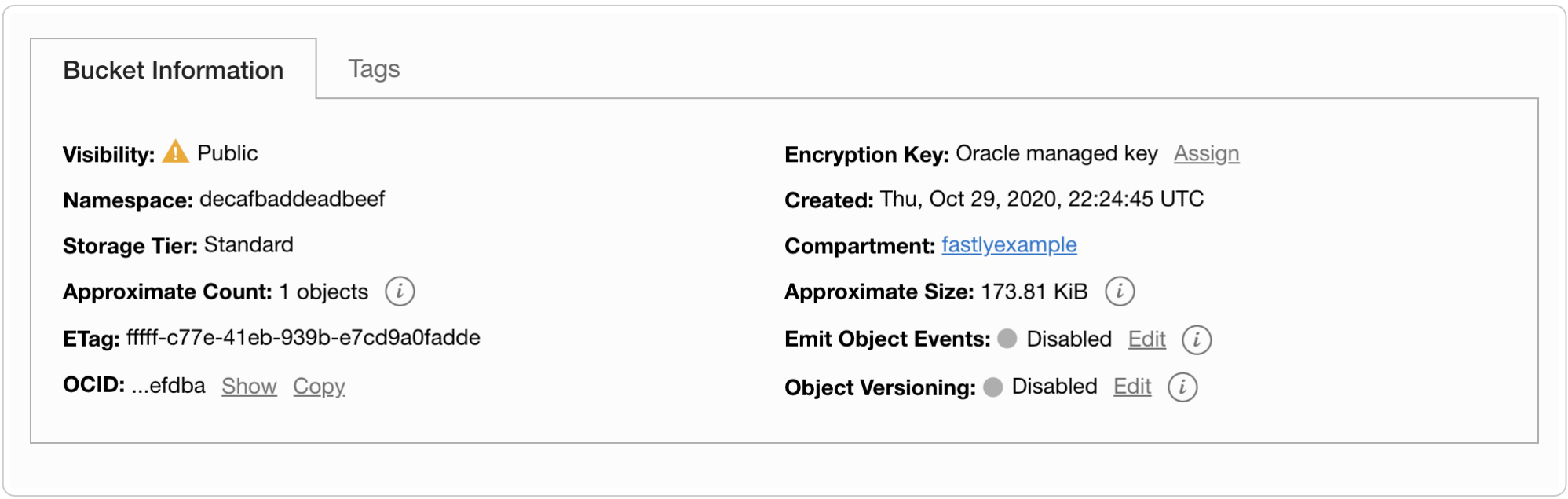
Fastly's Real-Time Log Streaming feature can send log files to Oracle Cloud Storage using Oracle Cloud's S3-compatible API connectivity option. Oracle Cloud Storage is a static file storage service used by developers and IT teams.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Oracle Cloud Storage as a logging endpoint for Fastly services, you need an Oracle Customer Secret Key, which consists of an **Access Key** and **Secret Key**. These need read and write permissions on the bucket.

You will also need to know the namespace identifier assigned to your bucket. You can find your namespace by clicking on the bucket and examining the **Bucket Information** tab (`decafbaddeadbeef` in this case).



Adding Oracle Cloud Storage as a logging endpoints

After you've registered for an Oracle Cloud Storage account and created an Access Key, follow these instructions to add Oracle Cloud Storage as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Amazon Web Services S3 logo. The **Create an Amazon S3 endpoint** page appears.
3. Fill out the **Create an Amazon S3 endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.

- In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
- In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
- In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
- In the **Access method** area, select **User Credentials**.
- In the **Bucket name** field, enter the name of the Oracle Cloud Storage bucket in which to store the logs.
- In the **Access key** field, enter the access key associated with the Oracle account.
- In the **Secret key** field, enter the secret key associated with the Oracle account.
- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a new S3 endpoint** page.

5. Fill out the rest of the **Advanced options** of the **Create an Amazon S3 endpoint** page as follows:

- In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
- In the **Domain** field, enter `<namespace>.compat.objectstorage.<region>.oraclecloud.com`.
- In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
- In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
- In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
- From the **Redundancy level** menu, select a setting. Valid values are **Standard** and **Infrequent Access**. This value defaults to **Standard**.
- In the **Server side encryption** area, optionally select an encryption method to protect files that Fastly writes to your Oracle Cloud Storage bucket. Valid values are **None** and **AES-256**. See Oracle's guide to this feature, which is functionally identical to Amazon's implementation except for lack of support for a key management service.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> Although Fastly continuously streams logs into Oracle Cloud, the Oracle Cloud website and API do not make files available for access until after their upload is complete.

Example format

The following is an example format string for sending data to Oracle Cloud Storage. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Papertrail

Last updated: 2021-06-09

https://docs.fastly.com/en/guides/log-streaming-papertrail

Fastly's Real-Time Log Streaming feature can send log files to Papertrail. Papertrail is a web-based log aggregation application used by developers and IT teams. Instructions for setting up remote log streaming via Papertrail are detailed in the Papertrail setup and configuration documentation.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Log streaming: Scalyr

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-scalyr

Fastly's Real-Time Log Streaming feature can send log files to Scalyr. Scalyr pulls all your server logs and metrics into a centralized, searchable system in real time.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

If you don't already have a Scalyr account, you'll need to register for one. Follow the signup instructions on the Scalyr website.

Once you've signed up, navigate to the **API Keys** area in the **Settings** on your Scalyr dashboard and make note of your Scalyr Write Token. Scaylr uses this to associate data you send them with your account. You'll need this token when you set up your endpoint with Fastly.

If you're adding the Scalyr endpoint via the command line, instead of the web interface, you should also have your Fastly API token and the service ID and version number of the Fastly service for which you'll be enabling Scalyr logging.

Adding Scalyr as a logging endpoint

Follow these instructions to add Scalyr as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Scalyr **Create endpoint** button. The Create a Scalyr endpoint page appears.

3. Fill out the **Create a Scalyr endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Logfile** field, optionally specify the log file name under which your logs will appear on Scalyr's Overview page. Defaults to `logplex`.
   - In the **Token** field, enter the Scalyr Write Token provided in the Scalyr dashboard.
   - From the **Region** menu, select the region to stream logs to.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Scalyr. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: SFTP

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-sftp

Fastly's Real-Time Log Streaming feature can send log files to SFTP, a secure file transfer subsystem for the Secure Shell (SSH) protocol. Our SFTP endpoint supports both password-based authentication and SSH public-key authentication, with SSH public-key authentication being preferred. To learn more about SSH public-key authentication, or to learn how to generate public and private key pairs, see this guide.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding SFTP as a logging endpoint

Follow these instructions to add SFTP as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the SFTP **Create endpoint** button. The Create an SSH File Transfer Protocol (SFTP) endpoint page appears.
3. Fill out the **Create an SSH File Transfer Protocol (SFTP) endpoint** fields as follows:

- In the **Name** field, enter a human-readable name for the endpoint.
- In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
- In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
- In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
- In the **Address** field, enter the hostname or IP address of the SFTP server. In the port field after the colon, enter the port number you're using for SFTP (the default is `22`).
- In the **Path** field, enter the path to use for storing log files. Leaving the default `/` in this field means the files will be saved in the root path. We describe this variable in more detail in our guide on changing where log files are written.

  > **TIP**
  >
  > If you save logs on the SFTP server, make sure the directory already exists.

- In the **User** field, enter the username used to authenticate to the SFTP server.
- In the **Known hosts** field, enter a Host key for each Host you can connect to over SFTP. Each Host key you enter must be on its own line. Known hosts entries should match what's stored in your known_hosts file located in your home directory (or the local account settings if you're working with a Mac or Windows operating system). A known hosts entry looks like this:

  ```
  1.2.3.4 ecdsa-sha2-nistp256 aBc123xYz…
  ```

  where the `1.2.3.4` is the SFTP IP address, `ecdsa-sha2-nistp256` is your Host key algorithm, and `aBc123xYz…` is your public key.
- In the **Secret key** field, enter the SSH secret key used to connect to the server. If both Secret key and Password are entered, the Secret key will be used in preference.
- In the **Password** field, enter the password used to authenticate to the SFTP server. If both Password and Secret key are entered, the Secret key will be used in preference.
- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create an SSH File Transfer Protocol (SFTP) endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create an SSH File Transfer Protocol (SFTP) endpoint** as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to an SFTP logging endpoint. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Shape Log Analysis

Last updated: 2021-06-09

https://docs.fastly.com/en/guides/log-streaming-shape-log-analysis

Fastly's Real-Time Log Streaming feature can send log files to Shape Security. Shape Log Analysis uses anonymized attack data to analyze HTTP and application logs for insight into fraudulent activity and various types of attack that attempt to bypass security measures protecting origin servers.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Shape Log Analysis as a logging endpoint, send an email to fastly@f5.com with the subject line "Fastly Log Streaming Setup" to request a secure S3 bucket from Shape. In return, you will receive an email with details to help you complete the configuration of the logging endpoint including:

1. a note about the Log format value
2. a Bucket name
3. an Access key
4. a Secret key
5. a Path
6. a Domain

Each item will be specifically numbered in the email and the images in the configuration details below reflect those numbers.

Adding Shape Log Analysis as a logging endpoint

IMPORTANT

Shape Log Analysis setup and configuration uses Fastly's Amazon S3 log streaming endpoint to enable logs for storage and analysis. It places them in a secure S3 bucket managed by Shape that only accepts traffic from Fastly IP addresses.

After you've contacted fastly@f5.com and received the prerequisite information email, complete the following steps:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Amazon Web Services S3 **Create endpoint** button. The Create an Amazon S3 endpoint page appears.
3. Fill out the **Create an Amazon S3 endpoint** fields as follows:

- In the **Name** field, enter a human-readable name for the endpoint.
- In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
- In the **Log format** field, copy and paste the following log format value to send log data to Shape's secure S3 bucket:

```
{
  "timestamp": "%{begin:%Y-%m-%dT%H:%M:%S%z}t",
  "ts": "%{time.start.sec}V",
  "id.orig_h": "%h", "status_code": "%>s",
  "method": "%m",
  "host": "%{Host}i",
  "uri": "%U%q",
  "accept_encoding": "%{Accept-Encoding}i",
  "request_body_len": "%{req.body_bytes_read}V",
  "response_body_len": "%{resp.body_bytes_written}V",
  "location": "%{Location}i",
  "x_forwarded_for": "%{X-Forwarded-For}i",
  "user_agent": "%{User-Agent}i",
  "referer": "%{Referer}i",
  "accept": "%{Accept}i",
  "accept_language": "%{Accept-Language}i",
  "content_type": "%{Content-Type}o",
  "geo_city": "%{client.geo.city}V",
  "geo_country_code": "%{client.geo.country_code}V",
  "is_tls": %{if(req.is_ssl, "true", "false")}V,
  "tls_version": "%{tls.client.protocol}V",
  "tls_cipher_request": "%{tls.client.cipher}V",
  "tls_cipher_req_hash": "%{tls.client.ciphers_sha}V",
  "tls_extension_identifiers_hash": "%{tls.client.tlsexts_sha}V"
}
```

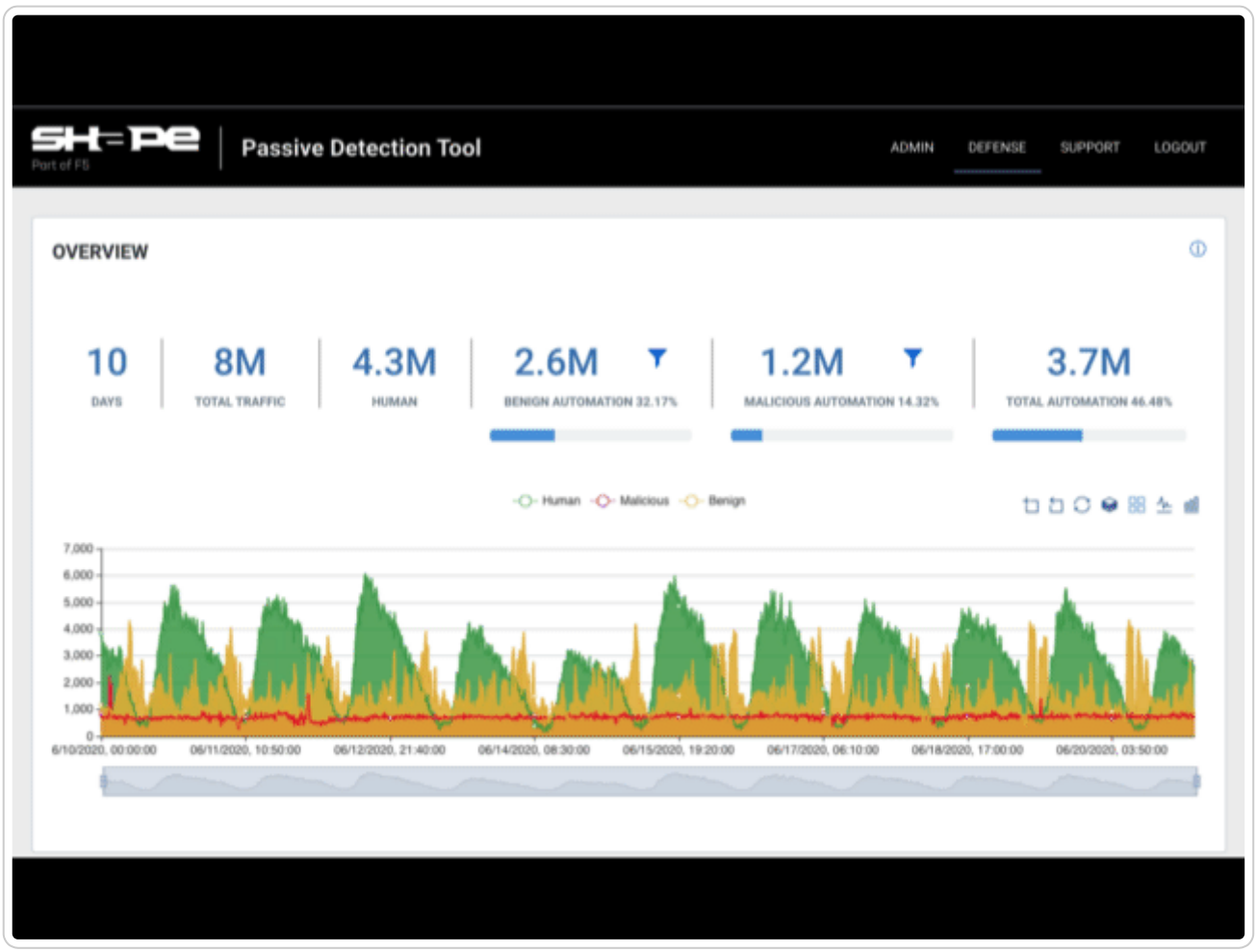
- In the **Access method** area, select **User Credentials**.
- In the **Bucket name** field, enter the name of value #2 “Bucket Name” received in the Shape email response.
- In the **Access key** field, enter the name of value #3 “Access Key” received in the Shape email response.
- In the **Secret key** field, enter the name of value #4 “Secret Key” received in the Shape email response.

> **NOTE**
>
> Password management software may mistakenly treat the **Secret Key** field as a password field because of the way your web browser works. As such, that software may try to auto-fill this field with your Fastly account password. If this happens to you, the integration with Fastly services won't work and you will need to enter **Secret Key** manually instead.

4. Click the **Advanced options** link of the **Create a new S3 endpoint** page and fill in the following:
5. Fill out the **Advanced options** of the **Create an Amazon S3 endpoint** page as follows:
   - In the **Path** field, enter the name of value #5 “Path” received in the email response.
   - In the **Domain** field, enter the name of value #6 “Domain” received in the email response.
   - In the **Select a log line format** area, select `Blank` to prevent prefixes from being added to log line messages and only report details in JSON log format. Our guide on changing log line formats provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Shape data analysis

Once Fastly logging configuration is complete, logs will be sent to Shape's secure S3 bucket for analysis. Typically, Shape collects approximately two weeks worth of log data to provide an analysis of attack traffic. After analysis is complete, Shape will send you a report with data on topics like Malicious Automation, Attack Surface (URLs), Account Takeover Bots, Suspicious Manual Fraud, and Top Bot Campaigns.



Log streaming: Splunk

Last updated: 2021-06-09

https://docs.fastly.com/en/guides/log-streaming-splunk

Fastly's Real-Time Log Streaming feature can send log files to Splunk. Splunk is a web-based log analytics platform used by developers and IT teams.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

To use Splunk as a logging endpoint, you'll need to enable the HTTP Event Collector (HEC), create a token, and enable it. Follow the instructions on Splunk's website:

1. Enable HEC.
2. Create an HEC token.
3. Enable the HEC token.
4. Disable indexer acknowledgment for tokens used by Fastly to stream logs.

You'll need to remember the HEC token and find the URL for your collector. The URL structure depends on the type of Splunk instance you're using. Use the table below to find the URL structure for your Splunk instance.

Type	URL
Self hosted	`https://<hostname>:8088/services/collector/event`
Self-service Splunk Cloud plans	`https://input-<hostname>:8088/services/collector/event`
All other Splunk Cloud plans	`https://http-inputs-<hostname>:8088/services/collector/event`

While logged in to Splunk, you can find the hostname for the URL in your web browser's address bar.

Adding Splunk as a logging endpoint

After you've created a Splunk account and obtained your customer token, follow these instructions to add Splunk as a logging endpoint for Fastly services:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Splunk **Create endpoint** button. The Create a Splunk endpoint page appears.
3. Fill out the **Create a Splunk endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, enter an Apache-style string or VCL variables to use for log formatting. You can use our recommended log format.
   - In the **URL** field, enter the URL to send data to (e.g., `https://<splunk_host>:8088/services/collector/event/1.0`).
   - In the **Token** field, enter the token for the HEC.
   - From the **Use TLS** controls, optionally select whether or not to enable TLS. When you select Yes, additional TLS fields appear.
   - In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported. If you are using Splunk Enterprise see the Splunk Enterprise section below for more information.
   - In the **TLS CA certificate** field, enter the CA certificate used to verify that the origin's certificate is valid. It must be in PEM format. This is not required if your origin-side TLS certificate is signed by a well-known CA. See the using TLS CA certificates section for more information.
   - In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection.
   - In the **Maximum logs** field, optionally enter the maximum number of logs to append to a batch, if non-zero.
   - In the **Maximum bytes** field, optionally enter the maximum size of the log batch, if non-zero.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Recommended log format

We recommend using the following log format to send data to Splunk.

> **NOTE**
>
> All JSON sent to the Splunk HEC must have an event field. The event field can be text or nested JSON. There can also be other meta data in the payload. See the Splunk documentation for more information.

```
{
  "time":%{time.start.sec}V,
  "host":"%{Fastly-Orig-Host}i",
  "event":  {
    "service_id":"%{req.service_id}V",
    "time_start":"%{begin:%Y-%m-%dT%H:%M:%S%Z}t",
    "time_end":"%{end:%Y-%m-%dT%H:%M:%S%Z}t",
    "time_elapsed":%D,
    "client_ip":"%h",
    "client_as_name":"%{client.as.name}V",
    "client_as_number":"%{client.as.number}V",
    "client_connection_speed":"%{client.geo.conn_speed}V",
    "request":"%m",
    "protocol":"%H",
    "origin_host":"%v",
    "url":"%{cstr_escape(req.url)}V",
    "is_ipv6":%{if(req.is_ipv6, "true", "false")}V,
    "is_tls":%{if(req.is_ssl, "true", "false")}V,
    "tls_client_protocol":"%{cstr_escape(tls.client.protocol)}V",
    "tls_client_servername":"%{cstr_escape(tls.client.servername)}V",
    "tls_client_cipher":"%{cstr_escape(tls.client.cipher)}V",
    "tls_client_cipher_sha":"%{cstr_escape(tls.client.ciphers_sha )}V",
    "tls_client_tlsexts_sha":"%{cstr_escape(tls.client.tlsexts_sha)}V",
    "is_h2":%{if(fastly_info.is_h2, "true", "false")}V,
    "is_h2_push":%{if(fastly_info.h2.is_push, "true", "false")}V,
    "h2_stream_id":"%{fastly_info.h2.stream_id}V",
    "request_referer":"%{Referer}i",
    "request_user_agent":"%{User-Agent}i",
    "request_accept_content":"%{Accept}i",
    "request_accept_language":"%{Accept-Language}i",
    "request_accept_encoding":"%{Accept-Encoding}i",
    "request_accept_charset":"%{Accept-Charset}i",
    "request_connection":"%{Connection}i",
    "request_dnt":"%{DNT}i",
    "request_forwarded":"%{Forwarded}i",
    "request_via":"%{Via}i",
    "request_cache_control":"%{Cache-Control}i",
    "request_x_requested_with":"%{X-Requested-With}i",
    "request_x_att_device_id":"%{X-ATT-Device-Id}i",
    "request_x_forwarded_for":"%{X-Forwarded-For}i",
    "status":"%s",
    "content_type":"%{Content-Type}o",
    "cache_status":"%{regsub(fastly_info.state, "^(HIT-(SYNTH)|(HITPASS|HIT|MISS|PASS|ERROR|PIPE)).*", "\\2\\3")}V",
    "is_cacheable":%{if(fastly_info.state ~"^(HIT|MISS)$", "true", "false")}V,
    "response_age":"%{Age}o",
    "response_cache_control":"%{Cache-Control}o",
    "response_expires":"%{Expires}o",
    "response_last_modified":"%{Last-Modified}o",
    "response_tsv":"%{TSV}o",
    "server_datacenter":"%{server.datacenter}V",
    "server_ip":"%A",
    "geo_city":"%{client.geo.city.utf8}V",
    "geo_country_code":"%{client.geo.country_code}V",
    "geo_continent_code":"%{client.geo.continent_code}V",
    "geo_region":"%{client.geo.region}V",
    "req_header_size":%{req.header_bytes_read}V,
    "req_body_size":%{req.body_bytes_read}V,
    "resp_header_size":%{resp.header_bytes_written}V,
    "resp_body_size":%B,
    "socket_cwnd":%{client.socket.cwnd}V,
    "socket_nexthop":"%{client.socket.nexthop}V",
    "socket_tcpi_rcv_mss":%{client.socket.tcpi_rcv_mss}V,
    "socket_tcpi_snd_mss":%{client.socket.tcpi_snd_mss}V,
    "socket_tcpi_rtt":%{client.socket.tcpi_rtt}V,
    "socket_tcpi_rttvar":%{client.socket.tcpi_rttvar}V,
    "socket_tcpi_rcv_rtt":%{client.socket.tcpi_rcv_rtt}V,
    "socket_tcpi_rcv_space":%{client.socket.tcpi_rcv_space}V,
    "socket_tcpi_last_data_sent":%{client.socket.tcpi_last_data_sent}V,
    "socket_tcpi_total_retrans":%{client.socket.tcpi_total_retrans}V,
    "socket_tcpi_delta_retrans":%{client.socket.tcpi_delta_retrans}V,
    "socket_ploss":%{client.socket.ploss}V
  }
}
```

Using TLS CA certificates

If you've installed your own TLS certificate in Splunk Enterprise or Splunk Cloud, you'll need to provide the corresponding CA certificate.

Splunk Cloud

For Splunk Cloud, the default set up has the following CA certificate:

```
-----BEGIN CERTIFICATE-----
MIIB/DCCAaGgAwIBAgIBADAKBggqhkjOPQQDAjB+MSswKQYDVQQDEyJTcGx1bmsg
Q2xvdWQgQ2VydGlmaWNhdGUgQXV0aG9yaXR5MRYwFAYDVQQHEw1TYW4gRnJhbmNp
c2NvMRMwEQYDVQQKEwpTcGx1bmsgSW5jMQswCQYDVQQIEwJDQTEVMBMGA1UECxMM
U3BsdW5rIENsb3VkMB4XDTE0MTExMDA3MDAxOFoXDTM0MTEwNTA3MDAxOFowfjEr
MCkGA1UEAxMiU3BsdW5rIENsb3VkIENlcnRpZmljYXRlIEF1dGhvcml0eTEWMBQG
A1UEBxMNU2FuIEZyYW5jaXNjbzETMBEGA1UEChMKU3BsdW5rIEluYzELMAkGA1UE
CBMCQ0ExFTATBgNVBAsTDFNwbHVuayBDbG91ZDBZMBMGByqGSM49AgEGCCqGSM49
AwEHA0IABPRRy9i3yQcxgMpvCSsI7Qe6YZMimUHOecPZWaGz5jEfB4+p5wT7dF3e
QrgjDWshVJZvK6KGO7nDh97GnbVXrTCjEDAOMAwGA1UdEwQFMAMBAf8wCgYIKoZI
zj0EAwIDSQAwRgIhALMUgLYPtICN9ci/ZOoXeZxUhn3i4wIo2mPKEWX0IcfpAiEA
8Jid6bzwUqAdDZPSOtaEBXV9uRIrNua0Qxl1S55TlWY=
-----END CERTIFICATE-----
```

Splunk Enterprise

Splunk Enterprise provides a set of default certificates, but we strongly recommend you configure your own certificates for your Fastly logging endpoint rather than relying on the default certificates. The certificates provided by Splunk Enterprise only specify a Common Name (CN), which cannot be used to properly verify the identity of the Splunk host presenting the certificate. Additionally, these certificates are less secure because the same root certificate is available in every Splunk Enterprise download. We encourage you to maintain the best possible security posture by configuring your own certificates rather than relying on the default certificates. The Splunk documentation provides a guide for configuring your own certificates.

Log streaming: Storj DCS

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-storj-dcs

Fastly's Real-Time Log Streaming feature can send log files to Storj DCS, a decentralized object storage service that is S3 compatible and end-to-end encrypted by default.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Storj DCS as a logging endpoint for Fastly services, you will need to create a Storj DCS account, project and access credentials, and a bucket that will store the log output.

Adding Storj DCS as a logging endpoint

Follow these instructions to add Storj DCS as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Amazon Web Services S3 logo. The **Create an Amazon S3 endpoint** page appears.
3. Fill out the **Create an Amazon S3 endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Bucket name** field, enter the name of the Storj DCS bucket in which to store the logs.

- In the **Access method** area, select **User Credentials**.
- In the **Access key** field, enter the access key associated with the Storj DCS bucket.
- In the **Secret key** field, enter the secret key associated with the Storj DCS bucket.
- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a new S3 endpoint** page.
5. Fill out the rest of the **Advanced options** of the **Create an Amazon S3 endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **Domain** field, enter the fully qualified hostname of any Storj DCS Gateway.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
   - From the **Redundancy level** menu, select a setting. This value defaults to **Standard**.
   - In the **Server side encryption** area, optionally select an encryption method to protect files that Fastly writes to your Storj DCS bucket. Valid values are **None** and **AES-256**.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Storj DCS. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Log streaming: Sumo Logic

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-sumologic

Fastly's Real-Time Log Streaming feature can send log files to Sumo Logic. Sumo Logic is a web-based log analytics platform used by developers and IT teams.

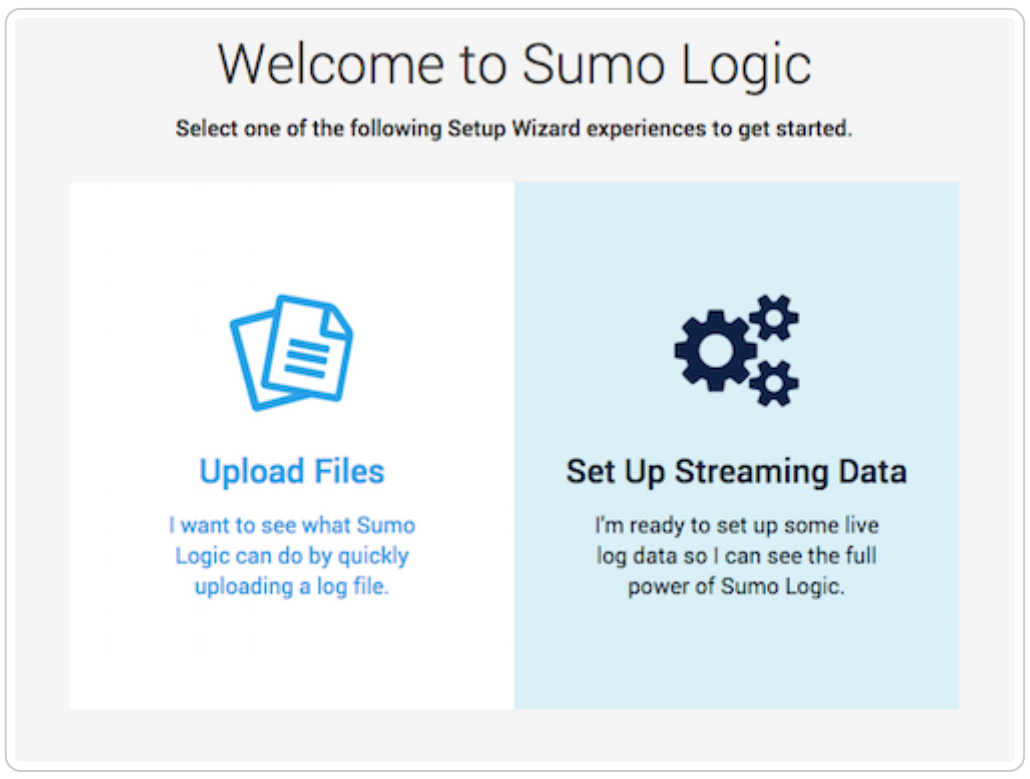
NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

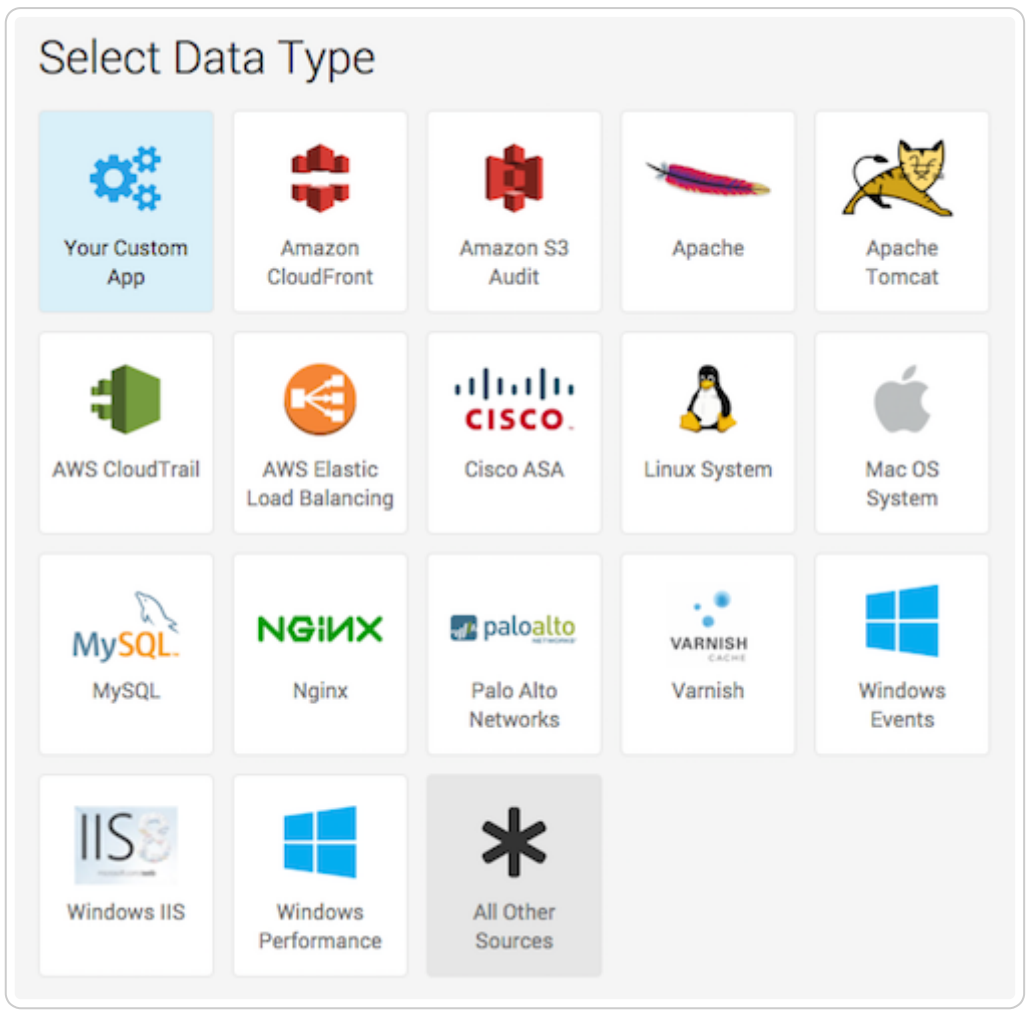
Setting up Sumo Logic

To use Sumo Logic as a logging endpoint, you'll need to create a Sumo Logic account, add a new source, and save the HTTP Source URL. Follow these instructions to add a new source in the Sumo Logic website:

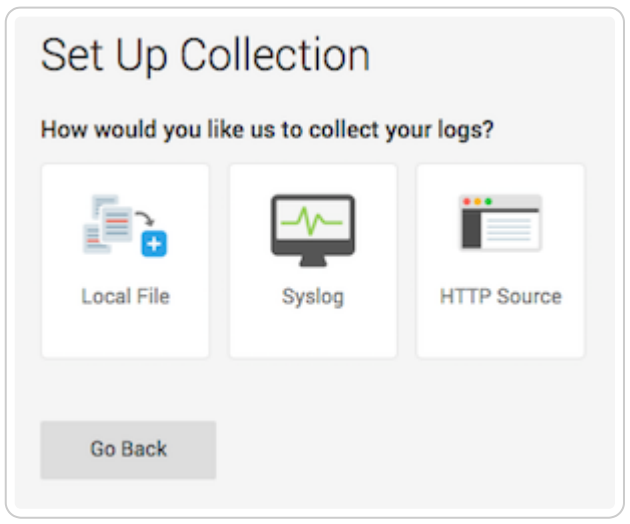
1. The process starts with the Sumo Logic Setup Wizard, which appears immediately after you create your Sumo Logic account. If you already have an account, you can access the wizard by selecting **Setup Wizard** from the **Manage** menu at the top of the Sumo Logic application.



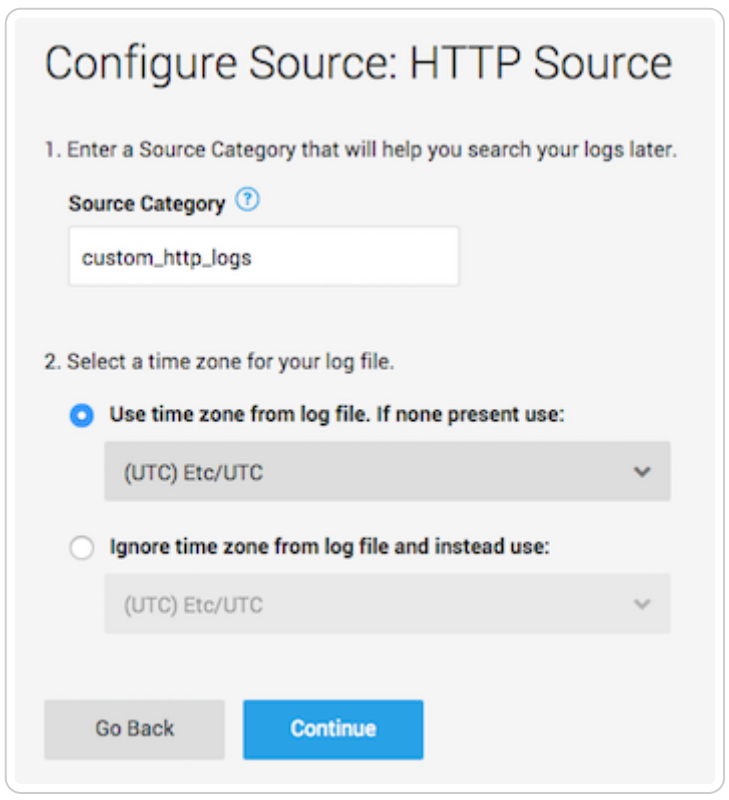
2. Click **Set Up Streaming Data**. The Select Data Type window appears.



3. Click **All Other Sources**. The Set Up Collection window appears.

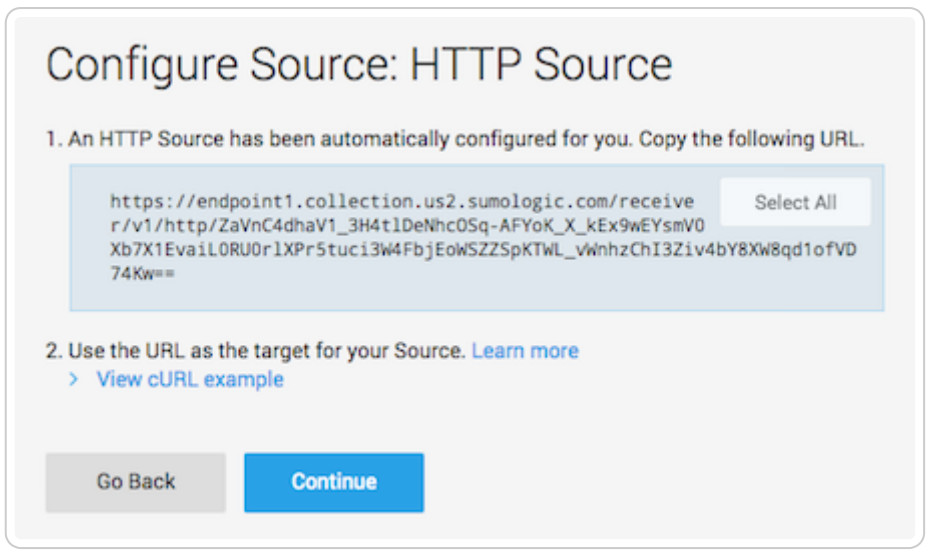


4. Click **HTTP Source**. The Configure Source: HTTP Source window appears.



5. In the **Source Category** field, enter a human-readable name for the category (e.g., `fastly_cdn`) and select a time zone for your log file.

6. Click **Continue**. The HTTP Source URL appears.



7. Copy the HTTP Source URL. You will enter this value in the Fastly web interface.

8. Click **Continue**. Sumo Logic will add the new source.

Adding Sumo Logic as a logging endpoint

After you've created a Sumo Logic account and obtained the HTTP Source URL, follow these instructions to add Sumo Logic as a logging endpoint for Fastly services:

1. Review the information in our Setting Up Remote Log Streaming guide.

2. Click the Sumo Logic **Create endpoint** button. The Create a Sumo Logic endpoint page appears.

3. Fill out the **Create a Sumo Logic endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Collector URL** field, enter the address of the HTTP Source URL you found in the Sumo Logic website.
4. Click the **Advanced options** link of the **Create a Sumo Logic endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Sumo Logic endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to Sumo Logic. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Troubleshooting

The Sumo Logic logging endpoint is designed for services with sustained levels of traffic. If you aren't seeing any logs in Sumo Logic, try waiting a bit.

Log streaming: Syslog

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-syslog

Fastly's Real-Time Log Streaming feature can send log files to syslog-based logging software. Syslog is a widely used standard for message logging.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding syslog as a logging endpoint

Follow these instructions to add syslog as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Syslog **Create endpoint** button. The Create a Syslog endpoint page appears.
3. Fill out the **Create a Syslog endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Syslog address** field, enter the domain name or IP address and port to which logs should be sent. Be sure this port can receive incoming TCP traffic from Fastly. See the firewall considerations section for more information.
   - In the **Token** field, optionally enter a string prefix (line prefix) to send in front of each log line.
   - From the **TLS** menu, select **No** to disable encryption for the syslog endpoint, or **Yes** to enable it. When you select Yes, additional TLS fields appear.
   - In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.
4. Click the **Advanced options** link of the **Create a Syslog endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Syslog endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Example format

The following is an example format string for sending data to syslog. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

Adding separators or static strings

To insert a separator or other arbitrary string into the syslog endpoint format:

1. Create a new header with the following fields:
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter any suitable header name (for example, `http.X-Separator`).
   - In the **Source** field, enter any special character or string you want (for example, `"|"`).
2. Reference the new header variable in the log format box for your specific provider (for example, `req.http.X-Separator`).

Syslog facility and severity

The syslog output includes the following facility and severity values:

```
facility: local0
severity: info
```

Firewall considerations

Syslog has limited security features. For this reason, it's best to create a firewall for your syslog server and only accept TCP traffic on your configured port from our address blocks. Our list of IP address blocks is dynamic, so we recommend programmatically obtaining the list whenever possible.

Log streaming: Wasabi Hot Cloud Storage

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/log-streaming-wasabi-hot-cloud-storage

Fastly's Real-Time Log Streaming feature can send log files to Wasabi Hot Cloud Storage using Wasabi's S3-compatible API connectivity option. Wasabi Hot Cloud Storage is a static file storage service used by developers and IT teams.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Wasabi Hot Cloud Storage as a logging endpoint for Fastly services, we recommend creating an Access Key to which you've given read and write permissions on the bucket.

Adding Wasabi Hot Cloud Storage as a logging endpoint

After you've registered for an Wasabi Hot Cloud Storage account and created an Access Key, follow these instructions to add Wasabi Hot Cloud Storage as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Amazon Web Services S3 logo. The **Create an Amazon S3 endpoint** page appears.
3. Fill out the **Create an Amazon S3 endpoint** fields as follows:
   - In the **Name** field, enter a human-readable name for the endpoint.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **waf_debug (waf_debug_log)**, and **None**. See our guide on changing log placement for more information.
   - In the **Log format** field, optionally enter an Apache-style string or VCL variables to use for log formatting. See the example format section for details.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Bucket name** field, enter the name of the Wasabi Hot Cloud Storage bucket in which to store the logs.
   - In the **Access method** area, select **User Credentials**.
   - In the **Access key** field, enter the access key associated with the Wasabi account. See Wasabi's Access Key Guide for more information.
   - In the **Secret key** field, enter the secret key associated with the Wasabi account. See Wasabi's Access Key Guide for more information.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
4. Click the **Advanced options** link of the **Create a new S3 endpoint** page.
5. Fill out the rest of the **Advanced options** of the **Create an Amazon S3 endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **Domain** field, enter `s3.wasabisys.com`.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
   - The **Redundancy level** option is not useful as Wasabi only provides a single storage class which is most like the standard AWS S3 storage class. Wasabi's Object Storage Class documentation provides more information on using reduced redundancy storage.
   - In the **Server side encryption** area, optionally select an encryption method to protect files that Fastly writes to your Wasabi Hot Cloud Storage bucket. Valid values are **None** and **AES-256**. See Wasabi's guide to this feature which is functionally identical to Amazon's implementation except for lack of support for a key management service.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> Although Fastly continuously streams logs into Wasabi, the Wasabi website and API do not make files available for access until after their upload is complete.

Example format

The following is an example format string for sending data to Wasabi. Our discussion of format strings provides more information.

```
{
  "timestamp": "%{strftime(\{"%Y-%m-%dT%H:%M:%S%z"\}, time.start)}V",
  "client_ip": "%{req.http.Fastly-Client-IP}V",
  "geo_country": "%{client.geo.country_name}V",
  "geo_city": "%{client.geo.city}V",
  "host": "%{if(req.http.Fastly-Orig-Host, req.http.Fastly-Orig-Host, req.http.Host)}V",
  "url": "%{json.escape(req.url)}V",
  "request_method": "%{json.escape(req.method)}V",
  "request_protocol": "%{json.escape(req.proto)}V",
  "request_referer": "%{json.escape(req.http.referer)}V",
  "request_user_agent": "%{json.escape(req.http.User-Agent)}V",
  "response_state": "%{json.escape(fastly_info.state)}V",
  "response_status": %{resp.status}V,
  "response_reason": %{if(resp.response, "%22"+json.escape(resp.response)+"%22", "null")}V,
  "response_body_size": %{resp.body_bytes_written}V,
  "fastly_server": "%{json.escape(server.identity)}V",
  "fastly_is_edge": %{if(fastly.ff.visits_this_service == 0, "true", "false")}V
}
```

§

These articles describe how non-Fastly services interoperate with Fastly.

https://docs.fastly.com/en/guides/integrations#_non-fastly-services

Alibaba Object Storage Service

Last updated: 2022-03-01

https://docs.fastly.com/en/guides/alibaba-object-storage-service

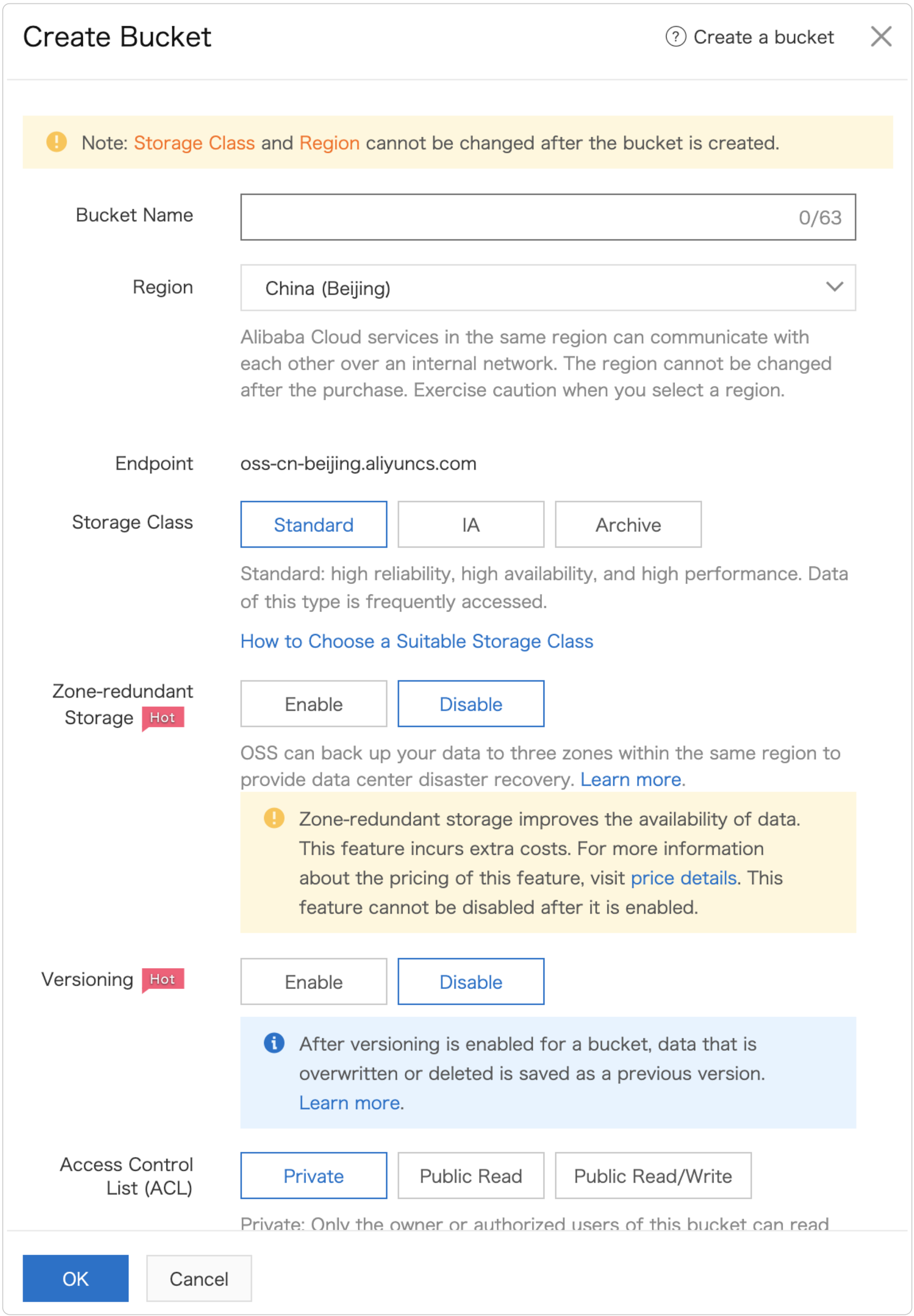
Alibaba Object Storage Service (OSS) can be used as an origin for Fastly for both public and private content.

Using OSS as an origin

To use OSS as an origin, follow the steps below.

Setting up and configuring your OSS account

1. Sign up for Alibaba Object Storage Service.
2. Create a bucket to store your origin's data. The Create Bucket window appears.



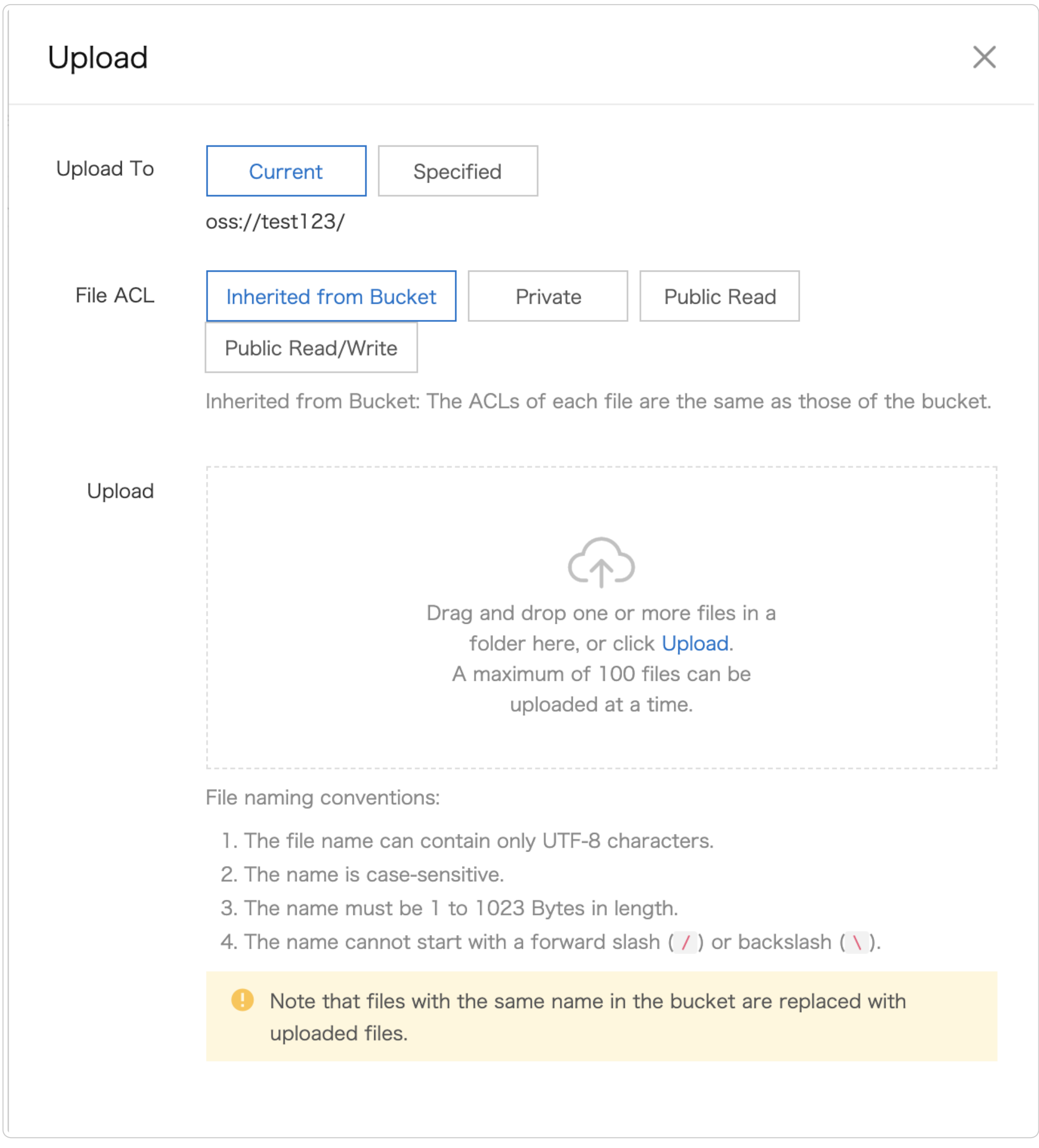
3. Fill out the **Create Bucket** fields as follows:
   - In the **Bucket Name** field, enter a name for your bucket. Remember the name you enter. You'll need it to connect your bucket to your Fastly service.

- From the **Region** menu, select a location to store your content. Most customers select a region close to the POP they specify for shielding.
- From the **Storage Class** options, select **Standard**.
- From the **Access Control List (ACL)** options, select **Public Read**.
- Optionally select other options, such as **Server-side Encryption** and **Scheduled Backup**.

4. Click the **OK** button.

Uploading files to your bucket

Once you've created your bucket, select it and then navigate to the Files tab to add files to it by clicking the **Upload** button.



You can make the files externally accessible by selecting the **Public Read** option for the bucket or you can use the **Inherited from Bucket** option next to each of the files.

Setting up Fastly to use OSS as an origin

To add your OSS bucket as an origin, follow the instructions for working with hosts. You'll add specific details about your origin server.

1. On the **Origins** page, click **Create Host** and enter the appropriate address for your Host using the format `<BUCKET>.<REGION>.aliyuncs.com`. For example, if your bucket name is `test123` and your region is Beijing (e.g., `oss-cn-beijing`) your hostname would be `test123.oss-cn-beijing.aliyuncs.com`. You can also find the hostname on the Bucket Overview page in the **Bucket Domain Name** area.
2. Click on the newly created Host to edit it.
3. In the **Name** field, enter a descriptive name for your service (e.g., `Alibaba Object Storage`).
4. If the **Address** field doesn't contain the `<BUCKET>.<REGION>.aliyuncs.com` hostname you provided in the first step, enter it now.
5. Fill out the **Transport Layer Security (TLS)** area fields as follows:
   - Leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
   - Leave the **Verify certificate?** default set to **Yes**.
   - Set the **Certificate hostname** field to the same address that appears in the Address field (e.g., `test123.oss-cn-beijing.aliyuncs.com`).
   - In the **SNI hostname** field, select the checkbox to **Match the SNI hostname to the Certificate hostname**. The hostname address you entered during Host creation appears.
6. From the **Shielding** menu below the TLS area, select a Fastly POP near the Alibaba region from the list of shielding locations.
7. In the **Override host** field, enter an appropriate address for your Host (e.g., `test123.oss-cn-beijing.aliyuncs.com`). You entered this information during Host creation.

Review our caveats of shielding and select a shield POP accordingly.

Using OSS with private objects

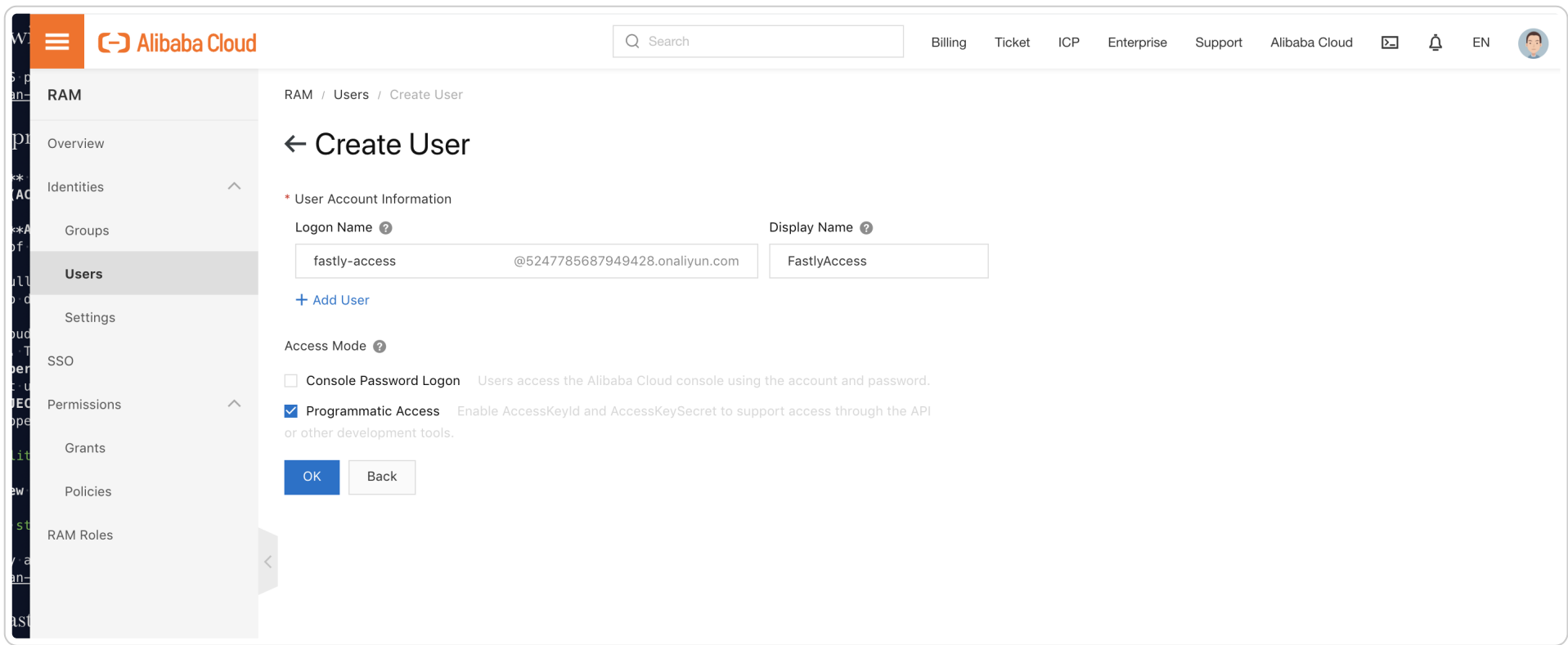
To use Fastly with OSS private objects, be sure you've already made your OSS data available to Fastly by pointing to the right OSS bucket, then follow the steps below.

Setting up a private bucket and sub user

Setting up a private bucket is the same as setting up a public bucket, except you select the **Private** option in the **Access Control List (ACL)** area of the OSS bucket settings.

You'll need an **AccessKey ID** and **Access Key Secret**. These can be linked to your account by clicking on your avatar in the top right corner of the Alibaba Cloud Console, selecting **Access Key**, and then creating a new key. Since this key has full access to the account, we recommend following Alibaba's procedure for creating a sub user. Follow the steps below.

1. Navigate to the Resource Access Management (RAM) page.
2. Click the **Users** tab.
3. Click **Create User**.
4. Enter an appropriate **Logon Name** and **Display Name**.
5. Select the **Programmatic Access** checkbox to enable access through the Alibaba API.



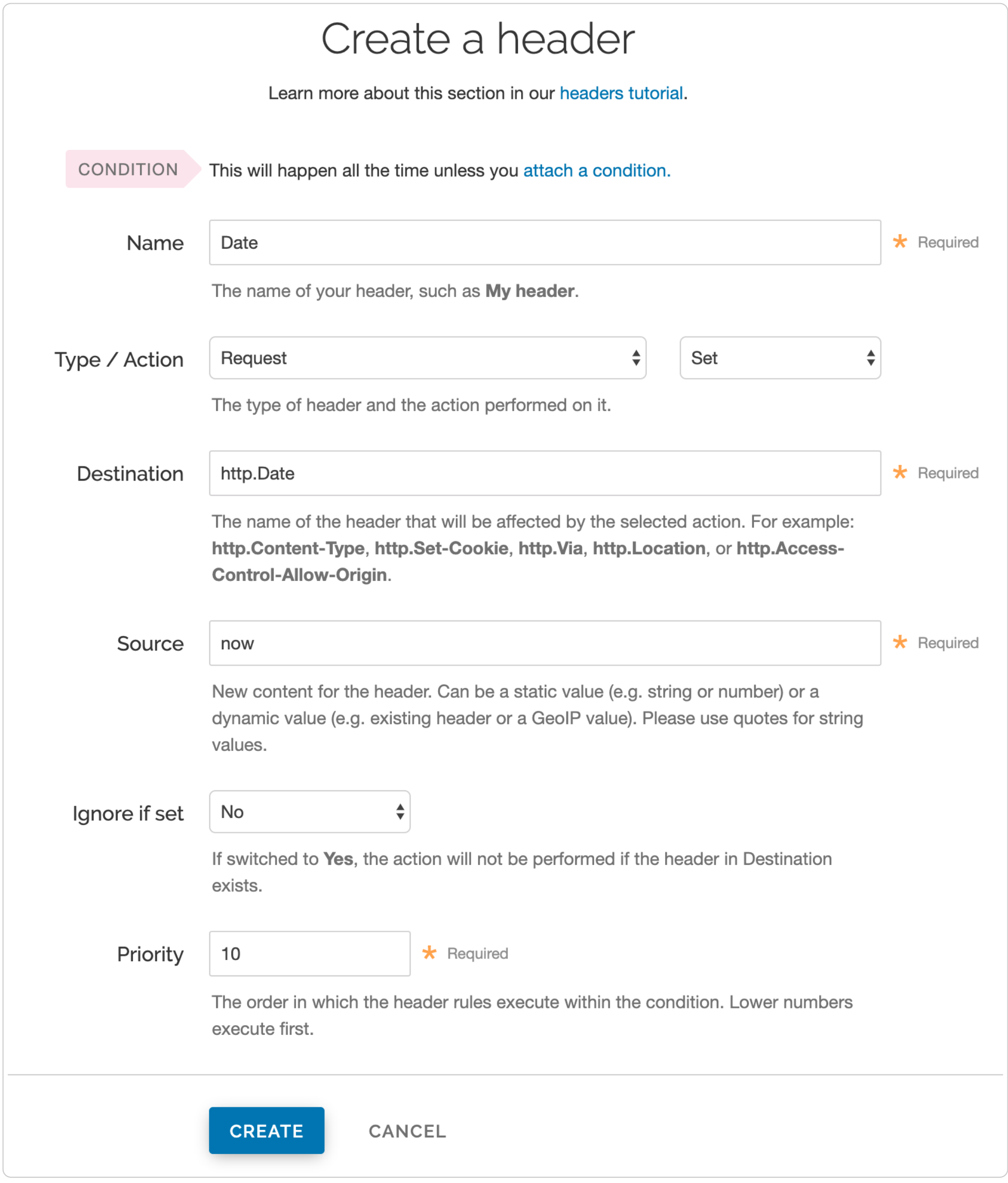
6. Click the **OK** button.
7. Copy the **AccessKeyId** and **AccessKeySecret**. You'll need these later when you're creating an Authorization header.
8. Go back to the bucket overview, click on the **Files** tab and then click on the **Authorize** button. You should see a list of authorized users. If this is a new bucket it should be empty.
9. Click on the **Authorize** button, filling out the fields as follows:
   - From the **Applied To** menu, select the **Whole Bucket** option. You can select **Specified Resources**, but this may lead to unexpected errors later if you don't update the permissions with new files.
   - From the **Accounts** menu, select **RAM Users** and then use the menu to select your newly created RAM user.
   - From the **Authorized Operation** menu, select **Read Only**.
   - You can leave **Condition** blank or customize it using **IP =**, Fastly's IP ranges, or setting **Access Method** to **HTTPS**.

Setting up Fastly to use OSS private content

To use OSS private content with Fastly, you'll need to create two headers: a Date header (required for authorization signature) and a Host header. You'll also need to add some authorization parameters.

Creating a Date header

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a new header page appears.



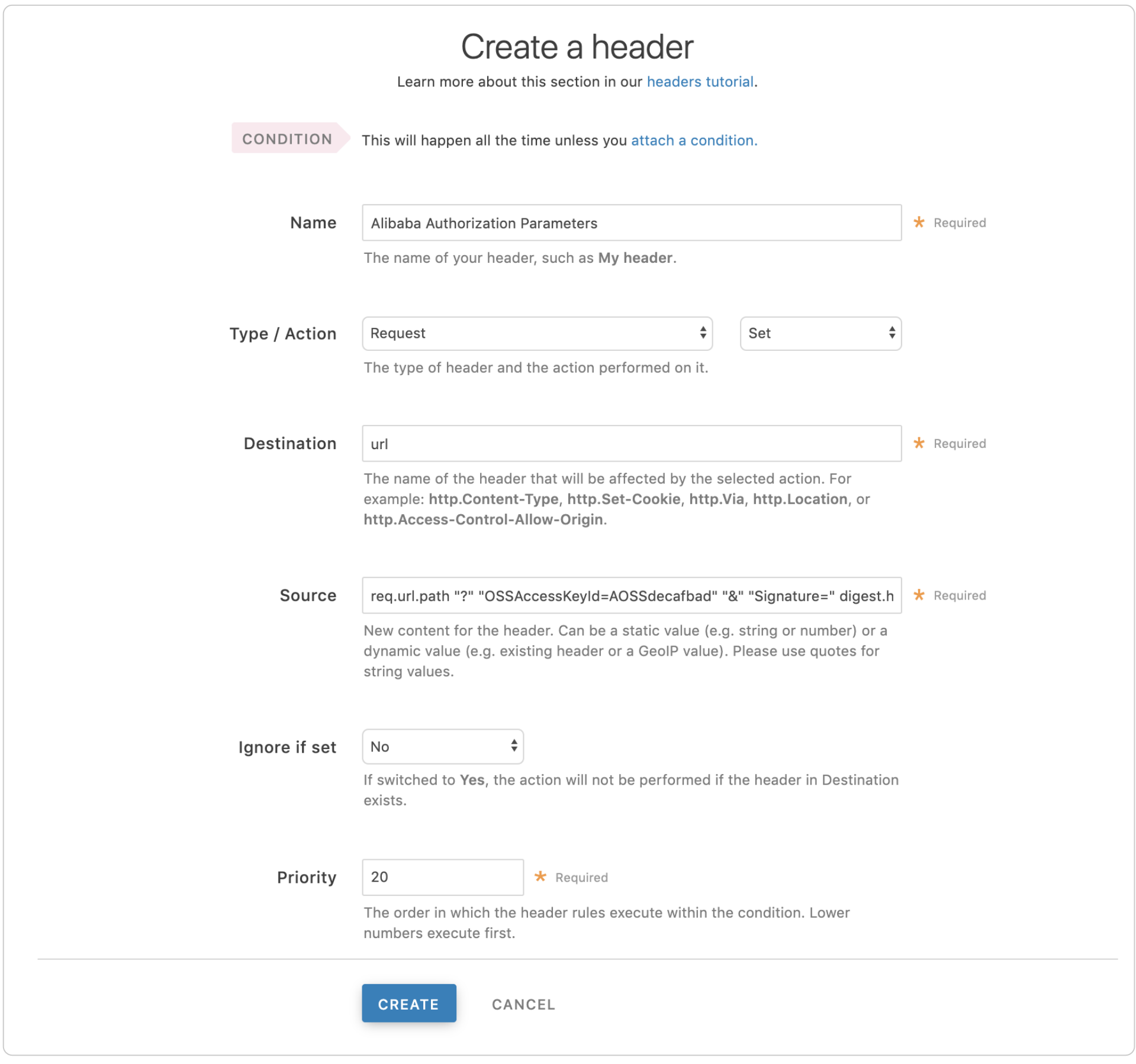
6. Fill out the **Create a new header** fields as follows:
   - In the **Name** field, enter `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Date`.
   - In the **Source** field, enter `var.ali_expires`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `19`.
7. Click the **Create** button. A new Date header appears on the Content page. You will use this later within the signature of the Authorization header.

Creating a Host header

1. Click the **Create header** button. The Create a new header page appears.
2. Fill out the **Create a new header** fields as follows:
   - In the **Name** field, enter `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Host`.
   - In the **Source** field, enter `"<your OSS domain>"`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `19`.
3. Click the **Create** button. A new Host header appears on the Content page.

Creating the Authorization header

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.

- In the **Destination** field, enter `url`.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, enter `20`.

3. In the **Source** field, enter the Authorization header information using the following format:

```
   req.url.path "?" "OSSAccessKeyId=<AccessKeyId>" "&" "Signature=" digest.hmac_sha1_base64("<AccessKeySecret>", if(re
q.method == "HEAD", "GET", req.method) LF LF LF req.http.Date LF "/<OSS bucket name>" req.url.path) "&" "Expires=" var.
ali_expires
```

   Replace `<AccessKeyId>`, `<AccessKeySecret>`, and `<OSS bucket name>` with the information you gathered before you began. For example:

```
   req.url.path "?" "OSSAccessKeyId=AOSSdecafbad" "&" "Signature=" urlencode(digest.hmac_sha1_base64("AOSSdeadbeef", if
(req.method == "HEAD", "GET", req.method) LF LF LF req.http.Date LF "/test123" req.url.path)) "&" "Expires=" var.ali_ex
pires
```

4. Click the **Create** button. A new Authorization header appears on the Content page.
5. Click the **Activate** button to deploy your configuration changes.

Setting up Fastly to use OSS private content using VCL Snippets

You can also put the configuration in a VCL Snippet with a priority of `20`.

```
declare local var.ali_bucket STRING;
declare local var.ali_region STRING;
declare local var.ali_access_key_id STRING;
declare local var.ali_access_key_secret STRING;
declare local var.ali_expires INTEGER;
declare local var.ali_canon STRING;
declare local var.ali_sig STRING;_

set var.ali_bucket = "test123";
set var.ali_region = "oss-cn-beijing";
set var.ali_access_key_id = "decafbad";
set var.ali_access_key_secret = "deadbeef";
set var.ali_expires  = std.atoi(now.sec);
set var.ali_expires += 60;


set req.http.Host = var.ali_bucket "." + var.ali_region + ".aliyuncs.com";
set req.http.Date = var.ali_expires;
set var.ali_canon = if(req.method == "HEAD", "GET", req.method) LF LF LF
                    req.http.Date LF "/" var.ali_bucket req.url.path;
set var.ali_sig   = digest.hmac_sha1_base64(var.alibaba_access_key_secret, var.ali_canon);

set req.url       = req.url.path;
set req.url       = querystring.set(req.url, "OSSAccessKeyId", var.alibaba_access_key_id);
set req.url       = querystring.set(req.url, "Signature",  var.ali_sig);
set req.url       = querystring.set(req.url, "Expires",  var.ali_expires);
```

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Amazon S3

Last updated: 2022-03-01

https://docs.fastly.com/en/guides/amazon-s3

Amazon S3 public and private buckets can be used as origins with Fastly.

Using Amazon S3 as an origin

To make your S3 data buckets available through Fastly, follow the steps below.

Creating a new service

Follow the instructions for creating a new service.

1. When you create the new domain and the new Host:
   - In the **Domain Name** field on the **Create a domain** page, enter the hostname you want to use as the URL (e.g., `cdn.example.com`).
   - In the **Hosts** field on the **Origins** page, enter the appropriate address for your Host using the format `<BUCKET>.s3.<REGION>.amazonaws.com`. Use the table in the Amazon S3 endpoints of the AWS general reference documentation as a guide. For example, if your bucket name is `fastlytestbucket` and your region is `us-east-2`, your hostname would be `fastlytestbucket.s3.us-east-2.amazonaws.com`.

     > **TIP**
     >
     > Most customers select a region close to the interconnect location they specify for shielding.

2. When you edit the Host details on the **Edit this host page**:
   - In the **Name** field, enter any descriptive name for your service if you haven't already done so.
   - In the **Address** field, ensure you've entered the appropriate address for your region (e.g., `fastlytestbucket.s3.us-east-2.amazonaws.com`). You entered this information during Host creation.
3. When you edit the Transport Layer Security (TLS) area information for your Host:
   - Leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.

     > **TIP**
     >
     > If you're using Amazon S3 to host a static website, **Enable TLS?** should be set to **No** instead of relying on the default. Amazon doesn't support TLS connections to S3 buckets with the static website hosting feature enabled. You can still use one of Fastly's TLS service options to secure connections between Fastly and clients.

   - Under the **SNI hostname** field, select the checkbox to **Match the SNI hostname to the Certificate hostname**. The address you entered during Host creation appears.
   - In the **Certificate hostname** field, enter `fastlytestbucket.s3.us-east-2.amazonaws.com`.
4. In the **Override host** field, enter an appropriate address for your Host (e.g., `fastlytestbucket.s3.us-east-2.amazonaws.com`). You entered this information during Host creation. If you're using an Amazon S3 private bucket leave this field blank.

Validating the DNS mapping

By default, we create a DNS mapping called **yourdomain.global.prod.fastly.net**. In the example above, it would be `cdn.example.com.global.prod.fastly.net`. Enter this URL in a browser to confirm it's working.

Creating a DNS alias

Create a DNS alias for the domain name you specified (e.g., CNAME `cdn.example.com` to `global-nossl.fastly.net`).

Verifying your results

Fastly will cache any content without an explicit `Cache-Control` header for 1 hour. You can verify whether you are sending any cache headers using curl. For example:

```
$ curl -I opscode-full-stack.s3.amazonaws.com

HTTP/1.1 200 OK
x-amz-id-2: ZpzRp7IWc6MJ8NtDEFGH12QBdk2CM1+RzVOngQbhMp2f2ZyalkFsZd4qPaLMkSlh
x-amz-request-id: ABV5032583242618
Date: Fri, 18 Mar 2012 17:15:38 GMT
Content-Type: application/xml
Transfer-Encoding: chunked
Server: AmazonS3
```

In this example, no Cache-Control headers are set so the default Time to Live (TTL) will be applied.

Enhanced cache control

If you need more control over how different types of assets are cached (e.g., Javascript files, images), check out our documentation on cache freshness.

Using an Amazon S3 private bucket

To use an Amazon S3 private bucket with Fastly, you must implement version 4 of Amazon's header-based authentication. You can do this using custom VCL. Start by obtaining the following information from AWS:

Item	Description
Bucket name	The name of your AWS S3 bucket. When you download items from your bucket, this is the string listed in the URL path or hostname of each object.
Region	The AWS region code of the location where your bucket resides (e.g., `us-east-1`).
Access key	The AWS access key string for an IAM account that has at least read permission on the bucket.
Secret key	The AWS secret access key paired with the access key above.

Once you have this information, you can configure your Fastly service to authenticate against your S3 bucket using header authentication by calculating the appropriate header value in VCL.

IMPORTANT

Consider leaving the **Override host** field for the origin blank in your service settings. This setting will override the Host header from the snippets shown here and may invalidate the signature that authenticates the information being sent.

Start by creating a regular VCL snippet. Give it a meaningful name, such as `AWS protected origin`. When you create the snippet, select **within subroutine** to specify its placement and choose **miss** as the subroutine type. Then, populate the **VCL** field with the following code (be sure to change specific values as noted to ones relevant to your own AWS bucket):

```
declare local var.awsAccessKey STRING;
declare local var.awsSecretKey STRING;
declare local var.awsS3Bucket STRING;
declare local var.awsRegion STRING;
declare local var.canonicalHeaders STRING;
declare local var.signedHeaders STRING;
declare local var.canonicalRequest STRING;
declare local var.canonicalQuery STRING;
declare local var.stringToSign STRING;
declare local var.dateStamp STRING;
declare local var.signature STRING;
declare local var.scope STRING;

set var.awsAccessKey = "YOUR_AWS_ACCESS_KEY";   # Change this value to your own data
set var.awsSecretKey = "YOUR_AWS_SECRET_KEY";   # Change this value to your own data
set var.awsS3Bucket = "YOUR_AWS_BUCKET_NAME";   # Change this value to your own data
set var.awsRegion = "YOUR_AWS_BUCKET_REGION";   # Change this value to your own data

if (req.method == "GET" && !req.backend.is_shield) {

  set bereq.http.x-amz-content-sha256 = digest.hash_sha256("");
  set bereq.http.x-amz-date = strftime({"%Y%m%dT%H%M%SZ"}, now);
  set bereq.http.host = var.awsS3Bucket ".s3." var.awsRegion ".amazonaws.com";
  set bereq.url = querystring.remove(bereq.url);
  set bereq.url = regsuball(urlencode(urldecode(bereq.url.path)), {"%2F"}, "/");
  set var.dateStamp = strftime({"%Y%m%d"}, now);
  set var.canonicalHeaders = ""
    "host:" bereq.http.host LF
    "x-amz-content-sha256:" bereq.http.x-amz-content-sha256 LF
    "x-amz-date:" bereq.http.x-amz-date LF
  ;
  set var.canonicalQuery = "";
  set var.signedHeaders = "host;x-amz-content-sha256;x-amz-date";
  set var.canonicalRequest = ""
    "GET" LF
    bereq.url.path LF
    var.canonicalQuery LF
    var.canonicalHeaders LF
    var.signedHeaders LF
    digest.hash_sha256("")
  ;

  set var.scope = var.dateStamp "/" var.awsRegion "/s3/aws4_request";

  set var.stringToSign = ""
    "AWS4-HMAC-SHA256" LF
    bereq.http.x-amz-date LF
    var.scope LF
    regsub(digest.hash_sha256(var.canonicalRequest),"^0x", "")
  ;

  set var.signature = digest.awsv4_hmac(
    var.awsSecretKey,
    var.dateStamp,
    var.awsRegion,
    "s3",
    var.stringToSign
  );

  set bereq.http.Authorization = "AWS4-HMAC-SHA256 "
    "Credential=" var.awsAccessKey "/" var.scope ", "
    "SignedHeaders=" var.signedHeaders ", "
    "Signature=" + regsub(var.signature,"^0x", "")
  ;
  unset bereq.http.Accept;
  unset bereq.http.Accept-Language;
  unset bereq.http.User-Agent;
  unset bereq.http.Fastly-Client-IP;
}
```

You may also remove the headers that AWS adds to the response. Do this by creating another VCL snippet. Give it a meaningful name, such as `Strip AWS response headers`. When you create the snippet, select **within subroutine** to specify its placement and choose **fetch** as the subroutine type. Then, place the following code in the **VCL** field:

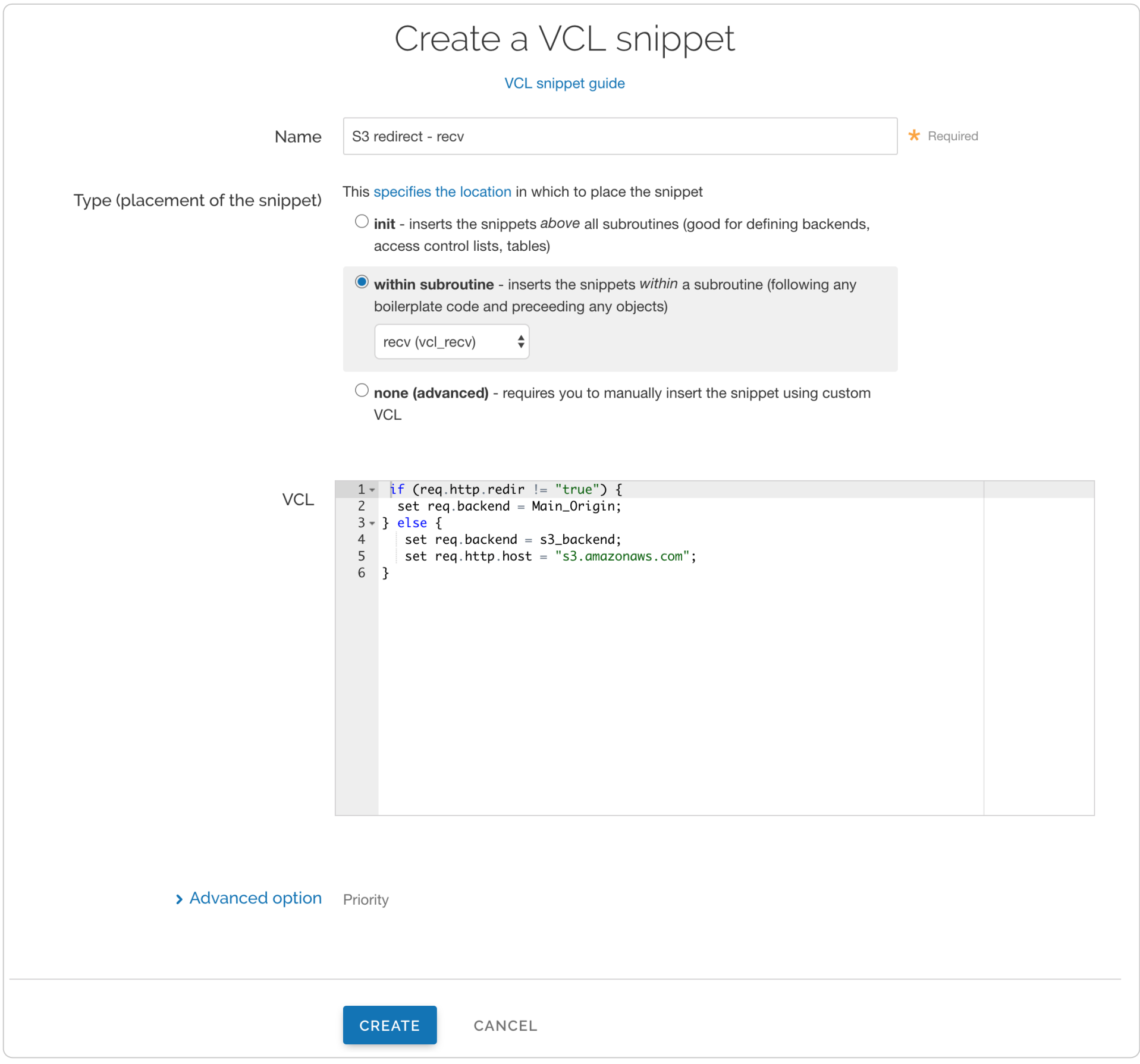
```
unset beresp.http.x-amz-id-2;
unset beresp.http.x-amz-request-id;
unset beresp.http.x-amz-delete-marker;
unset beresp.http.x-amz-version-id;
```

Following redirects to S3 objects and caching S3 responses

Using VCL Snippets, Fastly can follow redirects to S3 objects and cache the response.

To configure Fastly to follow redirects to S3 objects, follow the steps below:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the **Create snippet** button. The Create a VCL snippet page appears.



6. In the **Name** field, enter an appropriate name (e.g., `S3 redirect - recv`).
7. From the **Type (placement of the snippet)** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **recv (vcl_recv)**.
9. In the **VCL** field, add the following condition:

```
if (req.http.redir != "true") {
  set req.backend = Main_Origin;
} else {
  set req.backend = s3_backend;
  set req.http.host = "s3.amazonaws.com";
}
```

IMPORTANT

Be sure to set the `Main_Origin` and `s3_backend` to the actual name of your backends in the service to which you're applying these redirects. Find the exact names by clicking the VCL button at the top of the page while viewing the service and reviewing your VCL.

10. Click **Create** to create the snippet.
11. Click the **Create snippet** button again. The Create a VCL snippet page appears.

12. In the **Name** field, enter an appropriate name (e.g., `S3 redirect - deliver`).
13. From the **Type (placement of the snippet)** controls, select **within subroutine**.
14. From the **Select subroutine** menu, select **deliver (vcl_deliver)**.
15. In the **VCL** field, add the following condition:

```
if (resp.status == 302 || resp.status == 301) {
  set req.http.redir = "true";
  set req.url = regsub(resp.http.Location, "http://s3.amazonaws.com/(.*)$", "/\1");
  set req.http.Fastly-Force-Shield = "yes";
  restart;
}
```

16. Click **Create** to create the snippet.
17. Click the **Activate** button to deploy your configuration changes.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Backblaze B2 Cloud Storage

Last updated: 2022-03-01

https://docs.fastly.com/en/guides/backblaze-b2-cloud-storage

Backblaze B2 Cloud Storage (B2) public and private buckets can be used as origins with Fastly.

TIP

Backblaze offers an integration discount that eliminates egress costs to Fastly when using Backblaze B2 Cloud Storage as an origin. In addition, Backblaze also offers a migration program designed to offset many of the data transfer costs associated with switching from another cloud provider to Backblaze. To ensure your migration has minimal downtime, contact support@fastly.com.

Before you begin

Before you begin the setup and configuration steps required to use B2 as an origin, keep in mind the following:

- **You must have a valid Backblaze account.** Before you can create a new bucket and upload files to it for Fastly to use, you must first create a Backblaze account at the Backblaze website.
- **Backblaze provides two ways to set up and configure B2.** B2 can be set up and configured using either the Backblaze web interface or the B2 command line tool. Either creation method works for *public* buckets. To use *private* buckets, however, you must use the B2 command line tool. For additional details, including instructions on how to install the command line tool, read Backblaze's B2 documentation.
- **Backblaze provides two APIs for integrating with Backblaze B2 Cloud Storage.** You can use the B2 Cloud Storage API or the S3 Compatible API to make your B2 data buckets available through Fastly. The S3 Compatible API allows existing S3 integrations and SDKs to integrate with B2. Buckets and their specific application keys created prior to May 4th, 2020, however, cannot be used with the S3 Compatible API. For more information, read Backblaze's article on Getting Started with the S3 Compatible API.

Using Backblaze B2 as an origin

To use B2 as an origin, follow the steps below.

Creating a new bucket

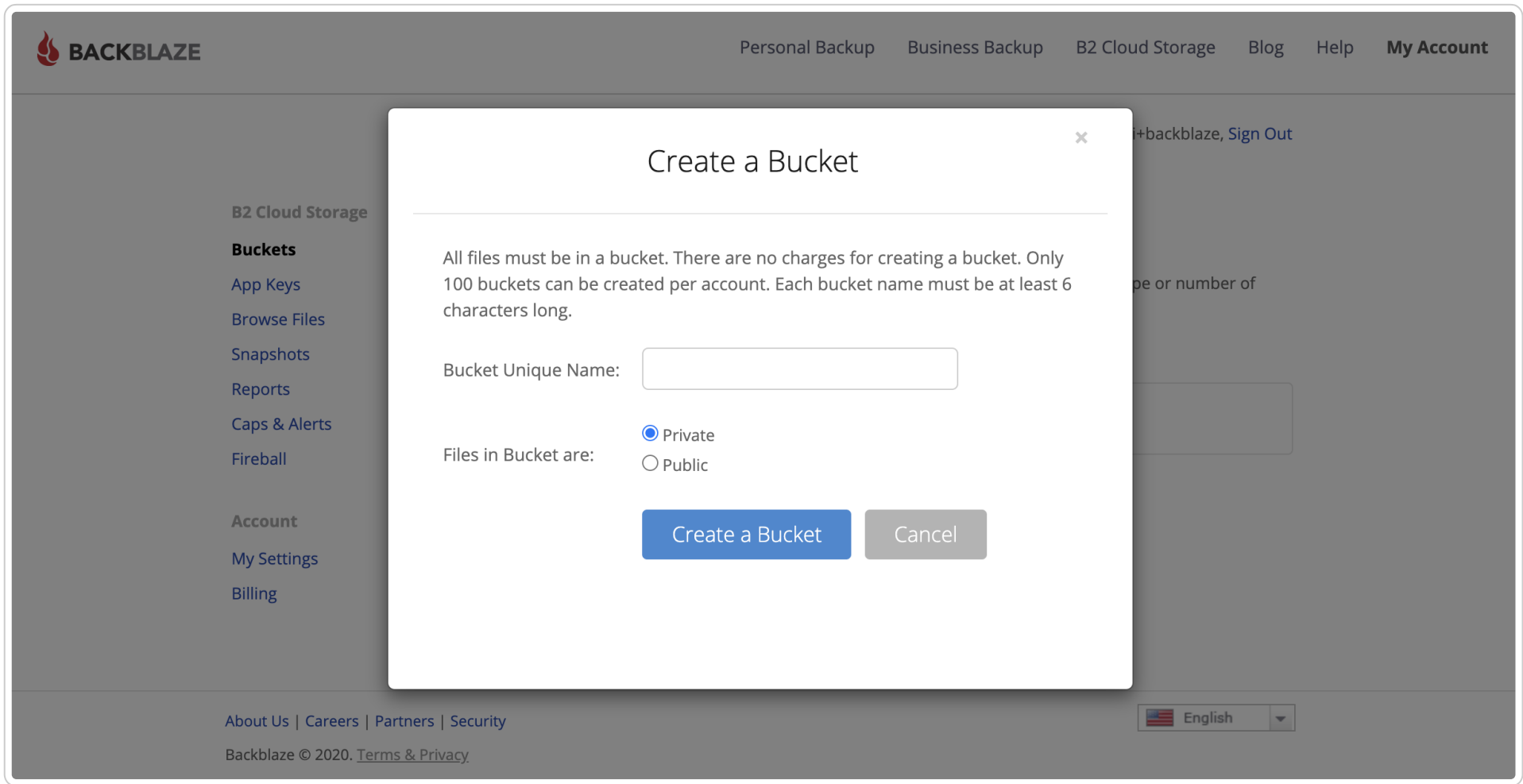
Data in B2 is stored in buckets. Follow these steps to create a new bucket via the B2 web interface.

TIP

The Backblaze Guide provides details on how to create a bucket using the command line tool.

1. Log in to your Backblaze account. Your Backblaze account settings page appears.
2. Click the **Buckets** link. The B2 Cloud Storage Buckets page appears.

3. Click the **Create a Bucket** link. The Create a Bucket window appears.



4. In the **Bucket Unique Name** field, enter a unique bucket name. Each bucket name must be at least 6 alphanumeric characters and can only use hyphens (`-`) as separators, not spaces.
5. Click the **Create a Bucket** button. The new bucket appears in the list of buckets on the B2 Cloud Storage Buckets page.
6. Upload a file to the new bucket you just created.

> **NOTE**
>
> Buckets created prior to May 4th, 2020 cannot be used with the S3 Compatible API. If you do not have any S3 Compatible buckets, Backblaze recommends creating a new bucket.

Uploading files to a new bucket

Once you've created a new bucket in which to store your data, follow these steps to upload files to it via the B2 web interface.

> **TIP**
>
> The Backblaze Guide provides details on how to upload files using the command line tool.

1. Click the **Buckets** link in the B2 web interface. The B2 Cloud Storage Buckets page appears.
2. Find the bucket details for the bucket you just created.
3. Click the **Upload/Download** button. The Browse Files page appears.
4. Click the **Upload** button. The upload window appears.
5. Either drag and drop any file into the window or click to use the file selection tools to find a file to be uploaded. The name and type of file at this stage doesn't matter. Any file will work. Once uploaded, the name of the file appears in the list of files for the bucket.
6. Find your bucket's assigned hostname so you can set up a Fastly service that interacts with B2.

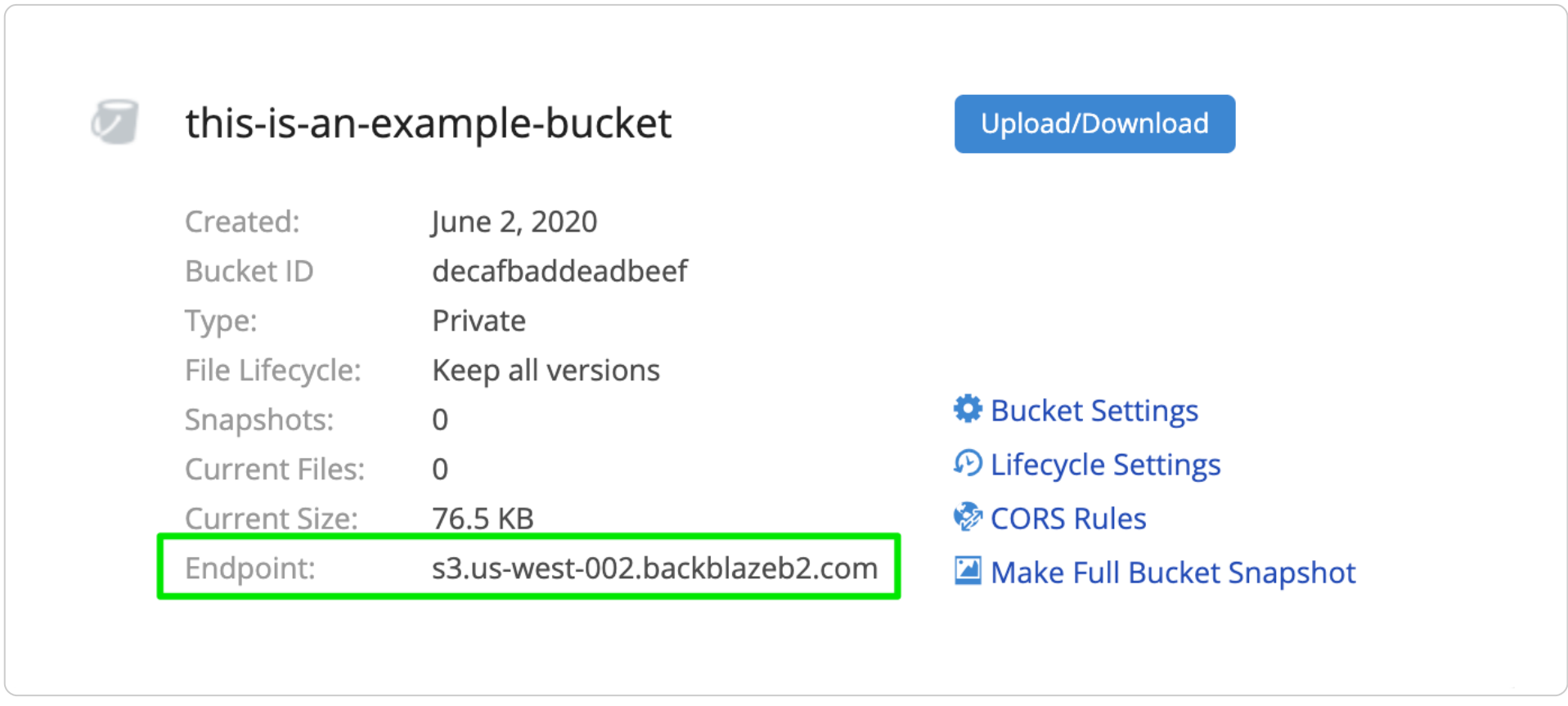
Finding your bucket's assigned hostname

To set up a Fastly service that interacts with your B2, you will need to know the hostname Backblaze assigned to the bucket you created and uploaded files to.

Find your hostname in one of the following ways:

- **Via the B2 web interface when you're using the standard B2 Cloud Storage API.** Click the name of the file you just uploaded and examine the **Friendly URL** and **Native URL** fields in the Details window that appears. The hostname is the text after the `https://` designator in each line that matches exactly.

- **Via the command line and the B2 Cloud Storage API.** Run the `b2 get-account-info` command on the command line and use the hostname from the `downloadUrl` attribute.
- **Via the B2 web interface when you're using the S3 Compatible API.** Click the **Buckets** link and find the bucket details for the bucket you just created. The hostname is the text in the Endpoint field.



Creating a Backblaze application key for private buckets

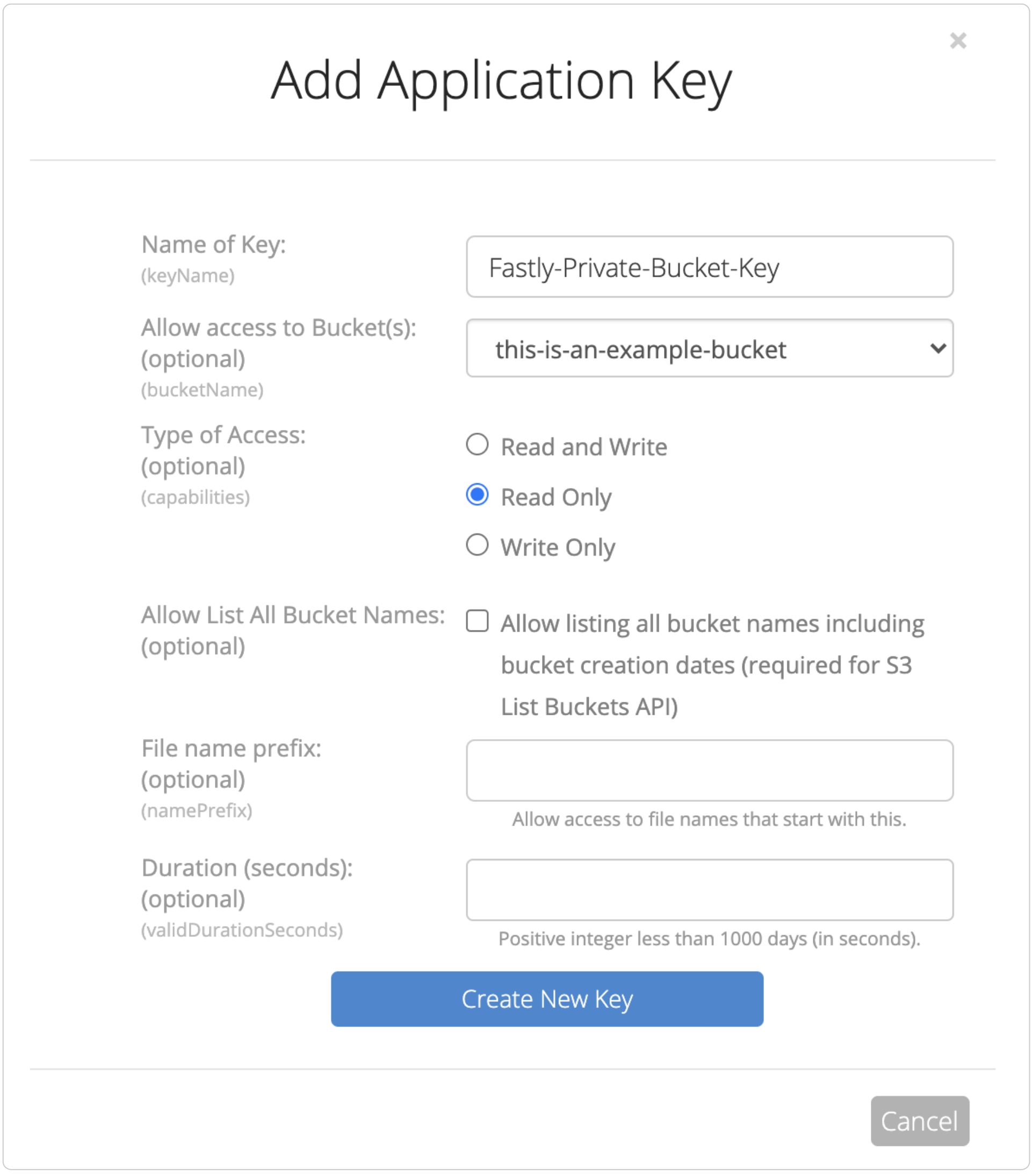
Your Backblaze master application key controls access to all buckets and files on your Backblaze account. If you plan to use a Backblaze B2 *private* bucket with Fastly, you should create an application key specific to the bucket.

> **NOTE**
>
> The Backblaze documentation provides more information about application keys. When creating application keys for your private bucket, we recommend using the least amount of privileges possible. You can optionally set the key to expire after a certain number of seconds (up to a maximum of 1000 days or 86,400,000 seconds). If you choose an expiration, however, you'll need to periodically create a new application key and then update your Fastly configuration accordingly each time.

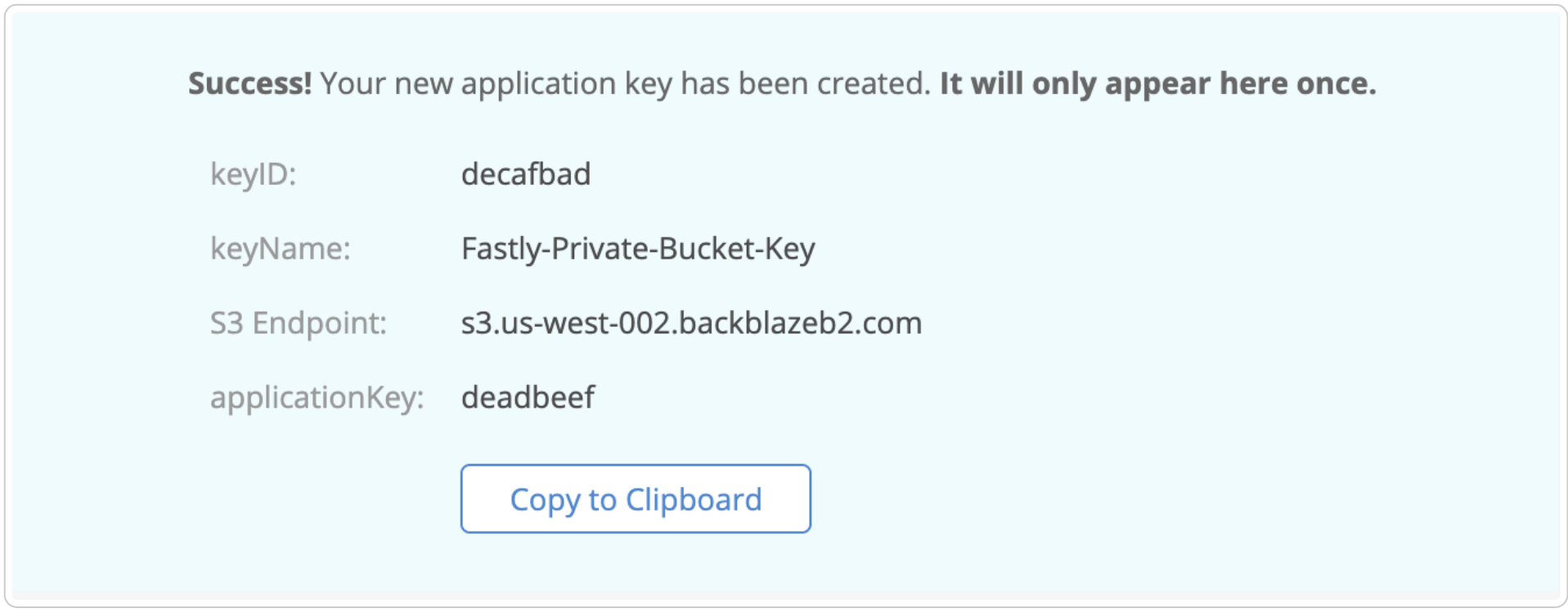
Via the web interface

To create an application key via the B2 web interface:

1. Click the **App Keys** link. The Application Keys page appears.
2. Click the **Add a New Application Key** button. The Add Application Key window appears.



3. Fill out the fields of the **Add Application Key** controls as follows:
   - In the **Name of Key** field, enter the name of your private bucket key. Key names are alphanumeric and can only use hyphens (`-`) as separators, not spaces.
   - From the **Allow access to Bucket(s)** menu, select the name of your private bucket.
   - From the **Type of Access** controls, select **Read Only**.
   - Leave the remaining optional controls and fields blank.
4. Click the **Create New Key** button. A success message and your new application key appear.



5. Immediately note the **keyID** and the **applicationKey** from the success message. You'll use this information when you implement header-based authentication with private objects.

Via the command line

To create an application key from the command line, run the `create-key` command as follows:

```
$ b2 create-key --bucket <bucketName> <keyName> shareFiles,listBuckets
```

where `<bucketName> <keyName>` represents the name of the bucket and key you created. For example:

```
$ b2 create-key --bucket this-is-an-example-bucket Fastly-Private-Bucket-Key shareFiles,listBuckets
```

The **keyID** and the **applicationKey** are the two values returned.

> **NOTE**
>
> Application keys created prior to May 4th, 2020 cannot be used with the S3 Compatible API.

Creating a new service

To create a new Fastly service, you must first create a new domain and then create a new host and edit it to accept traffic for B2. Instructions to do this appear in our guide to creating a new service. While completing these instructions, keep the following in mind:

1. When you create the new host, enter the B2 bucket's hostname in the **Hosts** field on the **Origins** page.
2. When you edit the host details on the **Edit this host page**, confirm the Transport Layer Security (TLS) area information for your host. Specifically, make sure you:
   - secure the connection between Fastly and your origin.
   - enter your bucket's hostname in the **Certificate hostname** field.
   - select the checkbox to match the SNI hostname to the Certificate hostname (it appears under the SNI hostname field).
3. Also when you edit the host, optionally enable shielding by choosing the appropriate shielding location from the **Shielding** menu. When using B2 Cloud Storage, this means you must choose a shielding location closest to the most appropriate Backblaze data center. For the data centers closest to:
   - Sacramento, California (in the US West region), choose **San Jose (SJC)** from the **Shielding** menu.
   - Phoenix, Arizona (in the US West region), choose **Palo Alto (PAO)** from the **Shielding** menu.
   - Amsterdam, Netherlands (in the EU central region), choose **Amsterdam (AMS)** from the **Shielding** menu.
4. Decide whether or not you should specify an override host:
   - If you're using the S3 Compatible API, skip this step and don't specify an override host.

- If you're not using the S3 Compatible API, in the **Override host** field, enter an appropriate address for your host (e.g., `s3-uswest-000.backblazeb2.com` or `f000.backblazeb2.com`).

Using the S3 Compatible API

Using the S3 Compatible API with public objects

To use the S3 Compatible API with public objects, you will need to make sure the `Host` header contains the name of your B2 Bucket. There are two ways to do this, both of which require you to get your region name which will be the 2nd part of your S3 Endpoint. So if your S3 Endpoint is `s3.us-west-000.backblazeb2.com`, this means your region will be `us-west-000`.

1. In the **Origin** you created set the **Override host** field to `<bucket>.s3.<region>.backblazeb2.com` (e.g., `testing.s3.uswest-000.backblazeb2.com`)

2. Create a VCL Snippet. When you create the snippet, select **within subroutine** to specify its placement and choose **miss** as the subroutine type. Then, populate the **VCL** field with the following code. Be sure to change specific values as noted to ones relevant to your own B2 bucket - in this case `var.b2Bucket` would be `"testing"` and `var.b2Region` would be `"uswest-000"`.

```
declare local var.b2Bucket STRING;
declare local var.b2Region STRING;
set var.b2Bucket = "YOUR_B2_BUCKET_NAME";   # Change this value to your own data
set var.b2Region = "YOUR_B2_BUCKET_REGION"; # Change this value to your own data

if (req.method == "GET" && !req.backend.is_shield) {
  set bereq.http.host = var.b2Bucket ".s3." var.b2Region ".backblazeb2.com";
}
```

Using the S3 Compatible API with private objects

To use a Backblaze B2 private bucket with Fastly, you must implement version 4 of Amazon's header-based authentication. You can do this using custom VCL.

Start by obtaining the following information from Backblaze (see Creating a Backblaze application key for private buckets):

Item	Description
Bucket name	The name of your Backblaze B2 bucket. When you download items from your bucket, this is the string listed in the URL path or hostname of each object.
Region	The Backblaze region code of the location where your bucket resides (e.g., `uswest-000`).
Access key	The Backblaze keyID for the App Key that has at least read permission on the bucket.
Secret key	The Backblaze applicationKey paired with the access key above.

Once you have this information, you can configure your Fastly service to authenticate against your B2 bucket using header authentication by calculating the appropriate header value in VCL.

Start by creating a regular VCL snippet. Give it a meaningful name, such as `AWS protected origin`. When you create the snippet, select **within subroutine** to specify its placement and choose **miss** as the subroutine type. Then, populate the **VCL** field with the following code (be sure to change specific values as noted to ones relevant to your own AWS bucket):

```
declare local var.b2AccessKey STRING;
declare local var.b2SecretKey STRING;
declare local var.b2Bucket STRING;
declare local var.b2Region STRING;
declare local var.canonicalHeaders STRING;
declare local var.signedHeaders STRING;
declare local var.canonicalRequest STRING;
declare local var.canonicalQuery STRING;
declare local var.stringToSign STRING;
declare local var.dateStamp STRING;
declare local var.signature STRING;
declare local var.scope STRING;

set var.b2AccessKey = "YOUR_B2_ACCESS_KEY";   # Change this value to your own data
set var.b2SecretKey = "YOUR_B2_SECRET_KEY";   # Change this value to your own data
set var.b2Bucket = "YOUR_B2_BUCKET_NAME";     # Change this value to your own data
set var.b2Region = "YOUR_B2_BUCKET_REGION";   # Change this value to your own data

if (req.method == "GET" && !req.backend.is_shield) {

  set bereq.http.x-amz-content-sha256 = digest.hash_sha256("");
  set bereq.http.x-amz-date = strftime({"%Y%m%dT%H%M%SZ"}, now);
  set bereq.http.host = var.b2Bucket ".s3." var.b2Region ".backblazeb2.com";
  set bereq.url = querystring.remove(bereq.url);
  set bereq.url = regsuball(urlencode(urldecode(bereq.url.path)), {"%2F"}, "/");
  set var.dateStamp = strftime({"%Y%m%d"}, now);
  set var.canonicalHeaders = ""
    "host:" bereq.http.host LF
    "x-amz-content-sha256:" bereq.http.x-amz-content-sha256 LF
    "x-amz-date:" bereq.http.x-amz-date LF
  ;
  set var.canonicalQuery = "";
  set var.signedHeaders = "host;x-amz-content-sha256;x-amz-date";
  set var.canonicalRequest = ""
    "GET" LF
    bereq.url.path LF
    var.canonicalQuery LF
    var.canonicalHeaders LF
    var.signedHeaders LF
    digest.hash_sha256("")
  ;

  set var.scope = var.dateStamp "/" var.b2Region "/s3/aws4_request";

  set var.stringToSign = ""
    "AWS4-HMAC-SHA256" LF
    bereq.http.x-amz-date LF
    var.scope LF
    regsub(digest.hash_sha256(var.canonicalRequest),"^0x", "")
  ;

  set var.signature = digest.awsv4_hmac(
    var.b2SecretKey,
    var.dateStamp,
    var.b2Region,
    "s3",
    var.stringToSign
  );

  set bereq.http.Authorization = "AWS4-HMAC-SHA256 "
    "Credential=" var.b2AccessKey "/" var.scope ", "
    "SignedHeaders=" var.signedHeaders ", "
    "Signature=" + regsub(var.signature,"^0x", "")
  ;
  unset bereq.http.Accept;
  unset bereq.http.Accept-Language;
  unset bereq.http.User-Agent;
  unset bereq.http.Fastly-Client-IP;
}
```

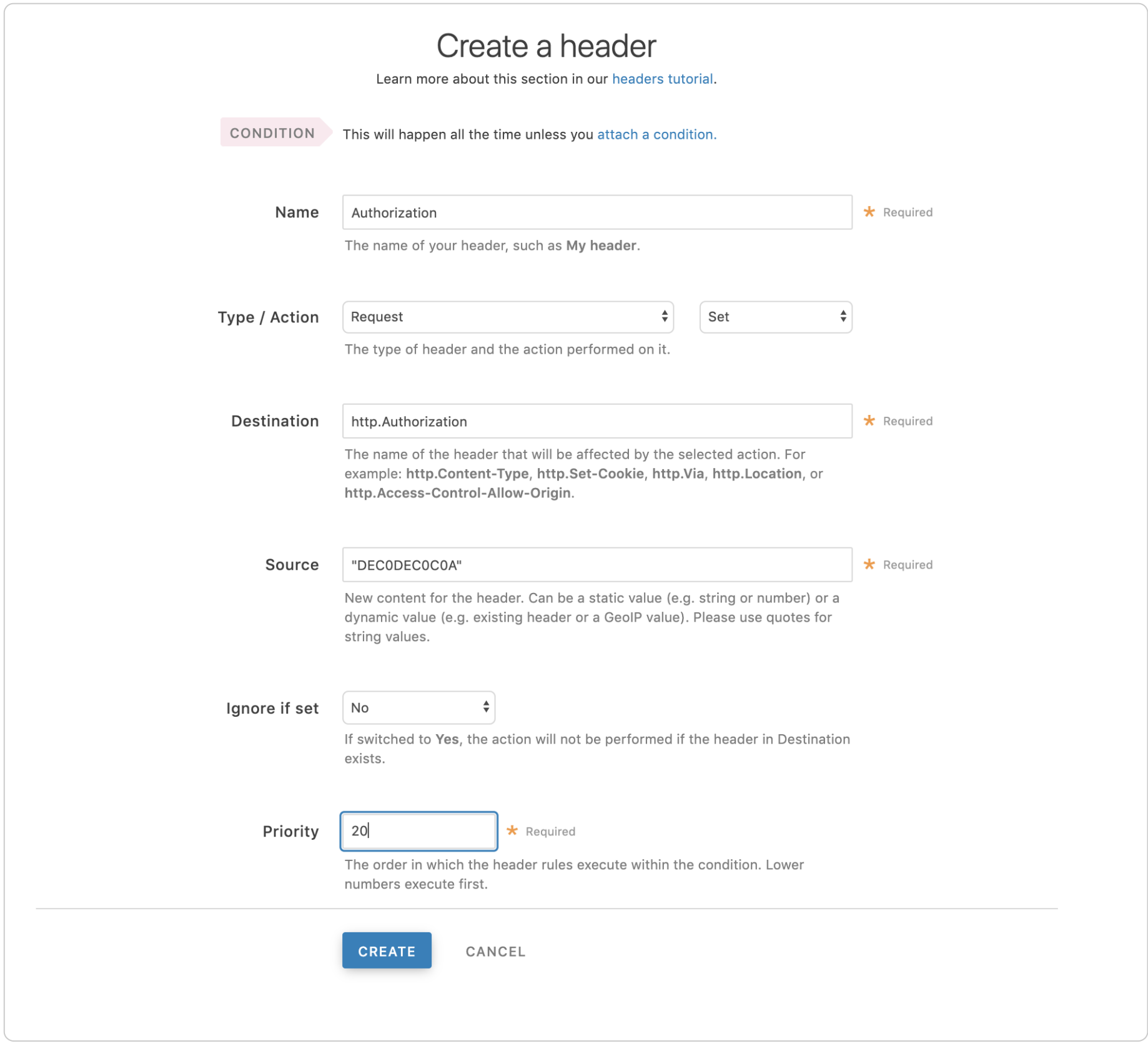
Using the B2 API

Public Objects

You'll need to make sure the URL contains your bucket name. There are two ways to do this.

- Using a **Header** object.

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Rewrite B2 URL`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `url`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `20`.
3. In the **Source** field, enter `"/file/<Bucket Name>" req.url`.
4. Click the **Create** button. A new Authorization header appears on the Content page.
5. Click the **Activate** button to deploy your configuration changes.

- Alternatively create a VCL Snippet. When you create the snippet, select **within subroutine** to specify its placement and choose **miss** as the subroutine type. Then, populate the **VCL** field with the following code. Be sure to change the variable to the name of your own B2 bucket.

```
declare local var.b2Bucket STRING;
set var.b2Bucket = "YOUR_B2_BUCKET_NAME";   # Change this value to your own data

if (req.method == "GET" && !req.backend.is_shield) {
  set bereq.url = "/file/" var.b2Bucket bereq.url;
}
```

Private Objects

1. To use a Backblaze B2 private bucket with Fastly, you must obtain an `Authorization Token`. This must be obtained via the command line.

2. You'll now need to authorize the command line tool with the application key you obtained.

```
$ b2 authorize-account <keyID> <applicationKey>
```

3. You will now need to get an authorization token for the private bucket.

```
$ b2 get-download-auth <bucket>
```

e.g

```
$ b2 get-download-auth testing
```

This will create a token that is valid for 86400 seconds (i.e 1 day), the default. You can optionally change the expiration time from anywhere between 1s and 604,800 seconds (i.e 1 week).

```
$ b2 get-download-auth --duration 604800 testing
```

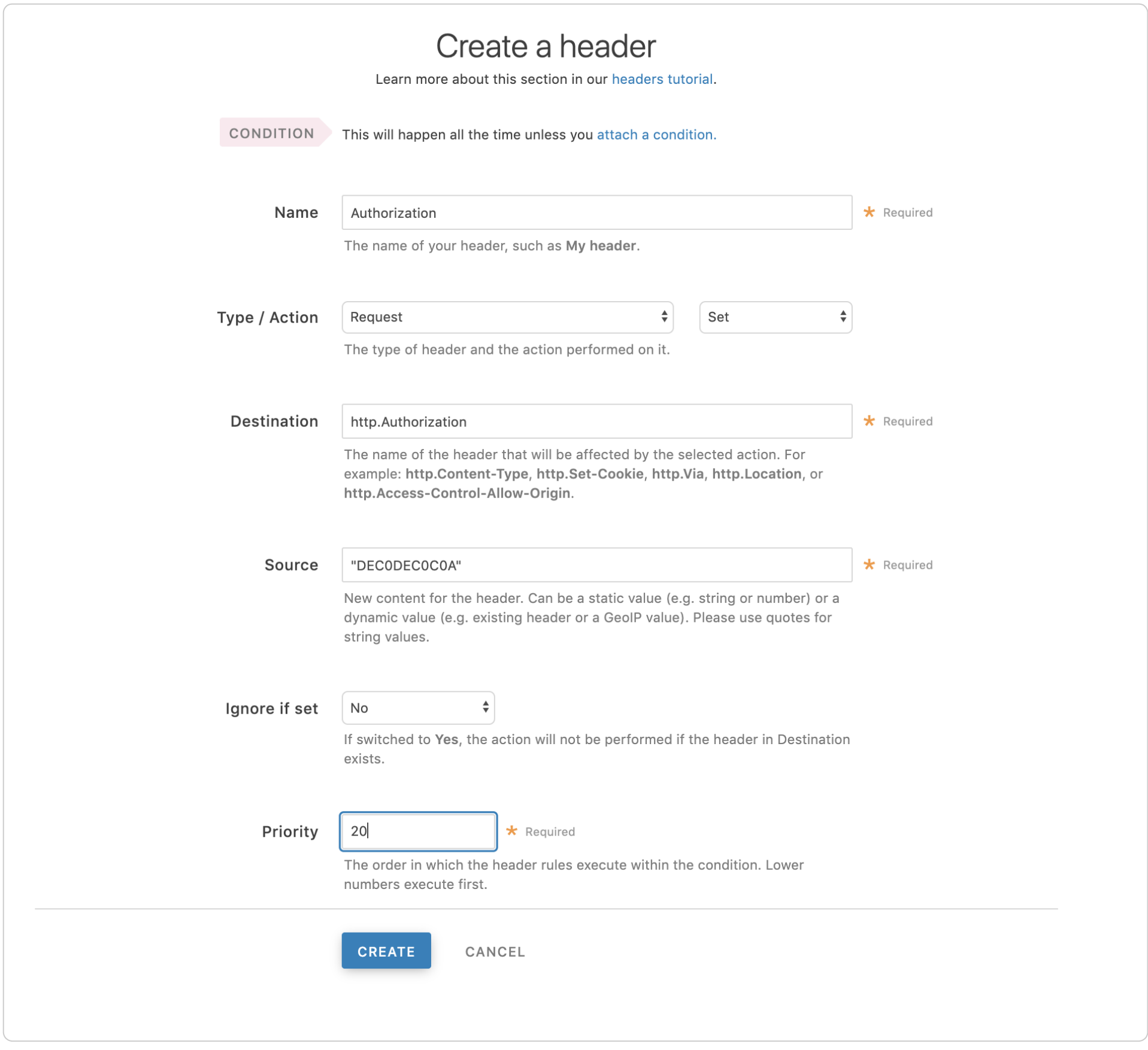
Take note of the generated token.

> **NOTE**
>
> You will need to regenerate an authorization token and update your Fastly configuration before the end of the expiration time. A good way to do this would be through Fastly's versionless Edge Dictionaries.

Passing a generated token to Backblaze

There are two ways you can pass the generated token to Backblaze. The first is using an `Authorization` header. This is the recommended method.

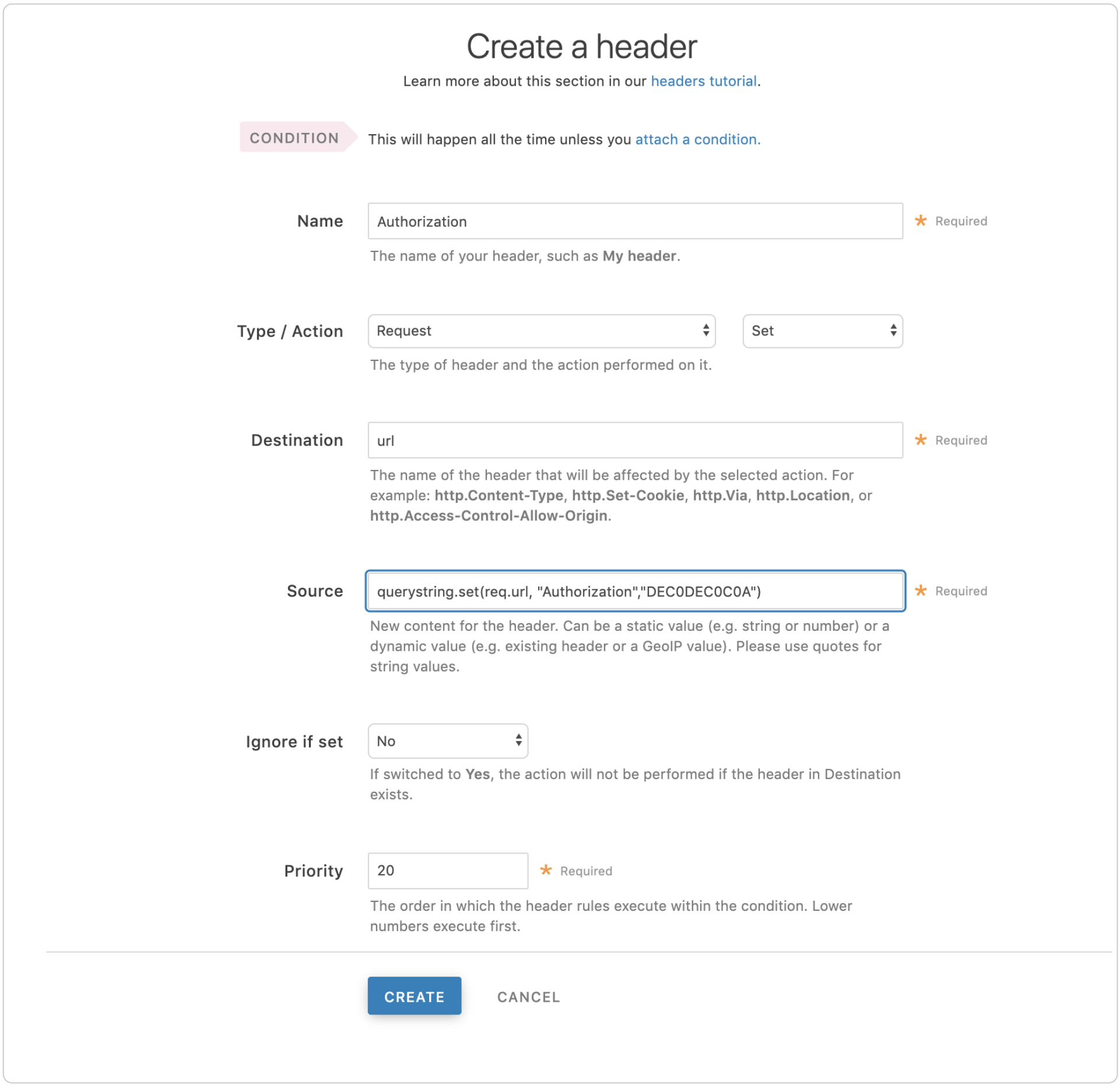
1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Authorization`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `20`.
3. In the **Source** field, enter the **Authorization Token** generated in the command line tool, surrounded by quotes. For example, if the token generated was `DEC0DEC0C0A`, then the **Source** field would be `"DEC0DEC0C0A"`
4. Click the **Create** button. A new Authorization header appears on the Content page.
5. Click the **Activate** button to deploy your configuration changes.

Alternatively, the second way is to pass an `Authorization` query parameter.

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `url`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `20`.
3. In the **Source** field, enter the header authorization information using the following format:

   ```
   querystring.set(req.url, "Authorization", "<Authorization Token>")
   ```

   Using the previous example, that would be:

   ```
   querystring.set(req.url, "Authorization", "DEC0DEC0C0A")
   ```

4. Click the **Create** button. A new Authorization header appears on the Content page.
5. Click the **Activate** button to deploy your configuration changes.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Data transfer with Backblaze B2

Last updated: 2020-09-15

https://docs.fastly.com/en/guides/data-transfer-with-backblaze-b2

Fastly has partnered with Backblaze to provide an integration between Fastly and Backblaze B2 Cloud Storage services. Specifically, there are no egress costs from Backblaze to Fastly when you configure a Backblaze B2 service as your Fastly origin.

To offset many of the data transfer costs associated with switching from another cloud provider, Backblaze offers a migration program. Backblaze will cover migration fees for customers migrating over 50 TB of data from the US, Canada, and Europe regions of other Cloud Providers, and storing it with them for at least 12 months. For customers migrating less than 50 TB of data, Backblaze offers discounted transfer rates.

Read more about the migration program and review estimated cost saving calculations in their guide to Migrate to B2 Cloud Storage. To ensure your migration has minimal downtime, contact support@fastly.com.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

DigitalOcean Spaces

Last updated: 2022-03-01

https://docs.fastly.com/en/guides/digitalocean-spaces

DigitalOcean Spaces public and private Spaces can be used as origins with Fastly.

Using DigitalOcean Spaces as an origin

To make your DigitalOcean Spaces available through Fastly, follow the steps below.

Creating a new service

Follow the instructions for creating a new service.

1. When you create the new domain and the new host:
   - In the **Domain Name** field on the **Create a domain** page, enter the hostname you want to use as the URL (e.g., `cdn.example.com`).
   - In the **Hosts** field on the **Origins** page, enter the appropriate address for your Host using the format `<SPACE>.<REGION>.digitaloceanspaces.com`. For example, if your space name is `test123` and your region is `nyc3`, your hostname would be `test123.nyc3.digitaloceanspaces.com`.
2. When you edit the host details on the **Edit this host page**:
   - In the **Name** field, enter any descriptive name for your service if you haven't already done so.
   - In the **Address** field, ensure you've entered the appropriate address for your Host (e.g., `test123.nyc3.digitaloceanspaces.com`). You entered this information during Host creation.
3. When you edit the Transport Layer Security (TLS) area information for your host:
   - Leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
   - In the **Certificate hostname** field, enter the same address that appears in the Address field (e.g., `test123.nyc3.digitaloceanspaces.com`).
   - Under the **SNI hostname** field, select the checkbox to **Match the SNI hostname to the Certificate hostname**. The address you entered during Host creation appears.
4. In the **Override host** field, enter an appropriate address for your Host (e.g., `test123.nyc3.digitaloceanspaces.com`). You entered this information during Host creation.

Testing your results

By default, we create DNS mapping called `yourdomain.global.prod.fastly.net`. In the example above, it would be `cdn.example.com.global.prod.fastly.net`. Create a DNS alias for the domain name you specified (e.g., CNAME `cdn.example.com` to `global-nossl.fastly.net`).

Fastly will cache any content without an explicit `Cache-Control` header for 1 hour. You can verify whether you are sending any cache headers using curl. For example:

```
$ curl -I opscode-full-stack.nyc3.digitaloceanspaces.com

HTTP/1.1 200 OK
x-amz-id-2: ZpzRp7IWc6MJ8NtDEFGH12QBdk2CM1+RzVOngQbhMp2f2ZyalkFsZd4qPaLMkSlh
x-amz-request-id: ABV5032583242618
Date: Fri, 18 Mar 2012 17:15:38 GMT
Content-Type: application/xml
Transfer-Encoding: chunked
```

In this example, no Cache-Control headers are set so default TTL will be applied.

Enhanced cache control

If you need more control over how different types of assets are cached (e.g., Javascript files, images), refer to our cache freshness documentation.

Using private DigitalOcean Spaces

To use a private DigitalOcean Space with Fastly, follow the instructions below.

Before you begin

Be sure you've already made your Spaces data available to Fastly by pointing to the right Space and setting your origin to port 443. This needs to be done before authenticating.

Be sure you've got the access key, secret key, and Space name on hand. The DigitalOcean Spaces Authorization header takes the following form:

```
Authorization: AWS `AWSAccessKeyId`:`Signature`
```

From the DigitalOcean website you will need the following information:

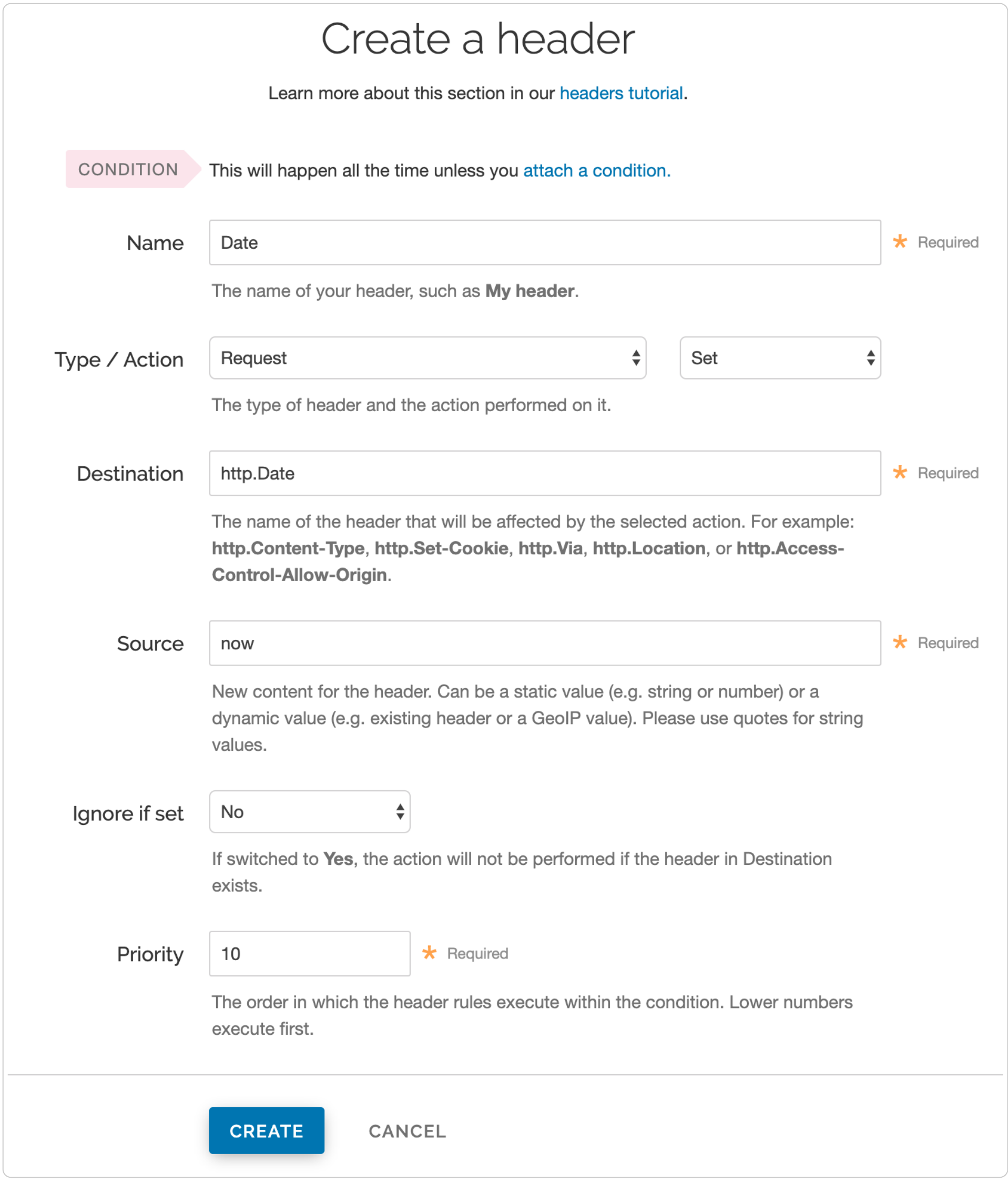
1. the **access key** and **secret key**
2. your **Space** name

Setting up Fastly to use a private DigitalOcean Space

In order to use a private DigitalOcean Space with Fastly, create two headers, a Date header (for use with the authorization Signature) and an Authorization header.

Create a Date header

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.

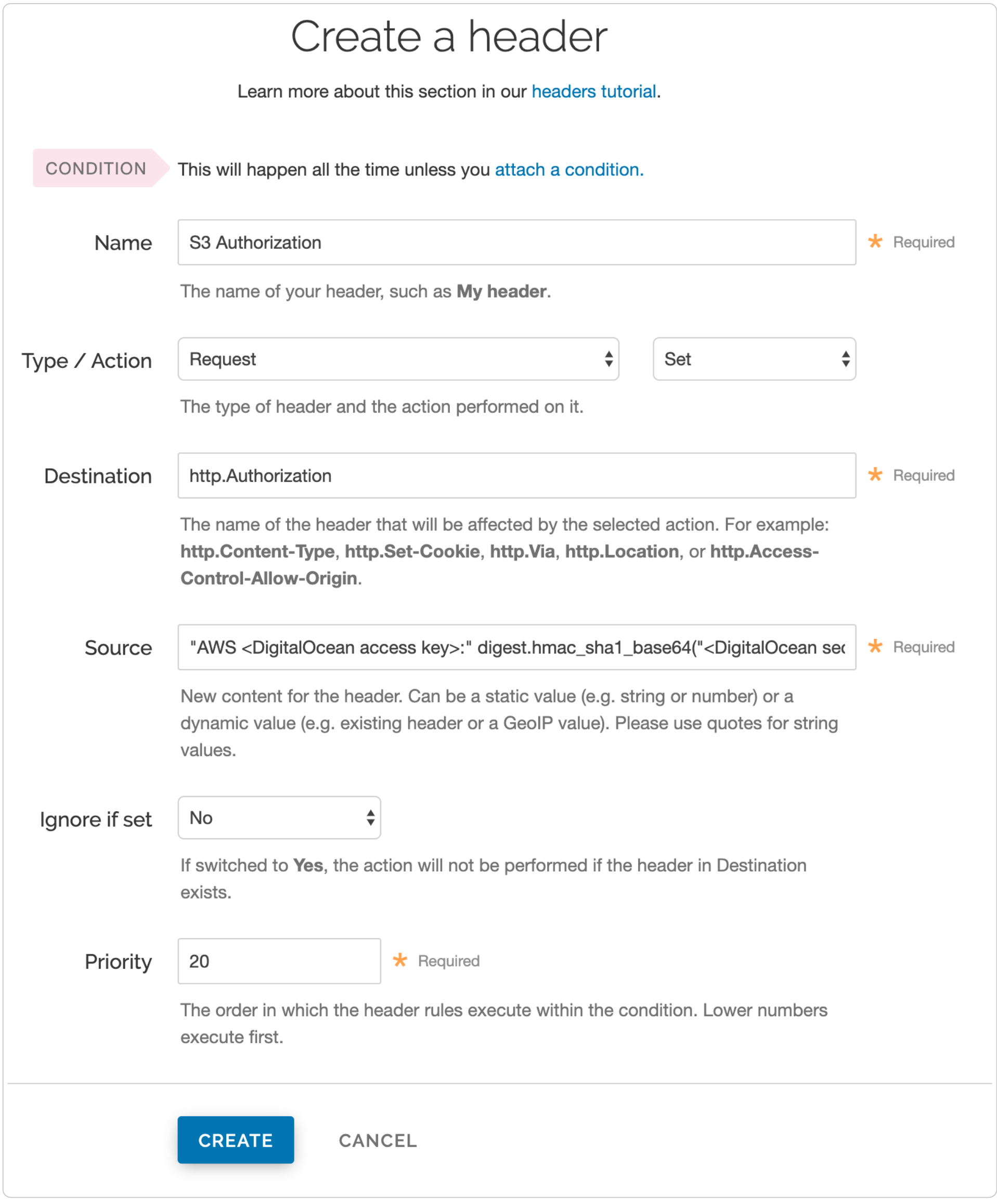


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Date`.
   - In the **Source** field, enter `now`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
7. Click the **Create** button. A new Date header appears on the Content page. You will use this later within the Signature of the Authorization header.

Create an Authorization header

Next, create the Authorization header with the specifications listed below.

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Spaces Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Authorization`.
   - From the **Ignore if set** menu, select **No**.

- In the **Priority** field, enter `20`.

3. In the **Source** field, enter the header authorization information using the following format:

```
"AWS <DigitalOcean access key>:" digest.hmac_sha1_base64("<DigitalOcean secret key>", if(req.method == "HEAD", "GET", r
eq.method) LF LF LF req.http.Date LF "/<Space name>" req.url.path)
```

replacing `<DigitalOcean access key>`, `<DigitalOcean secret key ID>`, and `<Space name>` with the information you gathered before you began. For example:

```
"AWS JKCAUEFV2ONFFOFMSSLA:" digest.hmac_sha1_base64("P2WPSu68Bfl89j72vT+bXYZB7SjlOwhT4whqt27", if(req.method == "HEAD",
"GET", req.method) LF LF LF req.http.Date LF "/test123" req.url.path)
```

4. Click the **Create** button. The new Authorization header appears on the Content page.

A detailed look at the Source field

So what's going on in the Source field of the Authorization header? Here's the basic format:

`AWS<Access Key><Signature Function><key><message>`

It tells us the following:

Element	Description
`AWS`	A constant placed before the access key. It's always AWS.
`access key`	The access key from your DigitalOcean account. We used `JKCAUEFV2ONFFOFMSSLA` in this example.
`signature function`	The algorithm used to validate the key and message of the signature. We used `digest.hmac_sha1_base64(<key>, <message>)` in this example.
`key`	The secret key from your DigitalOcean account. We used `P2WPSu68BfI89j72vT+bXYZB7SjIOwhT4whqt27` in this example.
`message`	The UTF-8 encoding of the StringToSign. See the table below for a break down of each portion of the message.

The message that's part of the Source field in the Authorization header takes on this basic format:

`<HTTP-verb></n><Content-MD5>/n<Content-Type></n><Date></n><CanonicalizedAmzHeader></n><CanonicalizedResource>`

It tells us the following:

Element	Description
`HTTP-verb`	The REST verb. We use `req.method` in this example. We rewrite HEAD to GET because Varnish does this internally before sending requests to origin.
`/n`	A newline indicator constant. It's always /n.
`Content-MD5`	The content-md5 header value, used as a message integrity check. It's often left blank. We use `LF` (line feed) in this example.
`Content-Type`	The content-type header value, used to specify the MIME-type. It's often left blank. We use `LF` in this example.
`Date`	The date and time stamp. We use `req.http.Date` (which we created first as a separate header in the steps above).
`CanonicalizedAmzHeader`	The x-amz headers, which customize your Spaces implementation. It's often left blank. We use `LF` in this example.
`CanonicalizedResource`	Your DigitalOcean Space name. We use `"/test123"` in this example.

Following redirects to Spaces objects and caching Spaces responses

With custom VCL, Fastly can follow redirects to Spaces objects and cache the Spaces response as well as the 301 or 302 response separately.

Be sure to read our instructions about mixing and matching Fastly VCL with custom VCL. It's important to include the entire VCL boilerplate if you do not intend to override the Fastly default settings.

To configure Fastly to follow redirects to Spaces objects, insert the following VCL in your custom VCL:

Within vcl_recv

```
sub vcl_recv {

  if (req.http.redir != "true") {
    set req.backend = Main_Origin;
  } else {
    set req.backend = spaces_backend;
    set req.http.host = "nyc3.digitaloceanspaces.com";
  }

#FASTLY recv

  if (req.method != "HEAD" && req.method != "GET" && req.method != "FASTLYPURGE") {
    return(pass);
  }
  return(lookup);

}
```

Within vcl_deliver

```
sub vcl_deliver {

  if (resp.status == 302 || resp.status == 301) {
    set req.http.redir = "true";
    set req.url = regsub(resp.http.Location, "http://nyc3.digitaloceanspaces.com/(.*)$", "/\1");
    set req.http.Fastly-Force-Shield = "yes";
    restart;
  }

#FASTLY deliver

  return(deliver);
}
```

Be sure to set the `Main_Origin` and `spaces_backend` to the actual name of your backends in the service to which you're applying these redirects. You can find the exact names by reviewing your VCL. Simply click on the VCL button at the top of the page while viewing the service.

Once you added these VCL snippets to your custom VCL, upload the VCL file and then activate the new version of your service to apply the changes.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Discounted egress from Google

Last updated: 2021-06-07

https://docs.fastly.com/en/guides/discounted-egress-from-google

Fastly has partnered with Google to provide an integration between Fastly and Google services. Specifically, the integration allows you to connect Google's Cloud Platform service directly to Fastly's content delivery network services via private network interconnections (direct PNIs), thus speeding up your content delivery and optimizing backend workload.

When you sign up for Fastly services and configure a Google Cloud Platform service as your origin server, you designate a specific point of presence (POP) to serve as an Origin Shield that handles cached content from their servers.

Requests from Fastly POPs to these specific interconnect locations are routed over Fastly's network, leveraging optimized TCP connection handling, quick-start, and opened connections to enable fast response times between POPs and through to the end-user. Fastly ensures requests go directly to the Origin Shield instead of the origin servers. Only requests that the entire network has never handled will go back to the Google Cloud Platform service.

WARNING

We encourage you to read Google's CDN interconnect pricing. Despite this connection to Fastly's services being in place, in certain circumstances your data may egress from Google over the public internet rather than via the private network interconnection. In such cases, your traffic to the public internet will be metered according to your commercial arrangement with Google.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Google Cloud Storage

Last updated: 2022-03-01

https://docs.fastly.com/en/guides/google-cloud-storage

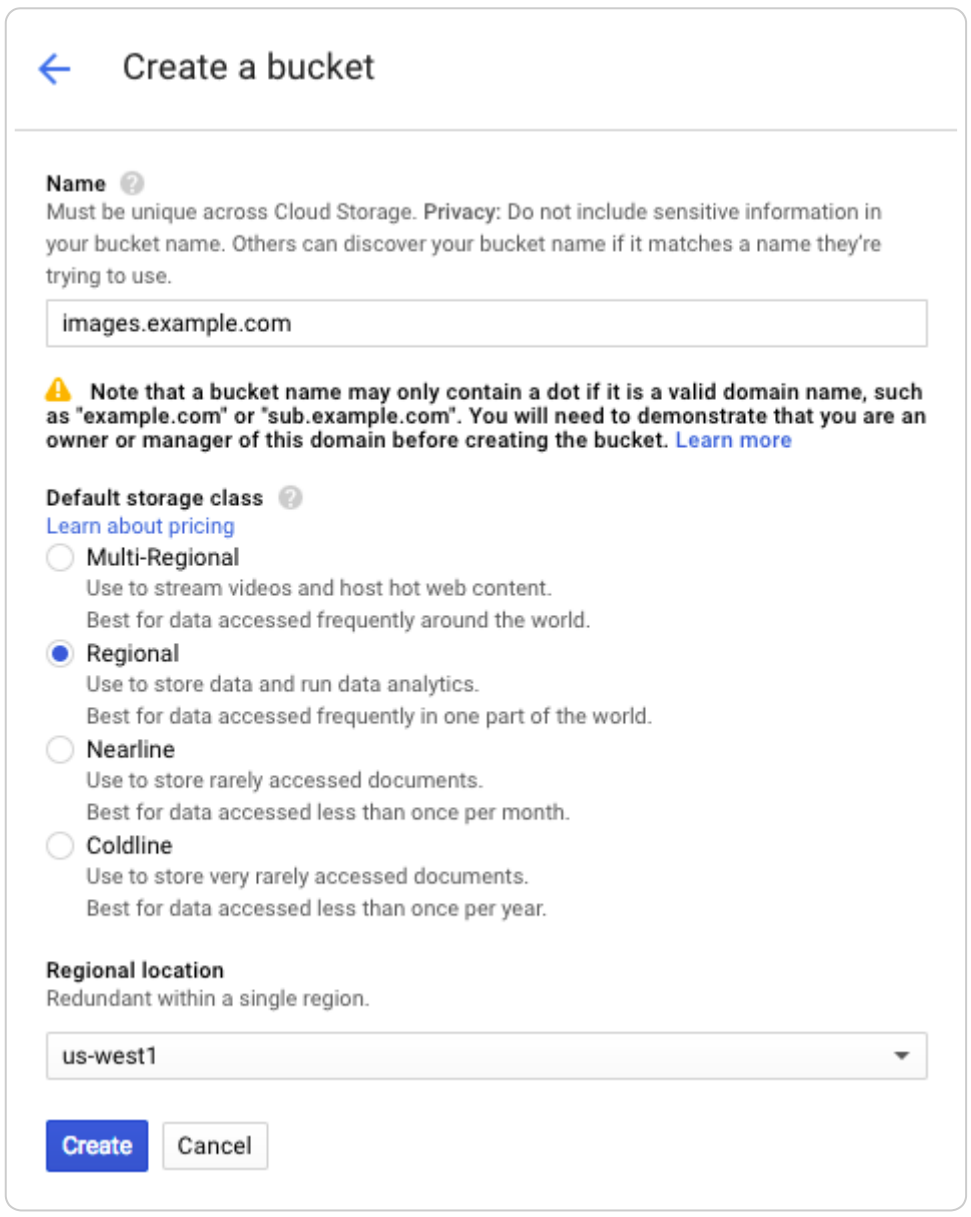
Google Cloud Storage (GCS) can be used as an origin server with your Fastly services once you set up and configure your GCS account and link it to a Fastly service. It can also be configured to use private content. This speeds up your content delivery and reduces your origin's workload and response times with the dedicated links between Google and Fastly's POPs.

Using GCS as an origin server

To make your GCS data available through Fastly, follow the steps below.

Setting up and configuring your GCS account

1. Sign up for Google Cloud Storage.
2. Create a bucket to store your origin's data. The Create a bucket window appears.



3. Fill out the **Create a bucket** fields as follows:
    - In the **Name** field, enter a name for your bucket (e.g., `mybucket`). You can also create a domain-named bucket (e.g., `images.example.com`), but you'll be required to verify your domain ownership using Google's Search Console, if you have not already done so. See the instructions on Google's website. Remember the name you type. You'll need it to connect your GCS bucket to your Fastly service.
    - In the **Default storage class** area, select **Regional**.
    - From the **Regional location** menu, select a location to store your content. Most customers select a region close to the interconnect location they specify for shielding.
4. Click the **Create** button.

You should now add objects to your bucket and make them externally accessible by selecting the **Public link** checkbox next to each of the objects.

Adding your GCS bucket as an origin server

To add your GCS bucket as an origin server, follow the instructions for working with hosts. You'll add specific details about your origin server.

1. In the **Hosts** field on the **Origins** page, enter the appropriate address for your Host using the format `<BUCKET>.storage.googleapis.com`. For example, if your bucket name is `test123`, your hostname would be `test123.storage.googleapis.com`.

For the initial **Edit this host** fields:

- In the **Name** field, enter any descriptive name for your service (e.g., `Google Cloud Storage`).
- In the **Address** field, enter the appropriate address for your Host using the format `<BUCKET>.storage.googleapis.com`. For example, if your bucket name is `mybucket`, your hostname would be `mybucket.storage.googleapis.com`.
    1. When you edit the Transport Layer Security (TLS) area information for your host:
- Leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
- In the **Certificate hostname** field, enter `storage.googleapis.com`.
- Under the **SNI hostname** field, select the checkbox to **Match the SNI hostname to the Certificate hostname**. The hostname address you entered during Host creation appears.
    1. From the **Shielding** menu below the TLS area, select an interconnect location from the list of shielding locations.
    2. In the **Override host** field, enter an appropriate address for your Host (e.g., `test123.storage.googleapis.com`). You entered this information during Host creation.

Interconnect locations

Interconnect locations allow you to establish direct links with Google's network edge when you choose your shielding location. By selecting one of the locations listed in our developer documentation, your traffic will be carried across our interconnections with Google and you will be eligible to receive Google's CDN partner pricing discount. Most customers select the interconnect closest to their GCS bucket's region. Review our caveats of shielding and select an interconnect accordingly.

Setting the Cache-Control header for your GCS bucket

By default, GCS performs its own caching for publicly readable objects, which may complicate efforts to purge cache. To avoid potential problems, we recommend using the gsutil command line utility to set the Cache-Control header for one or more objects in your GCS bucket:

```
$ gsutil setmeta -h "Cache-Control: no-store, max-age=86400" gs://<bucket>/*.html
```

Replace `<bucket>` in the example above with your GCS bucket's name. Note that `no-store` instructs GCS not to cache your content, while `max-age=86400` instructs Fastly to cache your content for one day. See Google's documentation on the `setmeta` command for more information.

Changing the default TTL for your GCS bucket

If you want to change the default TTL for your GCS bucket, if at all, keep the following in mind:

- Your GCS account controls the default TTL for your GCS content. GCS currently sets the default TTL to 3600 seconds. Changing the default TTL will not override the default setting in your GCS account.
- To override the default TTL set by GCS from within the Fastly web interface, create a new cache setting and enter the TTL there.
- To override the default TTL in GCS, download the gsutil tool and then change the Cache-Control headers to delete the default TTL or change it to an appropriate setting.

X-Http-Method-Override header behavior

GCS provides a unique functionality that allows clients to add a `X-Http-Method-Override` request header to override the request method being sent in the HTTP messages. For instance, a GET request with the `X-Http-Method-Override: HEAD` request header is treated as a HEAD request by GCS and returns a HEAD response (200 status code with an empty body).

This can cause unintended caching behavior, which is a security risk. For example, if an `X-Http-Method-Override` request header is received and an unexpected response is cached. In order to minimize this risk, we strongly recommend you unset the `X-Http-Method-Override` header in the `vcl_recv` subroutine as shown below:

```
unset req.http.X-Http-Method-Override;
```

Using GCS with private objects

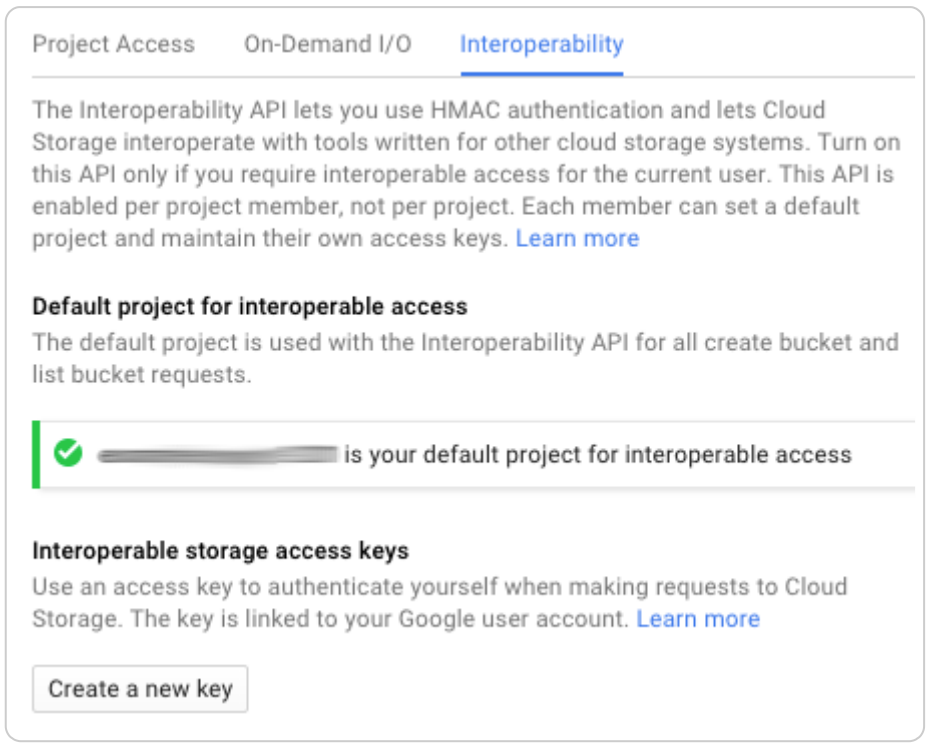
To use Fastly with GCS private objects, be sure you've already made your GCS data available to Fastly by pointing to the right GCS bucket, then follow the steps below.

Setting up interoperable access

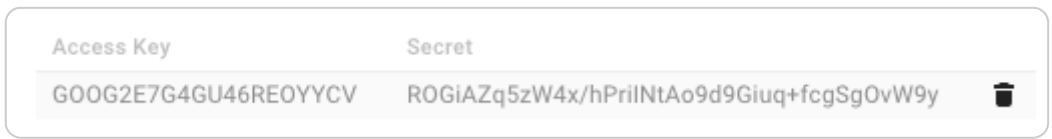
By default, GCS authenticates requests using OAuth2, which Fastly does not support. To access private objects on GCS, your project must have HMAC authentication enabled and interoperable storage access keys (an *Access Key* and *Secret* pair) created. Do this by following the steps below.

> **TIP**
>
> To limit access to your Google Cloud account, consider creating a service account and then creating HMAC keys for that service account. For more information, see Google's documentation on managing HMAC keys for service accounts.

1. Open the Google Cloud Platform console and select the appropriate project.
2. Click **Settings**. The Settings appear with the Project Access controls highlighted.
3. Click the **Interoperability** tab. The Interoperability API access controls appear.
4. If you have not set up interoperability before, click **Enable interoperability access**.
5. Click **Make `<PROJECT-ID>` your default project** for interoperable access. If that project already serves as the default project, that information appears instead.



6. Click **Create a new key**. An access key and secret code appear.



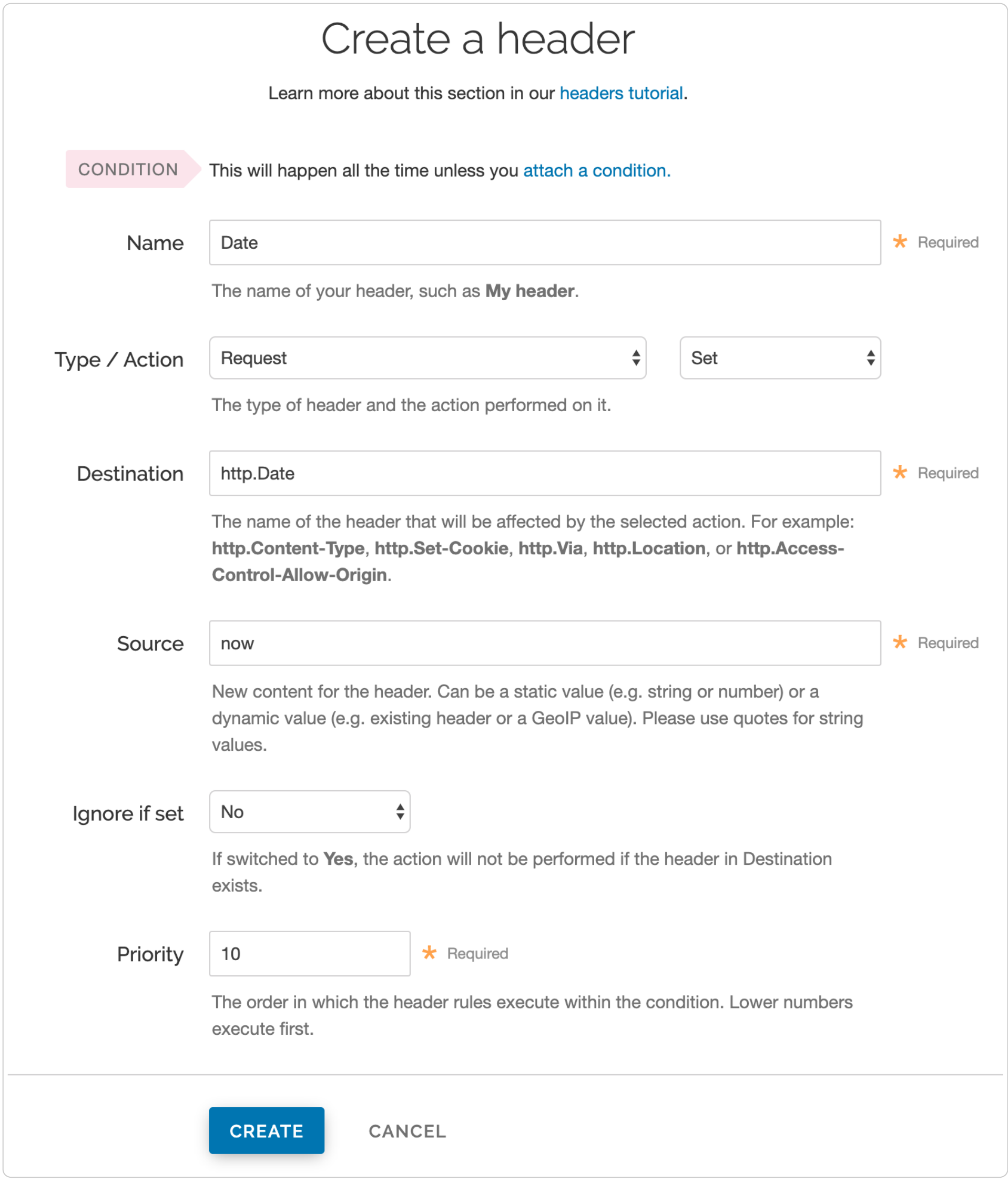
7. Save the access key and secret code that appear. You'll need these later when you're creating an authorization header.

Setting up Fastly to use GCS private content

To use GCS private content with Fastly, create two headers, a Date header (required Authorization Signature) and an Authorization header.

Creating a Date header

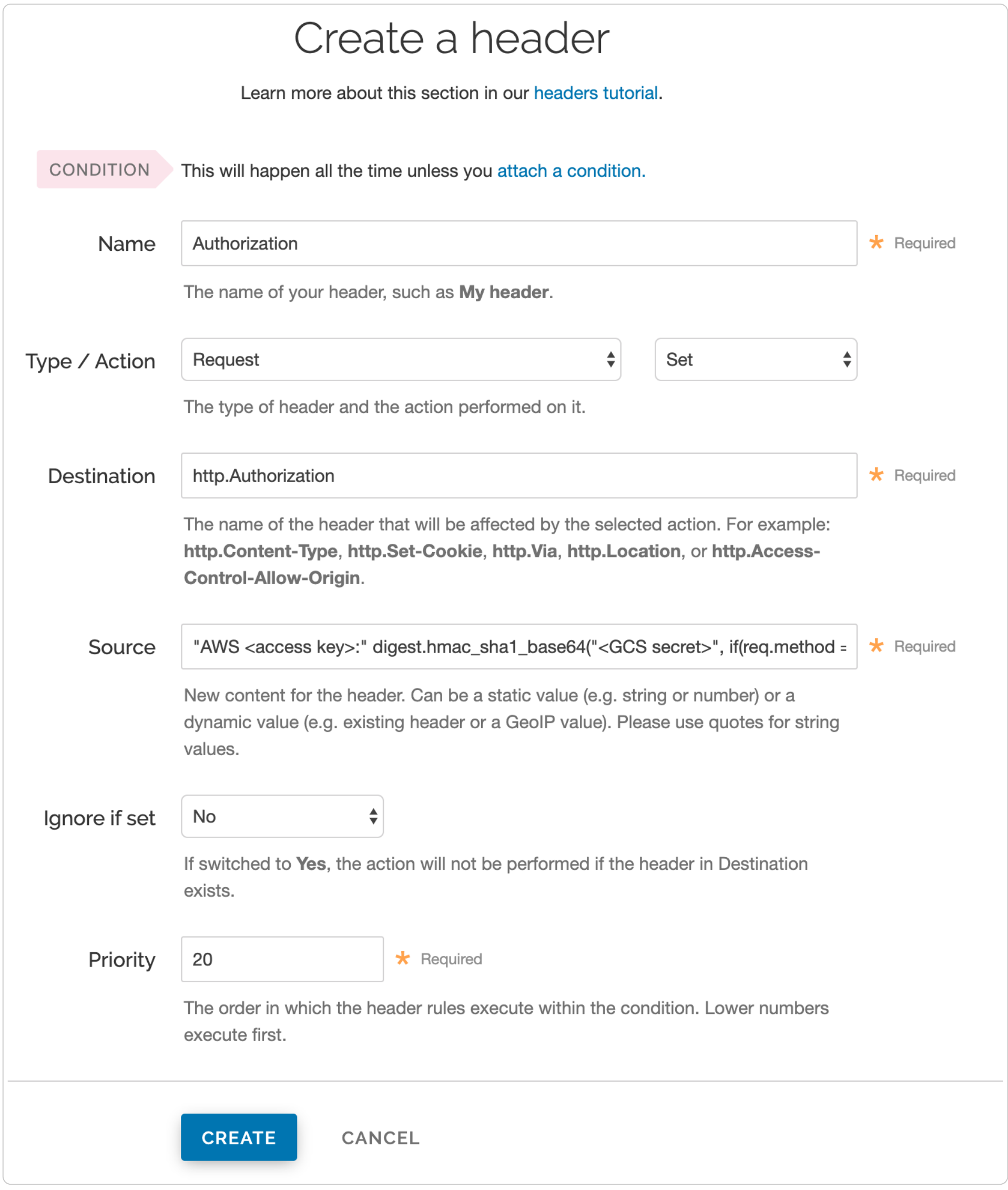
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a new header page appears.



6. Fill out the **Create a new header** fields as follows:
   - In the **Name** field, enter `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Date`.
   - In the **Source** field, enter `now`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
7. Click the **Create** button. A new Date header appears on the Content page. You will use this later within the Signature of the Authorization header.

Creating an Authorization header

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Authorization`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `20`.
3. In the **Source** field, enter the header authorization information using the following format:

```
  "AWS <access key>:" digest.hmac_sha1_base64("<GCS secret>", if(req.method == "HEAD", "GET", req.method) LF LF LF req.
http.Date LF "/<GCS bucket name>" req.url.path)
```

replacing `<access key>`, `<GCS secret>`, and `<GCS bucket name>` with the information you gathered before you began. For example:

```
  "AWS GOOGQORE5WOJJHLXH6OD:" digest.hmac_sha1_base64("oQb0hdmaxFOc5UmC6F833Cde0+ghRSgsr7CCnX62", if(req.method == "HEA
D", "GET", req.method) LF LF LF req.http.Date LF "/test123" req.url.path)
```

4. Click the **Create** button. A new Authorization header appears on the Content page.
5. Click the **Activate** button to deploy your configuration changes.

A detailed look at the Source field

So what's going on in the Source field of the Authorization header? Here's the basic format:

`AWS<access key><signature function><key><message>`

It tells us the following:

Element	Description
`AWS`	A constant placed before the access key. It's always AWS.
`access key`	The access key ID from your GCS developer's account. We used `GOOGQORE5WOJJHLXH6OD` in this example.
`signature function`	The algorithm used to validate the key and message of the signature. We used `digest.hmac_sha1_base64(<key>, <message>)` in this example.
`key`	The secret key ID from your GCS developer's account. We used `oQb0hdmaxFOc5UmC6F833Cde0+ghRSgsr7CCnX62` in this example.
`message`	The UTF-8 encoding of the StringToSign. See the table below for a break down of each portion of the message.

The message that's part of the Source field in the Authorization header takes on this basic format:

`<HTTP-verb><\n><Content-MD5>\n<Content-Type><\n><Date><\n><CanonicalExtensionHeaders><\n><CanonicalizedResource>`

It tells us the following:

Element	Description
`HTTP-verb`	The REST verb. We use `req.method` in this example.
`\n`	A newline indicator constant. It's always \n.
`Content-MD5`	The content-md5 header value, used as a message integrity check. It's often left blank. We use `LF` (line feed) in this example.
`Content-Type`	The content-type header value, used to specify the MIME-type. It's often left blank. We use `LF`in this example.
`Date`	The date and time stamp. We use `req.http.Date` (which we created first as a separate header in the steps above).
`CanonicalExtensionHeaders`	The x-amz- or x-goog- headers, which customize your GCS implementation. It's often left blank. We use `LF` in this example.
`CanonicalizedResource`	Your GCS resource path name. We're concatenating GCS bucket name `"/test123"` with object path `req.url.path` in this example.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Google Compute Engine
Last updated: 2021-10-25

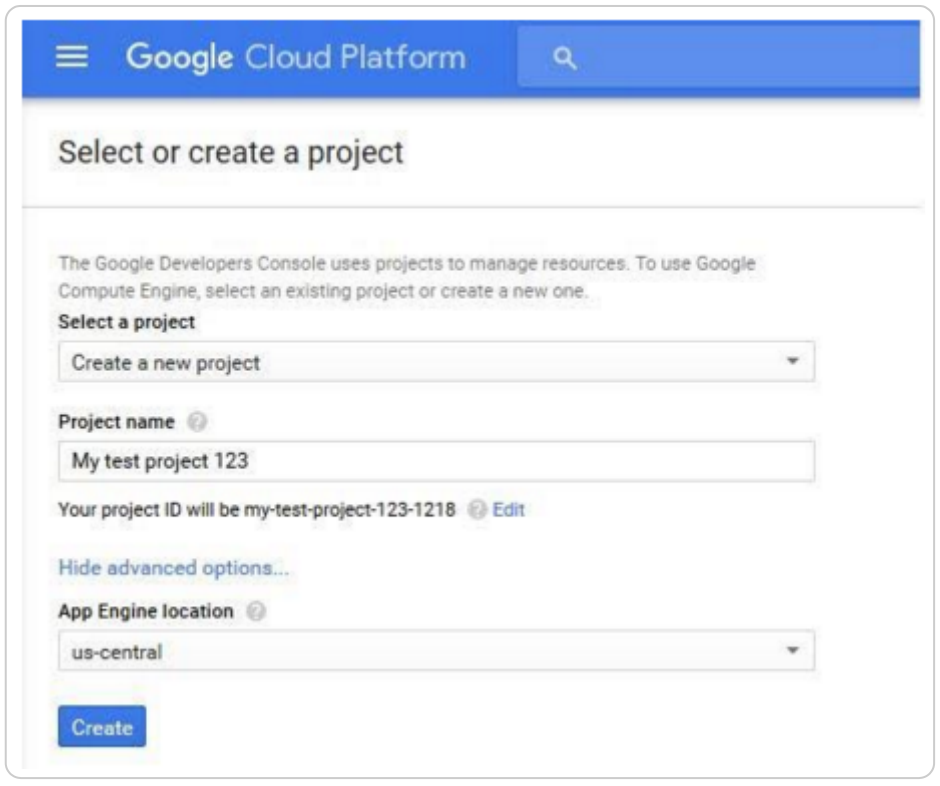
https://docs.fastly.com/en/guides/google-compute-engine

Google Compute Engine (GCE) lets you create and run a virtual machine (VM) on the Google infrastructure. The VM can be used as an origin server with your Fastly service once you set up and configure your VM instance and link your instance to a Fastly service.

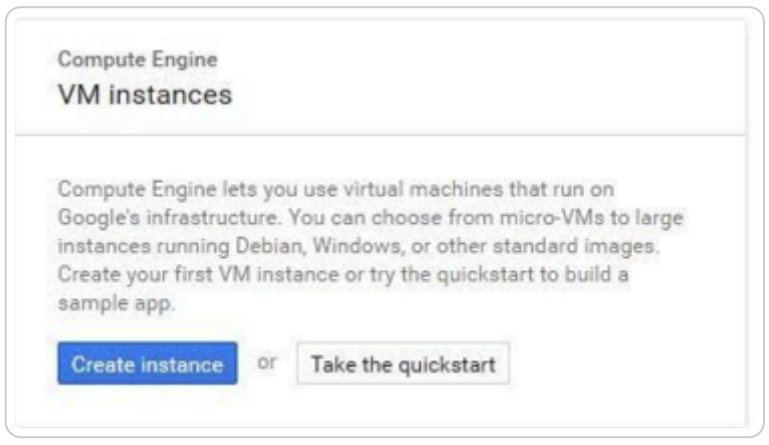
Creating and setting up your GCE instance

1. Sign up for Google Compute Engine and start the basic set up. If you are already signed up and at your dashboard, click the **Get started** link in the Try Compute Engine area.

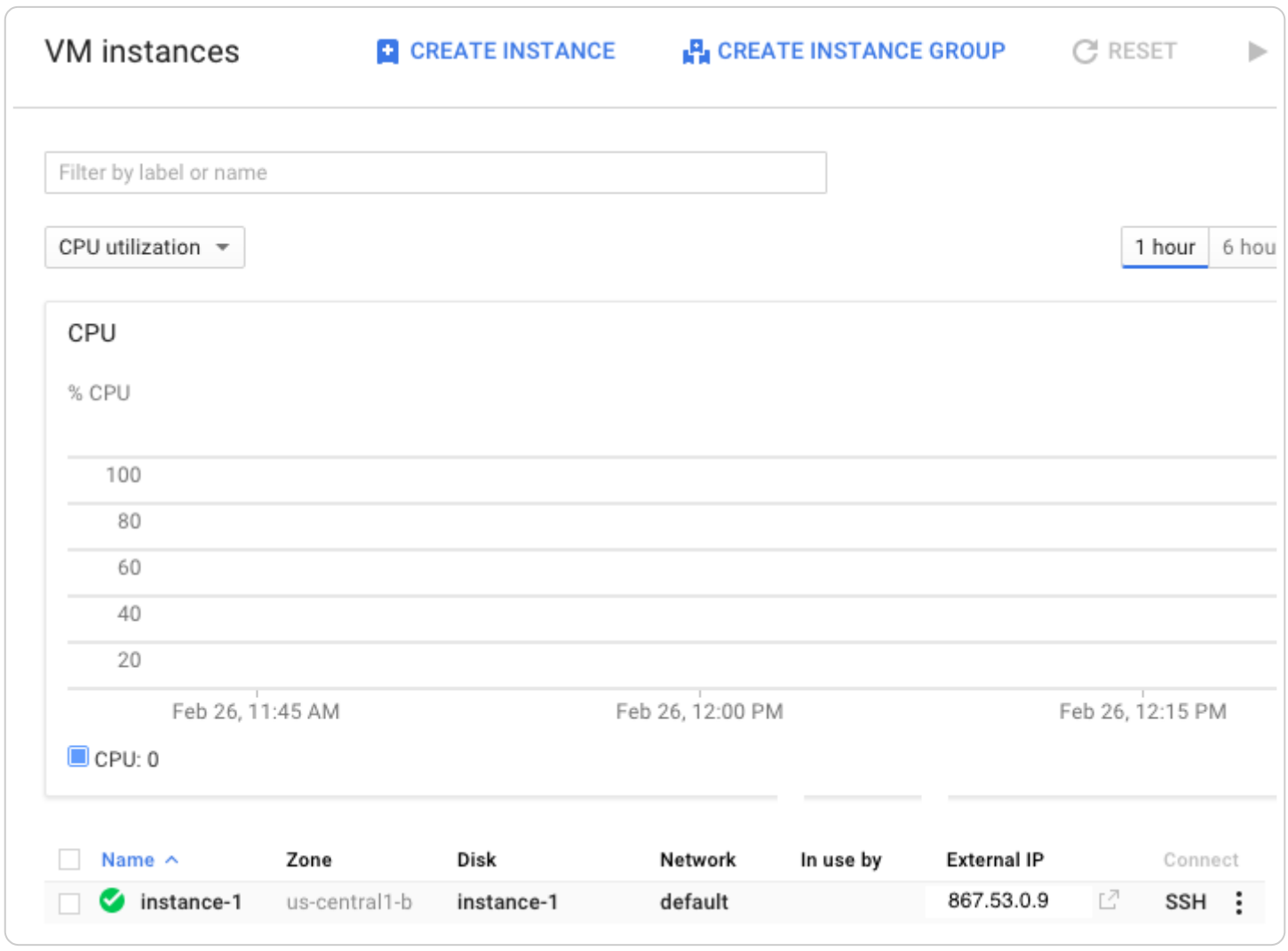
2. Create or select a project to hold your origin's data.



3. Click **Create instance** to set up your VM. You can set up your instance using either Windows or Linux.



4. Fill in the necessary fields and click **Create**. When making your firewall selection, select either **Allow HTTPS traffic** (port 443) or **Allow HTTP traffic** (port 80); you will use one of those ports when you create your new origin in your Fastly service. If you select HTTPS traffic, you need to configure the VM to respond on port 443 with a valid TLS certificate.



5. Make note of the following information for when you create your new origin in your Fastly service:
   - The instance's IP address (located in the External IP column at the bottom of the page). You'll use this in the **Address** field when you create your new origin.
   - The zone you are using (located in the Zone column at the bottom of the page). You'll use this to guide your selection of an appropriate shielding location for your origin.

Creating a new origin in your Fastly service for your GCE account

Link your GCE account to a Fastly service following the steps below.

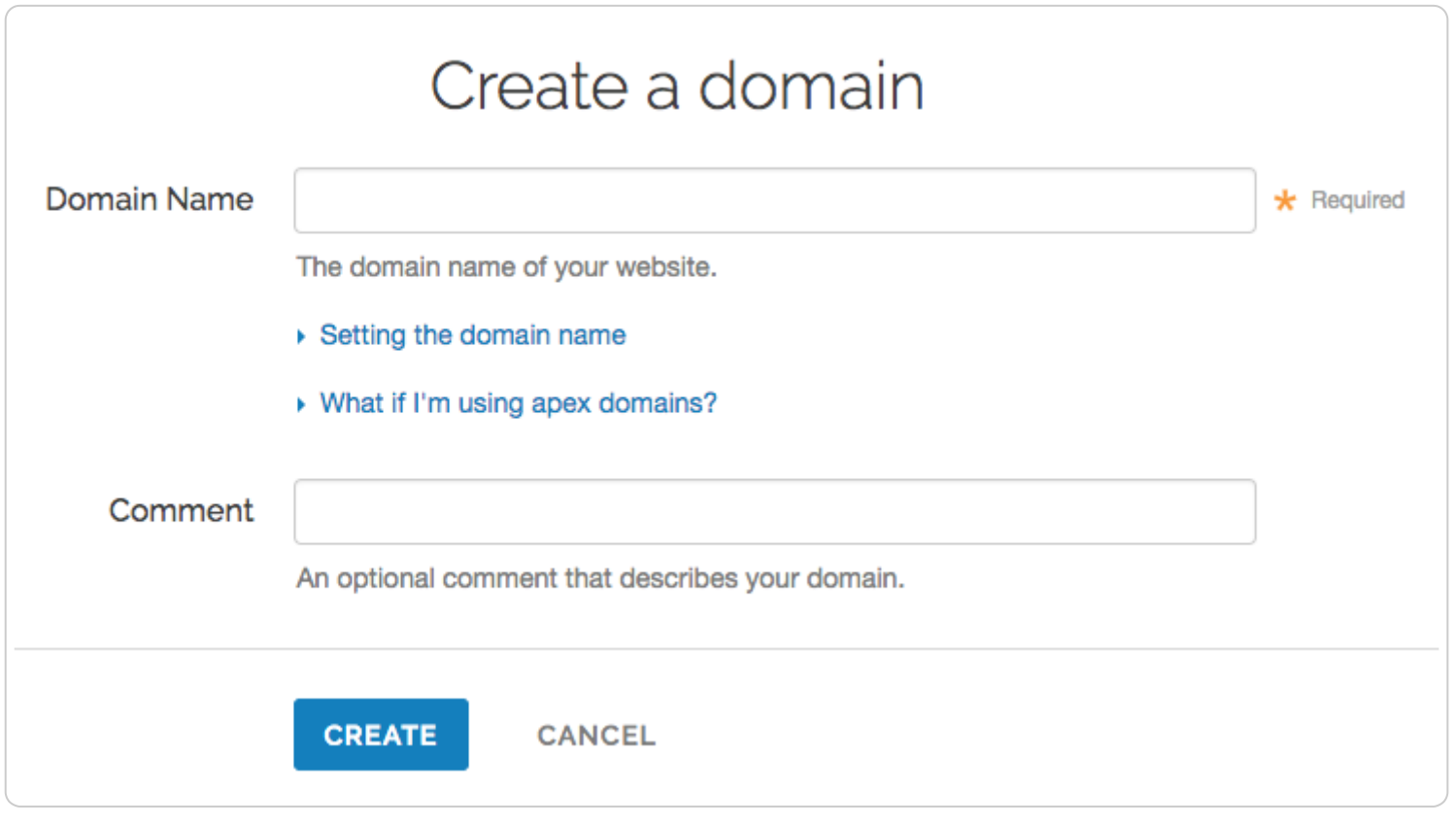
1. Log in to the Fastly web interface.
2. Create a new service if you don't already have one set up.
3. Follow the instructions for working with hosts. You'll add specific details about your origin server when you fill out the **Create a host** fields:
   - In the **Name** field, enter the name of your server (for example, `Google Compute Engine`).
   - In the **Address** field, enter the IP address of your server. This should match the port that you selected in the GCE interface.
   - From the **Shielding** menu, select an interconnect location from the list of shielding locations.

Interconnect locations

Interconnect locations allow you to establish direct links with Google's network edge when you choose your shielding location. By selecting one of the locations listed in our developer documentation, you will be eligible to receive discounted pricing from Google CDN Interconnect for traffic traveling from Google Cloud Platform to Fastly's network. Most customers select the interconnect closest to their origin. Review our caveats of shielding and select an interconnect accordingly.

Creating new domains for GCE to respond to

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Create domain** button. The Create a domain page appears.



5. In the **Domain Name** field, enter the name that users will enter in their browsers to access your site.
6. In the **Comment** field, optionally enter a comment that describes your domain.
7. Click the **Create** button. The new domain appears on the Domains page.
8. Click the **Activate** button to deploy your configuration changes.

Creating a CNAME record

You can now test your configuration. In the example above, your domain would appear as `www.example.com.global-nossl.fastly.net`. After you test and you're satisfied with the results, create a CNAME record for your domain (e.g., `www.example.com`) pointing to `global-nossl.fastly.net`.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Microsoft Azure Blob Storage

Last updated: 2021-05-13

https://docs.fastly.com/en/guides/microsoft-azure-blob-storage

Microsoft Azure Blob Storage public and private containers can be used as origins with Fastly.

TIP

With properly configured services in place, shared Fastly and Microsoft customers will benefit from Fastly's integration with Azure's ExpressRoute Direct Local, which results in Fastly including your outbound data transfer costs from Azure in your standard Fastly pricing. See our guide to outbound data transfers from Azure for more details.

Using Azure Blob Storage as an origin

When using Azure Blob Storage as an origin, we recommend using the most recent version of Azure storage services. Certain incompatibilities may occur if using too old a version or if the version is not specified by the request header. To enforce a version to use when not specified by the header, set the `DefaultServiceVersion` in your Blob storage service.

Alternatively, place a VCL snippet to `vcl_miss` and `vcl_pass` to set or enforce the version. For example, if the version is `2020-06-12`, the VCL snippet would look like the following:

```
1 {{if (req.backend.is_origin) { }}
2 {{ set bereq.http.x-ms-version = "2020-06-12"; }}
3 }
```

Once you've configured your Azure Blob Storage stores are ready to make them available through Fastly, follow the steps below.

Creating a new service

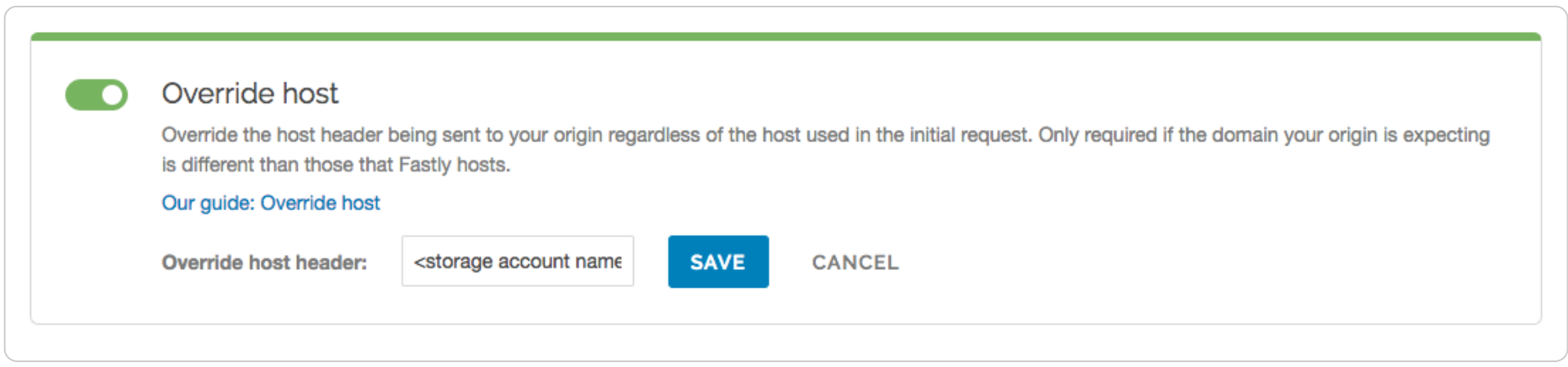
Follow the instructions for creating a new service. You'll add specific details about your origin when you fill out the **Create a new service** fields:

- In the **Name** field, enter any descriptive name for your service.
- In the **Domain** field, enter the hostname you want to use as the URL (e.g., `cdn.example.com`).
- In the **Address** field, enter `<storage account name>.blob.core.windows.net`.
- In the **Transport Layer Security (TLS)** area, leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
- In the **Transport Layer Security (TLS)** area, enter `<storage account name>.blob.core.windows.net` in the **Certificate hostname** field.

Setting the default Host and correct path

Once the new service is created, set the default Host to `azure` and then add your container path to the URL by following the steps below:

1. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
2. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
3. Click the **Settings** link. The Settings page appears.
4. Click the **Override host** switch. The Override host header field appears.



5. Enter the hostname of your Azure Blob Storage account. For example, `<storage account name>.blob.core.windows.net`.
6. Click the **Save** button. The new override host header appears in the Override host section.
7. Click the **Content** link. The Content page appears.
8. Click the **Create header** button. The Create a header page appears.
9. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Modify URL`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `url`.
   - In the **Source** field, enter `"/<your container name>" req.url`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
10. Click the **Create** button. The new Modify URL header appears on the Content page.
11. Click the **Activate** button to deploy your configuration changes.

Testing your results

By default, we create DNS mapping called **yourdomain.global.prod.fastly.net**. In the example above, it would be `cdn.example.com.global.prod.fastly.net`. Create a DNS alias for the domain name you specified (e.g., CNAME `cdn.example.com` to `global-nossl.fastly.net`).

Fastly will cache any content without an explicit `Cache-Control` header for 1 hour. You can verify whether you are sending any cache headers using curl. For example:

```
$ curl -I opscode-full-stack.blob.core.windows.net

HTTP/1.1 200 OK
Date: Fri, 04 May 2018 21:23:07 GMT
Content-Type: application/xml
Transfer-Encoding: chunked
Server: Blob Service Version 1.0 Microsoft-HTTPAPI/2.0
```

In this example, no Cache-Control headers are set so the default TTL will be applied.

Using an Azure Blob Storage private container

To use an Azure Blob Storage private container with Fastly, follow the instructions below.

Before you begin

Be sure you've already made your Azure Blob Storage containers available to Fastly by pointing to the right container and setting your origin to port 443. This needs to be done before authenticating.

To complete the setup, you'll also need your Azure Storage Account shared key and storage account name to construct the Azure Blob Storage Authorization header, which takes the following form:

```
Authorization: SharedKey `Account name`:`Signature`
```

Finally, you'll also need to know your Blob Storage container name.

Setting up Fastly to use an Azure Blob Storage private container with a Shared Key

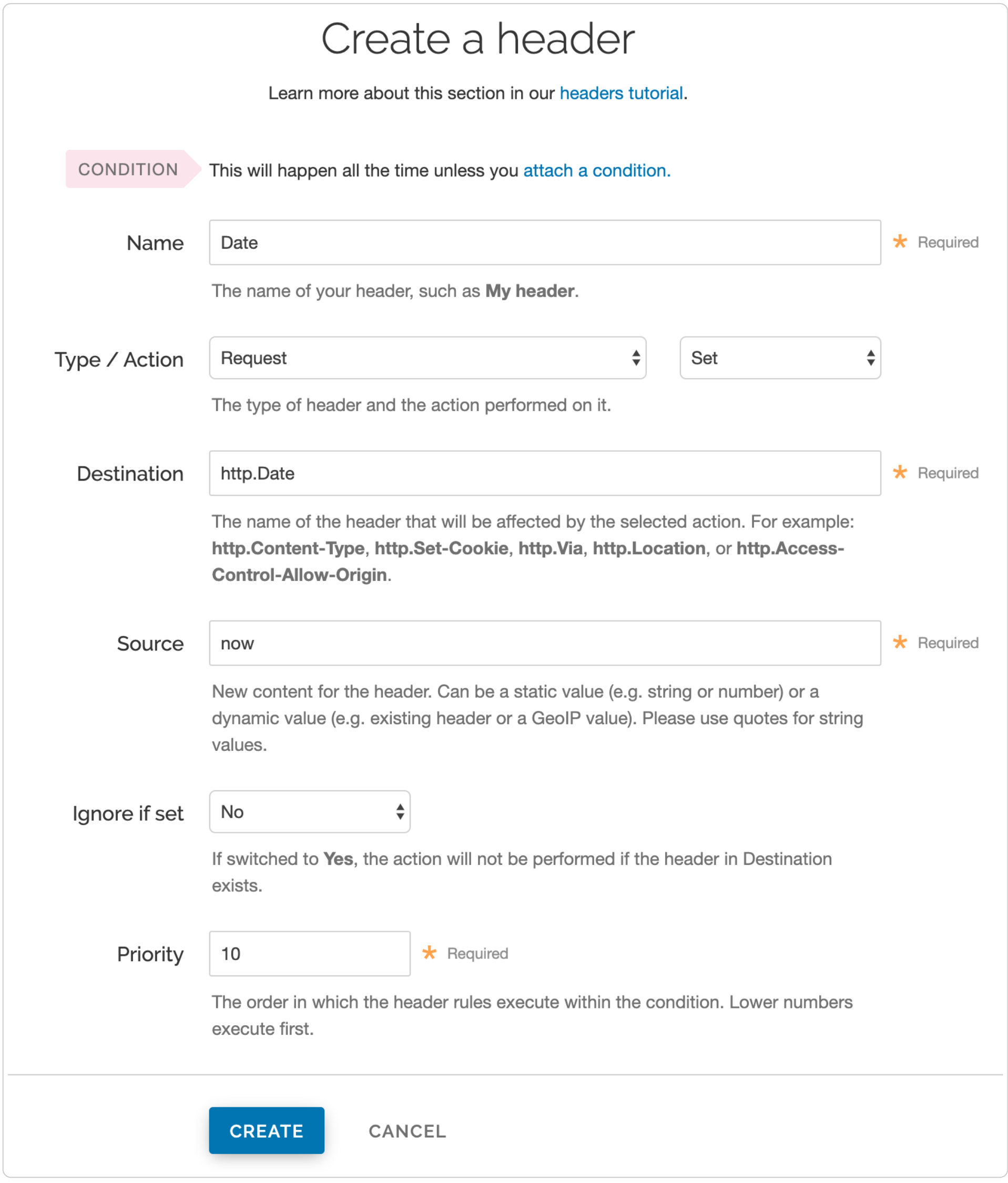
> **WARNING**
>
> Your account's Shared Key does not have detailed access control. Anyone with access to your Shared Key can read and write to your container. Consider using a Shared Access Signature (SAS) instead.

To access an Azure Blob Storage private container with Fastly using a Shared Key, first familiarize yourself with Microsoft's documentation on authorizing with Shared Key. Then, create two headers: a Date header (for use with the authorization Signature) and an Authorization header.

Create a Date header

Create the Date header using the steps below.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.

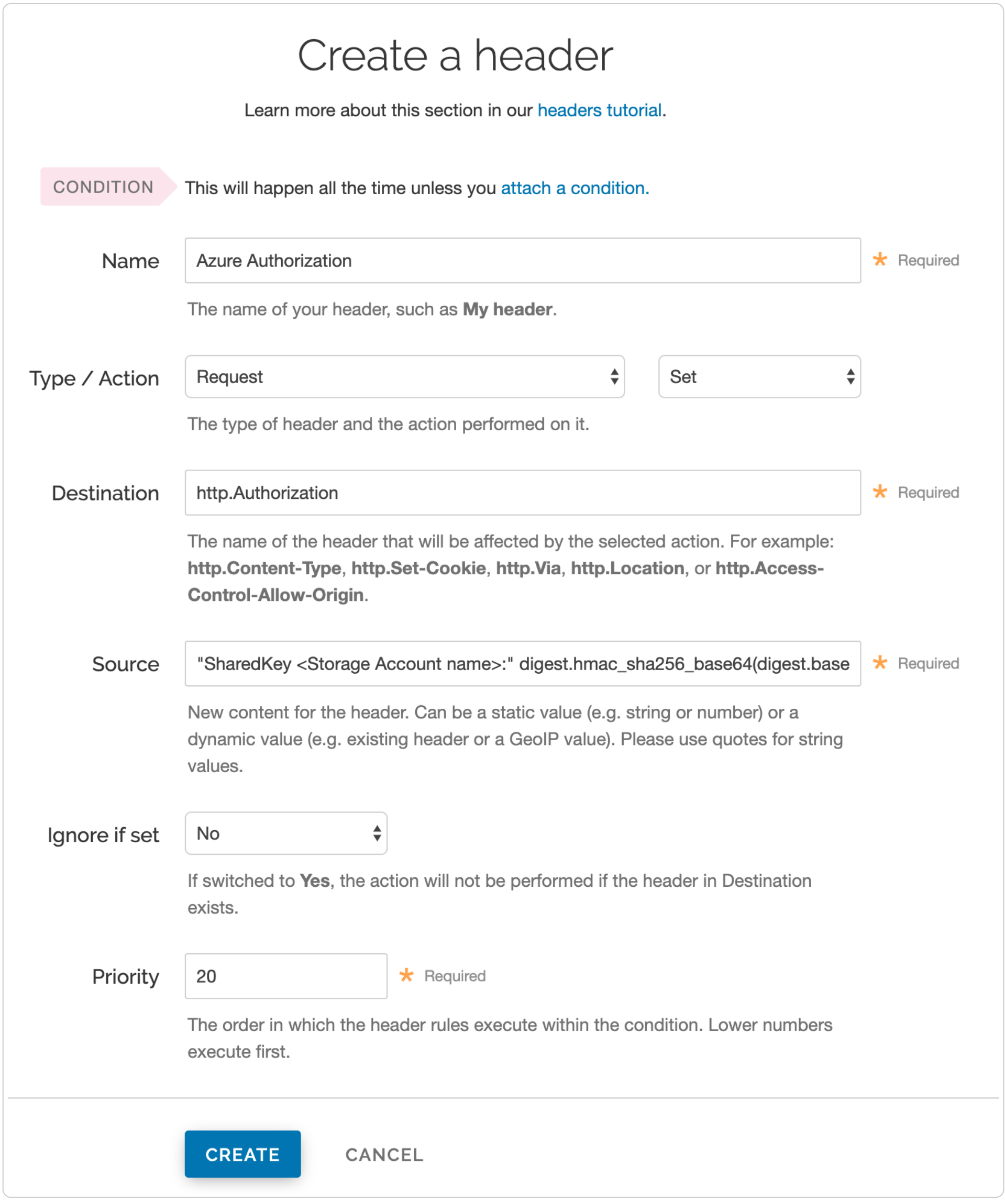


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Date`.
   - In the **Source** field, enter `now`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `10`.
7. Click the **Create** button. A new Date header appears on the Content page. You will use this later within the Signature of the Authorization header.

Create an Authorization header

Next, create the Authorization header with the specifications listed below.

1. Click the **Create header** button again to create another new header. The Create a header page appears.



2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Azure Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `http.Authorization`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, enter `20`.

3. In the **Source** field, enter the header authorization information using the following format:

```
"SharedKey <Storage Account name>:" digest.hmac_sha256_base64(digest.base64_decode("<Azure Storage Account shared key
>"), if(req.method == "HEAD", "GET", req.method) LF LF LF req.http.Date LF "/<Storage Account name>" req.url.path)
```

replacing `<Storage Account name>` and `<Azure Storage Account shared key>` with the information you gathered before you began. For example:

```
"SharedKey test123:" digest.hmac_sha256_base64(digest.base64_decode("UDJXUFN1NjhCZmw4OWo3MnZUK2JYWVpCN1NqbE93aFQ0d2hxdD
I3"), if(req.method == "HEAD", "GET", req.method) LF LF LF req.http.Date LF "/test123" req.url.path)
```

We provide a detailed look at the Source field parameters below.

4. Click the **Create** button. The new Authorization header appears on the Content page.

5. Click the **Activate** button to deploy your configuration changes.

A detailed look at the Source field

So what's going on in the Source field of the Authorization header? Here's the basic format:

`SharedKey<storage account name><Signature Function><key><message>`

It tells us the following:

Element	Description
`SharedKey`	A constant placed before the storage account name. It's always SharedKey.
`storage account name`	The name of your Azure Storage Account. We used `test123` in this example.
`signature function`	The algorithm used to validate the key and message of the signature. We used `digest.hmac_sha256_base64(<key>, <message>)` in this example.
`key`	The Azure Storage Account shared key from your Azure Storage developer's account. We used `UDJXUFN1NjhCZmw4OWo3MnZUK2JYWVpCN1NqbE93aFQ0d2hxdDI3` in this example. It must be Base64 decoded.
`message`	The UTF-8 encoding of the StringToSign. See the table below for a break down of each portion of the message.

The message that's part of the Source field in the Authorization header takes on this basic format:

`<HTTP-verb></n><Content-MD5>/n<Content-Type></n><Date></n><CanonicalizedAmzHeader></n><CanonicalizedResource>`

It tells us the following:

Element	Description
`HTTP-verb`	The REST verb. We use `req.method` in this example. We rewrite HEAD to GET because Varnish does this internally before sending requests to origin.
`\n`	A newline indicator constant. It's always \n.
`Content-MD5`	The content-md5 header value, used as a message integrity check. It's often left blank. We use `LF` (line feed) in this example.
`Content-Type`	The content-type header value, used to specify the MIME-type. It's often left blank. We use `LF` in this example.
`Date`	The date and time stamp. We use `req.http.Date` (which we created first as a separate header in the steps above).
`CanonicalizedHeaders`	The x-ms headers, which customize your Azure Blob Storage implementation. It's often left blank. We use `LF` in this example.
`CanonicalizedResource`	Your Storage Account Name. We use `"/test123"` in this example.

Setting up Fastly to use an Azure Blob Storage private container with a Shared Access Signature (SAS)

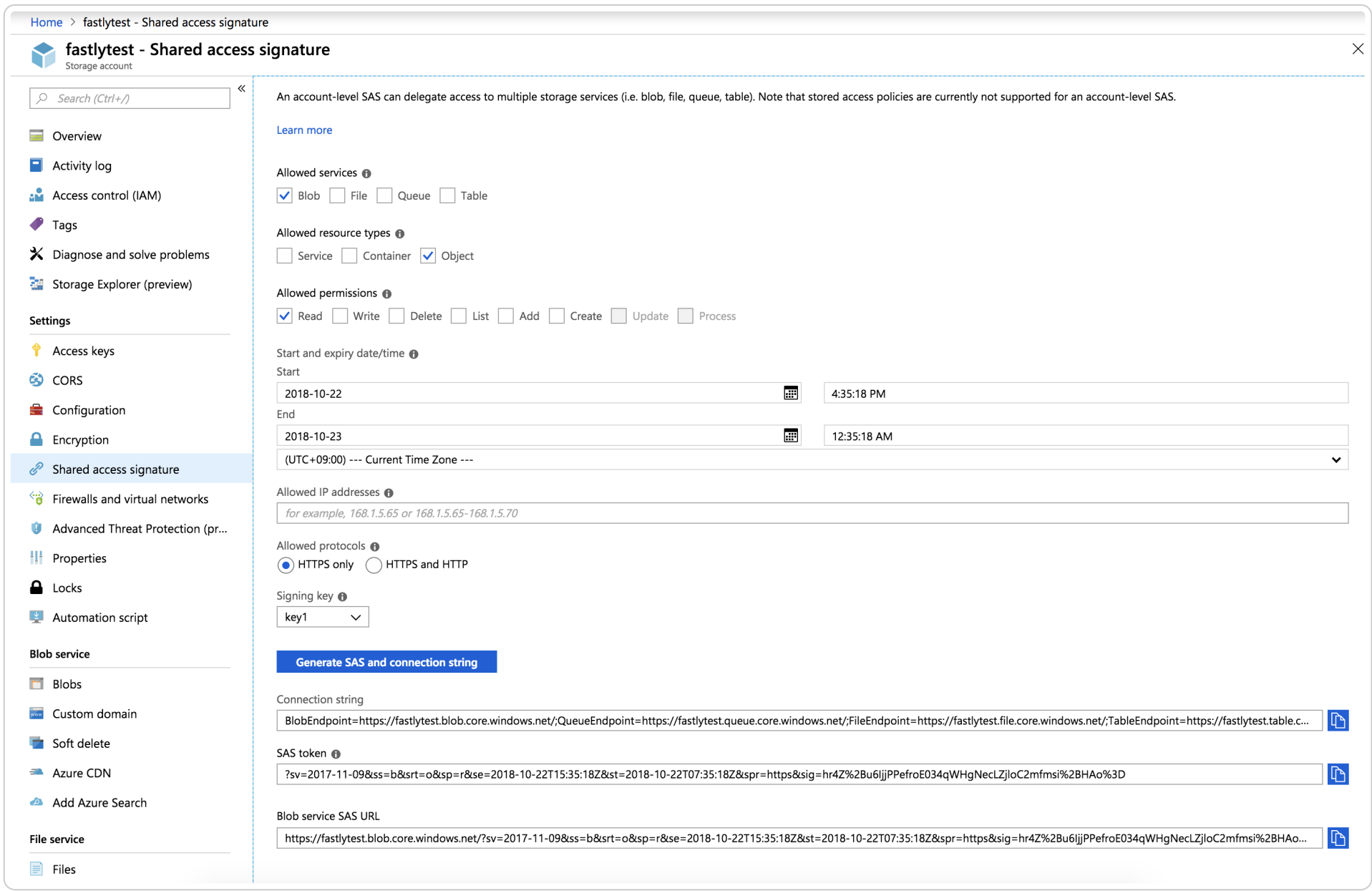
To access an Azure Blob Storage private container with Fastly using a Service Shared Access Signature (SAS), read Microsoft's Delegate access with a shared access signature page. Then, obtain the SAS and sign the access URL.

> **TIP**
>
> Using a Service Shared Access Signature gives you more detailed control over:
>
> - The interval during which the SAS is valid, including the start time and the expiry time.
> - The permissions granted by the SAS. For example, a SAS for a blob might grant read and write permissions to that blob, but not delete permissions.
> - An optional IP address or range of IP addresses from which Azure Storage will accept the SAS. For example, you might specify a range of IP addresses belonging to your organization.
> - The protocol over which Azure Storage will accept the SAS. You can use this optional parameter to restrict access to clients using HTTPS.

Obtaining the Shared Access Signature

Obtain the SAS using the steps below.

1. In the Azure portal, navigate to your storage account
2. Under settings navigate to **Shared access signature**. The Shared access signature controls appear.



3. From the **Allowed services** controls, select **Blob**.
4. From the **Allowed resource types** controls, select **Object**.
5. From the **Allowed permissions** controls, select **Read**.
6. Leave the **Start time** set to the current date and time.
7. Set the **End time** as far in the future as you are comfortable (see note below).
8. Ensure the **Allowed protocols** remain set to **HTTPS only**.

9. Click the **Generate SAS and connection string** button. The generated information appears.
10. Copy and save the contents of the **SAS token** field. It will look something like:

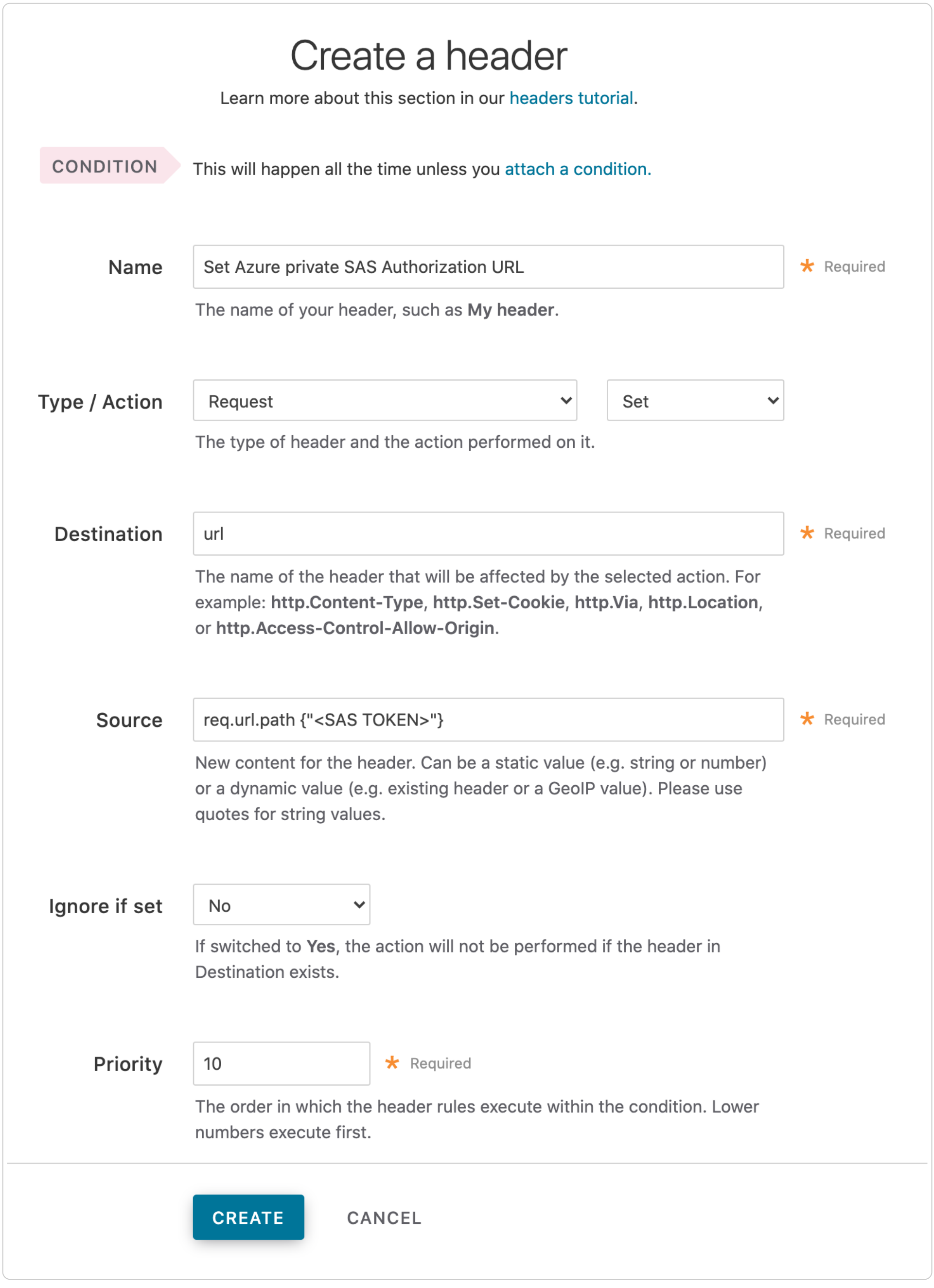
    ```
    ?sv=2017-11-09&ss=b&srt=o&sp=r&se=2019-10-22T15:41:23Z&st=2018-10-22T07:41:23Z&spr=https&sig=decafbaddeadbeef
    ```

    We provide a detailed look at the Shared Access Signature parameters below.

Signing the URL

Next, sign the access URL by creating an authorization header following the steps below.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter a meaningful name such as `Set Azure private SAS Authorization URL`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter `url`.

- In the **Source** field, enter `req.url.path {"<SAS TOKEN>"}` replacing `{"<SAS TOKEN>"}` with the token you obtained from the Azure Portal.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, enter `10`.

7. Click the **Create** button. A new header appears on the Content page.
8. Click the **Activate** button to deploy your configuration changes.

A detailed look at the Shared Access Signature parameters

Microsoft's Create a service SAS page provides more details on shared access signatures and how they are constructed.

Element	Description
`sv`	The `signedversion` field. This is required and should be whatever the Azure portal provided.
`ss`	The `signedservice` field. This is required and should be `b` for "blob storage."
`srt`	The `signedresourcetype` field. This is required and should be `o` for "object."
`sp`	The `signedpermissions` field. This is required and should be `r` for "read only."
`st`	The `signedstart` field. This is optional and specifies, in a UTC format compatible with ISO 8601, the time at which the shared access signature becomes valid. If omitted, the start time for this call is assumed to be the time when the storage service receives the request.
`se`	The `signedexpiry` field. This is required and specifies, in a UTC format compatible with ISO 8601, the time at which the shared access signature becomes invalid.
`spr`	The `signedprotocol` field. This is optional and specifies which HTTP protocol (`http` or `https`) the container should use for access. We recommend `https`.
`sig`	The `signature` field. This is required and should be whatever the Azure portal provided.

WARNING

Always keep track of the `se` expiry date. After it has passed, Fastly will not be able to access your private container.

TCP connection settings for improved performance

By default, Fastly keeps established TCP connections opened to your origin to improve performance. Azure's default behavior, however, closes idle connections. Specifically, the Azure Load Balancer's default behavior silently drops flows when the default idle timeout of a flow reaches four minutes. To ensure successful integration, we suggest tuning Azure in two ways:

- increase the Azure Load Balancer's TCP idle timeout setting
- configure Azure to send a bidirectional TCP Reset (RST packet) on idle timeout

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Oracle Cloud Storage

Last updated: 2021-08-12

https://docs.fastly.com/en/guides/oracle-cloud-storage

Oracle Cloud Storage public and private buckets can be used as origins with Fastly.

Before you begin

Before you begin the setup and configuration steps required to use Oracle Cloud as an origin, keep in mind the following:

- You must have a valid Oracle Cloud account. Before you can create a new bucket and upload files to it for Fastly to use, you must first create an Oracle Cloud account at the Oracle website.
- Oracle Cloud implements both its own proprietary API and an S3 Compatible API. Currently, Fastly supports private buckets only via the S3 Compatible API.

Using Oracle Cloud Storage as an origin

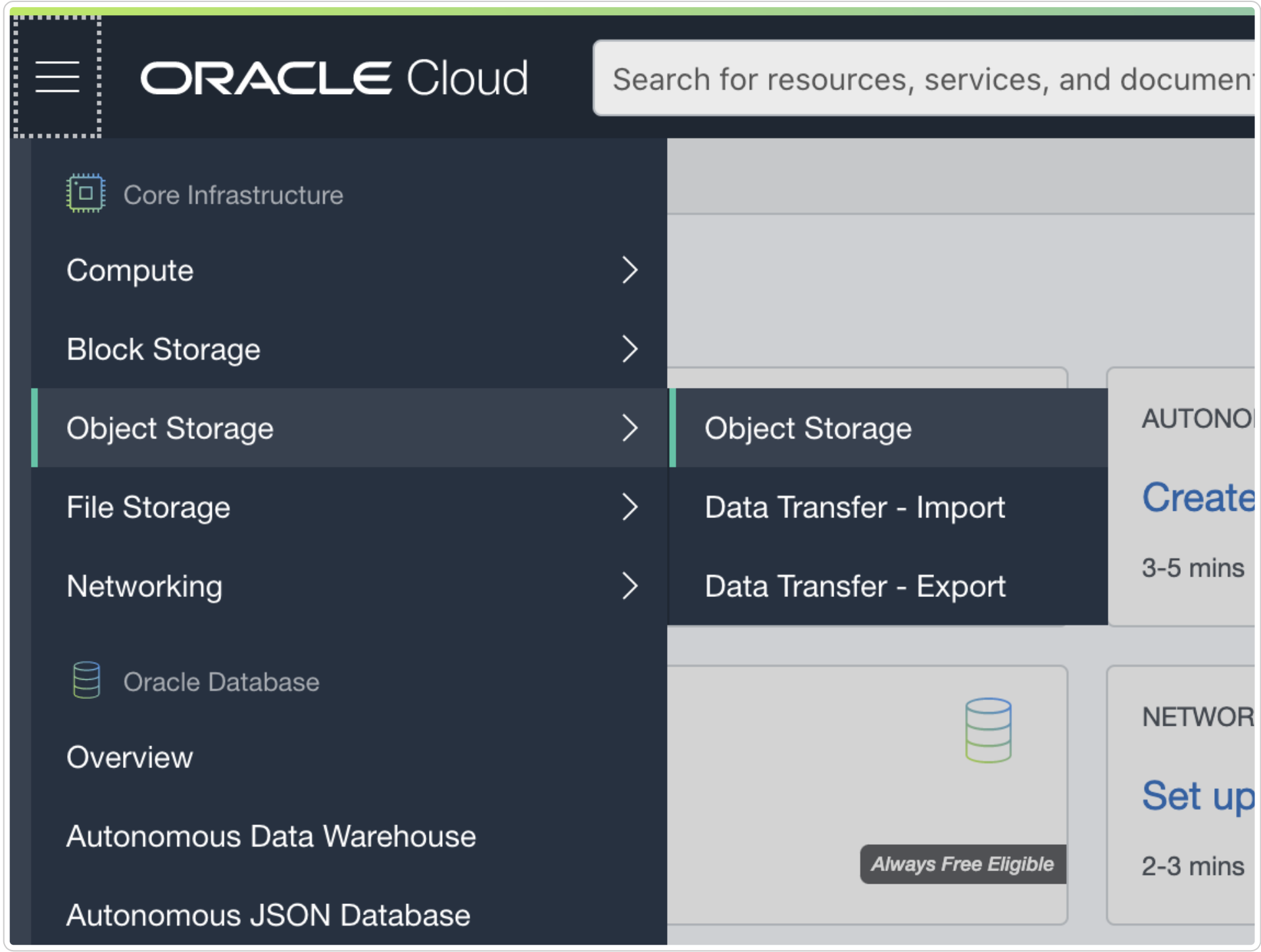
To use Oracle Cloud Storage as an origin, follow the steps below.

Creating a new bucket

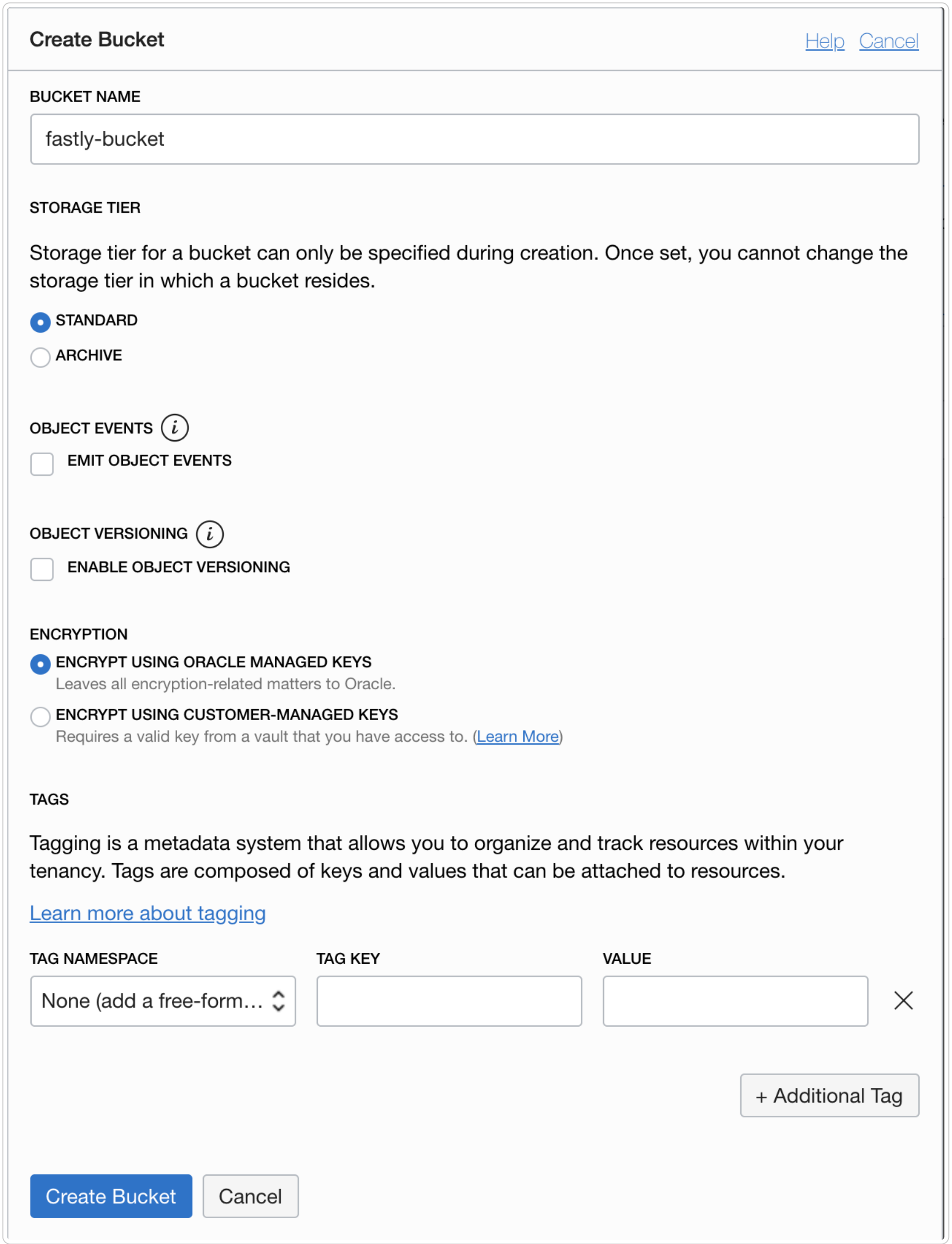
Data in Oracle Cloud Storage is stored in buckets. Follow these steps to create a new bucket via the Oracle Cloud web interface.

> **TIP**
>
> The Oracle Guide provides more details on how to create a bucket.

1. Log in to your Oracle account. Your Oracle account settings page appears.
2. Open the navigation menu in the upper left and navigate to **Object Storage**, then select **Object Storage**.



3. Select a compartment from the **Compartment** list on the left side of the page.
4. Click **Create a Bucket**. The Create a Bucket window appears.
5. In the **Bucket Name** field, enter a unique bucket name. Bucket names must be unique within the namespace and cannot be nested. The name can contain letters, numbers, dashes, and periods.

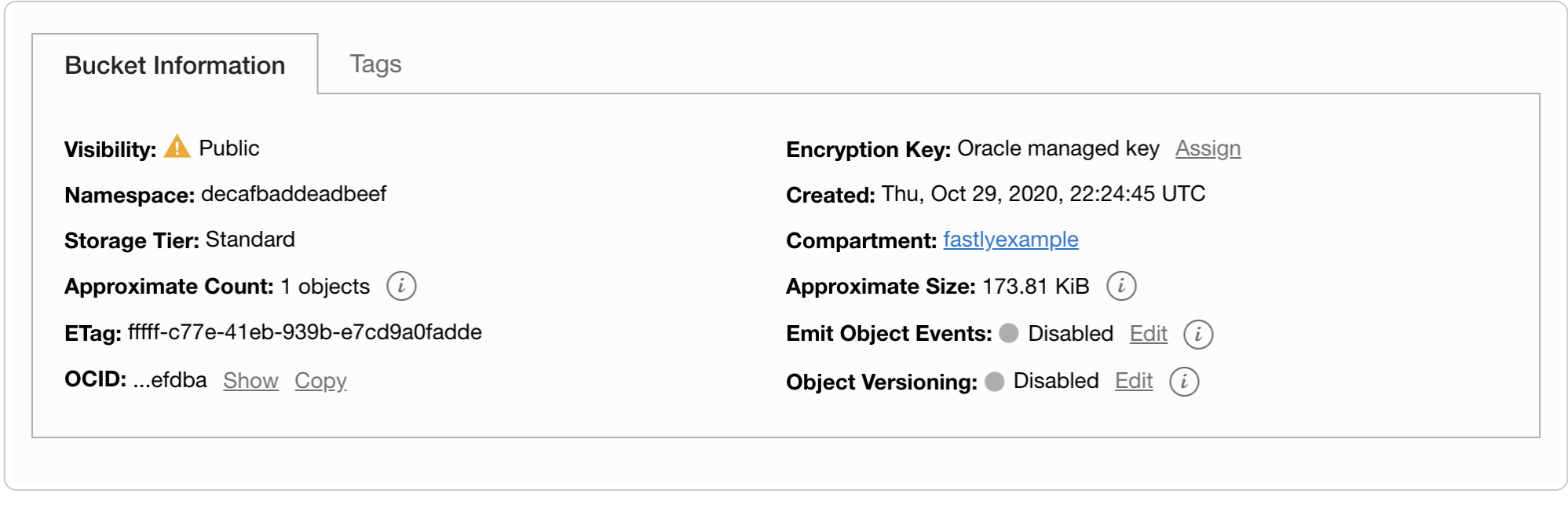


6. Click the **Create a Bucket** button. The new bucket appears in the list of buckets on the Oracle Cloud Storage Buckets page.
7. By default, new buckets are private. Click on three dots on the right side of the bucket and select **Edit Visibility**. Change the visibility to **Public** and deselect the **Allow users to list objects from this bucket** option.
8. Upload a file to the new bucket you just created.

Finding your bucket's namespace and hostname

To set up a Fastly service that interacts with your Oracle Cloud Storage, you will need to know the namespace identifier and hostname assigned to the bucket you created and uploaded files to.

To find your namespace, click on the bucket and examine the **Bucket Information** tab. In this example the namespace is `decafbaddeadbeef`.



To determine your bucket's hostname:

- If you're using the native Oracle API then the hostname takes the form of `objectstorage.<region>.oraclecloud.com` (e.g `objectstorage.us-ashburn-1.oraclecloud.com`).
- If you're using the S3 Compatible API then the hostname takes the form of `<namespace id>.compat.objectstorage.<region>.oraclecloud.com` (e.g `decafbaddeadbeef.compat.objectstorage.us-ashburn-1.oraclecloud.com`).

Creating a new service

To create a new Fastly service, you must first create a new domain and then create a new host and edit it to accept traffic for Oracle Cloud Storage. Instructions to do this appear in our guide to creating a new service. While completing these instructions, keep the following in mind:

- When you create the new host, enter the Oracle bucket's hostname in the **Hosts** field on the **Origins** page. See Finding your bucket's namespace and hostname.
- When you edit the host details on the **Edit this host page**, confirm the Transport Layer Security (TLS) area information for your host. Specifically, make sure you do the following:
  - Secure the connection between Fastly and your origin.
  - Enter your bucket's hostname in the **Certificate hostname** field.
  - Select the checkbox to match the SNI hostname to the Certificate hostname (it appears under the SNI hostname field).
- Also when you edit the host, optionally enable shielding by choosing the appropriate shielding location from the **Shielding** menu. When using Oracle Cloud Storage, this means you must choose a shielding location closest to the most appropriate Oracle region.
- Decide whether or not you should specify an override host in the **Advanced options** area which is the same as your bucket hostname.

Using the Oracle Cloud API with public objects

To use the Oracle Cloud API with public objects, you need to either create a new header, or a VCL Snippet. The purpose of the header or VCL snippet is to rewrite request URLs for your Oracle Cloud Storage instance.

Using a Header object

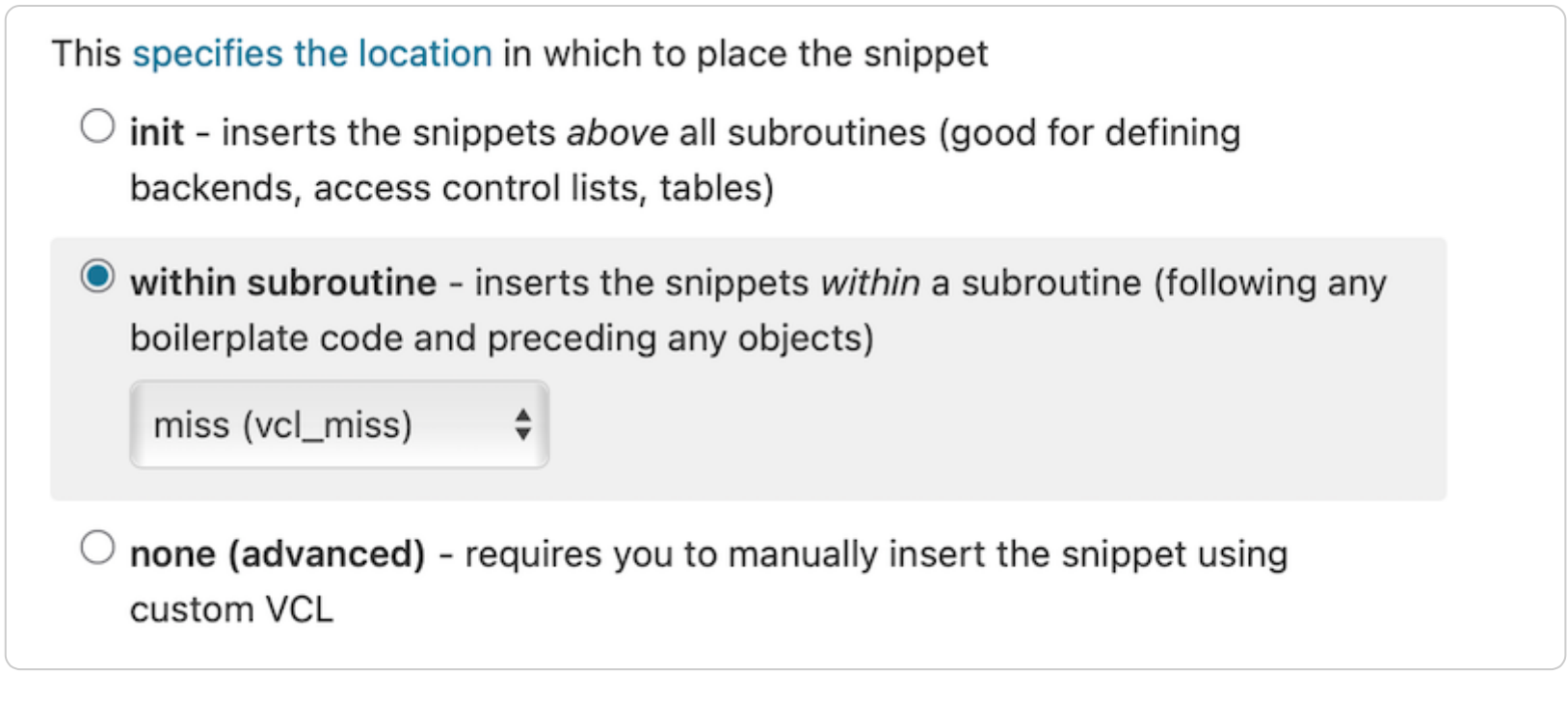
1. On your Fastly service's configuration page, click the **Create header** button to create a new header. The Create a header page appears.
2. Fill out the **Create a header** fields as follows:
   - In the **Name** field, enter `Rewrite Oracle Cloud Storage URL`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.

- In the **Destination** field, enter `url`.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, enter `20`.

3. In the **Source** field, enter `"/n/<namespace id>/b/<bucket name>/o" req.url` (e.g., `"/n/decafbaddeadbeef/b/fastly-bucket/o" req.url`).
4. Click the **Create** button. The new header appears on the Content page.
5. Click the **Add a condition** link next to the `Rewrite Oracle Cloud Storage URL` header. The Add a condition window appears.
6. Click the **Create a new request condition** button. The Create a new request condition window appears.
7. Fill out the condition fields as follows:
   - In the **Name** field, enter `Oracle Cloud Storage Shielding`.
   - In the **Apply if** field, enter `(req.method == "GET" && !req.backend.is_shield) {}`.
8. Click **Save and apply**.
9. Click the **Activate** button to deploy your configuration changes.

Using a VCL Snippet

1. Click **VCL Snippets** on your service's configuration page, then click **Create Snippet**.
2. In the Create a VCL Snippet page, enter a name for the snippet.
3. Select **within subroutine** to specify its placement, and **miss** as the subroutine type.



4. Add the following code to the **VCL** field. Change the values of the `oracleNamespace` and `oracleBucket` variables to match your Oracle namespace and bucket.

```
declare local var.oracleNamespace STRING;
declare local var.oracleBucket STRING;
set var.oracleNamespace = "YOUR_ORACLE_NAMESPACE_ID";   # Change this value to your own data
set var.oracleBucket = "YOUR_ORACLE_BUCKET_NAME";   # Change this value to your own data

if (req.method == "GET" && !req.backend.is_shield) {
  set bereq.url = "/n/" var.oracleNamespace "/b/" var.oracleBucket "/o/" bereq.url;
}
```

Using the S3 Compatible API with public objects

To use the S3 Compatible API with public objects you must create a new header, as explained below.

1. On your Fastly service's configuration page, click the **Create header** button to create a new header. The Create a header page appears.
2. Fill out the **Create a header** fields as follows:

- In the **Name** field, enter `Rewrite Oracle Cloud Storage URL`.
- From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
- In the **Destination** field, enter `url`.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, enter `20`.

3. In the **Source** field, enter `"/<bucket name>/" req.url` (e.g., `"/fastly-bucket/o" req.url`).
4. Click the **Create** button. A new header appears on the Content page.
5. Click the **Activate** button to deploy your configuration changes.

Private Buckets

> **IMPORTANT**
>
> Currently, Fastly can only support private objects using the S3 Compatible API.

To use an Oracle Cloud Storage private bucket with Fastly you must implement version 4 of Amazon's header-based authentication. You can do this using custom VCL. Keep in mind the following:

- You will need an Oracle Customer Secret Key which consists of an **Access Key** and **Secret Key**.
- You must use path-based access. Virtual host-style access (for example, accessing a bucket as `<bucketname>.<namespace>.compat.objectstorage.<region>.oraclecloud.com`) is not supported.

The following table lists the information you need to obtain from Oracle Cloud Storage before starting.

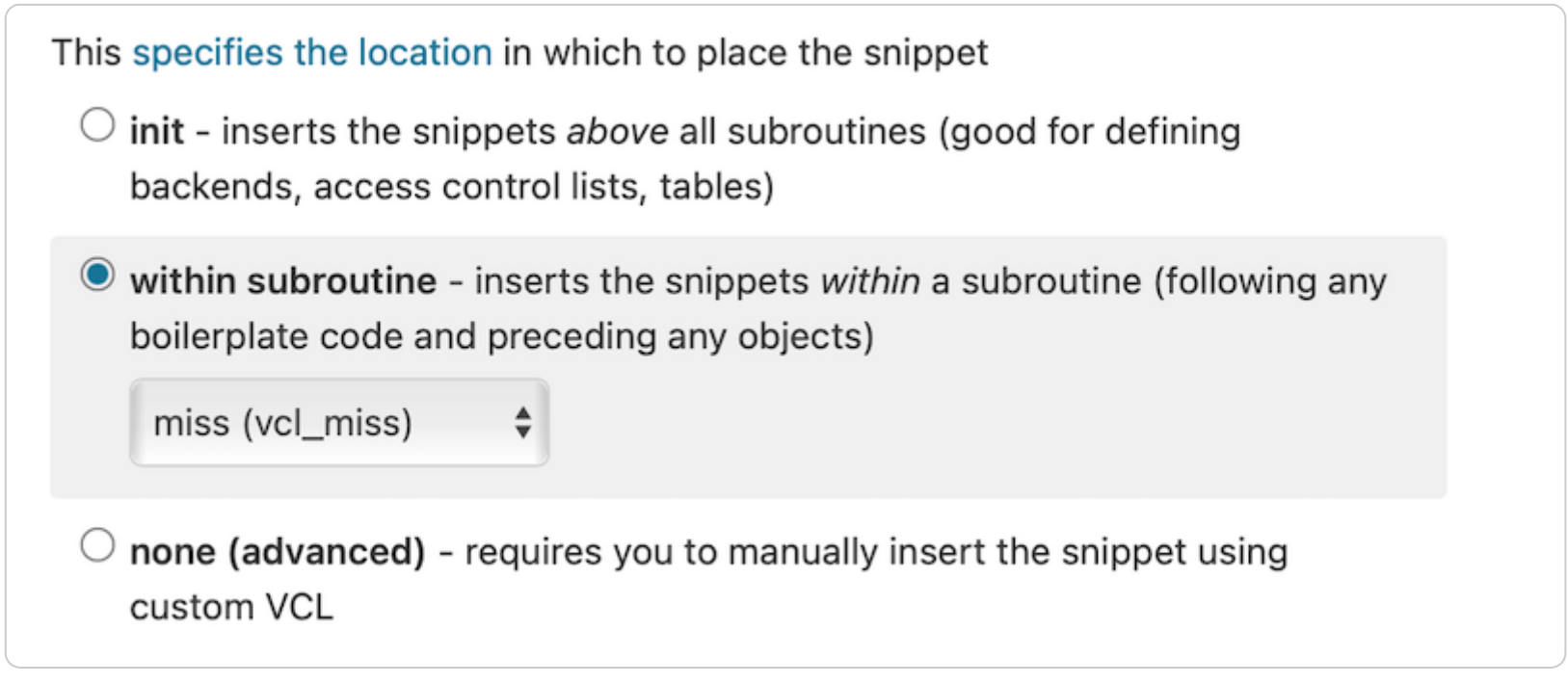
Item	Description
Namespace	The namespace identifier assigned to your bucket (see Finding your bucket's namespace and hostname.
Bucket name	The name of your OCS bucket. When you download items from your bucket, this is the string listed in the URL path or hostname of each object.
Region	The OCS region code of the location where your bucket resides (e.g., `us-east-1`).
Access key	The OCS access key string for your account that has at least read permission on the bucket.
Secret key	The OCS secret access key paired with the access key above.

Once you have this information, you can configure your Fastly service to authenticate against your S3 bucket using header authentication by calculating the appropriate header value in VCL.

> **IMPORTANT**
>
> Consider leaving the **Override host** field for the origin blank in your service settings. This setting will override the host header from the snippets shown here and may invalidate the signature that authenticates the information being sent.

Start by creating a regular VCL snippet. Give it a meaningful name, such as `AWS protected origin`. When you create the snippet, select **within subroutine** to specify its placement and choose **miss** as the subroutine type. Then, populate the **VCL** field with the following code (be sure to change specific values as noted to ones relevant to your own AWS bucket):

1. Click **VCL Snippets** on your service's configuration page, then click **Create Snippet**.
2. In the Create a VCL Snippet page give the snippet a meaningful name, such as `AWS protected origin`.
3. Select **within subroutine** to specify snippet placement, and **miss** as the subroutine type.



4. Add the following code to the **VCL** field. Be sure to change the values of the variables (`ocsNamespace`, `ocsAccessKey`, etc.) to match your Oracle environment.

```
declare local var.ocsNamespace STRING;
declare local var.ocsAccessKey STRING;
declare local var.ocsSecretKey STRING;
declare local var.ocsS3Bucket STRING;
declare local var.ocsRegion STRING;
declare local var.canonicalHeaders STRING;
declare local var.signedHeaders STRING;
declare local var.canonicalRequest STRING;
declare local var.canonicalQuery STRING;
declare local var.stringToSign STRING;
declare local var.dateStamp STRING;
declare local var.signature STRING;
declare local var.scope STRING;

set var.ocsNamespace = "YOUR_OCS_NAMESPACE"; # Change this value to your own data
set var.ocsAccessKey = "YOUR_OCS_ACCESS_KEY";   # Change this value to your own data
set var.ocsSecretKey = "YOUR_OCS_SECRET_KEY";   # Change this value to your own data
set var.ocsS3Bucket = "YOUR_OCS_BUCKET_NAME";   # Change this value to your own data
set var.ocsRegion = "YOUR_OCS_REGION";   # Change this value to your own data

if (req.method == "GET" && !req.backend.is_shield) {

  set bereq.http.x-amz-content-sha256 = digest.hash_sha256("");
  set bereq.http.x-amz-date = strftime({"%Y%m%dT%H%M%SZ"}, now);
  set bereq.http.host = var.ocsNamespace ".compat.objectstorage." var.ocsRegion ".oraclecloud.com";
  set bereq.url = querystring.remove(bereq.url);
  set bereq.url = regsuball(urlencode(urldecode(bereq.url.path)), {"%2F"}, "/");
  set var.dateStamp = strftime({"%Y%m%d"}, now);
  set var.canonicalHeaders = ""
    "host:" bereq.http.host LF
    "x-amz-content-sha256:" bereq.http.x-amz-content-sha256 LF
    "x-amz-date:" bereq.http.x-amz-date LF
  ;
  set var.canonicalQuery = "";
  set var.signedHeaders = "host;x-amz-content-sha256;x-amz-date";
  set var.canonicalRequest = ""
    "GET" LF
    bereq.url.path LF
    var.canonicalQuery LF
    var.canonicalHeaders LF
    var.signedHeaders LF
    digest.hash_sha256("")
  ;

  set var.scope = var.dateStamp "/" var.ocsRegion "/s3/aws4_request";

  set var.stringToSign = ""
    "AWS4-HMAC-SHA256" LF
    bereq.http.x-amz-date LF
    var.scope LF
    regsub(digest.hash_sha256(var.canonicalRequest),"^0x", "")
  ;

  set var.signature = digest.awsv4_hmac(
    var.ocsSecretKey,
    var.dateStamp,
    var.ocsRegion,
    "s3",
    var.stringToSign
  );

  set bereq.http.Authorization = "AWS4-HMAC-SHA256 "
    "Credential=" var.ocsAccessKey "/" var.scope ", "
    "SignedHeaders=" var.signedHeaders ", "
    "Signature=" + regsub(var.signature,"^0x", "")
  ;
  unset bereq.http.Accept;
  unset bereq.http.Accept-Language;
  unset bereq.http.User-Agent;
  unset bereq.http.Fastly-Client-IP;
}
```

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Outbound data transfer from Azure

Last updated: 2021-07-15

https://docs.fastly.com/en/guides/outbound-data-transfer-from-azure

Fastly has integrated local circuits with Microsoft Azure ExpressRoute Direct and Microsoft Routing Preference Unmetered to create private connections to Azure. If using Azure as your origin, you can take advantage of the improved reliability, faster speeds, lower latencies, and higher security that this private local circuit offers over typical public internet connections. Additionally, Fastly will include your outbound data transfer costs from Azure in your standard Fastly pricing.

To configure your Fastly service to use direct connectivity via this local circuit, enable shielding using your Azure service region to determine the shielding location. Additionally, configure the routing preference for your storage account based on the service region and shielding location combination. Our developer documentation lists the available shielding locations for the routing preference you've chosen.

NOTE

You can configure a routing preference after the storage account is created.

Once you configure your Fastly service and Azure storage account correctly, outbound data transfers from the appropriate Azure regions to Fastly will use this local circuit. Because Fastly has purchased this local circuit from Microsoft, Microsoft does not apply outbound data transfer rates to traffic traveling over it. Use of this local circuit should work with Azure services like Container Instances, Functions, and Media Services, as well as Blob Storage.

WARNING

We encourage you to use Microsoft Azure's Billing Tools for monitoring traffic not on the ExpressRoute Direct Local. Despite this connection to Fastly's services being in place, in certain circumstances your data may egress from Azure over the public internet rather than the ExpressRoute Direct connection. In such cases, your traffic to the public internet will be metered according to your commercial arrangement with Microsoft.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

PerimeterX Bot Defender

Last updated: 2018-05-11

https://docs.fastly.com/en/guides/perimeterx-bot-defender

Fastly provides direct integration between PerimeterX Bot Defender and Fastly edge servers. By placing a snippet of JavaScript (or HTML5) on your site and custom VCL directly into your Fastly service configuration, this integration allows you to gather behavioral data and statistics that may help you do things like detect invalid traffic and mitigate automated web attacks.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

How to get started

Integration with PerimeterX Bot Defender requires an account with PerimeterX. Once you have this account set up, contact your Fastly account manager or email sales@fastly.com to begin the integration process with Fastly. We'll work with you to configure your service to include the required code to enforce bot mitigation policies.

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Storj DCS Object Storage

Last updated: 2021-05-04

https://docs.fastly.com/en/guides/storj-dcs-object-storage

Storj DCS can be used as an origin for public and private Storj buckets via the Storj DCS S3 Gateway. Built on the Storj Network, Storj DCS is a decentralized object storage service that is S3 compatible and end-to-end encrypted by default.

Prerequisites

Before adding Storj DCS as an origin for Fastly services, you will need to create a Storj DCS account, project and access credentials, and a bucket that will serve as your origin.

Using Storj DCS as an origin

To use Storj DCS as an origin and make your Storj bucket available through Fastly via the Storj DCS S3 Gateway, follow the steps below.

Creating a new service

Follow the instructions for creating a new service.

1. When you create the new domain and the new host:
   - In the **Domain Name** field on the **Create a domain** page, enter the hostname you want to use as the URL (e.g., `cdn.example.com`).
   - In the **Hosts** field on the **Origins** page, enter the IP address or hostname of your Storj DCS Gateway Endpoint using the format `<BUCKET>.gateway.<REGION>.storjshare.io` including your bucket (e.g., `origin.gateway.us1.storjshare.io`).
2. When you edit the host details on the **Edit this host page**:
   - In the **Name** field, enter any descriptive name for your service if you haven't already done so.
   - In the **Address** field, ensure you've entered the IP address or hostname of your Storj DCS Gateway Endpoint. You entered this information during host creation.
3. When you edit the Transport Layer Security (TLS) area information for your host:
   - If you've set up TLS for your Storj DCS S3 Gateway, leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
   - Under the **SNI hostname** field, select the checkbox to **Match the SNI hostname to the Certificate hostname**. The address you entered during host creation appears.
   - In the **Certificate hostname** field, enter the IP address or hostname of your Storj DCS S3 Gateway.

Testing your results

By default, we create a DNS mapping called `yourdomain.global.prod.fastly.net`. In the example above, it would be `cdn.example.com.global.prod.fastly.net`. Create a DNS alias for the domain name you specified (e.g., CNAME `cdn.example.com` to `global-nossl.fastly.net`).

Fastly will cache any content without an explicit `Cache-Control` header for 1 hour. You can verify whether you are sending any cache headers using curl. For example:

```
$ curl -I https://cdn.example.com

Accept-Ranges: bytes
Content-Length: 250
Content-Type: application/xml
Server: MinIO/DEVELOPMENT.GOGET
Vary: Origin
Date: Wed, 07 Oct 2020 02:31:27 GMT
```

In this example, no Cache-Control headers are set so the default TTL will be applied.

Enhanced cache control

If you need more control over how different types of assets are cached (e.g., Javascript files, images), check out our documentation on cache freshness.

Using a Storj DCS bucket for origin hosting

To use a Storj DCS S3 Gateway as an origin with Fastly, you must implement version 4 of Amazon's header-based authentication. You can do this using custom VCL. Start by obtaining the following information from AWS:

Item	Description
Bucket name	The name of your private bucket. When you download items from your bucket, this is the string listed in the URL path or hostname of each object.
Access key	The access key string associated with a Storj DCS Access Grant that has at least read permissions on the bucket.
Secret key	The secret access key paired with the access key above.

Once you have this information, you can configure your Fastly service to authenticate against your private bucket using header authentication by calculating the appropriate header value in VCL.

IMPORTANT

Consider leaving the **Override host** field for the origin blank in your service settings. This setting will override the host header from the snippets shown here and may invalidate the signature that authenticates the information being sent.

Start by creating a regular VCL snippet. Give it a meaningful name, such as `Storj DCS Origin`. When you create the snippet, select **within subroutine** to specify its placement and choose **miss** as the subroutine type. Then, populate the **VCL** field with the following code (be sure to change specific values as noted to ones relevant to your own bucket):

```
declare local var.accessKey STRING;
declare local var.secretKey STRING;
declare local var.storjBucket STRING;
declare local var.storjGateway STRING;
declare local var.region STRING;
declare local var.canonicalHeaders STRING;
declare local var.signedHeaders STRING;
declare local var.canonicalRequest STRING;
declare local var.canonicalQuery STRING;
declare local var.stringToSign STRING;
declare local var.dateStamp STRING;
declare local var.signature STRING;
declare local var.scope STRING;


set var.accessKey = "YOUR_ACCESS_KEY";   # Change this value to your own data
set var.secretKey = "YOUR_SECRET_KEY";   # Change this value to your own data
set var.storjBucket = "YOUR_BUCKET_NAME";   # Change this value to your own data
set var.storjGateway = "STORJ-DCS_GATEWAY";   # Change this value to your own data
set var.region = "decentralized";


if (req.method == "GET" && !req.backend.is_shield) {

  set bereq.http.x-amz-content-sha256 = digest.hash_sha256("");
  set bereq.http.x-amz-date = strftime({"%Y%m%dT%H%M%SZ"}, now);
  set bereq.http.host = var.storjBucket "." var.storjGateway;
  set bereq.url = querystring.remove(bereq.url);
  set bereq.url = regsuball(urlencode(urldecode(bereq.url.path)), {"%2F"}, "/");
  set var.dateStamp = strftime({"%Y%m%d"}, now);
  set var.canonicalHeaders = ""
        "host:" bereq.http.host LF
        "x-amz-content-sha256:" bereq.http.x-amz-content-sha256 LF
        "x-amz-date:" bereq.http.x-amz-date LF
  ;
  set var.canonicalQuery = "";
  set var.signedHeaders = "host;x-amz-content-sha256;x-amz-date";
  set var.canonicalRequest = ""
        "GET" LF
        bereq.url.path LF
        var.canonicalQuery LF
        var.canonicalHeaders LF
        var.signedHeaders LF
        digest.hash_sha256("")
  ;

  set var.scope = var.dateStamp "/" var.region "/s3/aws4_request";


  set var.stringToSign = ""
        "AWS4-HMAC-SHA256" LF
        bereq.http.x-amz-date LF
        var.scope LF
        regsub(digest.hash_sha256(var.canonicalRequest),"^0x", "")
  ;

  set var.signature = digest.awsv4_hmac(
        var.secretKey,
        var.dateStamp,
        var.region,
        "s3",
        var.stringToSign
  );


 set bereq.http.Authorization = "AWS4-HMAC-SHA256 "
        "Credential=" var.accessKey "/" var.scope ", "
        "SignedHeaders=" var.signedHeaders ", "
        "Signature=" + regsub(var.signature,"^0x", "")
  ;

  unset bereq.http.Accept;
  unset bereq.http.Accept-Language;
  unset bereq.http.User-Agent;
  unset bereq.http.Fastly-Client-IP;
}
```

[Wasabi Hot Cloud Storage]

Last updated: 2022-03-01

https://docs.fastly.com/en/guides/wasabi-hot-cloud-storage

Wasabi Hot Cloud Storage public and private buckets can be used as origins with Fastly.

Using Wasabi as an origin

To make your Wasabi Hot Cloud Storage bucket available through Fastly, follow the steps below.

Creating a new service

Follow the instructions for creating a new service.

1. When you create the new domain and the new backend host:
   - In the **Domain Name** field on the **Create a domain** page, enter the hostname you want to use as the URL (e.g., `cdn.example.com`).
   - In the **Hosts** field on the **Origins** page, enter the appropriate address for your Wasabi Hot Cloud Storage bucket's region. For the us-east-1 region, enter `<BUCKET>.s3.wasabisys.com`. For all other regions, enter `<BUCKET>.s3.<REGION>.wasabisys.com`, replacing `<REGION>` as appropriate (e.g., `<BUCKET>.s3.eu-central-1.wasabisys.com`).
2. When you edit the host details on the **Edit this host page**:
   - In the **Name** field, enter any descriptive name for your service if you haven't already done so.
   - In the **Address** field, ensure you've entered the appropriate address for your Host (e.g., `<BUCKET>.s3.wasabisys.com`). You entered this information during Host creation.
3. When you edit the Transport Layer Security (TLS) area information for your host:
   - Leave the **Enable TLS?** default set to **Yes** to secure the connection between Fastly and your origin.
   - In the **Certificate hostname** field, enter the same address that appears in the Address field (e.g., `<BUCKET>.s3.wasabisys.com`).
   - Under the **SNI hostname** field, select the checkbox to **Match the SNI hostname to the Certificate hostname**. The address you entered during Host creation appears.
4. In the **Override host** field, enter an appropriate address for your Host (e.g., `<BUCKET>.s3.wasabisys.com`). You entered this information during Host creation.
5. From the **Shielding** menu below the TLS area, select an appropriate shielding location. For more information about this setting and which locations to select, see our enabling shielding information.

Enabling shielding

We strongly encourage you to enable shielding for your origin server. Wasabi imposes soft caps on free egress. Without shielding enabled, Fastly will request the same objects from all Fastly edge POPs instead of just one, which may not follow Wasabi's free egress guidelines.

When you select a shielding location from the **Shielding** menu, choose the location appropriate for your Wasabi Hot Cloud Storage bucket as follows:

Wasabi bucket region	Shielding location
eu-central-1	Amsterdam, NL
us-east-1	Ashburn, VA
us-west-1	Seattle, WA

Read our guidance on choosing a shield location for more information.

Testing your results

By default, we create a DNS mapping called **yourdomain.global.prod.fastly.net**. In the example above, it would be `cdn.example.com.global.prod.fastly.net`. Create a DNS alias for the domain name you specified (e.g., CNAME `cdn.example.com` to `global-nossl.fastly.net`).

Fastly will cache any content without an explicit `Cache-Control` header for 1 hour. You can verify whether you are sending any cache headers using curl. For example:

```
$ curl -I opscode-full-stack.s3.wasabisys.com

HTTP/1.1 200 OK
x-amz-id-2: ZpzRp7IWc6MJ8NtDEFGH12QBdk2CM1+RzVOngQbhMp2f2ZyalkFsZd4qPaLMkSlh
x-amz-request-id: ABV5032583242618
Date: Fri, 18 Mar 2012 17:15:38 GMT
Content-Type: application/xml
Transfer-Encoding: chunked
```

In this example, no Cache-Control headers are set so the default TTL will be applied.

Enhancing cache control

If you need more control over how different types of assets are cached (e.g., Javascript files, images), check out our documentation on cache freshness.

Using private Wasabi Hot Cloud Storage buckets

To use a Wasabi Hot Cloud Storage private bucket with Fastly, you must implement version 4 of Amazon's header-based authentication. You can do this using custom VCL and following the instructions below.

Before you begin

Make your Wasabi Hot Cloud Storage bucket available to Fastly. Be sure you've set your origin to port 443. This needs to be done before implementing header-based authentication with the instructions below.

Gathering Wasabi information

Start by obtaining the following information from Wasabi:

Item	Description
Bucket Name	The unique name of your Wasabi Hot Cloud Storage bucket. When you download items from your bucket, this is the string listed in the URL path or hostname of each object (e.g., `widget-project`).
Region	The Wasabi region code of the location where your bucket resides (e.g., `us-east-1`).
Access Key ID	The Wasabi access key ID string for an IAM account that has at least read permission on the bucket.
Secret Access Key	The Wasabi secret access key paired with the access key above.

You should review the user access separation document to make sure you are not inadvertently exposing files you didn't intend e.g. allowing ListBucket operations etc. Alternatively you can use the VCL snippet from the bottom of the document to block bucket listing.

Once you have this information, you can configure your Fastly service to authenticate against your Wasabi bucket using header authentication by calculating the appropriate header value in VCL.

Creating a VCL snippet for authentication

Create a regular VCL snippet.

- In the **Name** field, enter `Wasabi protected origin`.
- In the **Type (placement of the snippet)** field, select **within subroutine** then choose `miss (vcl_miss)`.

- In the **VCL** field, place the following code (be sure to change specific values as noted to ones relevant to your own Wasabi bucket):

```
if ( req.request == "GET" && req.backend.is_origin) {

  declare local var.wasabiAccessKey STRING;
  declare local var.wasabiSecretKey STRING;
  declare local var.wasabiBucket STRING;
  declare local var.wasabiRegion STRING;
  declare local var.canonicalHeaders STRING;
  declare local var.signedHeaders STRING;
  declare local var.canonicalRequest STRING;
  declare local var.canonicalQuery STRING;
  declare local var.stringToSign STRING;
  declare local var.dateStamp STRING;
  declare local var.signature STRING;
  declare local var.scope STRING;

  # Supply your own credentials
  set var.wasabiAccessKey = "YOUR_BUCKET_ACCESS_KEY";   # Change this value to your own data
  set var.wasabiSecretKey = "YOUR_BUCKET_SECRET";       # Change this value to your own data
  set var.wasabiBucket = "YOUR_BUCKET_NAME";            # Change this value to your own data
  set var.wasabiRegion = "YOUR_BUCKET_REGION";          # Change this value to your own data

  set bereq.http.x-amz-content-sha256 = digest.hash_sha256("");
  set bereq.http.x-amz-date = strftime({"%Y%m%dT%H%M%SZ"}, now);
  set bereq.http.host = var.wasabiBucket ".s3." var.wasabiRegion ".wasabisys.com";
  set bereq.url = querystring.remove(bereq.url);
  set var.dateStamp = strftime({"%Y%m%d"}, now);
  set var.canonicalHeaders = ""
    "host:" bereq.http.host LF
    "x-amz-content-sha256:" bereq.http.x-amz-content-sha256 LF
    "x-amz-date:" bereq.http.x-amz-date LF
  ;
  set var.canonicalQuery = "";
  set var.signedHeaders = "host;x-amz-content-sha256;x-amz-date";
  set var.canonicalRequest = ""
    "GET" LF
    bereq.url.path LF
    var.canonicalQuery LF
    var.canonicalHeaders LF
    var.signedHeaders LF
    digest.hash_sha256("")
  ;

  set var.scope = var.dateStamp "/" var.wasabiRegion "/s3/aws4_request";

  set var.stringToSign = ""
    "AWS4-HMAC-SHA256" LF
    bereq.http.x-amz-date LF
    var.scope LF
    regsub(digest.hash_sha256(var.canonicalRequest),"^0x", "")
  ;

  set var.signature = digest.awsv4_hmac(
    var.wasabiSecretKey,
    var.dateStamp,
    var.wasabiRegion,
    "s3",
    var.stringToSign
  );

  set bereq.http.Authorization = "AWS4-HMAC-SHA256 "
    "Credential=" var.wasabiAccessKey "/" var.scope ", "
  "SignedHeaders=" var.signedHeaders ", "
    "Signature=" + regsub(var.signature,"^0x", "")
  ;
  unset bereq.http.Accept;
  unset bereq.http.Accept-Language;
  unset bereq.http.User-Agent;
  unset bereq.http.Fastly-Client-IP;
 }
```

Creating a VCL snippet to remove added response headers

You may also remove the headers that Wasabi adds to the response. Do this by creating another VCL snippet.

- In the **Name** field, enter `Strip Wasabi response headers`.

- In the **Type (placement of the snippet)** field, select **within subroutine** then select `deliver (vcl_deliver)`.
- In the **VCL** field, place the following code:

```
if ( !req.http.Fastly-Debug ) {
  unset resp.http.x-amz-id-2;
  unset resp.http.x-amz-request-id;
  unset resp.http.server;
}
```

Blocking directory listing

If you don't set up correct IAM privileges you may allow users to list contents of your bucket folders. If you want to disallow that on Fastly please create following snippet

- In the **Name** field, enter `Disallow bucket listing`.
- In the **Type (placement of the snippet)** field, select **within subroutine** then select `recv (vcl_recv)`.
- In the **VCL** field, place the following code:

```
if ( req.url.path ~ "/$" ) {
  error 403;
}
```

This article describes an integration with a service provided by a third party. See our note on integrations for details.

Diagnostics

These articles describe how to log data, troubleshoot problems, and tune performance.

https://docs.fastly.com/en/guides/diagnostics

§

These articles describe common errors you may encounter when setting up and configuring Fastly services.

https://docs.fastly.com/en/guides/diagnostics#_common-errors

Common 503 errors on Fastly

Last updated: 2020-07-06

https://docs.fastly.com/en/guides/common-503-errors

Varnish, the software that powers the Fastly CDN, will sometimes return standardized 503 responses due to various issues that can occur when attempting to fetch data from your origin servers. The generic status text associated with a 503 error is "Service Unavailable." It can mean a wide variety of things. The most common reasons this generic text appears include:

1. The origin server generated a 503 error and Fastly passed it through as is.
2. The origin returned a 503 error without a response header, so Fastly used the default response.
3. The status line of the HTTP response from the origin was not parsable.
4. VCL code was run that used the "error" statement without an appropriate response status (e.g., `error 503` instead of `error 503 "broken thing"`).

The following list provides the most common non-generic, standardized 503 responses and basic explanations for each.

> **WARNING**
>
> If you are seeing 503 errors, do not purge all cached content. Purge all overrides stale-if-error and increases the requests to your origin server, which could result in additional 503 errors.

Timeout errors

The following describes typical timeout errors you may encounter.

Error 503 backend read error

This error typically appears if a timeout error occurs when Fastly cache servers attempt to fetch content from your origins. It can also be due to a variety of transient network issues that might cause a read to fail (e.g., router failovers, packet loss) or an origin overload.

Benchmarking your backend response times. Many outside factors cause backends response times to vary. Repeated, consistent backend read errors frequently can be prevented by changing your backend timeout settings in the Fastly web interface. Start by running the following command to estimate response time for benchmarking purposes:

```
$ curl -s -w "%{time_total}\n" -o /dev/null http://example.com/path/to/file
```

Increasing your backend timeout settings. After benchmarking some of the slower paths in your application, you should have an idea of your ideal backend response time. Adjust the backend timeout values on the Edit this host page in the Advanced options area. Also, if there is an external interface in front of the origin (such as a load balancer or firewall), review the timeouts for these interfaces.

Error 503 connection timed out

This error occurs if the request times out while waiting for Fastly to establish a TCP connection to your origin or waiting for your origin to respond to the request. Similar to backend read errors, connection timeouts can be caused by transient network issues, long trips to origin, and origin latency. Two common ways to alleviate these timeout errors include:

- Increasing the connection timeout values set for the Fastly Host.
- Setting up an origin shield. Setting up an origin shield provides two advantages:
  - Shortening the distance needed to establish a connection.
  - Reducing TCP handshakes resulting from using multiple POPs. This allows the origin to avoid slowdowns and to process only requests on a few connections from the shield.

> **NOTE**
>
> Fastly enforces a 60 second timeout between nodes unless you're passing requests in `vcl_recv`.

Error 503 backend write error

This error is similar to the backend read error but occurs when Fastly sends information in the form of a POST request to the backend. This error can be resolved the same way as the backend read error.

Error 503 client read error

This error generally occurs because of a network issue between the client and Fastly. It can also occur when a user abandons the loading of a page (e.g., a page is loading too slowly and the user clicks stop in the browser). It is similar to the backend read error but occurs when reading information from a client. If you get this error, contact Fastly support for help identifying the network issue.

Error 503 backend fetch failed

This error occurs when the connection closes before Fastly cache servers are done reading the response. This error can occur when there is a missing or invalid `Content-Length` header on the response, although there may be other causes. To resolve this, verify that your origin includes either a `Content-Length` or a `Transfer-Encoding: chunked` header along with the response. If

neither header is present, first ensure that one of them is added. If one of the headers is present, verify that the whole resource can be received from the origin directly. If either header is present and the whole resource can be received from the origin, contact Fastly support for more assistance.

Error 503 first byte timeout

This error occurs when Fastly establishes a connection to your origin, but the origin doesn't start sending the response within the time you've configured for your first byte timeout. To resolve this, extend your first byte timeout for your origin.

By default, the first byte timeout is set to 15 seconds. You can extend the maximum timeout possible to 600s. However, be aware of the following:

- If your origin is configured with a shield, the timeout maximum should be reduced to 60s.
- Clustering limits the maximum timeout to 60s. If an object is cacheable, then to increase the maximum timeout of 60s you must disable clustering by adding the `Fastly-No-Shield` header in `vcl_recv`. If you decide to add the `Fastly-No-Shield` header, make sure your condition precisely targets the requirements that take more than 60 seconds as adding it will affect your cache hit ratio.

Origin and service configuration errors

The following describes typical origin and Fastly service configuration errors you may encounter.

503 Response object too large

If Fastly determines the object being fetched exceeds the resource size limit of your Fastly service, we will generate a `503 Response object too large` response to the client. You can use the Segmented Caching feature to eliminate these errors.

Error 503 connection refused

This error occurs when Fastly attempts to make a connection to your origin over a specific port and the server refuses the connection. It typically appears when the wrong port is specified for the Host in the Fastly web interface. To resolve this error, you may need to adjust your port number to ensure you're using the port needed to connect to your origin. If adjusting your port number doesn't work, you may also need review your origin configurations to ensure you're allowing connections from Fastly specific IPs.

Error 503 illegal Vary header from backend

This error occurs when a backend returns a malformed Vary header with its response. A well-formed Vary header tells Fastly to serve a different version of an object based on the value of the request header included within it.

Error 503 network unreachable

This error appears when Fastly can't find a route to the given IP range. This generally occurs because of misconfigured or non-operational routers. To resolve this error, check your routers to ensure they are operational or configured correctly.

Origin health errors

The following describes typical origin health errors you may encounter.

Error 503 backend is unhealthy

This error appears when custom health checks report a backend as down. It typically occurs when a Fastly edge server receives a client request and must make a request to your origin, but because the backend is considered unhealthy, Fastly doesn't try to send the request at all. Some of the reasons this error may occur are:

- the origin took too long to respond to the request
- there are transient network issues and the health check couldn't get to the origin
- the health check was misconfigured, or the resource the health check is checking against was removed or altered in some way

To resolve this error, check to make sure your origin is configured correctly and the object the health check is requesting exists at the specified location.

Error 503 no stale object available

This error occurs when you configure Fastly to serve stale objects in the event of a backend failure but the stale object has expired and your backend is still failing for some reason (thus, no stale object is available). To resolve this error, you will need either to fix your origin or check your network.

Connection limit errors

The following describes typical connection limit errors you may encounter.

Error 503 backend.max_conn reached

This error occurs when Varnish makes a request to a backend in your Fastly service that has reached its defined maximum number of connections. By default, Fastly limits you to 200 origin connections from a single edge node to protect the origins from overload. For the majority of sites, this should be enough. If you get this error message with less than 10,000 non-hit requests per second, make sure your origin is responding normally (e.g., there are no origin slow downs). If you just increase the number of maximum connections, you may be exacerbating the problem. If you have determined that your origin is not the issue, increase the maximum connections limit to your origin or reach out to Fastly support for further help with this issue. This error may also appear as "Error 503 maximum threads for service reached."

Error 503 maximum threads for service reached

This error occurs when Varnish detects that a service has exceeded a safety limit on the number of concurrent requests. Typically this indicates that a service is experiencing an unusually high load, that an origin is slow, or that features like request collapsing are being intentionally avoided.

Director errors

The following describes typical Director errors you may encounter.

Error 503 no healthy backends

This error occurs when a Director used for balancing requests among a group of backends (only available via the Fastly API) can't cache the specified content because there are no healthy backends available in its group.

Error 503 all backends failed or unhealthy

This error occurs when a Director used for balancing requests among a group of backends (only available via the Fastly API) fails because all the backends are unhealthy or multiple backends from which the Director tried to fetch information failed with the same error.

Error 503 quorum weight not reached

This error occurs when a Director used for balancing requests among a group of backends (only available via the Fastly API) can't serve traffic based on its configuration because it does not have enough available backends in its group.

To resolve any of these errors, you should either check for and resolve any issues with your origin or make sure the quorum setting is correct. Also, make sure you are setting the quorum setting correctly. For example, in a five backend director, 85% of the quorum will mark the director unhealthy if a single backend is unhealthy.

TLS errors

The following describes typical TLS errors you may encounter. You also can find information about other common TLS errors at your origin in the TLS origin configuration messages guide.

Error 503 SSL handshake error

This error occurs when TLS negotiation between Fastly and your origin fails. To fix this error, review and correct your host's TLS configurations.

Error 503 unable to get local issuer certificate

This error occurs when a certificate in the certificate chain is missing or invalid. To better determine which of these issues is the cause of the error, we suggest running an SSL test on your origin to highlight any issues with the certificate installed there. There are two common ways you can resolve this error:

- For missing or invalid certificates, download and replace the missing or incorrect certificate.
- If both the intermediate and root certificates are correct, insert a valid Server Name Indication (SNI) hostname in the origin TLS options of your Fastly service.

Error 503 hostname doesn't match against certificate

This error occurs when the certificate hostname specified in your service's origin TLS settings does not match either the Common Name (CN) or available Subject Alternate Names (SANs). To resolve this error, enter a certificate hostname value that matches the CN or SAN entries on your origin's certificate.

Error 503:14077410:SSL routines:SSL23_GET_SERVER_HELLO:sslv3 alert

This error occurs when Server Name Indication (SNI) is required in the TLS handshake to origin, but the SNI hostname field is either blank or incorrect. To correct this error, enter a hostname value in the SNI hostname field. Often this will match the value specified in the certificate hostname field.

Error 503 certificate has expired

This error occurs when a certificate installed at the origin expires. To resolve this, renew your certificate or download a new one.

Common OTFP errors

Last updated: 2020-01-26

https://docs.fastly.com/en/guides/common-otfp-errors

This page lists some possible error values that Fastly's On-The-Fly Packaging service (OTFP) service will send in the `X-Fastly-Package-Error` response header when attempting to fetch video data from your origin server or when repackaging your MP4 files. For help troubleshooting, send the full URL of the response to support@fastly.com.

NOTE

There are certain cases where you may see one of these messages as a warning instead of an error.

404 responses

In a 404 response from OTFP, the following header value is returned:

`Origin 404` - This error occurs if the source MP4 does not exist on origin.

5xx responses

In a 5xx response from OTFP, header values might include the following.

500 responses

- `Service configuration error` - This error typically occurs when something is wrong with one of the cache control headers that determine the behavior of an OTFP service.
- `Incompatible or corrupt origin media` - This error typically occurs when the remote MP4 being requested is corrupt or the requested file doesn't actually use the MP4 container format.

- `Slice out of bounds` - This error occurs when the start and end markers of a video do not align with the actual duration of that video.
- `Video track has no key frames` - This error occurs when the source video MP4 file does not include an `stss` box, which has information about key frames.
- `Flapping origin entity` - This error occurs if conflicting entity tags (ETags) were seen when handling a request. It's possible that the file changed legitimately during the request, and a follow-up request should work as expected. If the problem persists, it indicates that the origin can't handle ETags correctly.
- `File is not a WebVTT` - This error occurs when Fastly is unable to parse the remote WebVTT file.
- `Unexpected EOF on origin fetch` - This error occurs because an attempt to fetch segment media failed due to the file's size being smaller than its index indicated.
- `Track media is not in order` - This error occurs if Fastly cannot handle the MP4 because the media in its media data box (`mdat` box) is out of order, as indicated by the `stco` or `co64` box, which has information about the location of data in the media track's data stream.
- `moov box is too big` - This error occurs when the index portion of the MP4 is larger than 500MiB.
- `Mp4 has too many frames` - This error occurs when the total number of samples (between both the audio and video track) exceeds 40,000,000.
- `ftyp box is too big` - This error occurs if the file type box (`ftyp`) is larger than necessary.
- `Track lacked valid track id` - This error occurs if the Track ID was not set to a value greater than 0 in the `tkhd` box.
- `Track lacks valid timescale` - This error occurs if the `timescale` field in the media header box (`mdhd` box) did not have valid settings.
- `Track lacks valid duration` - This error occurs if the `duration` field in the media header box (`mdhd` box) did not have valid settings.
- `Track XXXX and XXXX do not agree` - This error indicates that the MP4 index is corrupt.
- `Empty XXXX, missing XXXX, illegal XXXX, ....` - This error indicates that the MP4 index is corrupt.
- `Unable to unmarshal expressplay response` - This error occurs when ExpressPlay is responding unexpectedly.
- `expressplay: params unable to match expected format` - This error indicates that the `X-Fastly-Drm` header has invalid base64 encoded data after `expressplay:`.
- `Origin segment media too large` - This error indicates that the file must be repackaged.
- `Unsupported audio codec 0x<##>` - This error indicates that the video/audio is encoded with an audio codec that's not supported by OTFP. Example: `Unsupported audio codec 0x69`.
- `No audio/video samples found. Fragmented or malformed MP4 detected.` - This error indicates that the source MP4 provided is likely a fragmented MP4 (fmp4) or not an MP4.
- `HEVC in HLS transport stream is not supported` - This error indicates that you need to use fragmented MP4 segments (.fmp4) instead of transport stream (.ts) requests. (Transport stream requests are not compatible with HEVC source video files.) You can specify use of fragmented MP4 segments by including the `X-Fastly-Hls-Fragmented-Mp4: true` header in your VCL request. If, after setting that header, you are experiencing playback issues on Apple devices using HEVC, you can try changing the header value to `force-hvc1`, which will ensure that OTFP is using `hvc1` sample boxes instead of `hev1`: `X-Fastly-Hls-Fragmented-Mp4: force-hvc1`. This can sometimes fix mislabeled streams.

502 responses

- `Invalid response from origin` - This error typically occurs when the response from the server is something other than an expected 206, 404, or 5xx response. For example, returning a 403 error instead of a 404 when a file is missing.
- `Invalid Content-Range from origin` - This error occurs if the origin responded to a Range request with a Content-Range header that was illegal or did not match the request Fastly made.
- `Failed to reach origin` - This error typically indicates that a path or bucket is incorrectly set in X-Fastly-Origin.
- `Validator supplied but ignored` - This error indicates that the origin is not handling ETags correctly. Specifically, when asked If-None-Match, instead of responding with a 304 status, the origin responded with a 2xx status that had the same

ETag. See RFC 7232, section 3.2 for details.

504 responses

`Origin timeout` - This error occurs if the OTFP was unable to connect or fetch content from origin inside of 15 seconds.

Common service and domain errors

Last updated: 2018-04-24

https://docs.fastly.com/en/guides/common-service-and-domain-errors

Exceeding max number of domains

We currently limit the maximum number of services and domains you can configure (including when you create domains programmatically). Once you reach that limit, error messages may appear that look something like this:

```
{
  "msg": "An error occurred while connecting to the fastly API, please try your request again.",
  "detail": "Exceeding max number of domains: 10"
}
```

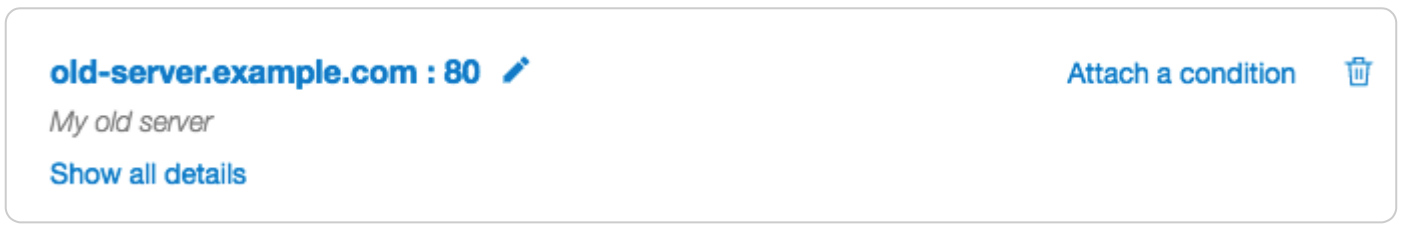
If you're receiving a limit message and need to create more services or domains, contact support@fastly.com for assistance. Fastly support engineers can not only increase the number of services that you can use, they can suggest other ways to design what you are trying to achieve.

Error 1000 with CloudFlare DNS

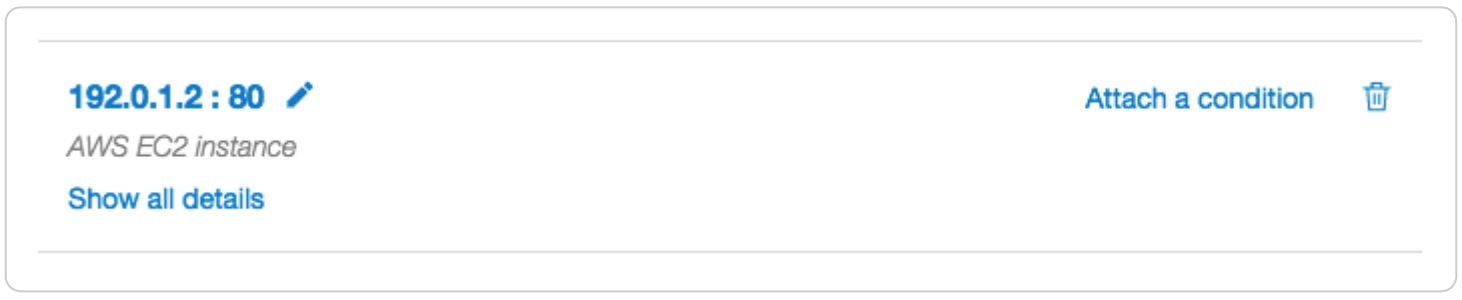
Last updated: 2017-11-03

https://docs.fastly.com/en/guides/error-1000-with-cloudflare-dns

Using CloudFlare for DNS and other CDNs can cause CloudFlare to show an Error 1000 indicating that your DNS points to prohibited IP addresses. This occurs when the hostnames are CNAMEed to Fastly and an origin server is configured as a fully qualified domain name (FQDN) within Fastly:



To solve this error, direct Fastly to use the IP address as the Host for any backend origin servers. This removes the need to resolve the hostname for traffic to the servers:



You can also change this by modifying the VCL configuration files directly. For example, this VCL:

```
backend F_Hosting_server_Example_Backend {
  ...
  .port = "80";
  .host = "exampleserver.exampledomain.tld";
}
```

would become:

```
backend F_Hosting_server_Example_Backend {
  ...
  .port = "80";
  .host = "12.34.56.78";
}
```

Fixing cross-domain errors

Last updated: 2021-09-08

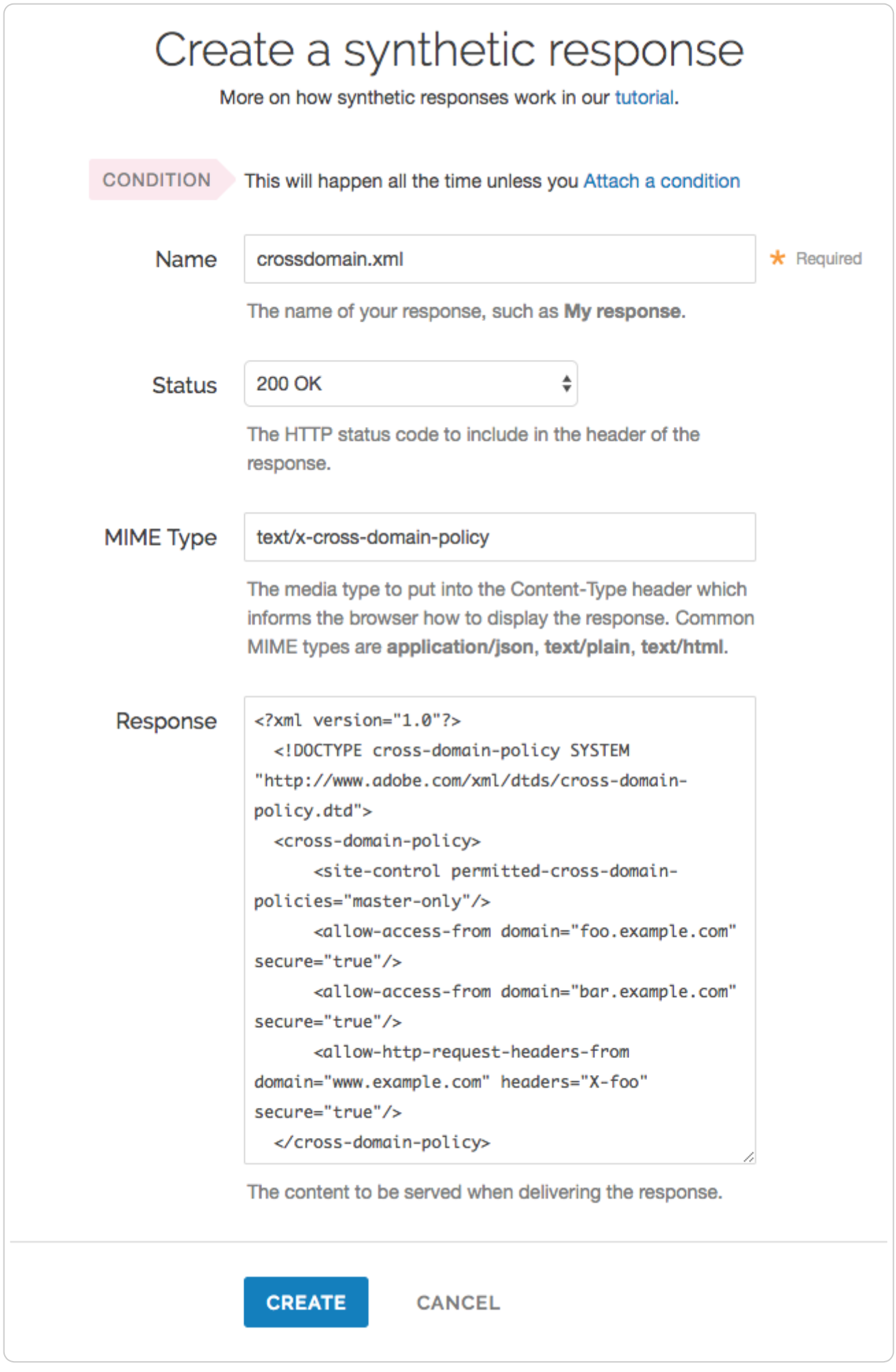
https://docs.fastly.com/en/guides/fixing-cross-domain-errors

Browser plugins, like Adobe Flash, often require permissions to play content hosted on domains other than from which they are hosted. The crossdomain policy file grants this permission and needs to be present in many cases to allow the content to be played. This guide shows you how to create a synthetic crossdomain.xml response to resolve cross-domain errors.

TIP

`Error #2048` is a common indicator of a crossdomain.xml issue.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Set up advanced response** button. The Create a synthetic response page appears.



6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, enter a human-readable name for the response. For example `crossdomain.xml`.
   - From the **Status** menu, select an HTTP code to return to the client. For example, `200 OK`.
   - In the **MIME Type** field, enter `text/x-cross-domain-policy` for the MIME type of the response.
   - In the **Response** field, add the correctly-formatted crossdomain.xml content you want the request to respond with. See cross-domain permissiveness and restrictiveness for additional details.
7. Click the **Create** button. Your new response appears in the list of responses.
8. Click the **Attach a condition** link to the right of the name of your new response. The Create a new condition window appears.
9. Fill out the **Create a new condition** fields as follows:
   - From the **Type** menu, select **Request**.
   - In the **Name** field, enter a human-readable name for the response condition. For example `crossdomain.xml`.
   - In the **Apply if** field, enter `req.url == "/crossdomain.xml"`.
10. Click the **Save and apply to** button to create the new request condition.

11. Click the **Activate** button to deploy your configuration changes.

Cross-domain permissiveness and restrictiveness

A crossdomain.xml policy file grants these browser plugins permissions to allow content to be played from domains other than that which they are hosted. This file usually has the name `crossdomain.xml` and gets placed by default in the root directory of the domain on which it is hosted. You use this file to define how permissive or restrictive access will be when attempting to play the content being requested.

The following example policy allows the `foo.example.com` and `bar.example.com` domains to pull data, and the `www.example.com` domain to push data via the `X-foo` header:

```
<?xml version="1.0"?>
  <!DOCTYPE cross-domain-policy SYSTEM "http://www.adobe.com/xml/dtds/cross-domain-policy.dtd">
  <cross-domain-policy>
      <site-control permitted-cross-domain-policies="master-only"/>
      <allow-access-from domain="foo.example.com" secure="true"/>
      <allow-access-from domain="bar.example.com" secure="true"/>
      <allow-http-request-headers-from domain="www.example.com" headers="X-foo" secure="true"/>
  </cross-domain-policy>
```

NOTE

This example uses `secure="true"` to force access via HTTPS. You can use `secure="false"` to allow access via HTTP.

Loop detection

Last updated: 2018-04-24

https://docs.fastly.com/en/guides/loop-detection

Fastly automatically detects loops resulting from service configuration errors. When a loop is detected, Fastly blocks the requests and generates an error message. Loops can occur when the same hostname is configured as both the domain and the origin server, and the CNAME record for the domain is pointed at Fastly. For example, loop detection will be triggered if you set `www.example.com` as the domain and the origin server in your Fastly service and you add a CNAME DNS record for `www.example.com` that points at Fastly.

How to avoid triggering loop detection

To avoid triggering loop detection, you should verify the hostname of your origin server is not the same as the domain using one of the following two options:

- Create a DNS hostname (`origin.example.com`) with the appropriate A and AAAA DNS records for your origin server, and use that origin DNS hostname in your Fastly service configuration. This ensures the origin (`origin.example.com`) is different than the domain (`www.example.com`) on your service. We recommend this option. If you make changes to the DNS records for `origin.example.com` in the future, Fastly will automatically detect and use those changes.
- Use an IPv4 address instead of a DNS hostname for your origin's address within your Fastly service's configuration. If the origin server's IP address changes in the future, you'll need to update and activate a new version of your Fastly service configuration.

Example error message

When Fastly detects a loop, an error message similar to the one displayed below will appear in the headers.

```
HTTP/1.1 503 Loop detected
Error-Reason: loop detected
Connection: close
Content-Type: text/plain
Fastly-Host: <hostname>
Fastly-FF: <hostname>
Server: Varnish
```

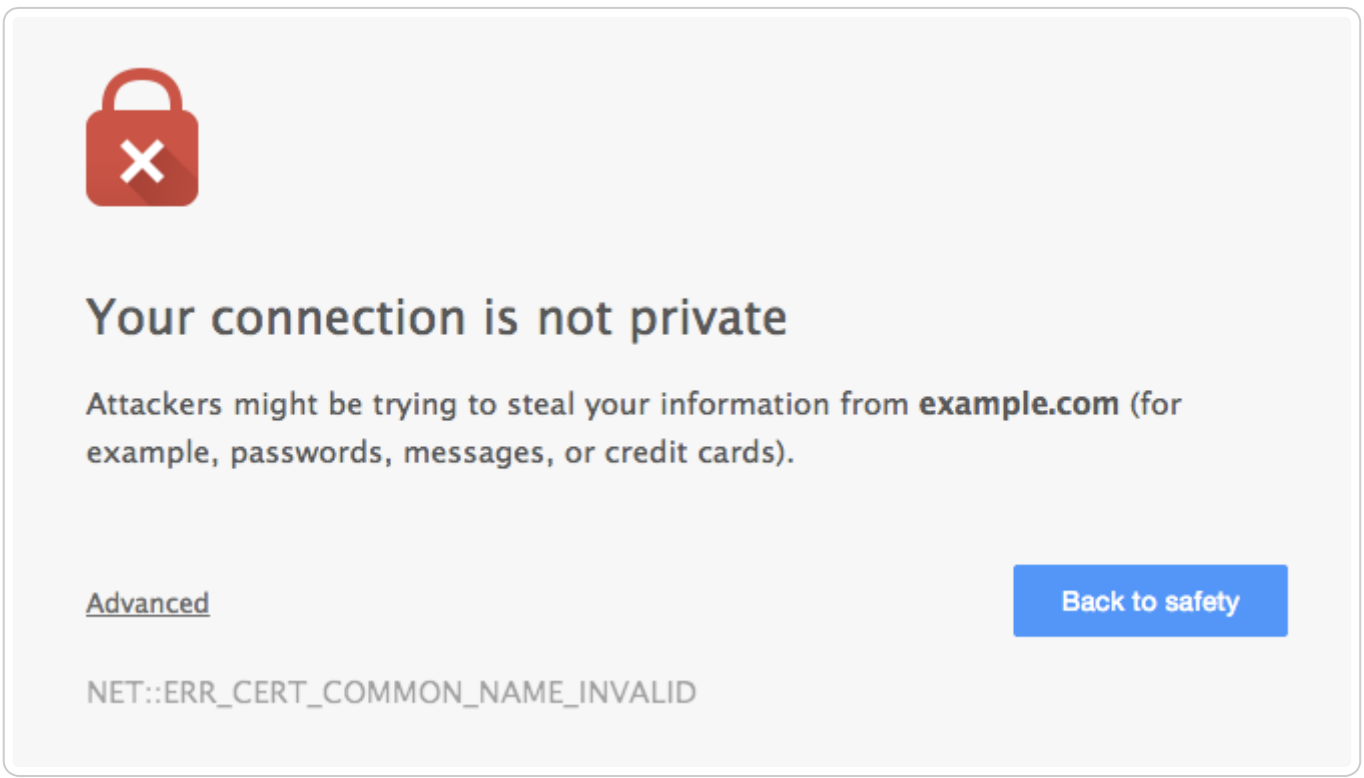
TLS certificate errors

Last updated: 2018-10-03

https://docs.fastly.com/en/guides/tls-certificate-errors

"Your connection is not private"

If you've recently started testing Fastly services, you may see errors like the following:



These errors appear because your domain has not been provisioned with TLS across the Fastly network. We offer a number TLS options that may work for you. Contact support@fastly.com to begin the provisioning process.

If you don't want to use TLS for your site, set the CNAME DNS record for your domain to point to `global-nossl.fastly.net`. This network endpoint only accepts requests over port 80, and will not expose your users to these certificate errors.

Errors when using Wget

When connecting to a Fastly service using Wget, you may see errors along the lines of

```
ERROR: Certificate verification error for mysite.example.com: unable to get local issuer certificate
ERROR: certificate common name `*.a.ssl.fastly.net' doesn't match requested host name `mysite.example.com'.
To connect to mysite.example.com insecurely, use `--no-check-certificate'.
Unable to establish TLS connection.
```

Checking with a browser or curl will show that there really is no problem, however. The errors appear because a previous version of Wget (wget-1.12-2.fc13) that shipped with some versions of Red Hat Enterprise Linux (RHEL) was buggy and failed to check Subject Alternative Names (SAN) properly.

Upgrading Wget will correct this problem and eliminate the errors. For more information you can read this Red Hat bug report or this Debian one.

TLS origin configuration messages

Last updated: 2020-06-29

https://docs.fastly.com/en/guides/tls-origin-configuration-messages

When you are connecting to origins over TLS, you may have errors.

Hostname mismatches

- `Error: Hostname mismatch`

Why the error appears

Your origin server is serving a TLS certificate with a Common Name (CN) or list of Subject Alternate Names (SAN) that does not match the origin host or the origin's SSL hostname setting.

How to fix it

You can fix this by telling Fastly what to match against in the CN or SAN field in your origin's certificate.

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the pencil icon to edit the affected host. The Edit this host page appears.
6. In the **Certificate Hostname** field, enter the hostname associated with your TLS certificate. This value is matched against the certificate common name (CN) or a subject alternate name (SAN) depending on the certificate you were issued. For example, if your certificate's CN field is `www.example.com`, enter that value for your hostname.
7. Click the **Update** button.
8. Click the **Activate** button to deploy your configuration changes.

When using custom VCL, you can specify the hostname to match against the certificate by using the `.ssl_cert_hostname` field of your origin's definition. For example: `.ssl_cert_hostname = www.example.com;`.

Certificate chain mismatches

- `Error: unable to verify the first certificate`
- `Error: self signed certificate`
- `Error: unable to get local issuer certificate`
- `Error: self signed certificate in certificate chain`
- `Error: unable to get issuer certificate`

Why the errors appear

Your origin server is serving a certificate chain that can not be validated using any of the certification authorities (CAs) that Fastly knows. This can happen for two reasons:

- Your certificate is self-signed or self-issued and you did not provide your generated CA certificate to Fastly for us to use for verification.
- Your certificate is issued by a CA that isn't in Fastly's CA certificates bundle.

How to fix them

In both cases, you can fix your configuration by adding the CA certificate that Fastly should use to verify the certificate to your service configuration:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the pencil icon to edit the affected host. The Edit this host page appears.
6. In the **TLS CA certificate** field, copy and paste a PEM-formatted CA certificate.

7. Click the **Update** button.

8. Click the **Activate** button to deploy your configuration changes.

If you are using custom VCL, you can specify the CA for Fastly to use by setting the `.ssl_ca_cert` backend parameter to a PEM encoded CA certificate.

Alternatively, you can get a new certificate issued by a CA in Fastly's CA certificate bundle (e.g., Globalsign).

Connection failures

- `Error: Gethostbyname`
- `Error: Connection timeout`
- `Error: Connection refused`

Why each error appears and how to fix it

For `Gethostbyname` failures, the configured backend Host domain is returning NXDOMAIN. Double check that the DNS settings for your backend are correct.

For `Connection time out` failures, the connection to your server is timing out. Double check that your backend is accessible and responding in a timely fashion.

For `Connection refused` failures, the connection to your server is being refused, potentially by a firewall or network ACL. Double check that you have allowlisted the Fastly IP addresses and that your backend is accessible from our network.

Certificate expirations

`Error: Certificate has expired`

Why the error appears

The certificate your backend server is presenting Fastly has expired and needs to be reissued with an updated validity period.

How to fix it

If this is a self-signed certificate you can perform this update on your own by issuing a new CSR with your private key, creating the corresponding certificate, and installing it on the server.

If this is a CA signed certificate you will need to issue a new CSR with your private key, submit it to your CA, and install the signed certificate they provide you.

SSL and old TLS protocol errors

- `Error: Unknown protocol`
- `Error: SSL handshake failure`
- `Error: TLSv1 alert internal error`

Why the errors appear

Either your origin server is not configured to use TLS or it only supports older, outdated versions of the protocol. We do not support SSLv2 or SSLv3.

How to fix them

If the origin server is configured to use TLS, use the following information to troubleshoot the problem:

- Make sure your server software is up to date and running a recent version of the TLS libraries for your platform or operating system. You may have to explicitly enable a newer protocol version. Fastly supports TLS 1.3, 1.2, 1.1, and 1.0.
- Confirm that you can connect to your origin. For example, if you're using TLS 1.3, enter a command like:

```
$ echo Q | openssl s_client -connect ${IP}:443 -tls1_3
```

To test other versions of TLS, you can replace `-tls1_3` with `tls1_2`, `-tls1_1`, or `-tls1_0`. If the TLS handshake is successful, you should see output showing the certificate, the subject, the issuer, and additional diagnostic information.

- Use `sslscan` to list the TLS protocols and ciphers supported by the TLS server.

If the origin server is not configured to use TLS, change your service configuration to disable TLS and communicate with it on port 80 instead of port 443:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the pencil icon to edit the affected host. The Edit this host page appears.
6. From the **Connect to backend using TLS** menu, select **No**.
7. Click the **Update** button.
8. Click the **Activate** button to deploy your configuration changes.

RC4 cipher error

- `Error: Using RC4 Cipher`

Why the error appears

When Fastly connects to your origin server using TLS, the only cipher suite your server supports for establishing a connection is the RC4 cipher. This cipher is considered to be unsafe for general use and should be deprecated.

How to fix it

You can fix this on your origin by using the latest version of both the server and the TLS library (e.g., OpenSSL) and ensuring the cipher suites offered are tuned to best practices. You may need to explicitly blocklist the RC4 cipher.

---

§

Because Fastly doesn't store customer logs, we provide information about debugging techniques that help you gain insights into your service configurations.

https://docs.fastly.com/en/guides/diagnostics#_debugging-techniques

Changing connection timeouts to your origin

Last updated: 2021-08-18

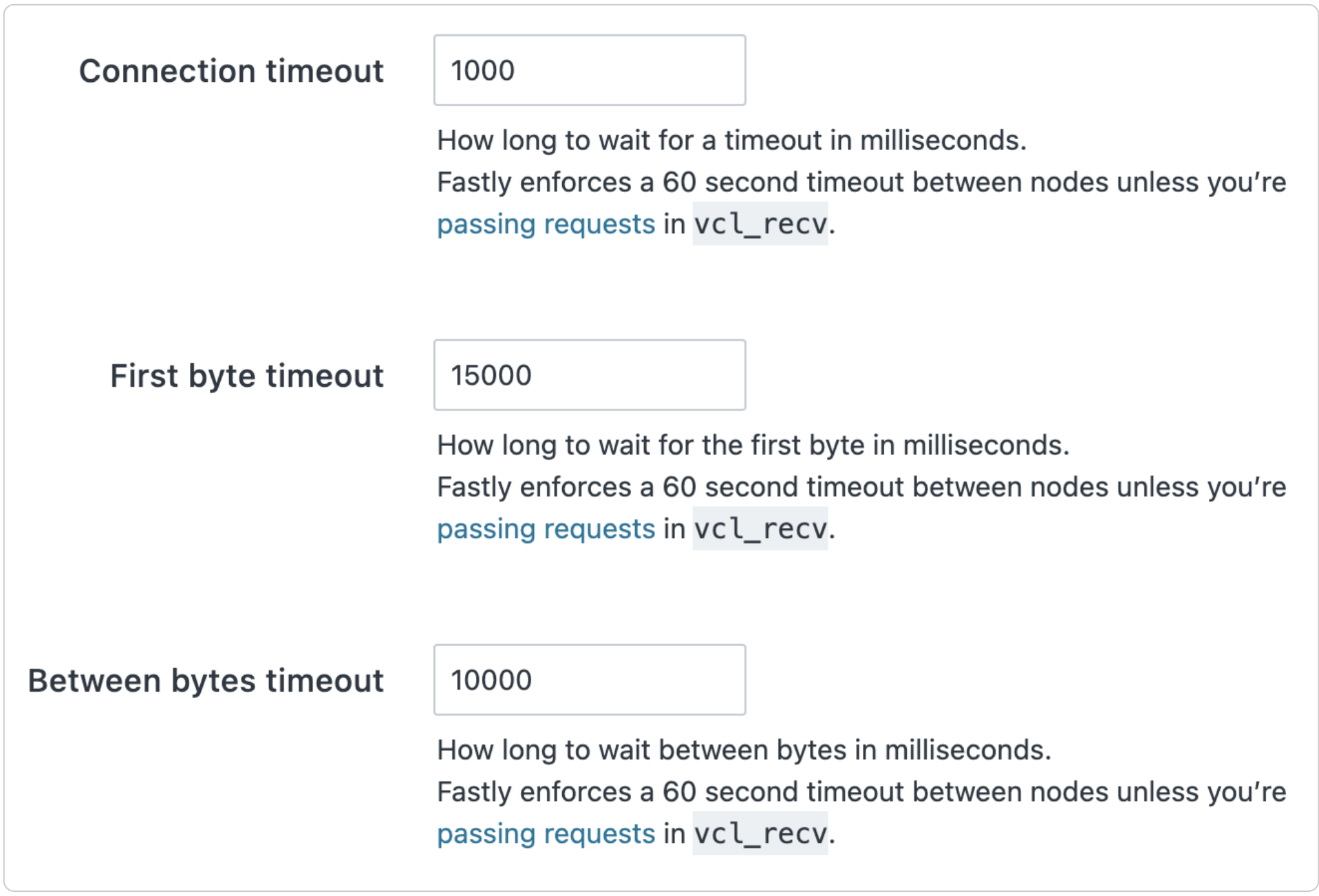
https://docs.fastly.com/en/guides/changing-connection-timeouts-to-your-origin

Connection timeouts to your origin server control how long Fastly will wait for a response from your origin server before exiting with an error. Changing the connection timeout is a good way to start troubleshooting 503 backend read errors. Follow the steps below to change the connection timeouts to your origin server:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.

5. Click the link of the Host that you want to edit. The Edit this host page appears.

6. Click the **Advanced options** link.



7. Type the new timeout in the appropriate field of the **Timeouts** section.

8. Click the **Update** button.

> **TIP**
>
> Additional techniques that help you gain insights into your service configurations can be found in our debugging guides.

Checking cache

Last updated: 2021-10-25

https://docs.fastly.com/en/guides/checking-cache

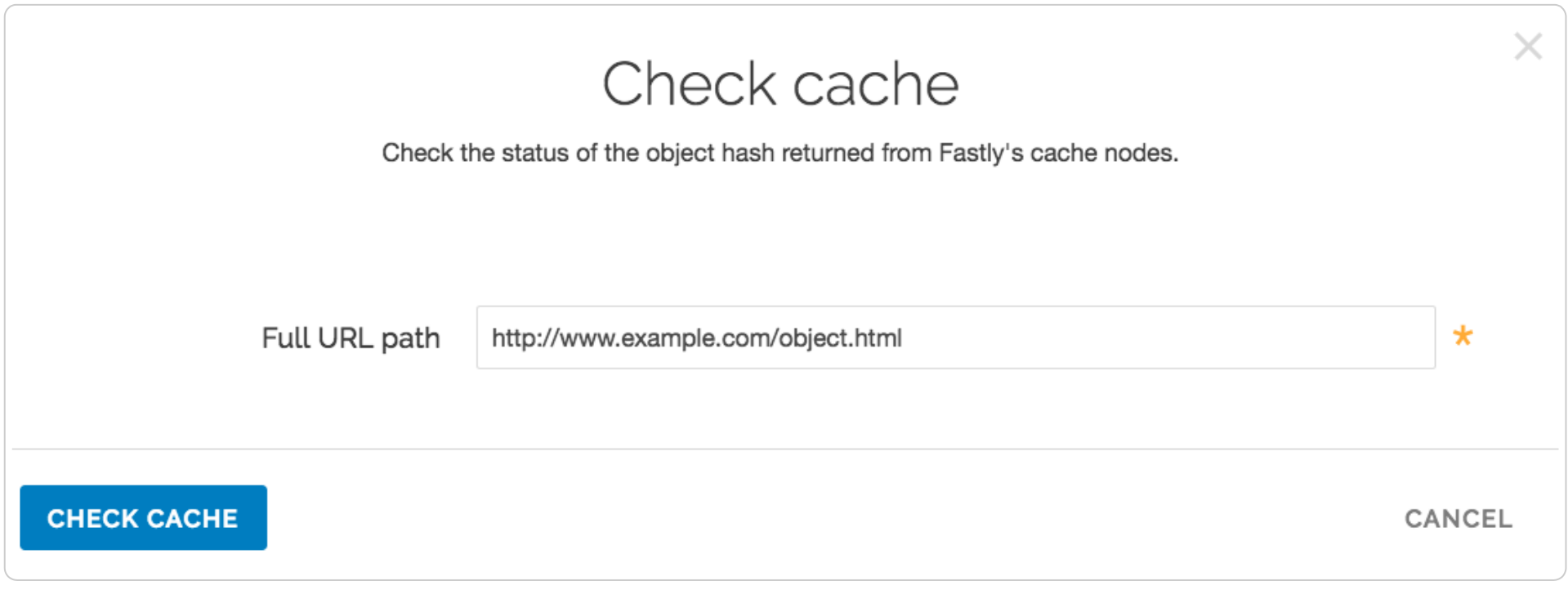
Checking the cache status of an object on your website can help when troubleshooting problems. You can use the web interface or the curl command (an open-source command line tool for transferring data with URL syntax from or to a server using one of many supported protocols) to check Fastly's cache nodes for a cached object, and you can use the information provided to examine the objects's status, response time, and content hash.

Using the web interface

Follow the steps below to check the cache status of an object using the Fastly web interface:

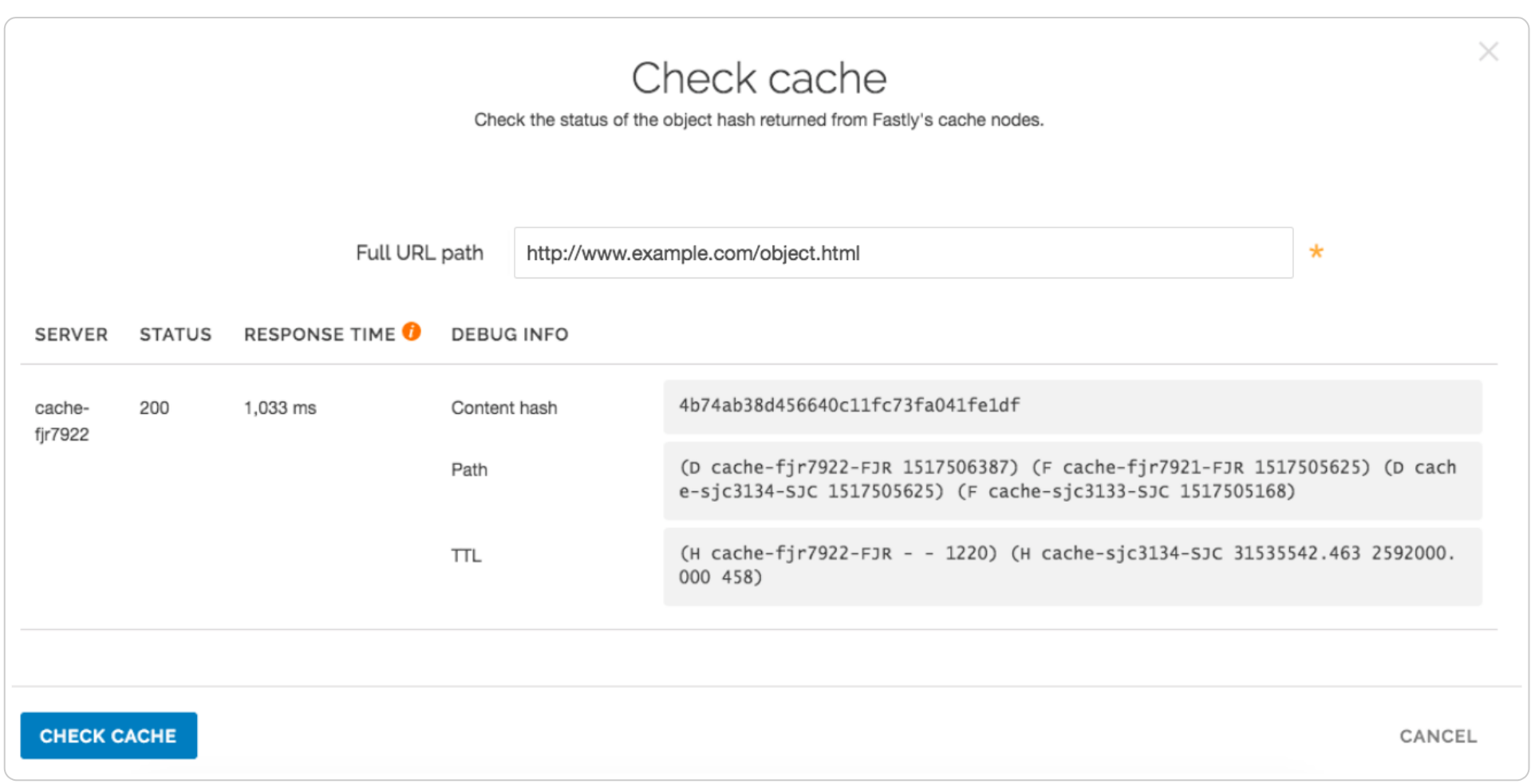
1. Log in to the Fastly web interface and click the **Deliver** link.

2. Click the **Check Cache** button. The Check Cache window appears.



3. In the **Full URL path** field, enter the full path to the object (e.g., `http://www.example.com/object.html`).

4. Click the **Check Cache** button. The results are displayed in the Check Cache window.



You can use this information to verify that the same copy of an object is stored on all of our servers. If the content hash is different across nodes, that usually indicates that there's a caching problem.

Using curl

The easiest way to tell if your request is caching in the Fastly network is to use the check cache feature in the Fastly web interface, but if you prefer command line utilities, you can also use one of two curl commands for debugging purposes:

- a simple curl command that displays the request and response headers for a given object
- a slightly more complex curl command that uses the `Fastly-Debug` header to expose information normally stripped by the simple curl

Using the simple curl command

The following curl command displays the request and response headers for a given object:

```
$ curl -svo /dev/null www.example.com/index.html
```

where `www.example.com/index.html` is replaced with the full object path of the object you're testing.

For example, using `curl -svo /dev/null www.example.com` produces something like the following section of output:

```
[...]

< Age: 142
< X-Served-By: cache-jfk1041-JFK, cache-ord1720-ORD
< X-Cache: HIT, HIT
< X-Cache-Hits: 1, 7

[...]
```

This output tells us the current age of the object in cache. It also shows shielding is enabled because two cache nodes display in `X-Served-By`. However, we're most interested in the output of the `X-Cache` header. A properly caching object displays a value of `X-Cache: HIT`, `X-Cache: HIT, HIT`, `X-Cache: HIT, MISS`, or `X-Cache: MISS, HIT`.

Using a Fastly-Debug header with curl

The `Fastly-Debug` header provides additional information for debugging by exposing specific information that is normally stripped when using a simple curl command:

```
$ curl -svo /dev/null -H "Fastly-Debug:1" www.example.com/index.html
```

where `www.example.com/index.html` is replaced with the full object path of the object you're testing.

For example, with optional shielding being used and a TTL set to 86400 (24 hours) using `Surrogate-Control`, the command `curl -svo /dev/null -H "Fastly-Debug:1" www.example.com` produces something like the following section of output:

```


[...]

< Surrogate-Control: max-age=86400
< Surrogate-Key: articles articles/1 articles/2

[...]
< Age: 403
< Fastly-Debug-Path: (D cache-ord1722-ORD 1470672957) (F cache-ord1743-ORD 1470672629) (D cache-jfk1041-JFK 1470672629)
(F cache-jfk1030-JFK 1470672554)
< Fastly-Debug-TTL: (H cache-ord1722-ORD 85997.246 0.000 403) (H cache-jfk1041-JFK - - 75)
< X-Served-By: cache-jfk1041-JFK, cache-ord1722-ORD
< X-Cache: HIT, HIT
< X-Cache-Hits: 1, 6

[...]


```

Because surrogate keys are present, the `Fastly-Debug` header exposes them. As with the simple curl command, this section of output tells us the current age of the object in cache. In addition, `Fastly-Debug` exposes specific header details to help with debugging as noted below.

Information exposed by the Fastly-Debug header

`Fastly-Debug Path` contains information about which cache server handles fetching and delivery of an object. The edge POP appears first in the sequence and the shield POP appears second.

- `D` represents which cache by name in the edge or shield ran `vcl_deliver`
- `F` represents which cache by name in the edge or shield ran `vcl_fetch`
- the number following each specific server name is a timestamp in seconds

With shielding enabled, you should generally see four cache servers listed in this header. In rare cases where a cache server exists as both an edge and a shield within the cluster for that object, you may see two or three caches listed.

`Fastly-Debug-TTL` provides information on HIT and MISS timings.

- `H` represents a HIT, meaning the object was found in the cache
- `M` represents a MISS, meaning the object was not cached at the time of the query

For each of these timings:

- the first number specifies the TTL remaining for the object
- the second number specifies the grace period
- the third number specifies the current age of the object in cache

It may take a few requests to see these numbers populate as expected because they need to either hit the cluster node or a node where the content already exists in temporary memory.

`X-Served-By` indicates the shield and edge servers that were queried for the request. The shield POP appears first in the sequence and the edge POP appears second.

`X-Cache` indicates whether the request was a HIT or a MISS for the data center.

> **NOTE**
>
> Our shielding debugging documentation provides in depth details key to understanding the `X-Served-By`, `X-Cache`, and `X-Cache-Hits` headers with shielded services.

Debugging with mtr

Last updated: 2021-04-20

https://docs.fastly.com/en/guides/debugging-with-mtr

For diagnostics and debugging in the Fastly network, we think the mtr tool offers a great way to test network speed, evaluate performance, and perform connection diagnostics. The mtr tool combines traceroute and ping programs in a single network diagnostic tool. The program's source and installation instructions live in GitHub.

While mtr provides a number of practical uses for network engineering needs, the following command works well:

```
$ mtr -c 20 -w -r www.example.com
```

Be sure to replace `www.example.com` with the hostname of the domain you're working with. The command will generate the network hops to the destination you specify, any packet loss experienced, and aggregate connection statistics.

For example, if we wanted to test the network connection from Fastly's San Francisco office to the CDN, we would use the above command for `www.fastly.com`. The following output would appear:

```
$ mtr -c 20 -w -r www.fastly.com
Start: Mon Feb  2 15:27:20 2015
HOST: test-local-machine.local              Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- 10.100.20.2                          0.0%    20    2.1   2.2   1.6   4.3   0.5
  2.|-- ge-4-3-4.mpr4.sfo7.us.zip.zayo.com    0.0%    20    2.3   2.4   1.8   5.2   0.6
  3.|-- ae5.cr2.sjc2.us.zip.zayo.com          0.0%    20    4.6   6.5   2.9  35.3   7.7
  4.|-- ae10.mpr4.sjc7.us.zip.zayo.com        0.0%    20    4.7   4.8   3.6  14.5   2.3
  5.|-- be6461.ccr21.sjc03.atlas.cogentco.com 5.0%    20    5.1   5.9   4.2  15.3   2.6
  6.|-- fastly-inc.edge2.sanjose3.level3.net  0.0%    20    5.0   4.7   4.2   8.2   0.8
  7.|-- ???                                  100.0    20    0.0   0.0   0.0   0.0   0.0
  8.|-- 23.235.47.184                         0.0%    20    4.7  14.3   3.8  74.6  20.3
```

Debugging with WebPageTest

Last updated: 2017-10-18

https://docs.fastly.com/en/guides/debugging-with-webpagetest

It's important to establish good habits of testing and performance before, during, and after migrating to Fastly. This allows you to clearly measure the impact of tests and changes to your infrastructure for accurate and informed debugging.

One tool that Fastly recommends for this purpose is WebPageTest.org. WebPageTest provides a free and open source testing tool for deep performance analysis. It is built on browser technology to accurately replicate what your end users encounter when visiting a website.

We recommend using the WebPageTest defaults for basic testing, but keep a few rules in mind:

- On the Test Settings tab under Advanced Settings, Connection should always be set to Native Connection during initial benchmarks.
- Two to three test runs may be required before a site is properly caching in Fastly.
- Using WebPageTest's Visual Comparison feature offers an ideal way to A/B test potential changes.

Fastly's network status

Last updated: 2020-12-07

https://docs.fastly.com/en/guides/fastlys-network-status

Fastly continuously monitors the status of our global network and all related services. In the event of a service interruption, an update will be posted on the Fastly status page at status.fastly.com. If you are experiencing problems and do not see a notice posted, email support@fastly.com for assistance.

Overall system status

The current system status appears at the top of the Fastly status page and includes the last time the status was refreshed so that you know how current the information is.

Individual component statuses

The status of each Fastly point of presence (POP), grouped by region, appears immediately below the overall status. Click the **+** icon next to the region's name to see the status of all POPs in the region. You can also see the status of edge cloud services like the Fastly API and web interface as well as Fastly statistics collection and delivery systems. Click the **+** icon next to each section to see all statuses.

Scheduled maintenance

Information regarding upcoming maintenance to Fastly POPs appears below the individual component statuses. In addition to the impacted POP, you can see a description of the planned maintenance and the date and time (UTC) it will occur.

Past incident statuses

Fastly keeps track of past incidents. Past incidents, if any, for approximately the past two weeks appear at the bottom of the status page.

In addition to the textual description, each incident status appears in a color that indicates the level of service impact. The color indicators are as follows:

- Green - operational
- Blue - scheduled maintenance
- Yellow - minor performance degradation or traffic rerouting
- Orange - partial outage
- Red - major outage

We also keep track of all past incidents in an incident history page. Click **Filter Components** to filter the incident history by a specific POP, edge cloud service, or Fastly system.

Subscribing to notifications

Fastly provides multiple options for subscribing to status notifications. Click the **Subscribe to Updates** button in the upper right corner of the status page screen and then select the icon for the method of notification. Once subscribed, we'll notify you any time we create or update an incident.

Unsubscribing from notifications

If you subscribe via email notifications, you can unsubscribe at any time by clicking the unsubscribe link that appears at the bottom of every status email.

If you subscribe via SMS text messaging, you can unsubscribe at any time by replying STOP to any status message you receive.

Maintenance events that aren't included on the status page

We're always making improvements to our network, our data centers, and the software stack that powers our platform. Fastly uses modern application development processes that are agile and we also use continuous integration and deployment (CI/CD) methods, which means we update the Fastly cache fleet almost daily. This allows us to release new code, features, and updates more frequently.

We strive to perform maintenance and updates in a way that minimizes the impact on our customers' services. For redundancy, we use a backup cache, which is another member of the cluster in a data center. We apply updates to POPs in off-peak hours and do rigorous testing and quality control.

Google Pagespeed module errors

Last updated: 2017-10-18

https://docs.fastly.com/en/guides/google-pagespeed-module-errors

If you are using the Google Pagespeed module and notice constant MISSes for HTML pages, check the Cache-Control settings in the module's .htaccess file.

By default, Google Pagespeed serves all HTML with `Cache-Control: no-cache, max-age=0`. This setting conflicts with Fastly's default configuration. If your origin sends the headers `Cache-Control: private` or `Cache-Control: max-age=0`, Fastly will pass requests straight to the origin.

To change the Google Pagespeed directive and leave the original HTML caching headers, update your origin's .htaccess file with:

```
ModPagespeedModifyCachingHeaders off
```

More details about the Pagespeed Module can be found within Google Developers directory. For additional information about controlling how long Fastly caches your resources, start with our documentation on cache freshness.

Googlebot crawl stats

Last updated: 2018-10-19

https://docs.fastly.com/en/guides/googlebot-crawl-stats

Any time you notice any major changes in your SEO stats, indexing, or crawler behavior, start troubleshooting by asking these questions:

- Did you read the Google FAQs for indexing, crawling, and ranking?
- Is your robots.txt file still accessible and were there any changes to it?
- Is your sitemap testing without errors?
- Did you adjust your Googlebot crawl rate?
- Have you had Google's URL Inspection Tool to check for errors and request reindexing the URLs?

We recommend exploring Google's Webmaster Tools if you're experiencing issues. Their article about the Fetch as Google feature and their article on troubleshooting sitemap errors offer specific help for debugging Googlebot crawl stats in this situation. Google also includes an entire section in their tools documentation on getting additional support if you're experiencing trouble.

TIP

Our debugging articles contain a variety of troubleshooting tips.

Temporarily disabling caching
Last updated: 2018-08-09
https://docs.fastly.com/en/guides/temporarily-disabling-caching

Caching can be disabled:

- at the individual URL level,
- at the browser level, and
- at the site level.

Disabling caching at the individual URL level

To disable caching at the individual URL level:

1. Create a request setting that always forces a pass.
2. Add a condition to the request setting that looks for specific URLs.
3. Activate the new version of your service to enable the setting.

Disabling caching at the browser level

Theoretically, all browsers should follow the stated rules of the HTTP standard. In practice, however, some browsers don't strictly follow these rules. The following combination of headers seems to force absolutely no caching with every browser we've tested.

```
Cache-Control: no-cache, no-store, private, must-revalidate, max-age=0, max-stale=0, post-check=0, pre-check=0
Pragma: no-cache
Expires: 0
```

In addition, IE8 has some odd behavior to do with the back button. Adding `Vary: *` to the headers seems to fix the problem.

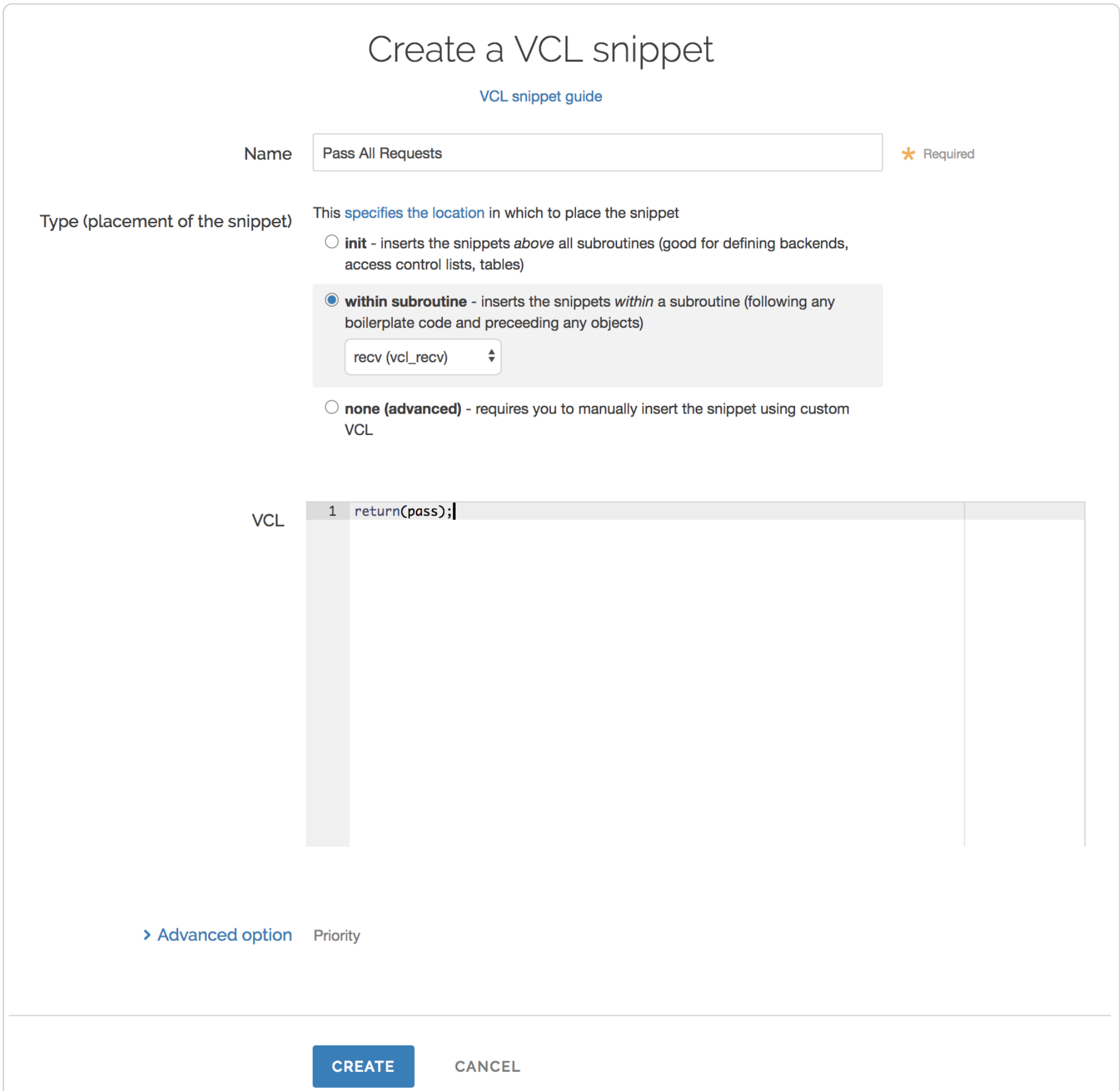
IMPORTANT

If you want your content cached in Fastly but not cached on the browser, you must not add these headers on your origin server. Instead, add these as new Headers on the Content page and be sure the Type is set to Response.

Disabling caching at the site level

You can disable caching at the site level by creating a VCL Snippet to pass on all requests to your service:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click **Create Snippet**. The Create a VCL snippet page appears.



6. In the **Name** field, enter an appropriate name (e.g., `Pass All Requests`).
7. From the **Type** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **recv (vcl_recv)**.
9. In the **VCL** field, add the following condition:

```
return(pass);
```

10. Click **Create** to create the snippet.
11. Click the **Activate** button to deploy your configuration changes.

All requests will continue to be passed until you remove return(pass); from vcl_recv in your VCL or you delete this snippet.

Using GET instead of HEAD for command line caching tests

Last updated: 2018-08-15

https://docs.fastly.com/en/guides/using-get-instead-of-head-for-command-line-caching-tests

If you're testing on the command line to determine an object's caching status, then use GET instead of HEAD. For example:

```
$ curl -svo /dev/null www.example.com
```

Default caching behavior of HTTP verbs

By default, the results of GET requests are cached. HEAD requests are not proxied as is, but are handled locally if an object is in cache or a GET is done to the backend to get the object into the cache. Anything other than HEAD or GET requests are proxied and not cached by default.

---

§

These articles describe how we support real-time log streaming of data that passes through Fastly.

https://docs.fastly.com/en/guides/diagnostics#_streaming-logs

About Fastly's real-time log streaming features

Last updated: 2021-03-17

https://docs.fastly.com/en/guides/about-fastlys-realtime-log-streaming-features

To help you tune the performance of your Fastly services, we support real-time log streaming of data that passes through Fastly. We support a number of protocols that allow you to stream logs to a variety of locations, including third-party services, for storage and analysis.

Supported protocols and logging providers

Fastly supports a variety of syslog-compatible logging providers, such as Sumo Logic, Papertrail, and Logentries. In addition, we provide a syslog endpoint specifically for sending log files to other syslog-based software (for example, to Logstash, part of the ELK stack, which supports input via syslog).

We also support other methods of sending logs besides the syslog protocol. We allow pushing of log files to Amazon S3 buckets as well as any S3-compatible providers (such as DreamHost's DreamObjects). And we support FTP uploading.

As part of our third-party integrations, Fastly offers a number of endpoints to which you can stream logs. If the logging endpoint you're looking for isn't here, contact support@fastly.com for suggestions on another endpoint that might provide the same functionality.

Supported log streaming features

Fastly's real-time log streaming supports the following specific features:

- **TLS support.** Fastly allows logging configuration information to be sent over TLS (Transport Layer Security) for certain endpoints. This means that logging information can be encrypted while in transit, which allows you to send potentially sensitive information to log files without exposing data.
- **Encryption.** Fastly allows you to encrypt log files for certain endpoints before they are written to disk. We encrypt files using OpenPGP (Pretty Good Privacy). For our Amazon S3 endpoint in particular, we also support server-side encryption.
- **Customized log formats.** Fastly allows you to change the format of your logs by providing variables compatible with the Apache Common Log Format (NCSA Common log format).
- **Log file locations.** Fastly provides two different ways for you to change where your log files are written for certain endpoints. You can change a log file's timestamp format (for example, if you wanted to remove characters from the log file name) and you can control the specific path to which those files are written.
- **Multiple endpoints.** Fastly allows you to send logs to multiple endpoints.
- **Allowlisting.** Fastly's publicly available list of IP ranges allows you to enable Fastly-only access to your logging servers through your firewall.

How real-time log streaming works

Varnish sends all streaming log records to a log aggregator, which streams them in near-real-time to the logging endpoint you configure.

Changing log compression formats
Last updated: 2021-02-03
https://docs.fastly.com/en/guides/changing-log-compression-options

Fastly's Real-Time Log Streaming feature allows you to specify compression format and options for file-based logging endpoints. These include the Azure Blob, FTP, Google Cloud Storage, Kafka, OpenStack, Amazon S3, SFTP, Digital Ocean, and Cloud Files logging endpoints.

Available log compression formats

Although the default is to use no compression, we allow you to choose one of several compression mechanisms:

- Zstandard, a compression algorithm defined by RFC 8478
- Snappy, a compression and decompression library used by many Google products as referenced in the Snappy compressed format description
- Gzip, a compression utility as defined in RFC 1952 and RFC 1951

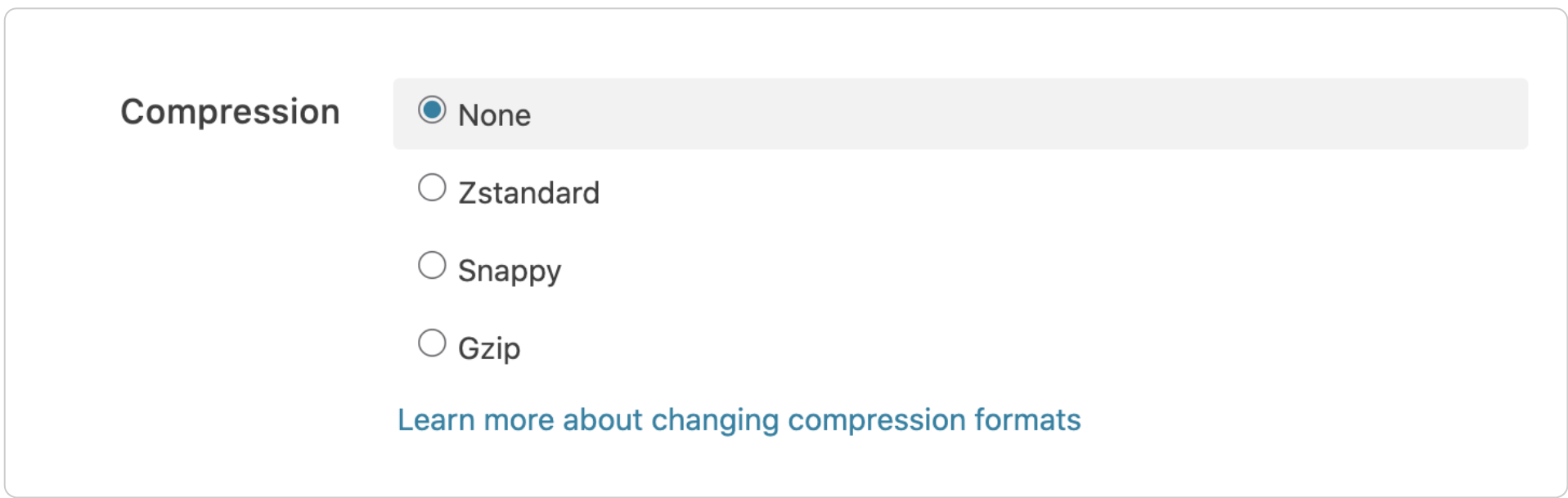
Using the web interface to change log compression formats

The web interface lets you select a compression codec to apply to log files.

Keep in mind that if you're using Gzip compression the web interface defaults to a Gzip compression level of 3, and can only be changed using the Logging API. If the Gzip compression level has been set to a value other than `3` via an API call, then that level is displayed as a read-only value.

Follow these instructions to update a file-based logging endpoint's compression format using the web interface:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears.
5. Click the name of a file-based logging endpoint you want to edit. The Edit this endpoint page appears.
6. Click the **Advanced options** link near the bottom of the page. The Advanced options appear.
7. In the **Compression** section, select a compression format for the logging endpoint.



8. Click the **Update** button.

9. Click the **Activate** button to deploy your configuration changes.

Using the API to change log compression formats

You can use the Logging API for any file-based logging endpoint to update the compression format.

The API provides two fields for specifying log file compression options: `compression_codec` and `gzip_level`.

- `compression_codec` is a string that specifies the codec used to compress your logs. One of the following:
  - `zstd` Use Zstandard compression.
  - `snappy` Use Snappy compression.
  - `gzip` Use Gzip. Defaults to gzip compression level 3.
- `gzip_level` is an integer in the range of 0, signifying no compression, to 9, signifying maximum compression.

> **NOTE**
>
> Use of `compression_codec` is recommended over use of `gzip_level` unless you need to set the gzip compression level to a value other than 3.

Keep in mind the following with respect to the request payload:

- Only one of `compression_codec` or `gzip_level` should be included in a request payload. Specifying both will result in an error.
- If `compression_codec` is set to "gzip", `gzip_level` will default to 3.
- If `compression_codec` is not specified in the request payload and a non-zero value for `gzip_level` is set, then `compression_codec` will default to "gzip".
- If neither `compression_codec` nor `gzip_level` is specified in the request, no compression is applied to log files and `compression_codec` will default to `null`.

For example, to update the compression format for a service's SFTP logging endpoint to use the Snappy compression codec, the curl command would look something like this:

```
$ curl -X PUT -H "Fastly-Key: YOUR_FASTLY_TOKEN"  -H "Accept: application/json" -d "compression_codec=snappy" https://api.fastly.com/service/SU1Z0isxPaozGVKXdv0eY/version/5/logging/sftp/sftp-log
```

To apply gzip compression to an Azure Blob Storage logging endpoint using the `compression_codec` field, the curl command would look like this:

```
$ curl -X PUT -H "Fastly-Key: YOUR_FASTLY_TOKEN"  -H "Accept: application/json" -d "compression_codec=gzip" https://api.fastly.com/service/SU1Z0isxPaozGVKXdv0eY/version/7/logging/azureblob/azure-log
```

To update a Google Cloud Storage logging endpoint's compression format to 1 using the `gzip_level` field, the curl command would look like this:

```
$ curl -X PUT -H "Fastly-Key: YOUR_FASTLY_TOKEN"  -H "Accept: application/json" -d "gzip_level=1" https://api.fastly.com/service/SU1Z0isxPaozGVKXdv0eY/version/4/logging/gcs/gcs-log
```

Changing log line formats

Last updated: 2021-10-25

https://docs.fastly.com/en/guides/changing-log-line-formats

Fastly's Real-Time Log Streaming feature allows you to change the format that your log messages are delivered in on select logging endpoints. We allow you to choose one of several formats:

- **Blank** is the default. There's no prefix — just your log message. This is useful when writing to JSON and CSV files.
- **Classic** is a standard syslog prefix format as defined by RFC 3164.
- **Loggly** is a structured syslog prefix format based on RFC 5424.

- **Logplex** is a Heroku-style prefixed syslog format.

Updating endpoints to use a different format

A number of logging endpoints can be updated to use a message format other than the default via either the web interface or the API.

Using the web interface

Follow these instructions to update a logging endpoint using the web interface:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears.
5. Click the name of a logging endpoint you want to edit. The Edit this endpoint page appears.
6. Click the **Advanced options** link near the bottom of the page. The Advanced options appear.
7. In the **Select a log line format** section, select a log line format for the logging endpoint.
8. Click the **Update** button.
9. Click the **Activate** button to deploy your configuration changes.

Using the API

Run the following command to update a logging endpoint using the API:

```
$ curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/<your Fastly service ID>/version/<version number>/logging/<logging endpoint>/<log name>' --data-binary '{"message_type":"<type>"}'
```

Keep in mind that the `message_type` field is a per-object field. Updating it on one logging object *will not* change it on any other objects.

For example, to update a Google Cloud Storage logging endpoint named `GCS Test` to use the `blank` message type, the curl command would look something like this:

```
$ curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/SU1Z0isxPaozGVKXdv0eY/version/1/logging/gcs/GCS%20Test' --data-binary '{"message_type":"blank"}'
```

NOTE

The `log name` included a space that needed to be URL encoded (the space into `%20`).

An example

If you select **classic** format, we send log messages out in standard syslog format. The prefix for this format (as defined in RFC 3164) appears as follows:

```
<134>2016-07-04T22:37:26Z cache-sjc3128 LogTest[62959]: <your log message>
```

The prefix begins with the message priority (always `<134>`, which means `Facility=Local0, Severity=Informational`), followed the date and time the log was sent (`2016-07-04T22:37:26Z`), the cache node it came from (in this case, `cache-sjc3128`), the name of your log (`LogTest`) and the ID of the process sending it (`62959`).

Changing log placement

Last updated: 2021-06-15

https://docs.fastly.com/en/guides/changing-log-placement

Fastly's Real-Time Log Streaming feature allows you to specify where the logging call should be placed in the generated VCL.

Available log placement options

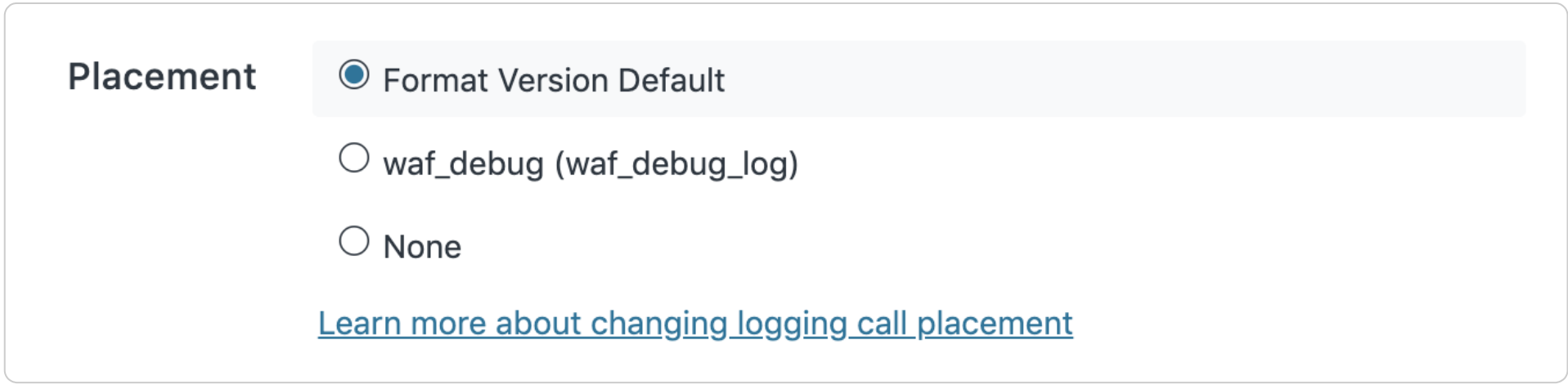
You can choose one of the following log placement options:

- **Format Version Default** puts the log statement in `vcl_log` if the logging endpoint is using version 2 log format. If the logging endpoint is using version 1 log format, puts the log statement in `vcl_deliver`.
- waf debug (waf_debug_log) puts the log statement in `waf_debug_log`, which allows for logging of WAF variables.
- **None** prevents the log statement from being rendered in VCL. Use this option if you intend to write a log statement manually in custom VCL.

Using the web interface to change log placement

Follow these instructions to update a logging endpoint's VCL placement using the web interface:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears.
5. Click the name of the logging endpoint you want to edit. The Edit this endpoint page appears.
6. In the **Placement** section, select a placement for the logging endpoint.



7. Click the **Update** button.
8. Click the **Activate** button to deploy your configuration changes.

Using the API to change log placement

You can use the API to update a logging endpoint's VCL placement. The API provides a `placement` field for specifying where in the generated VCL the logging call should be placed. It can be one of the following:

- `""` uses the default version placement when not set. Logging endpoints using version 2 log format are placed in `vcl_log`. Logging endpoints using version 1 log format are placed in `vcl_deliver`.
- `waf_debug` puts the log statement in the `waf_debug_log` subroutine, which allows for logging of WAF-specific variables.
- `none` prevents the log statement from being rendered in VCL. Use this option if you intend to write a log statement manually in custom VCL.

For example, to update the logging placement to `none`, the curl command would look like this:

```
$ curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/<your Fastly service ID>/version/<version_id>/logging/<logging_integration>/<logging_name>' --data-binary '{"placement":"none"}'
```

Changing where log files are written

Last updated: 2019-10-23

https://docs.fastly.com/en/guides/changing-where-log-files-are-written

For supported logging endpoints that write files to remote services, Fastly uses a combination of factors to ensure log files aren't overwritten, including:

- Using the file creation timestamp.
- Generating a unique ID.
- If a file with the same timestamp and UID combination exists, incrementing a counter and adding that to the end of the filename.

To change where log files are written, you can modify the `path` and `timestamp_format` variables on select endpoints. The logging system combines the `path`, `timestamp_format`, and `uid` variables to create the file name:

```
<path><timestamp>-<uid>.log<suffixes>
```

This guide explains how to use the `path` and `timestamp_format` variables to control where log files are written.

Timestamp format

You may want to consider changing the timestamp format to remove characters from the log filenames. For example, if you're working with Elastic MapReduce, you might need to remove the colons in the filename.

The `timestamp_format` variable is provided as a strftime compatible format. The default format is ISO 8601 Combined Date/Time Format:

```
%Y-%m-%dT%H:%M:%S.000
```

The variables are expanded when the file is created. For example, `%Y` will be replaced by the current year and `%m` by the current month number:

```
<year>-<2 digit month number>-<2 digit day number>T<hour>:<minute>:<second>
```

The timestamp for a file created at midnight on January 1st, 1970 would be `1970-01-01T00:00:00.000`.

Path

The `path` variable acts differently depending on whether or not it ends in a trailing `/`.

If the variable does end in a trailing `/`, then it's treated as a directory. For example, if the variable is set to `my_logs/`, the files are written in the directory `my_logs`. If the variable is set to `my_logs` without the trailing `/`, the files are written in the top-level directory and are prefixed with `my_logs`.

The two approaches can also be combined. For example, if the variable is set to `my_logs/foo`, the files are written in the `my_logs` directory and are prefixed with `foo`.

Logs can also be nested. For example, if the variable is set to `my_logs/sub_logs/`, the files are written in the `sub_logs` directory in the `my_logs` directory.

TIP

The path can also be a `strftime` compatible string. For example, if the variable is set to `%Y/%m/%d`, the files are written to a directory based on the year, month, and date.

Directories are created automatically when possible.

Suffixes

Fastly's logging system automatically adds suffixes to files as appropriate.

Suffix	File type
.log	Plain log file
.log.gz	Gzipped log file
.log.gpg	PGP encrypted log file
.log.gz.gpg	PGP encrypted, Gzipped log file

Creating an AWS IAM role for Fastly logging

Last updated: 2021-03-16

https://docs.fastly.com/en/guides/creating-an-aws-iam-role-for-fastly-logging

Before adding Amazon S3 or Amazon Kinesis as a logging endpoint for Fastly services, we recommend creating an Identity and Access Management (IAM) role in AWS specifically for Fastly. Using your Fastly customer account ID and the Fastly AWS account number, you can set up a role to give Fastly access to your S3 bucket or Kinesis data stream for log delivery using temporary credentials instead of long-term credentials like an access key and secret key pair.

You can do this through the AWS Management Console or the AWS CLI.

Creating an IAM role through the AWS Management Console

Follow the steps below to create an IAM role through the AWS Management Console.

1. Log in to the AWS Management Console and open the IAM console.
2. Create a permission policy that gives Fastly permission to write objects to AWS. Click **Create policy**. The Create policy window appears.
3. Select the JSON tab. Copy and paste one of the following templates, replacing the name in the resource field with the Amazon Resource Name (ARN) of the Amazon S3 bucket or Kinesis data stream you want Fastly to write logs to.

Amazon S3 template

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "s3:PutObject",
        "Resource": "arn:aws:s3:::YourS3BucketName/*"
    }
}
```

Amazon Kinesis template

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
    "Action": [
       "kinesis:PutRecords",
       "kinesis:ListShards"
       ],
    "Resource": "arn:aws:kinesis:::YourKinesisStreamName"
    }
}
```

4. Click **Create policy**. Your new policy appears in the list on the Policies tab.
5. Create a role with the trust and permissions policies attached. Select **Roles** from the navigation panel, and then click **Create role**. The Create role window appears.
6. For **Select type of trusted entity**, select **Another AWS account**.

7. For **Account ID**, enter the Fastly AWS account ID (`717331877981`).
8. Select **Require external ID**.
9. In the **External ID** field, enter your Fastly customer account ID.
10. Click **Next: Permissions**.
11. Select the checkbox next to the permission policy you created above.
12. Click **Next: Tags**.
13. Optionally, add metadata to the role by attaching tags as key–value pairs. For more information about using tags in IAM, see Amazon's documentation on tagging IAM resources.
14. Click **Next: Review**. Complete the following fields:
    - In the **Role name** field, enter a name for your role. Role names must be unique within your AWS account. They are not distinguished by case. Because other AWS resources might reference the role, you can't edit the name of the role after it has been created.
    - Optionally, in the **Role description** field, enter a description for the new role.
15. Review the role and then click **Create role**. The role you created appears in the list of roles on the **Roles** tab.

After you create your new role, select the role from the **Roles** tab to view details about the role, including the Role ARN. You will need this ARN to create your logging endpoint.

Creating an IAM role through the AWS CLI

Follow the steps below to create an IAM role through the AWS CLI:

1. Create a trust policy in JSON using your Fastly customer account ID and the Fastly AWS account number using one of the following templates. Copy the template to a text editor and replace `FastlyCustomerAccountID` with your customer account ID and `Sid` with the name of your policy. Save the file to your file system.

Amazon S3 template

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Condition": {
            "StringEquals": {
                "sts:ExternalId": "FastlyCustomerAccountID"
            }
        },
        "Action": "sts:AssumeRole",
        "Principal": {
            "AWS": "717331877981"
        },
        "Effect": "Allow",
        "Sid": "S3LoggingTrustPolicy"
    }
}
```

Amazon Kinesis template

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Condition": {
            "StringEquals": {
                "sts:ExternalId": "FastlyCustomerAccountID"
            }
        },
        "Action": "sts:AssumeRole",
        "Principal": {
            "AWS": "717331877981"
        },
        "Effect": "Allow",
        "Sid": "KinesisLoggingTrustPolicy"
    }
}
```

2. Create a permission policy in JSON using the Amazon Resource Name (ARN) of the S3 bucket or Kinesis data stream you want Fastly to write logs to. Copy the template to a text editor and replace the name in the resource field with the name of the S3 bucket or Kinesis data stream. Save the file to your file system.

Amazon S3 template

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "s3:PutObject",
        "Resource": "arn:aws:s3:::YourS3BucketName/*"
    }
}
```

Amazon Kinesis template

```
 {
     "Version": "2012-10-17",
     "Statement": {
         "Effect": "Allow",
         "Action": [
                    "kinesis:PutRecords",
                    "kinesis:ListShards"
                ],
          "Resource": "arn:aws:kinesis:::YourKinesisStreamName"
      }
 }
```

3. From the command line, create a role and attach the trust policy to the role. Replace `YourRoleName` with the name of your role and `file://trust-policy-file.json` with the name and location of the file in which you created your trust policy.

```
$ aws --profile personal iam create-role --role-name YourRoleName --assume-role-policy-document file://trust-policy-fil
e.json
```

Here's what the successful response to this command looks like:

```
    {
    "Role": {
        "Path": "/",
        "RoleName": "YourRoleName",
        "RoleId": "ABCD1234ZHHQGKDRUMGFH",
        "Arn": "arn:aws:iam::AmazonResourceName:role/YourRoleName",
        "CreateDate": "2021-03-19T23:14:27+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": {
                "Condition": {
                    "StringEquals": {
                        "sts:ExternalId": "abc12345-defg-6789-hijk-lmno10111213"
                    }
                },
                "Action": "sts:AssumeRole",
                "Principal": {
                    "AWS": "717331877981"
                },
                "Effect": "Allow",
                "Sid": "RoleForS3"
                }
            }
        }
    }

```

> **NOTE**
>
> Take note of the value in the `Arn` field. This is the ARN for the role, which you will need to create your logging endpoint.

4. Create the permission policy. Replace `YourPolicyName` with the name of your policy and `file://permission-policy-file.json` with the name and location of the file in which you created your trust policy.

```
$ aws --profile personal iam create-policy --policy-name YourPolicyName --policy-document file://permissions-policy-file.json
```

Here's what the successful response to this command looks like:

```
{
"Policy": {
    "PolicyName": "YourPolicyName",
    "PolicyId": "ABCDJH5Z123CTIKFWXYZ",
    "Arn": "arn:aws:iam::AmazonResourceName:policy/YourPolicyName",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2021-03-19T23:17:42+00:00",
    "UpdateDate": "2021-03-19T23:17:42+00:00"
    }
}
```

5. Attach the permissions policy to the role. Replace `YourRoleName` with the name of your role and the value after `--policy-arn` with the ARN of your permission policy.

```
$ aws --profile personal iam attach-role-policy --role-name YourRoleName --policy-arn arn:aws:iam::123453306678:policy/AllowLoggingBucketWrites
```

Custom log formats

Last updated: 2021-08-16

https://docs.fastly.com/en/guides/custom-log-formats

Fastly provides two versions of custom log formats for use when you set up remote log streaming. All new logging endpoints use the version 2 custom log format by default. You can upgrade version 1 logging endpoints to the version 2 custom log format. You can also make version 2 look like version 1 for the sake of continuity. We've described the key advantages of the version 2 custom log format below.

Version 2 log format

This table details version 2 of Fastly's custom log formats. All variables should be prefixed by a percent sign (`%`), as indicated in the table.

Format String	Description
`%%`	The percent sign.
`%a`	The client IP address of the request. Equivalent to `req.http.Fastly-Client-IP`.
`%A`	The local IP address. Equivalent to `server-ip`.
`%B`	The size of response in bytes, excluding HTTP headers. Equivalent to `resp.body_bytes_written`.
`%b`	The size of response in bytes, excluding HTTP headers. In Common Log Format (CLF), that means a `"-"` rather than a `0` when no bytes are sent.
`%{Foobar}C`	The contents of cookie `Foobar` in the request sent to the server.
`%D`	The time taken to serve the request, in microseconds. Equivalent to `time.elapsed.usec`.
`%{FOOBAR}e`	Not supported. Always returns `"-"`.
`%f`	The filename. Equivalent to `req.url` with any query string removed, because Varnish has no notion of a file being served.
`%h`	The remote IP address. Equivalent to `req.http.Fastly-Client-IP`.
`%H`	The request protocol. Equivalent to `req.proto`.

Format String	Description
`%{Foobar}i`	The contents of `Foobar:` header lines in the request sent to the server.
`%I`	Bytes received, including request and headers. Cannot be zero Equivalent to `req.bytes_read`.
`%k`	The number of keepalive requests handled on this connection. Always returns `0`.
`%l`	Not supported. Always returns `"-"`.
`%m`	The request method. Equivalent to `req.request`.
`%{Foobar}n`	Not supported. Always returns `"-"`.
`%{Foobar}o`	The contents of `Foobar:` header lines in the reply.
`%O`	Bytes sent, including headers. Cannot be zero. Equivalent to `resp.bytes_written`.
`%p`	The canonical port of the server serving the request. Always returns `80`. Equivalent to `server.port`.
`%{format}p`	The canonical port of the server serving the request. Valid formats are `canonical`, `local`, or `remote`. Returns `80` for HTTP requests and `443` for HTTPS requests.
`%P`	Not supported. Always returns `"-"`.
`%{format}P`	Not supported. Always returns `"-"`.
`%q`	The query string (prepended with a `?` if a query string exists, otherwise an empty string). Equivalent to `req.url`.
`%r`	The first line of the request.
`%R`	Not supported. Always returns `"-"`.
`%s`	The status. For requests that got internally redirected, this is the status of the *original* request. Use `%>s` for the final status. Equivalent to `resp.status`.
`%t`	The time the request was received, in Standard English format (e.g., `01/Jan/1970:00:00:00 -0700`). The last number indicates the timezone offset from GMT.
`%{format}t`	The time, in the form given by `format`, which should be in `strftime(3)` format (potentially localized). If the format starts with `begin:` (the default) the time is taken at the beginning of the request processing. If it starts with `end:` it is the time when the log entry gets written, close to the end of the request processing. In addition to the formats supported by `strftime(3)`, the following format tokens are supported: `sec` (number of seconds since the Epoch), `msec` (number of milliseconds since the Epoch), `usec` (number of microseconds since the Epoch), `msec_frac` (millisecond fraction), and `usec_frac` (microsecond fraction). Equivalent to `time.start`.
`%T`	The time taken to serve the request, in seconds. Equivalent to `time.elapsed.sec`.
`%u`	Not supported. Always returns `"-"`.
`%U`	The URL path requested, not including any query string. Equivalent to `req.url`.
`%v`	The domain name of the request. Equivalent to `req.http.host`.
`%V`	The same as `%v`. Equivalent to `req.http.host`.
`%{vcl}V`	The literal VCL to include without quoting. This can be used to write VCL variables to your logs (e.g., `%{client.geo.country_code}V` or `%{tls.client.cipher}V`). This %-directive is a Fastly extension and is not found in Apache.
`%X`	The connection status when response is completed. Always set as `+` (connection may be kept alive after the response is sent).

Version 1 log format

This table details version 1 of Fastly's custom log formats. All variables should be prefixed by a percent sign (`%`), as indicated in the table.

Format String	Description
`%b`	The content size of the response, calculated using the `Content-Length` header rather than actually checking the length of the response (and may therefore be wrong).
`%h`	The remote IP address.
`%l`	The remote log name. Always returns the hardcoded value `"-"`.
`%r`	The HTTP verb and request path (e.g., `GET /index.html`). Unlike Apache and version 2 log formats, the protocol version is not included.
`%>s`	The status of the last request.
`%t`	The time the request was received, in Unix `ctime` format (e.g., `Thu, 01 Jan 1970 00:00:00 GMT`) rather than Apache's Standard English format (e.g., `01/Jan/1970:00:00:00 -0700`).
`%u`	Always returns `"-"`.

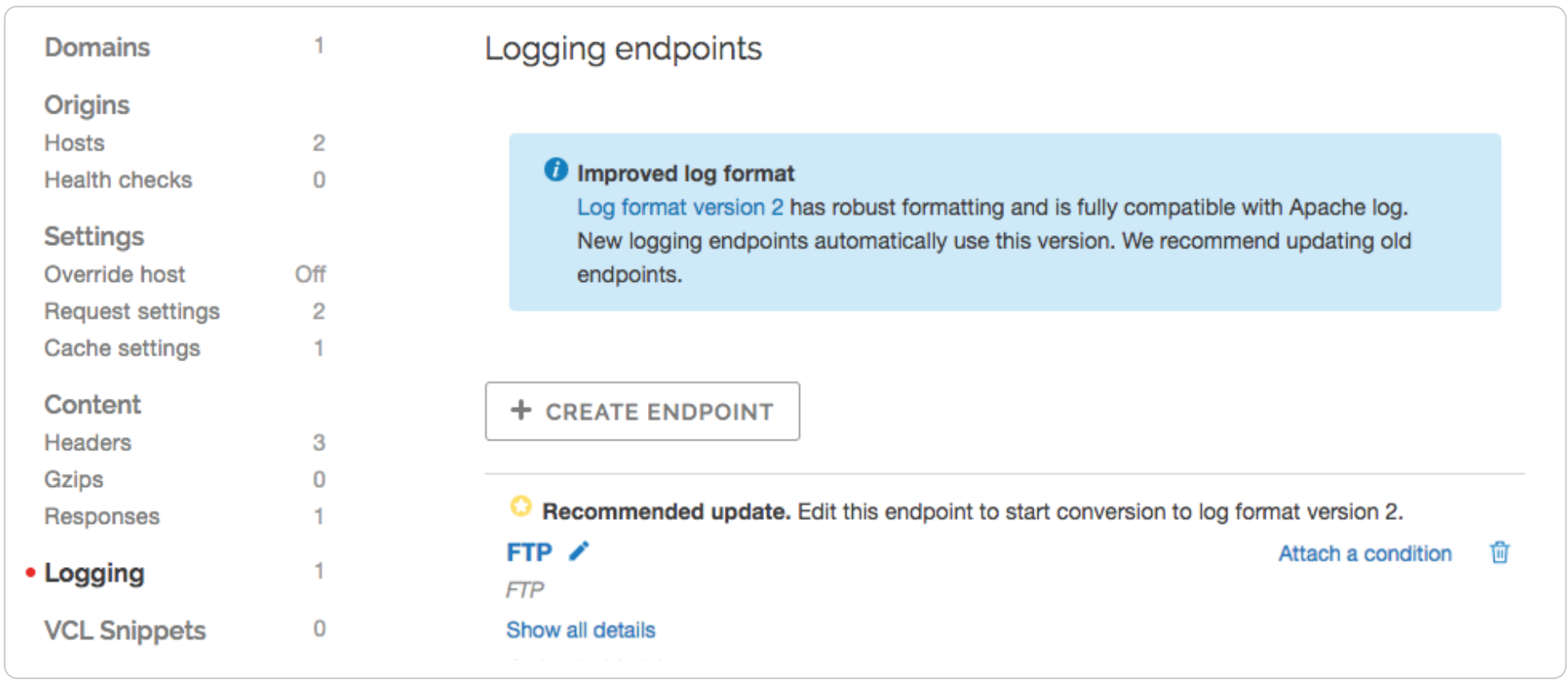
Upgrading endpoints to use version 2 log format

WARNING

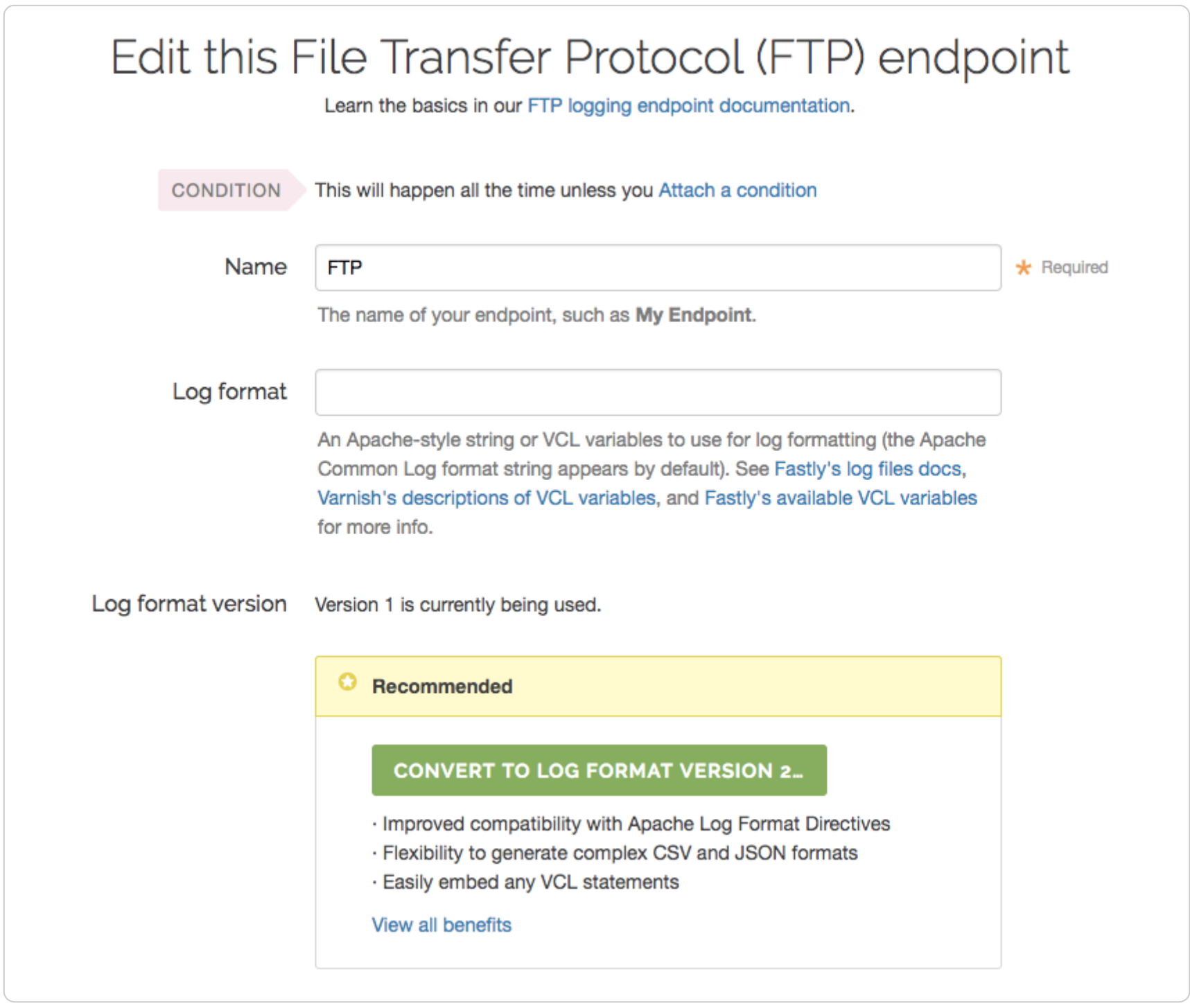
Upgrading is a permanent change. Logging objects using version 2 formatting cannot be downgraded to version 1.

Follow these instructions to upgrade a logging endpoint to the version 2 custom log format:

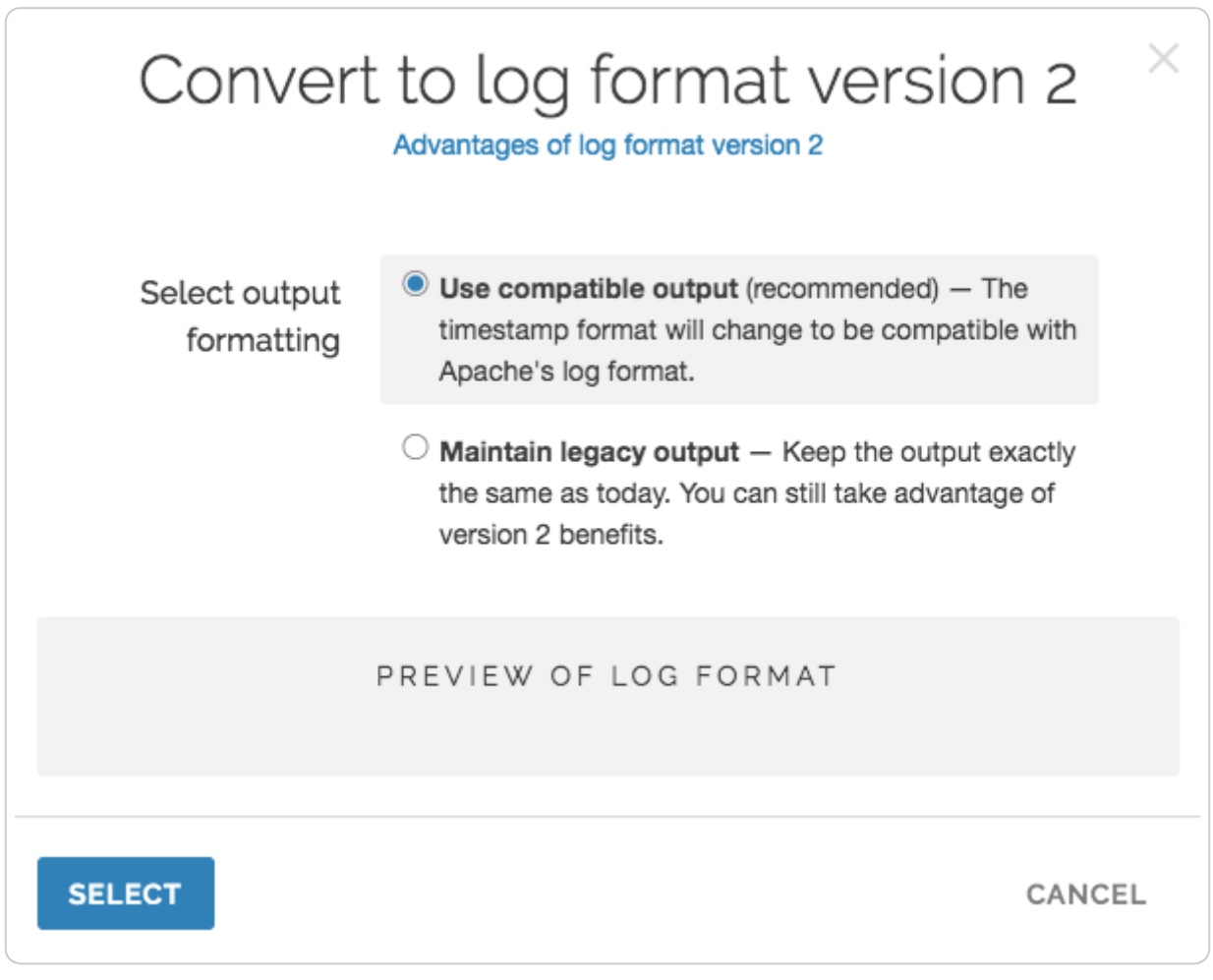
1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears. If you have any logging endpoints using the version 1 custom log format, a message appears indicating that they can be updated.



5. Click the name of a logging endpoint you want to edit. The Edit this endpoint page appears.



6. Click the **Convert to Log Format Version 2** button. The Convert to log format version 2 window appears.



7. Select an output format:
   - **Use compatible output** is the recommended setting. This setting won't modify your timestamp format string, but your logs will be formatted differently. The new format will be compatible with Apache's log format.
   - **Maintain legacy output** uses the version 2 parser, but the generated log string will be the same. This means that any instances of `%t` need to be turned into `%{now}V`, any instances of `%r` need to be turned into `%{req.url}V`, and any instances of `%b` need to be turned into `%{resp.http.Content-Length}V`.

8. Click the **Select** button. The Edit this endpoint page appears.
9. Click the **Update** button to upgrade the logging endpoint to the version 2 custom log format.
10. Click the **Activate** button to deploy your configuration changes.

Using the API to upgrade

To upgrade a logging endpoint using the Fastly API, either clone the active version of the service you need upgraded or choose a version of the service that is unlocked and not active, then run the following command on that in development version:

```
$ curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/<your Fastly service ID>/version/<version number>/logging/<logging endpoint>/<log name>' --data-binary '{"format_version":"2"}'
```

Keep in mind that the `format_version` field is a per-object field. Updating it on one logging object will *not* change it on any other objects.

For example, to update a Google Cloud Storage logging endpoint named "GCS Test," the curl command would look something like this:

```
$ curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -H 'Content-Type: application/json' 'https://api.fastly.com/service/SU1Z0isxPaozGVKXdv0eY/version/1/logging/gcs/GCS%20Test' --data-binary '{"format_version":"2"}'
```

NOTE

The `log name` included a space that needed to be URL encoded (the space into `%20`).

Determining which logging version is being used

To determine which logging version your service currently uses, issue the following curl command:

```
$ curl -X GET -H 'Fastly-Key: FASTLY_API_TOKEN'  'https://api.fastly.com/service/<your Fastly service ID>/version/<version number>/logging/<logging endpoint>/<log name>'
```

The curl command will produce JSON output detailing the configuration of your service version. For example:

```
{
    "address": "logs.papertrailapp.com",
    "created_at": "2016-04-01T15:37:30+00:00",
    "deleted_at": null,
    "format": "time.start.msec time.to_first_byte time.elapsed req.body_bytes_read req.bytes_read resp.http.content-length server.region client.ip %>s \"req.method req.url req.proto\" \"req.http.referer\" \"req.http.user-agent\"",
    "format_version": "2",
    "hostname": "logs.papertrailapp.com",
    "name": "fastly",
    "port": "11111",
    "public_key": null,
    "response_condition": "LOG /",
    "service_id": "1a2b3c4d5e6f7g8h9j0k",
    "updated_at": "2016-04-01T19:47:47+00:00",
    "version": "123"
}
```

The `format_version` field displays either a `1` or a `2` as appropriate for the custom log format being used.

Advantages of using the version 2 custom log format

The key advantages of using the version 2 custom log format include the following:

- Log lines are generated in `vcl_log` instead of `vcl_deliver` to allow us to accurately set the various size variables because `vcl_log` is run after the object has been delivered to the browser.
- The `%t` time directive is compatible with Apache log format. In version 1, we used a non-standard time format.

- The `%r` "first line of request" directive is compatible with Apache log format. In version 1, we incorrectly left off the protocol.
- When using the `%b` directive, which represents the size of a response in bytes, excluding HTTP headers is more accurate. In version 1, we used the reported Content-Length from the origin, which could be inaccurate (especially with ESI.
- We've added all Apache logging directives that make sense. In version 1, we used a smaller subset.

Making version 2 logs look like version 1

The default logging format for version 1 is as follows:

```
%h %l %u %t %r %>s
```

Most of the directives in version 2 are exactly the same - only `%t` and `%r` are different. After you upgrade to version 2 log formats, you can recreate the appearance of version 1 logs using the new `%{...}V` directive, which allows you to specifically include VCL in logging directives:

```
%h %l %u %{now}V %{req.method}V %{req.url}V %>s
```

In addition, if you are using the `%b` directive in version 1, then you can use this directive instead:

```
%{resp.http.Content-Length}V
```

Encrypting logs

Last updated: 2019-10-23

https://docs.fastly.com/en/guides/encrypting-logs

For supported logging endpoints, Fastly allows you to encrypt your log files before they are written to disk. The files are encrypted using OpenPGP (Pretty Good Privacy).

IMPORTANT

Be sure to take into account security, privacy, and compliance requirements when making configuration and endpoint decisions for the data you intend to include in streamed logs.

Generating a PGP key pair

To use this feature, you'll need to use a PGP implementation (such as GPG) to generate a public and private PGP key pair. Typically, this involves running the following command in a terminal application on your personal computer:

```
$ gpg --gen-key
```

Follow the instructions shown in your terminal application. Enter your email address and set a passphrase when prompted. Remember the values you enter.

WARNING

Keep your private key safe! If you lose it, your encrypted log files will be permanently unreadable.

Exporting the PGP public key

After you generate the PGP key pair, you'll need to export your public key. Typically, this involves running the following command in a terminal application on your personal computer:

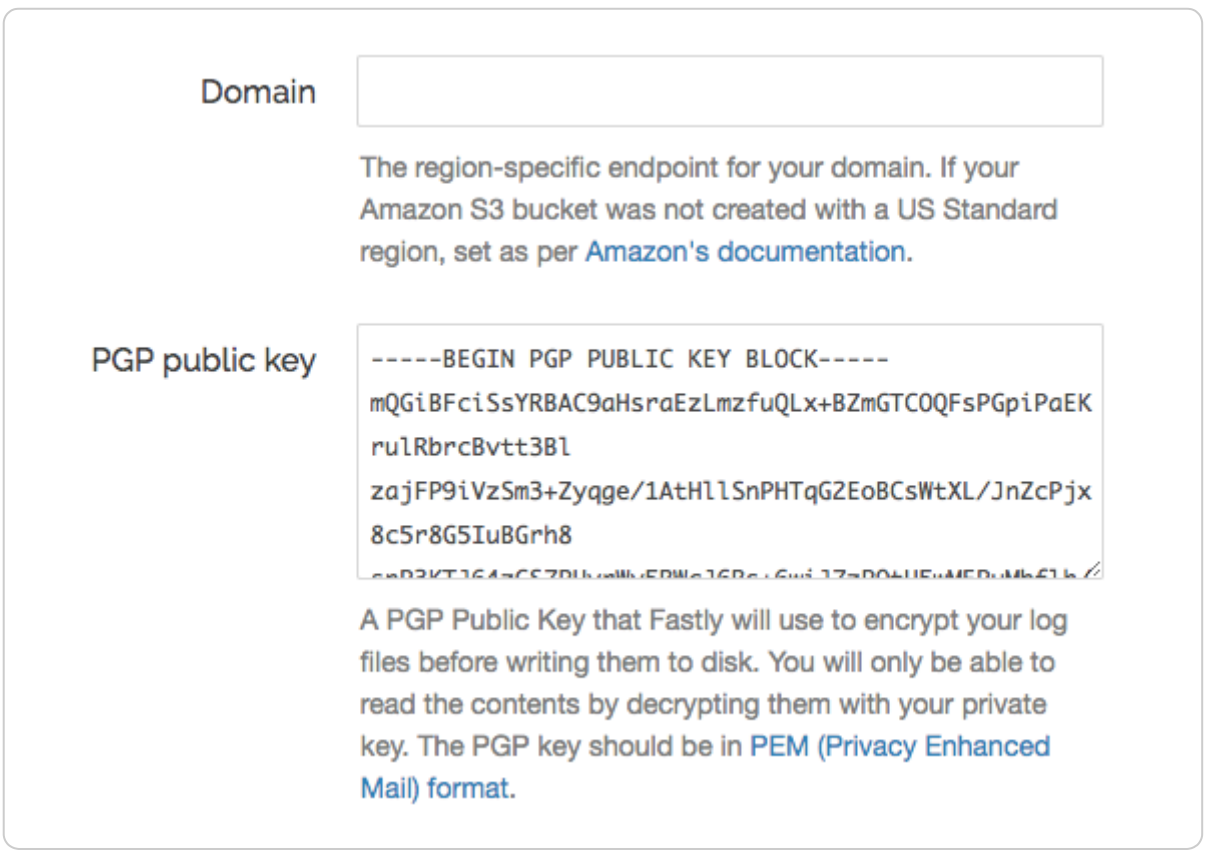
```
$ gpg --armor --export <your email>
```

The output will be in PEM (Privacy-Enhanced Mail) format and will look similar to the following:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQGiBFciSsYRBAC9aHsraEzLmzfuQLx+BZmGTCOQFsPGpiPaEKrulRbrcBvtt3Bl
zajFP9iVzSm3+Zyqge/1AtHllSnPHTqG2EoBCsWtXL/JnZcPjx8c5r8G5IuBGrh8
snP3KTJ64zCS7PUvrWy5RWcJ6Rs+6wiJ7zPOtU5wMEPuMbflh/soy50zrwCg74XN
[...REDACTED...]
-----END PGP PUBLIC KEY BLOCK-----
```

Enabling log encryption

To enable PGP encryption for a logging endpoint that supports it, copy and paste your public PGP key into the **PGP public key** field in the Fastly web interface when creating or editing a supported logging endpoint.



Decrypting log files

To read an encrypted log file, you'll need to download and decrypt it. Typically, this involves running the following command in a terminal application on your personal computer:

```
$ gpg --decrypt <encrypted log file>
```

Enter your passphrase to decrypt the log file.

Setting up remote log streaming

Last updated: 2021-09-28

https://docs.fastly.com/en/guides/setting-up-remote-log-streaming

Fastly's Real-Time Log Streaming feature allows you to automatically save logs to a third-party service for storage and analysis. Logs provide an important resource for troubleshooting connectivity problems, pinpointing configuration areas that could use performance tuning, and identifying the causes of service disruptions. We recommend setting up remote log streaming when you start using Fastly services.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Before you begin

Before setting up remote log streaming, keep the following in mind:

- Some third-party logging endpoints are disabled by default, so you won't see them in the Fastly web interface until they've specifically been enabled for your account. To enable these services, contact support@fastly.com.

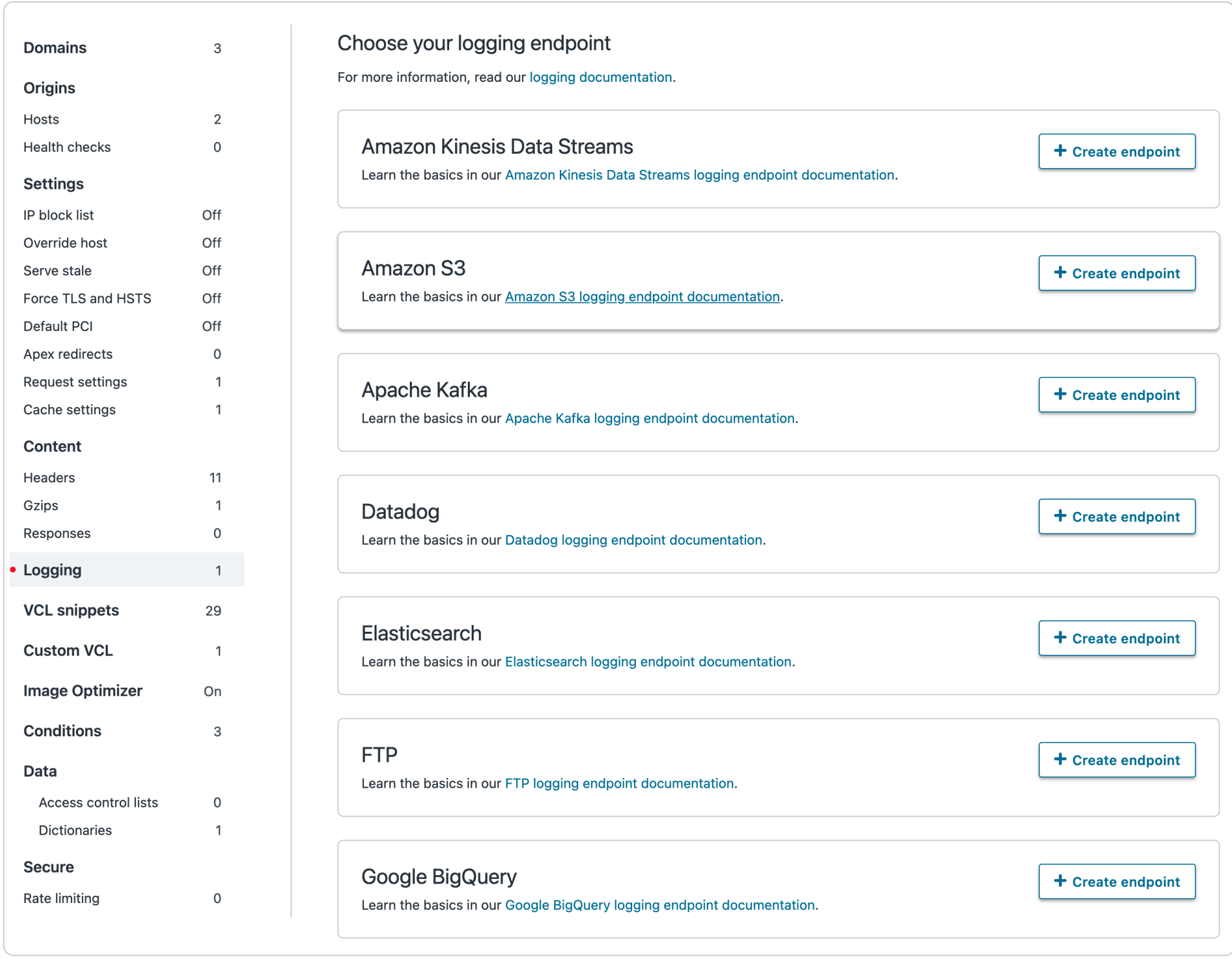
- Be sure to double-check the delivery formats required by your logging provider and what you're delivering to them. Some providers have strict formatting requirements for the formats they allow (e.g., JSON).

> **IMPORTANT**
>
> Be sure to take into account security, privacy, and compliance requirements when making configuration and endpoint decisions for the data you intend to include in streamed logs.

Configuring logging endpoints

You can configure one or more logging endpoints for Fastly services. Follow these instructions to access the logging settings:

1. Log in to the Fastly web interface.
2. From the **Home** page, select the appropriate service. You can use the search box to search by ID, name, or domain.
3. Click the **Edit configuration** button and then select the option to clone the active version. The Domains page appears.
4. Click the **Logging** link. The logging endpoints page appears. If you've already added a logging endpoint, click the **Create Endpoint** button. The list of available logging endpoints appears.



5. Follow the instructions in one of our logging endpoint guides to complete the set up process and deploy your changes.

Once you've clicked Activate to deploy your changes, events will begin being logged immediately. The logs may take a few moments to appear on your log server.

How, when, and where logs are streamed

To control log streaming, Fastly provides two versions of custom log formats, each of which uses Apache-style logging directives. The logging format strings in each of these versions are based on the Common Log Format (CLF).

Logs are streamed over TCP, not UDP, optionally using TLS for security with supported endpoints. Additionally, if you are using custom VCL be sure to include the `#FASTLY log` macro in your `vcl_log` handler.

By default, logs are placed in your root directory every hour using the file naming format `YYYY-mm-ddThh:mm:ss-<uid>`. You can change both the frequency and path of these files. Our guide on changing where log files are written provides more information.

If you've configured multiple logging endpoints for your service, the logs will be sent to all of the logging endpoints.

Fastly uses several different log-server aggregation points and each will send logs files, none of which contain duplicate entries. These log files are created as soon as streaming starts and they're written to over the entire time period you specify (or the default). Once that time has passed, the files aren't touched any more and the logging process creates a new batch of files.

The number of log-server aggregation points may change over time in line with our capacity requirements. If you're sending logs to a storage endpoint and are concerned about the number of log files that will be created on your disk, consider choosing a logging endpoint that supports real-time ingestion, which will eliminate a need for pre-processing log files.

Escaping characters in logs

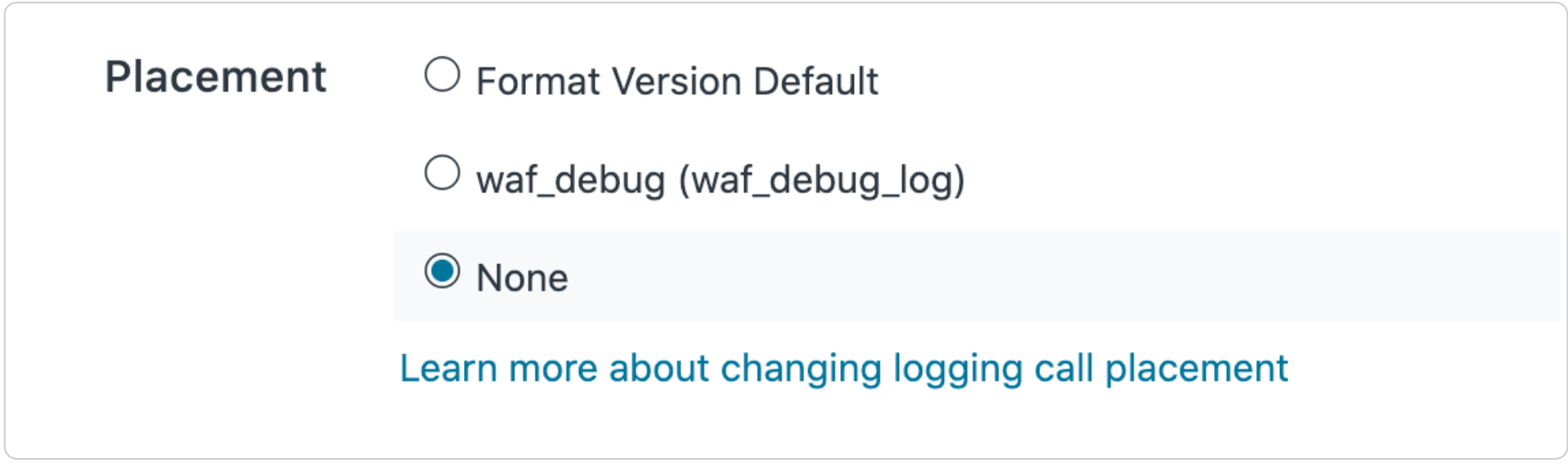
Logs respond to VCL like any other object. For example, the following code can escape quotes from User-Agent your log stream:

```
log {"syslog serviceid endpointname :: "} {"""} cstr_escape(req.http.user-agent);
```

Preventing duplicate log entries when using custom VCL

If you use custom VCL commands for logging, you may notice duplicate entries in your logs. This happens because logs are being generated by both Fastly and the custom VCL logging commands. You can eliminate the duplicate entries by following these steps:

1. On the Logging endpoints page, click the name of the logging endpoint you want to edit. The Edit this endpoint page appears.
2. From the **Placement** controls, select **None**.



3. Click the **Update** button.
4. Click the **Activate** button to deploy your configuration changes.

Fastly will stop generating log entries, and your logs will only contain entries generated by the custom VCL logging commands.

Troubleshooting common logging errors

The Fastly web interface displays errors with your logging configuration. You can also use the `logging_status` API endpoint to troubleshoot problems with your service's logging configuration:

```
$ curl -sg -H "Fastly-Key:$token" \
  "https://api.fastly.com/service/:SERVICE_ID/logging_status"
```

The output will indicate whether Fastly has detected an error. If `BrokenNow` is set to `false`, Fastly hasn't detected a problem with your logging configuration:

```
{"1234567890ABCDEF/my-service":{"LastErrorTime":null,"LastError":null,"BrokenNow":false}}
```

If an error in the Fastly web interface suggests that your logging configuration appears to be broken for the currently activated service version but you're still receiving some logs, not all of Fastly's log aggregators may be able to connect to your endpoint's server. It's likely the maximum number of concurrent connections has been reached. Try configuring your logging endpoint's server to allow a higher maximum number of inbound connections and then see if the error clears up after a couple of hours.

Useful conditions for logging

Last updated: 2019-07-31

https://docs.fastly.com/en/guides/useful-conditions-for-logging

In addition to the standard logging directives, the following conditions can be used for logging when you set up remote log streaming.

IMPORTANT

Be sure to take into account security, privacy, and compliance requirements when making configuration and endpoint decisions for the data you intend to include in streamed logs.

Logging errors only

You can log errors only if you want a general purpose log that catches everything and a more detailed log if there's an error:

```
fastly_info.state == "ERROR"
```

You can also log only 500 errors:

```
resp.status >= 500 && resp.status < 600`
```

Logging only specific URLs using an Edge Dictionary

Using an Edge Dictionary (e.g., `urls_to_log`), you can log specific URLs having issues:

```
table.lookup(urls_to_log, req.url.path) == "log"
```

If a URL becomes a problem, you can start logging it by using the API to add the URL's path to the dictionary as a key with the value "log".

Logging samples

If you have a high-volume service, you might want to log only a proportion of requests by using the `randombool` VCL function in a condition. The following example will log only one percent of all requests:

```
randombool(1,100)
```

You could combine that with an Edge Dictionary to change the percentage of requests logged without having to deploy a new version of your service. The following example uses an Edge Dictionary named `service_variables`:

```
randombool(std.atoi(table.lookup(service_variables, "logging_percentage", "0")), 100)
```

In the example above, if the key `logging_percentage` doesn't exist, nothing will be logged.

Using `false` to construct a log string in custom VCL

To construct a log string in custom VCL, you can use the `false` condition. This condition makes sure nothing is sent to Fastly logging objects.

Useful log formats

Last updated: 2022-02-17

https://docs.fastly.com/en/guides/useful-log-formats

Different systems have standardized on different logging formats over time. Fastly believes logging should be as customizable as possible, working with whichever infrastructure you already have in place. This guide details some of the more complicated custom logging strings (e.g., JSON, Key/Value, CSV, and URL-encoded) you can use to implement the logging formats mentioned in the Apache logging module.

NOTE

Fastly provides two versions of custom log formats. The version 2 logging formats, used by default when you create a new logging endpoint, improve compatibility with Apache's logging directives.

IMPORTANT

Be sure to take into account security, privacy, and compliance requirements when making configuration and endpoint decisions for the data you intend to include in streamed logs.

TIP

You can log any Varnish variable or Fastly's extensions to VCL. Consider reading our guide to useful variables to log.

Common Log Format (CLF)

```
%h %l %u %t "%r" %>s %b
```

This is the default for many of our logging providers.

Common Log Format with Virtual Host

```
%v %h %l %u %t "%r" %>s %b
```

NCSA extended/combined log format

```
%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-agent}i"
```

Referer log format

```
%{Referer}i -> %U
```

Agent (Browser) log format

```
%{User-agent}i
```

Custom Tags to Loggly or RFC 5424 to another provider

You'll have to create a regular Syslog logging object pointing at your RFC 5424 compatible endpoint. For example, to send custom tags to Loggly, create a new Syslog object and set `hostname` to `logs-01.loggly.com`, `port` to `6514`, `use_tls` to `true`, and the `message format` field to `blank`. You should also make sure the `token` field is blank. Then, in the `format` field, you would put the following:

```
<134>1 %{%Y-%m-%dT%TZ}t %{server.datacenter}V <log name> - - [<token>@<PEN> tag="fastly" tag="other-tag" id="12345" key="som
e-value" <tags>] <regular format string>
```

The various fields you need to replace are:

- *log name* - this can be whatever you want but we recommend using the same name as you've used for the logging object in Fastly.
- *token* - the private token for your RFC5424 endpoint (if sending to Loggly this is your Customer Token).
- *PEN* - this is a Private Enterprise Number. For example the Loggly PEN (Private Enterprise Number) is 41058. If you want to send to another provider then you can look up their PEN on the IANA registry and use that.
- *tags* - these can be any key/value pairs you want. Two appear in the above example: `fastly` and `other-tag`. Valid tag values include all alpha-numeric characters, the dash (`-`), the period (`.`), and the underscore (`_`). Tag values with spaces aren't valid and will be dropped by many logging providers such as Loggly. Additional information about tags, their restrictions, and details about how Loggly parses tags, can be found in the Loggly documentation but is useful information for sending data to all RFC 5424 compatible endpoints.
- *regular format string* - this is regular Fastly logging directives, put whatever you want here (for example: the Common Log Format mentioned above).

Structured data

The examples below demonstrate different representations of the same variables and variable types:

Name	VCL Value	Type	Description
Protocol	`req.protocol`	string	The HTTP protocol version.
Epoch Seconds	`time.start.sec`	number	The time at the start of the request in seconds.
Start Time	`begin:%Y-%m-%dT%H:%M:%S%z`	time	The time at the start of the request in ISO 8601. format
User Agent	`req.http.User-Agent`	escaped string	The User-Agent request header.
Is IPv6	`req.is_ipv6`	boolean	Whether the request was made over IPv6 or not.
ID	`deadbeef`	literal string	A generic ID.
Some String	`dwayne "the rock" johnson`	escaped literal string	A string with quotation marks in it.
Version	`1.1`	literal number	A generic version number.

JSON

```
{
  "protocol" : "%H",
  "epoch_seconds" : %{time.start.sec}V,
  "time_start" : "%{begin:%Y-%m-%dT%H:%M:%S%z}t",
  "user_agent" : "%{User-Agent}i",
  "is_ipv6" : %{if(req.is_ipv6, "true", "false")}V,
  "some_string":"%{json.escape(\{"dwayne "the rock" johnson"\})}V",
  "id" : "deadbeef",
  "version" : 1.1
}
```

IMPORTANT

When logging to BigQuery, any TIMESTAMP field requires using the timestamp format in the JSON above. When using the `DATETIME` type instead, the timezone designator should be omitted (i.e., use `"%{begin:%Y-%m-%dT%H:%M:%S}t",` instead).

CSV

```
%H, %{time.start.sec}V, %{begin:%Y-%m-%dT%H:%M:%S%z}t, %{regsub(req.http.User-Agent, \{"""\}, \{""""""\})}V, %{if(req.is_ipv
6, "true", "false")}V, deadbeef, %{regsub(\{"dwayne "the rock" johnson"\}, \{"""\}, \{""""""\})}V, 1.1
```

Key/Value

```
protocol:%H, epoch_seconds:%{time.start.sec}V, time_start:%{begin:%Y-%m-%dT%H:%M:%S%z}t, user_agent:%{User-Agent}i, is_ipv
6:%{if(req.is_ipv6, "true", "false")}V, id:deadbeef, some_string:%{json.escape(\{"dwayne "the rock" johnson"\})}V, version:
1.1
```

URL Encoded

```
protocol=%H&epoch_seconds=%{time.start.sec}V&time_start=%{begin:%Y-%m-%dT%H:%M:%S%z}t&user_agent=%{urlencode(req.http.User-A
gent)}i&is_ipv6=%{if(req.is_ipv6, "true", "false")}V&some_string=%{urlencode(\{"dwayne "the rock" johnson"\})}V&id=deadbeef&
version=1.1
```

Useful variables to log

Last updated: 2021-05-11

https://docs.fastly.com/en/guides/useful-variables-to-log

In addition to the standard logging directives, the following request and response variables can be used for logging when you set up remote log streaming. You can also log any Varnish variable. Consider taking advantage of some of Fastly's extensions to VCL as well.

IMPORTANT

Be sure to take into account security, privacy, and compliance requirements when making configuration and endpoint decisions for the data you intend to include in streamed logs.

Time-related logging variables

These are the time-related variables that can be used for logging.

Variable	Description
`%{begin:%Y-%m-%dT%H:%M:%S%z}t`	The time of the start of the request in ISO 8601 format.
`%{end:%Y-%m-%dT%H:%M:%S%z}t`	The time of the end of the request in ISO 8601 format.
`%{time.elapsed.usec}V`	How long the request took in microseconds.
`%{time.start.sec}V`	When the request started in Epoch seconds.

Connection-related logging variables

These are the connection-related variables that can be used for logging.

Variable	Description
`%{if(req.is_ipv6, "true", "false")}V`	Whether the request was over IPv6 or not.
`%{if(req.is_ssl, "true", "false")}V`	Whether the request was over HTTPS or not.
`%{cstr_escape(tls.client.protocol)}V`	Which version of TLS was used by the client.
`%{cstr_escape(tls.client.servername)}V`	Which SNI server name the client sent.
`%{cstr_escape(tls.client.cipher)}V`	Which cipher the TLS request used.
`%{cstr_escape(tls.client.ciphers_sha)}V`	Which cipher the TLS request used.
`%{cstr_escape(tls.client.tlsexts_sha)}V`	A SHA of the TLS extension identifiers sent from the client as part of the TLS handshake, represented in Base64.
`%{if(fastly_info.is_h2, "true", "false")}V`	Whether or not this was an HTTP/2 request.

Variable	Description
`%{if(fastly_info.h2.is_push, "true", "false")}V`	Whether or not this was an HTTP/2 Push response.
`%{fastly_info.h2.stream_id}V`	What the HTTP/2 Stream ID was.

Request- and response-related logging variables

These are the request- and response-related variables that can be used for logging.

Variable	Description
`%{Fastly-Orig-Host}i`	The original Host requested if a Host header override is present.
`%{Host}i`	The current Host request header (because it could have been modified to send to the origin).
`%{Referer}i`	The Referer request header. Specifically, which URL linked to this page.
`%{User-Agent}i`	The User-Agent request header. Specifically, which browser requested this page.
`%{Accept}i`	The Accept request header. Specifically, the types of content the client can accept.
`%{Accept-Language}i`	The Accept-Language request header. Specifically, the human languages the client can respond with.
`%{Accept-Encoding}i`	The Accept-Encoding request header. Specifically, the content encoding the client is able to understand.
`%{Accept-Charset}i`	The Accept-Charset request header. Specifically, the character set encodings the client accepts.
`%{Connection}i`	The Connection request header. Specifically, whether or not the client can do keep-alive connections.
`%{DNT}i`	The DNT request header. Specifically, whether or not the client is sending a "Do Not Track" header.
`%{Forwarded}i`	The Forwarded request header. Specifically, the originating IP address of a request if this request is proxied.
`%{Via}i`	The Via request header. Specifically, the intermediate protocols and recipients between the user agent and the server on proxied requests.
`%{X-Requested-With}i`	The X-Requested-With request header. Generally used to identify Ajax requests that will send the value XMLHttpRequest.
`%{X-Requested-For}i`	The X-Requested-For request header. Specifically, the originating IP address of a request if this request is proxied.
`%{X-ATT-DeviceId}i`	The X-ATT-DeviceId request header. Specifically, the make, mode, or firmware of AT&T devices.
`%{Content-Type}o`	The Content-Type response header. Specifically, the MIME type of the content.
`%{TSV}o`	The TSV response header. Specifically, the Tracking Status Value suggested for sending in response to a DNT request.

Cache-related logging variables

These are the cache-related variables that can be used for logging.

Variable	Description

Variable	Description
`%{If-Modified-Since}i`	The If-Modified-Since request header. Specifically, the server will send back the requested resource, with a 200 status, only if it has been last modified after the given date.
`%{If-None-Match}i`	The If-None-Match request header. Specifically, the server will send back the requested resource, with a 200 status, only if it doesn't have an ETag matching the given ones.
`%{Cache-Control}o`	The Cache-Control response header. Specifically, whether or not all caching mechanisms from server to client may cache this object in seconds.
`%{Age}o`	The Age response header. Specifically, the age the object has been in a proxy cache in seconds.
`%{Expires}o`	The Expires response header. Specifically, the date and time after which the response is considered stale in "HTTP-date" format as defined by RFC 7231.
`%{Last-Modified}o`	The Last-Modified response header. Specifically, the last modified date for the requested object in "HTTP-date" format as defined by RFC 7231. Used in conjunction with the If-Modified-Since request header.
`%{ETag}o`	The ETag response header. Specifically, an identifier for a specific version of a resource. Used in conjunction with the If-None-Match Request header.
`%{obj.hits}V`	The number of hits this object has (cache specific).
`%{obj.lastuse}V`	The last time this object was used (cache specific).

And these Fastly-specific ones:

Variable	Description
`%{if(fastly_info.state ~"^(HIT|MISS)(?:-|$)", "true", "false")}V`	Whether this object is cacheable or not.
`%{regsub(fastly_info.state, "^(HIT-(SYNTH)|(HITPASS|HIT|MISS|PASS|ERROR|PIPE)).*", "\2\3") }V`	Whether the response was a HIT, MISS, PASS, ERROR, PIPE, HITPASS, or SYNTH(etic).

Geographic logging variables

These are the geographic variables that can be used for logging.

Variable	Description
`%{server.datacenter}V`	Which Fastly data center this request hit.
`%{client.geo.city}V`	Which city Fastly thinks the request originated from.
`%{client.geo.city.ascii}V`	An alias of `client.geo.city`.
`%{client.geo.city.utf8}V`	The city or town name associated with the IP address, encoded using the UTF-8 character encoding.
`%{client.geo.country_code}V`	Which country Fastly thinks the request originated from.
`%{client.geo.continent_code}V`	Which continent Fastly thinks the request originated from.
`%{client.geo.region}V`	Which region Fastly thinks the request originated from.

Size-related logging variables

These are the size-related variables that can be used for logging.

Variable	Description
`%{req.header_bytes_read}V`	The size of the request headers.
`%{req.body_bytes_read}V`	The size of the request body.
`%{resp.header_bytes_written}V`	The size of the response headers.
`%{resp.body_bytes_written}V`	The size of the response body.

Socket-related logging variables

These are the socket-related variables that can be used for logging.

Variable	Description
`%{client.socket.cwnd}V`	The client socket congestion window.
`%{client.socket.nexthop}V`	The IP address of the next gateway.
`%{client.socket.tcpi_rcv_mss}V`	The client socket max segment size for receiving.
`%{client.socket.tcpi_snd_mss}V`	The client socket max segment size for sending.
`%{client.socket.tcpi_rtt}V`	The client socket smoothed round-trip time in microseconds.
`%{client.socket.tcpi_rttvar}V`	The client socket round-trip time variance in microseconds.
`%{client.socket.tcpi_rcv_rtt}V`	The client socket receiver-side estimation of round-trip time in microseconds.
`%{client.socket.tcpi_rcv_space}V`	The current buffer space available for receiving data.
`%{client.socket.tcpi_last_data_sent}V`	The time since last data sent on client socket in microseconds.
`%{client.socket.tcpi_total_retrans}V`	The total number of packet retransmissions on the client socket.
`%{client.socket.tcpi_delta_retrans}V`	The change in number of packet retransmissions on the client socket.
`%{client.socket.ploss}V`	The client socket packet loss.

Miscellaneous logging variables

These are the miscellaneous variables that can be used for logging.

Variable	Description
`%{LF}V`	Literal new line (i.e., `"\n"`)

Account info

These articles describe how to manage account access, billing, and security.

https://docs.fastly.com/en/guides/account-info

§

These articles describe how to manage account access.

https://docs.fastly.com/en/guides/account-info#_account-management

Account lockouts

Last updated: 2021-12-09

https://docs.fastly.com/en/guides/account-lockouts

Why is my account locked?

For security reasons, Fastly limits the number of times someone can try logging in to an account. We don't want to give people unlimited attempts at guessing your password, so we stop them from trying after a limited number of failed attempts to sign in. You can change your password at any time when you're logged in to your account.

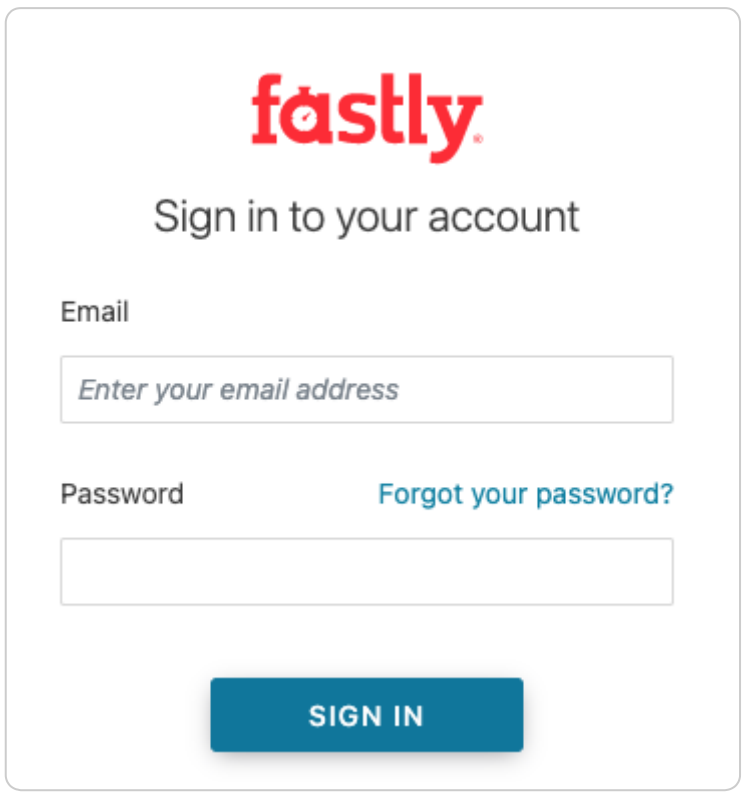
I am not using two-factor authentication. How can I access my account?

Once locked, you will not be able to sign in to your account, even with the correct password. To unlock your account because you exceeded the number of guesses you were allowed:

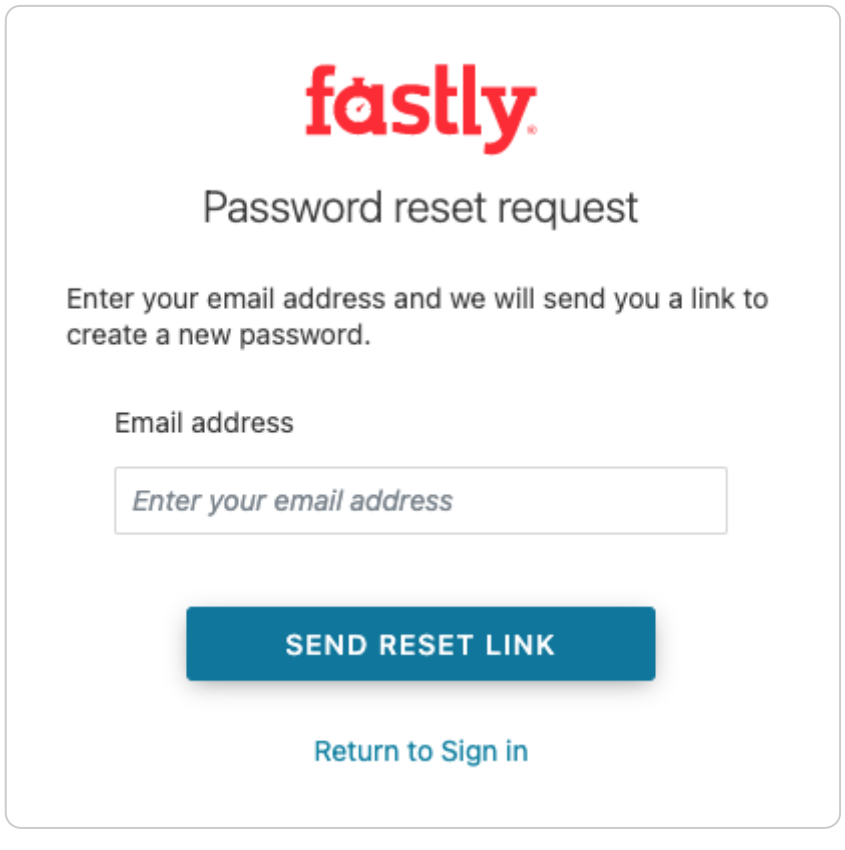
TIP

A superuser associated with your account can also issue an email with password reset instructions.

1. Point any standard web browser to the Fastly login page. The login controls appear.



2. Click the **Forgot your password** button above the password field. The Password reset request page appears.



3. In the **Email address** field, enter the email address you normally use to log in to your Fastly account.
4. Click the **Send Reset Link** button. Password reset instructions will be emailed to you.
5. Click on the password reset link in the emailed instructions that the system sends you. The Reset Your Password page appears.
6. Click the **Reset Password** button. The system sends you a temporary password to the email address you supplied.
7. Using the temporary password you receive, log in to your account. The controls to create a new password appear.
8. Fill out the **Reset Password** fields as follows:
   - In the **Current Password** field, enter the temporary password that the system emailed to you when you requested a password reset.
   - In the **New Password** field, enter a new password to replace the temporary password you were sent.
   - In the **Confirm Password** field, enter the new password a second time to confirm it.
9. Click the **Change Password** button. The system changes your password and logs you into your account.

I am using two-factor authentication. How can I access my account?

I don't have my mobile device.

If you do not have access to your mobile device, you can complete the login process using one of your recovery codes. These were the recovery codes you saved in a secured location outside of your Fastly account when two-factor authentication was first enabled. You can continue to use your recovery codes until your device is once again accessible. Recovery codes can only be used once, however, so remember to regenerate a new set to avoid running out before you recover your mobile device.

If you don't believe you will be able to recover your lost mobile device and you still have at least two recovery codes left, you can log in with one recovery code and disable two-factor authentication with a second code. Once two-factor authentication is disabled, you can re-enable it with a new mobile device at a later time and regenerate a new set of codes.

I don't have my mobile device and I don't have my recovery codes.

If you don't have your mobile device and didn't save any recovery codes, have another user at your company with the superuser role contact Customer Support at support@fastly.com. Have them inform Customer Support which user needs assistance with their login. After Customer Support verifies that the request is from a superuser, we will provide them with your recovery code. The superuser will then send you this information and reset your password so that you can access your account.

I don't have my phone, I didn't save my recovery codes, and I am the only superuser for the account.

Contact Customer Support at support@fastly.com. We will verify that you are associated with the company by phone. We will use the contact information located on the company website or under the Fastly account tab. Upon verification, we will send you a recovery code and reset your password.

Was my account compromised?

If a user's account appears to be hacked or phished, we may proactively reset the passwords for the affected accounts to revoke access to the hacker. In these cases, we send an email to the account's real owner (you) with additional information on how to reset the password. If you received one of these emails, follow the instructions in the email.

If you think your account has been hacked or phished, contact Customer Support at support@fastly.com immediately.

How is a locked account different from a blocked account?

Fastly allows you to restrict who can access your Fastly account based on the IP address of the person attempting to log in. This means that even with the correct login name and password, access to your Fastly account may be blocked if the IP doesn't match your company's list of allowed addresses.

If your company enables this optional IP allowlisting, they must keep the list of restricted IP addresses up to date. Only users with the role of superuser can make changes to the IP allowlist settings (your account owner is always a superuser), and your account owner must have a valid telephone number on file to do so.

If your IP addresses change after allowlisting is enabled and you forget to update your allowlist configuration, you will be locked out of your account. You will need to contact support@fastly.com to request that a Customer Support representative contact your account's owner via telephone during Fastly's regular business hours. To protect your account's security, we will not unlock your account based on an email request alone.

Changing company profile details

Last updated: 2022-02-09

https://docs.fastly.com/en/guides/changing-company-profile-details

Fastly allows you change most of the details about your company after your account has been created, including the company name and address, its owner, and any of the contacts you've created for specific company communications.

IMPORTANT

Users assigned the role of engineer can view company profile details. You must be the account owner or be assigned the role of superuser to change company profile details. You cannot change your customer ID.

Changing an account's company settings

Fastly allows you to change an account's company settings like company name, owner, and company address at any time after your account has been created.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. In the **Company name** field of the **Company settings** area, replace the current company name with the new one.
3. From the **Owner** menu, select a new company owner from the available names.
4. In the **Company HQ address** field, replace the existing company address with the new one.
5. To define the specific range of IP addresses authorized to access your account, follow the instructions for enabling or disabling an IP allowlist.
6. Click the **Update Company** button.

Creating company contacts

We allow you to set up a variety of company contacts to help Fastly route conversations and requests to the correct humans in your organization. To create a new company contact:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.

2. In the **Company contacts** area, click **Add contact**. The company contact controls appear.

3. From the **Select contact type** menu, select the type of company contact to create:

   - **Primary:** Select this option to specify who Fastly should contact when account renewals and decisions (e.g., new product adoption) must be discussed with someone in your organization.
   - **Technical:** Select this option to specify who Fastly should contact when product changes (e.g., retirements and deprecations) must be communicated to someone in your organization who can make changes to your services.
   - **Security:** Select this option to specify who Fastly should contact when security-related vulnerabilities or incidents must be communicated to someone in your organization.
   - **Emergency:** Select this option to specify who Fastly should contact when urgent, high-severity issues must be communicated to someone in your organization.
   - **Billing:** Select this option to specify who Fastly should contact when sending out invoices to someone in your organization or for confirmation of cost-based training that will be billed at the account level.
   - **Third-Party Reports:** Select this option to specify who Fastly should contact when compliance-related issues (e.g., DMCA, GDPR) need to be addressed by someone in your organization.

4. From the **Name** menu, select the name of the human contact to specify for this contact type. The name of the contact appears, along with the email address associated with their account.

   > **TIP**
   >
   > Some contacts for your organization may not have nor require Fastly accounts and, thus, can't be selected from the controls. You can enter contact information manually by instead clicking **Enter contact manually** to enable manual entry of information in all Company contacts fields.

5. In the **Phone number** field, optionally enter a telephone number at which to contact the selected user.

6. Click **Add**. The contact's information appears.

> **IMPORTANT**
>
> Company contact information can only be added or removed, not edited. Once you've added at least one company contact of a particular type, you cannot remove that contact until you've added another of the same type to replace it. To update a contact's information, remove their entry via the **Remove** link and re-add them to the company contacts list instead. Contacts are automatically removed from the system if the assigned contact is deleted as an account user.

Enabling an IP allowlist for account logins through the web interface

Last updated: 2022-01-21

https://docs.fastly.com/en/guides/enabling-an-ip-allowlist-for-account-logins-through-the-web-interface

Fastly allows you to define the range of IP addresses authorized on your Fastly account from which users are able to login to the Fastly web interface. This optional IP allowlisting functionality is not enabled by default. It can restrict access to most of Fastly's API endpoints.

Limitations and considerations

If you decide to use optional IP allowlisting, keep the following things in mind:

- **Metrics and stats API endpoints are exempt.** Metrics and stats API endpoints, such as historical stats or real-time analytics, do no support allowlisting.
- **An account owner telephone number is required.** During setup, Fastly checks your current IP address against the list you provide to ensure you don't lock yourself out of your account. If your IP addresses change at a later date (for example, because you move offices) and you forget to update your allowlist configuration, you will be locked out of your account.

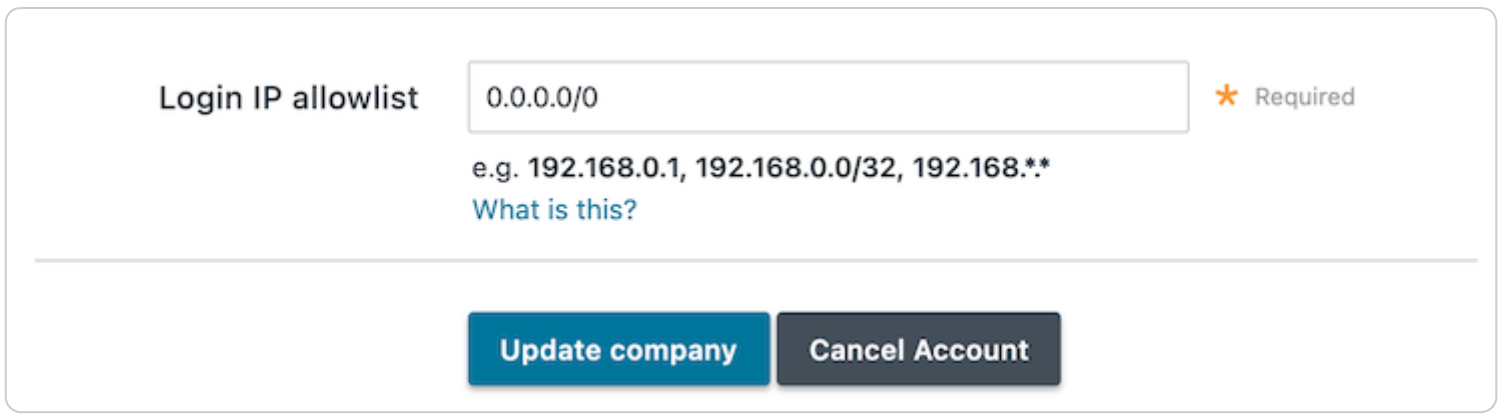
WARNING

If you are locked out of your account, you will need to email support@fastly.com and request that a Customer Support representative contact your account's owner via telephone during Fastly's regular business hours. To protect your account's security, ***we will not unlock your account based on an email request alone***.

Enabling an IP allowlist for account logins through the web interface

To restrict logins to the Fastly web interface based on a specific list or range of IP addresses, follow these steps.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. In the **Login IP allowlist** field of the **Company settings** area, replace `0.0.0.0/0` (the default IP range indicating no allowlisting) with the IP addresses for which web interface access to your account should be restricted.



You can include single or multiple IP addresses or IP ranges (separated by commas) as follows:

- a single IPv4 address (e.g., replace the default with `192.168.0.1`)
- an IPv4 CIDR range (e.g., replace the default with `192.168.0.0/32`)
- an IPv4 Wildcard range (e.g., replace the default with `192.168.0.*`, `192.168.*.1`, `192.168.*.*`)

3. Click the **Update Company** button.

Disabling an IP allowlist for account logins through the web interface

To disable IP allowlisting on your Fastly account and allow web interface access to any IP range, follow these steps.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. In the **Login IP allowlist** field of the **Company settings** area, enter `0.0.0.0/0` (the default IP range indicating no allowlisting).
3. Click the **Update Company** button.

Enabling and disabling two-factor authentication

Last updated: 2020-01-24

https://docs.fastly.com/en/guides/enabling-and-disabling-two-factor-authentication

Fastly supports two-factor authentication, a two-step verification system, for logging in to the web interface. In a two-factor authentication security process, users provide two means of identifying themselves to the system, typically by providing the system with something they know (for example, their login ID and password combination) and something they have (such as an authentication code). Organizations can enable company-wide two-factor authentication to require all users within the organization to use two-factor authentication.

Before you begin

You'll need to enter an authentication code regularly. Once two-factor authentication has been enabled, an authentication code will be requested upon login at least every 14 days for each computer and browser you use to access the Fastly web interface.

A mobile device is required. Using this security feature with a Fastly account requires a mobile device capable of scanning a barcode or QR code using a downloadable authenticator application. We recommend the following:

- For Android, iOS, and Blackberry: Google Authenticator
- For Android and iOS: Duo Mobile
- For Windows Phone: Authenticator

There are special requirements for using this feature with API tokens. See the API token documentation for more information.

Managing two-factor authentication as a user

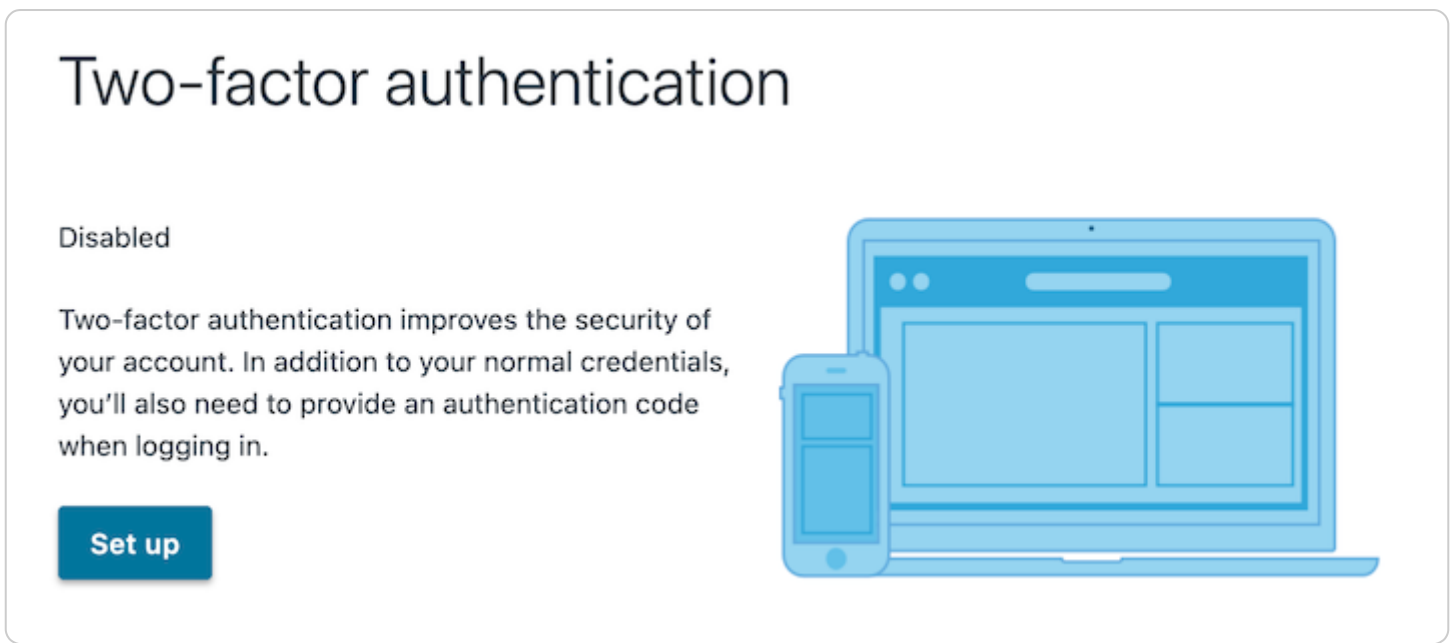
Depending on whether or not your organization has enabled company-wide two-factor authentication, you may be able to enable and disable two-factor authentication for your personal account. We also have instructions for recovering access to your account if you lose your mobile device.

Enabling two-factor authentication

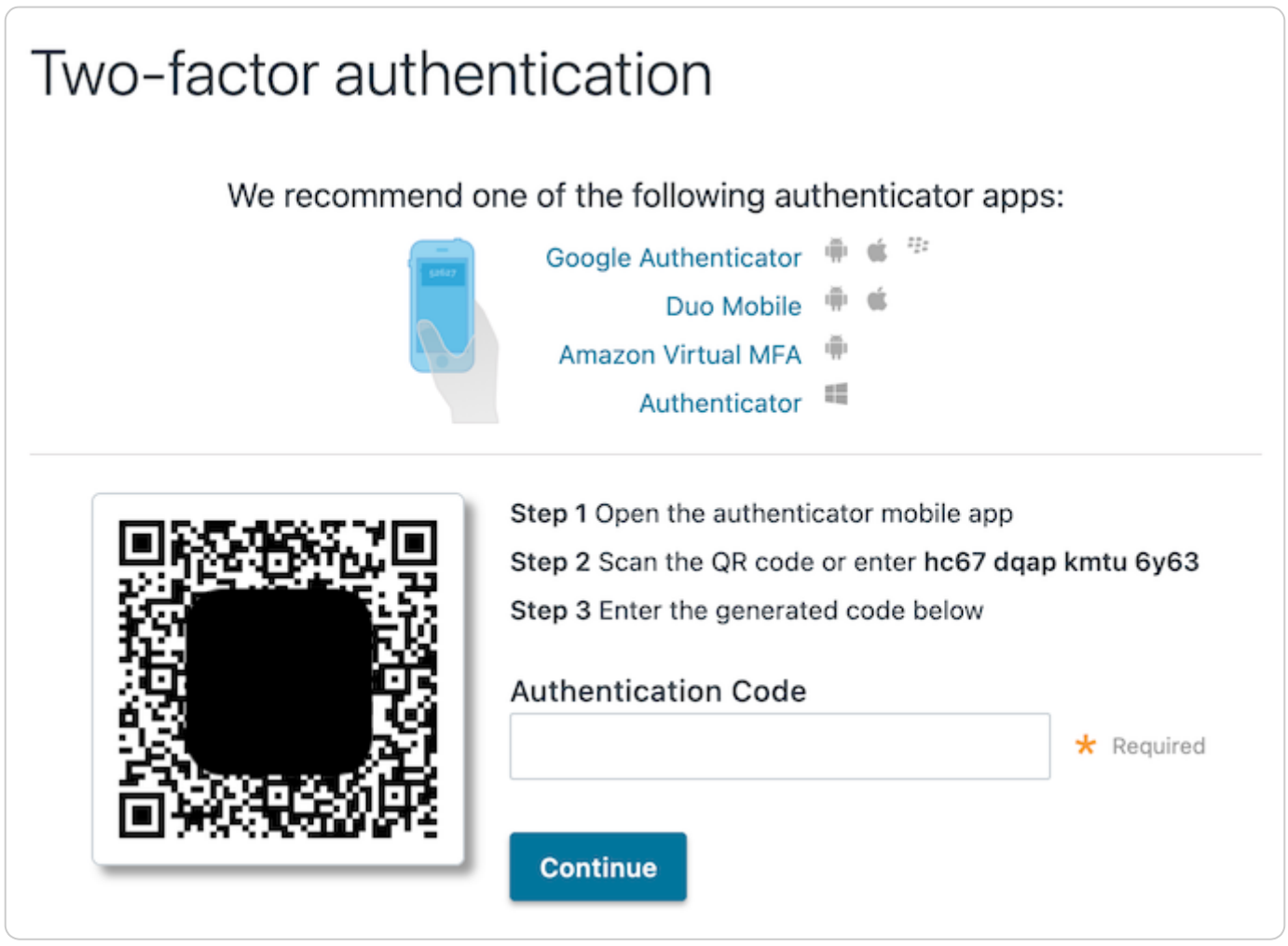
To enable two-factor authentication for your user account, follow the steps below.

> **IMPORTANT**
>
> If your organization has enabled company-wide two-factor authentication, you will be required to set up two-factor authentication when you log in to the Fastly web interface. Skip to step six for instructions.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Two-factor authentication** link. The Two-Factor authentication page appears.



3. Click the **Set Up** button. The password verification screen appears.
4. Verify your Fastly password and then click **Continue**. The authentication QR code appears.



IMPORTANT

The QR code above is an example. Scan the one that appears in the Fastly application, not in this guide.

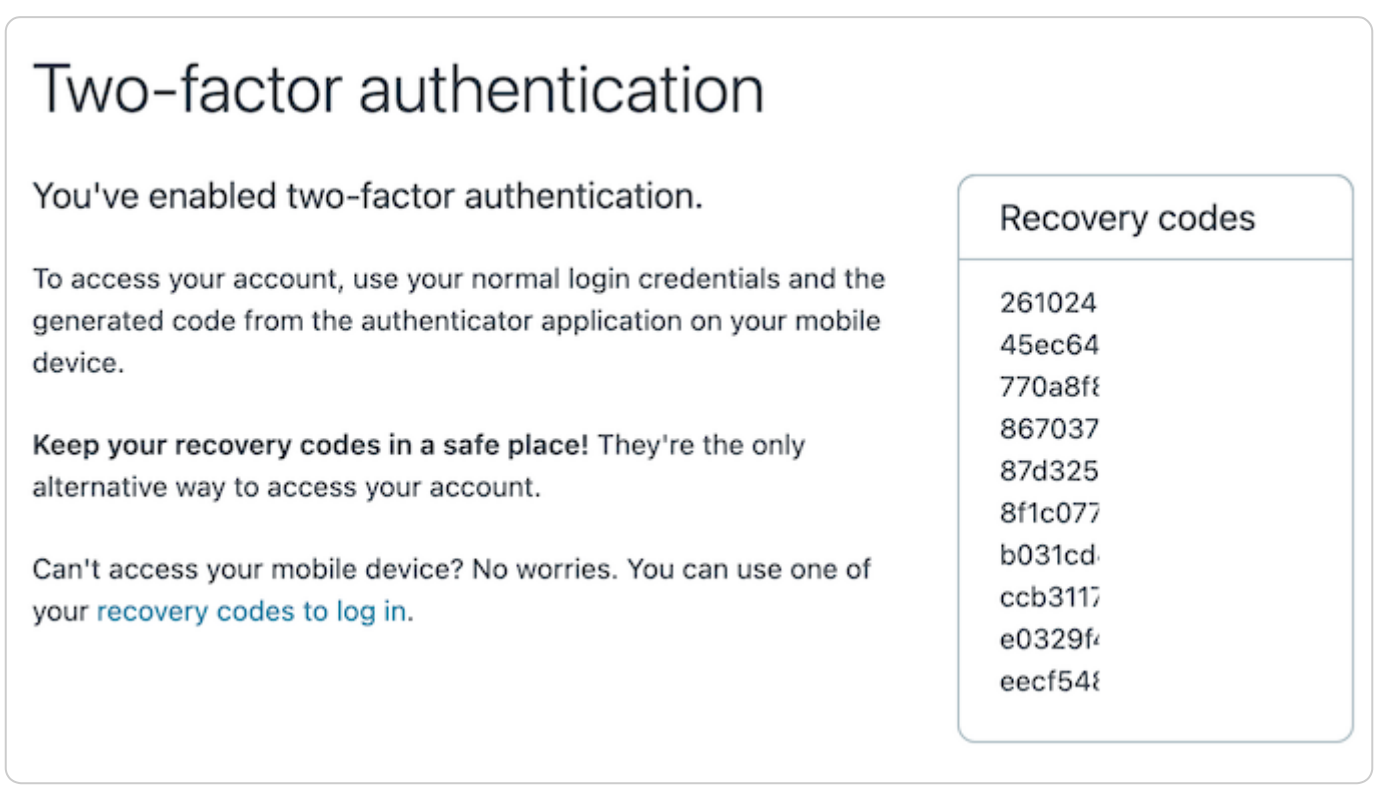
5. Launch the authenticator application installed on your mobile device and scan the displayed QR code or manually enter the key displayed in the setup window. A time-based authentication code appears on your mobile device. Depending on your device, however, a browser link may first appear. You need to click this link to save it. When you do, the words `Secret saved` appear briefly.

6. In the **Authentication Code** field in the Fastly application, enter the time-based authentication code displayed on your mobile device.

ANDROID USERS

A common time syncing issue may cause your authenticator codes to fail. You can correct this using Google's instructions for your authenticator application.

7. Click **Continue**. The confirmation screen appears along with your recovery codes.



IMPORTANT

If you're ever unable to access your mobile device, the displayed recovery codes can be used to log in when your account has two-factor authentication enabled. Each of these recovery codes can only be used once, but you can regenerate a new set of 10 at any time (any unused codes at that time will be invalidated). Store your recovery codes in a safe place.

Once you enable two-factor authentication for your account, any other open sessions will require reauthentication. For example, if you enable two-factor authentication in one browser window and you're viewing various aspects of your service through multiple additional browser windows, you will be required to reauthenticate in those additional windows, this time using an authentication code generated by the authenticator application installed on your mobile device (in addition to your email and password). Future logins will also require an authenticator code. By default, the system requires you to authenticate your login using an authentication code at least every two weeks for each computer and browser you use to access the Fastly web interface.

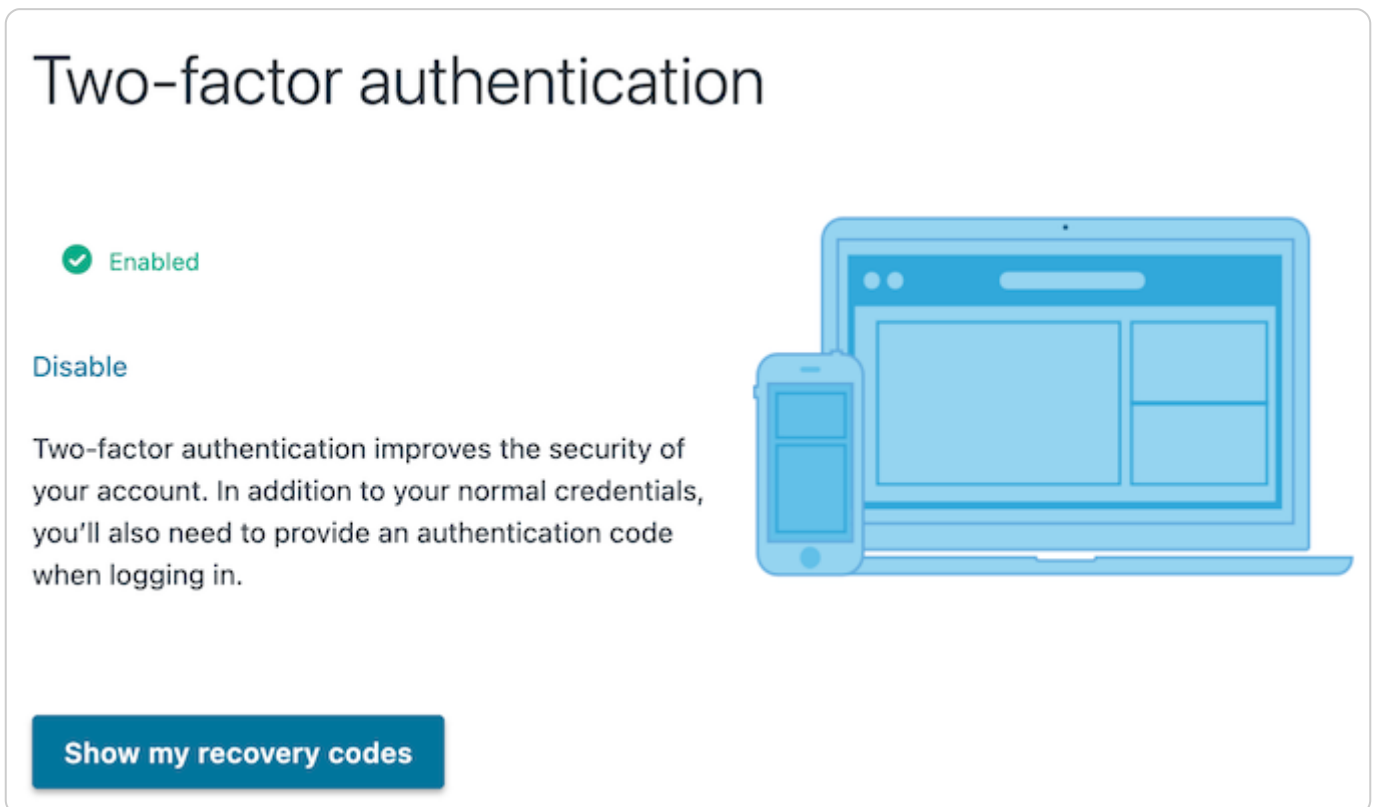
Disabling two-factor authentication

Once two-factor authentication is enabled for your account, you can disable it at any time by following the steps below.

IMPORTANT

If your organization has enabled company-wide two-factor authentication, you cannot disable two-factor authentication for your account.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Two-factor authentication** link. The Two-Factor authentication page appears.



3. Click **Disable**. The verification screen appears.
4. In the **Authentication Code** field, enter the time-based authentication code displayed in the authenticator application on your mobile device, then click **Confirm and Disable**.

NOTE

If you have lost your mobile device, you can enter a recovery code in the **Authentication Code** field. For more information, see the section on what to do if you lose your mobile device.

What to do if you lose your mobile device

If you lose your mobile device after enabling two-factor authentication, use a recovery code to log in to your Fastly account. You can continue to use recovery codes to log in until you get your mobile device back. Recovery codes can only be used once, however, so remember to regenerate a new list of codes to avoid running out before you recover your mobile device.

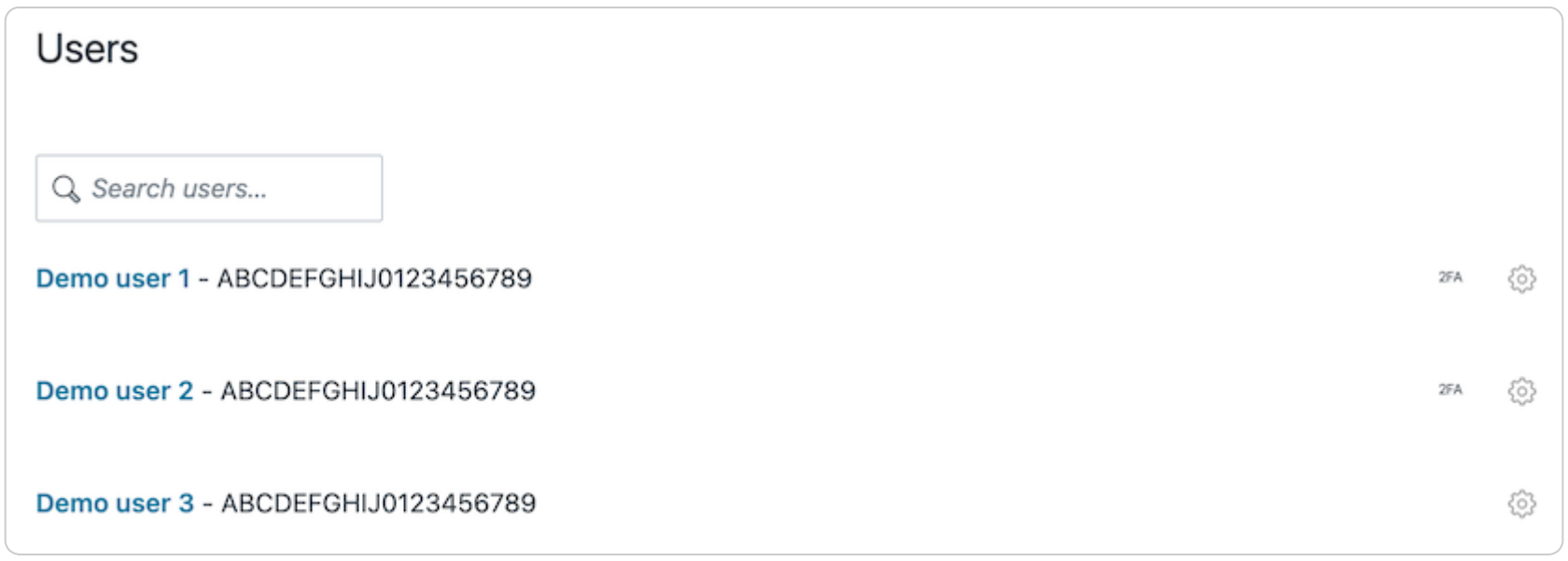
If you do not believe you will be able to recover your lost mobile device and you still have at least two recovery codes left, you can log in with one recovery code and disable two-factor authentication with a second code. Once two-factor authentication is disabled, you can re-enable it with a new mobile device at a later time and regenerate a new set of codes.

If your organization has enabled company-wide two-factor authentication, you can contact a superuser for your organization and ask them to reset your two-factor authentication.

Locked out of your account? See our article on what you can do about it.

Managing two-factor authentication as a superuser

If you are assigned the superuser role for your organization, you can view who has two-factor authentication enabled the User management settings for your Account. Users with this feature enabled have 2FA displayed next to their names.



To disable two-factor authentication for any user within your organization, select **Disable 2FA** from the menu that appears when you click the gear icon next to that user's name.

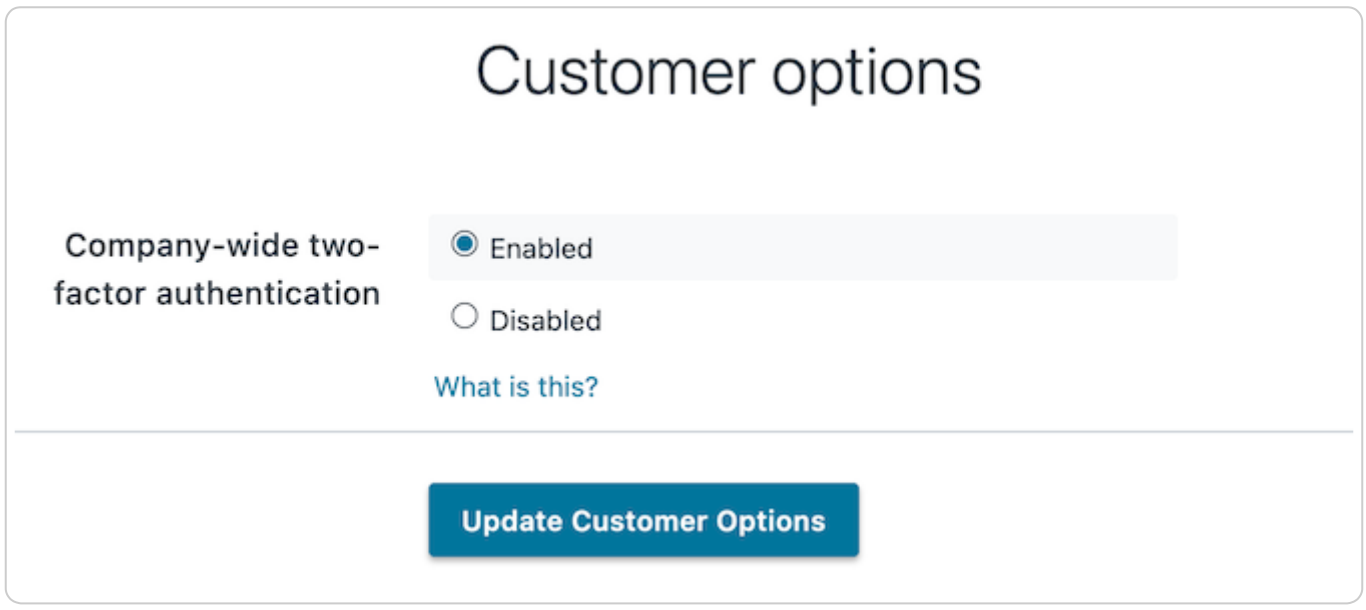
Managing two-factor authentication as a company

Organizations can enable two-factor authentication for all of their users. When the company-wide two-factor authentication feature is enabled, all users within the organization are required to use two-factor authentication to log in to the Fastly web interface, and they cannot disable two-factor authentication for their accounts.

Enabling company-wide two-factor authentication

Users assigned the superuser role can enable this feature on the Account page. To enable company-wide two-factor authentication for all users within your organization, follow the steps below.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. In the **Customer options** area, select **Enabled** from the **Company-wide two-factor authentication** controls.



3. Click **Update Customer Options**. A warning message appears stating that login sessions from non-2FA users in your company will immediately expire.
4. Click **Continue**. Two-factor authentication becomes required for all users in your company. Anyone currently logged in and not previously using 2FA on their account will be logged out of the Fastly web interface. Anyone who has not already enabled two-factor authentication for their account will be prompted to do so the next time they log in to the Fastly web interface.

Resetting a user's two-factor authentication

If company-wide two-factor authentication is enabled, and a user within the organization gets locked out of their account or needs to enable a new device, an account superuser can reset their two-factor authentication. To reset a user's two-factor authentication, follow the steps below.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link.
3. In the **Users** area, click the gear icon next to a user and then select **Reset 2FA**. A warning message appears.
4. Click **Reset**. The user will need to set up two-factor authentication for their account the next time they log in.

Disabling two-factor authentication for a single user's account

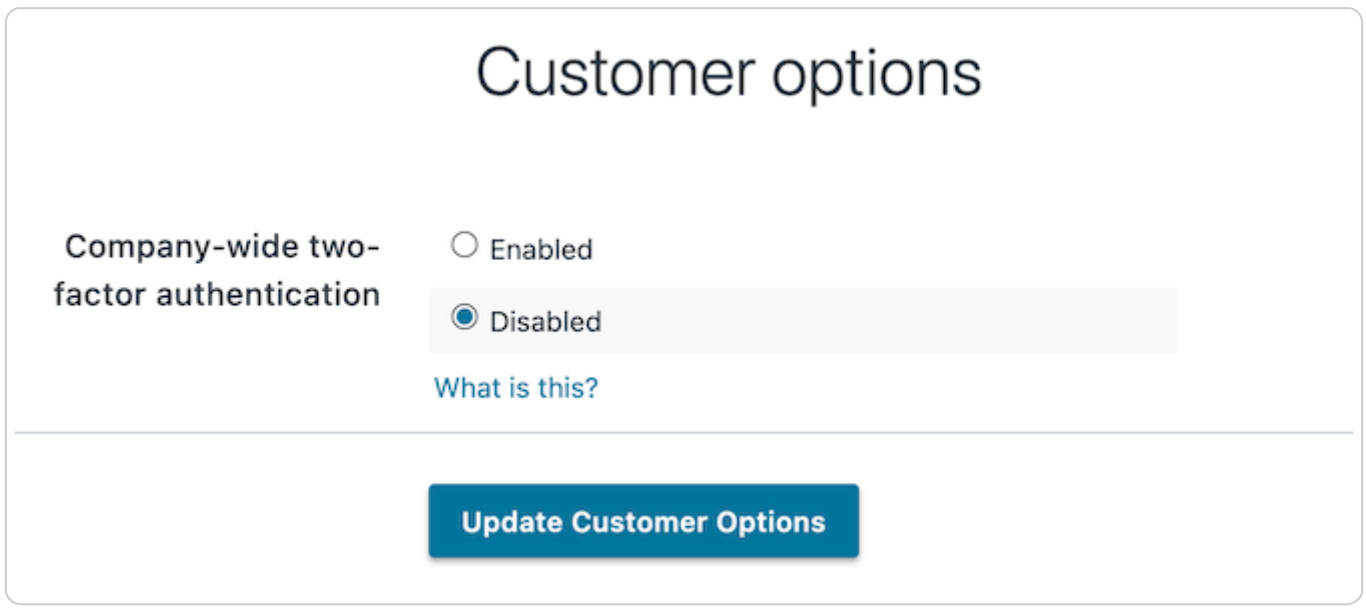
If company-wide two-factor authentication is enabled, a superuser can disable two-factor authentication for a single user's account. This is typically done for user accounts being used for scripts and session authentication. To disable two-factor authentication for a single user's account, follow the steps below.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link.
3. In the **Users** area, click the gear icon next to a user and then select **Ignore 2FA**. A warning message appears.
4. Click **Ignore**. Two-factor authentication will no longer be required for the selected user.

Disabling company-wide two-factor authentication

A superuser can disable company-wide two-factor authentication. Once this feature is disabled, existing users within the organization will be able to manage their own two-factor authentication settings, and new users will not be required to set up two-factor authentication to log in to the Fastly web interface. To disable company-wide two-factor authentication, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. In the **Customer options** area, select **Disabled** from the **Company-wide two-factor authentication** controls.



3. Click **Update Customer Options**. A warning message appears.
4. Click **Continue**. Company-wide two-factor authentication becomes disabled.

Finding and managing your account info

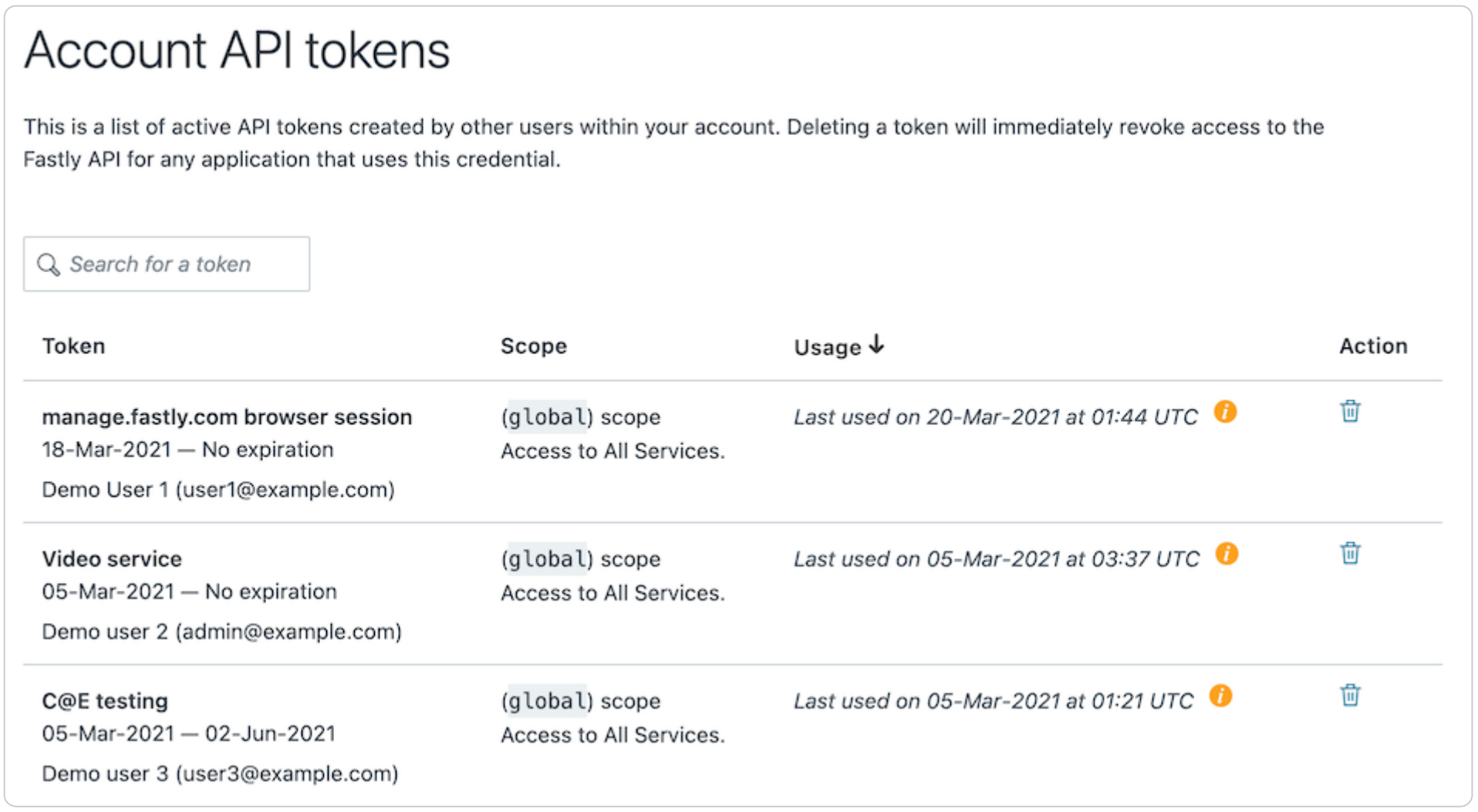
Last updated: 2021-10-25

https://docs.fastly.com/en/guides/finding-and-managing-your-account-info

Account information, including your service ID and your customer ID can be accessed directly from the Fastly web interface.

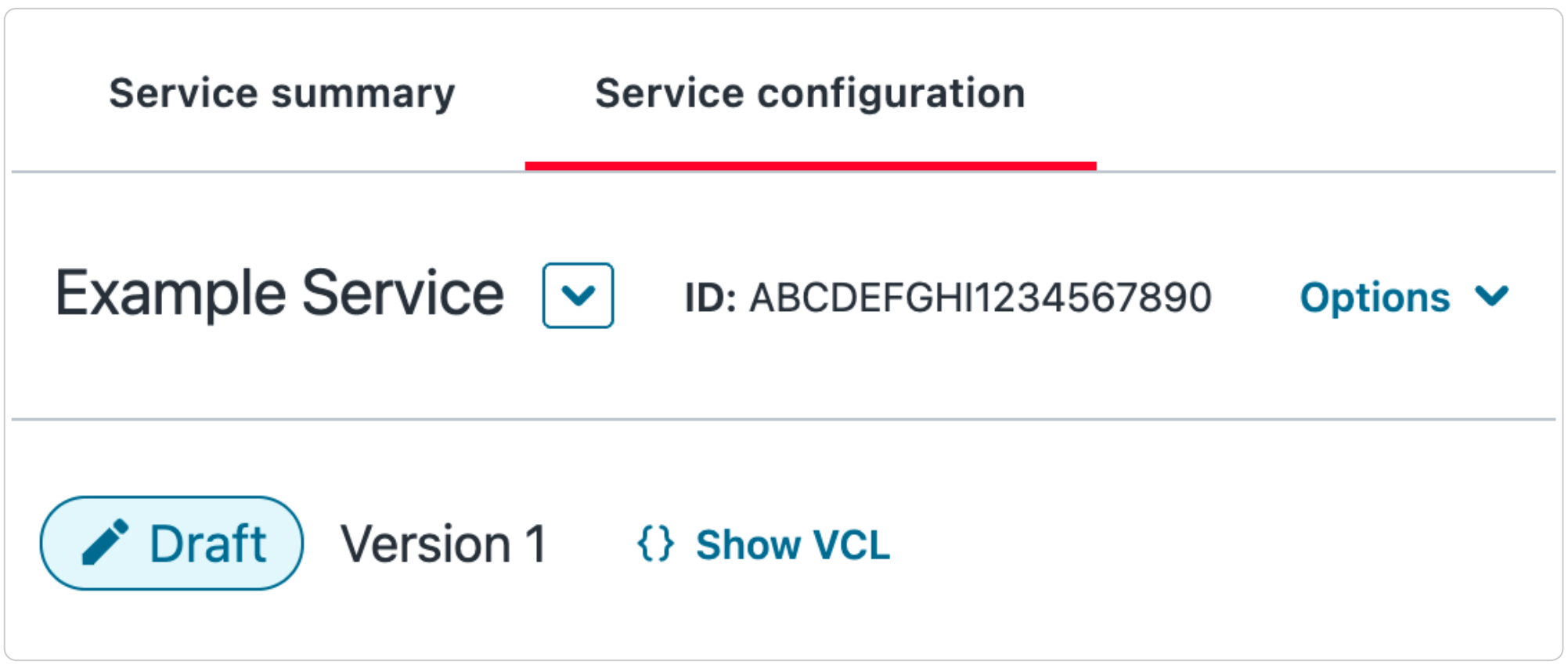
Finding your API tokens

Your account's API tokens appear in the **Account API tokens** of your **Account** page, which you access by clicking the **Account** link in the user menu.



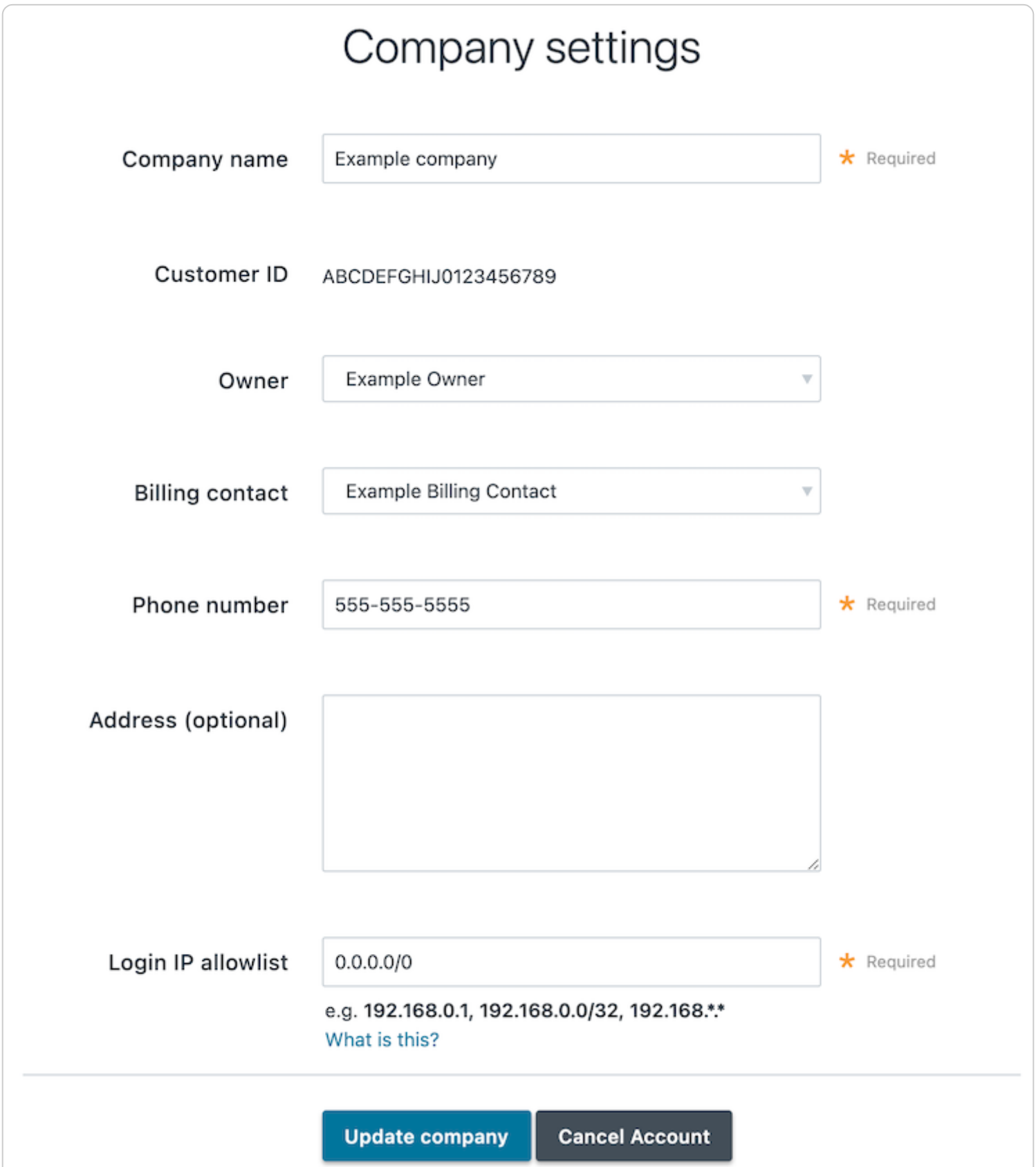
Finding your service ID

Your **Service ID** appears next to the name of your service on any page.



Finding your customer ID

Your **Customer ID** appears in the **Company settings** of your **Account** page, which you access by clicking the **Account** link in the user menu:



Finding your account owner

The name of your account **Owner** appears in the **Company settings** of your **Account** page, which you access by clicking the **Account** link in the user menu.

Monitoring account activity with the audit log

Last updated: 2021-06-22

https://docs.fastly.com/en/guides/monitoring-account-activity-with-the-audit-log

The audit log keeps track of events related to your account, users, and services. You can use the audit log to determine which changes were made and by whom. For example, you can use the audit log to:

- see when API tokens were created
- review who logged in to your account via the web interface
- view recent service configuration setting changes
- learn when users make changes to their account security settings

You can use the web interface and the Fastly API to view the audit log.

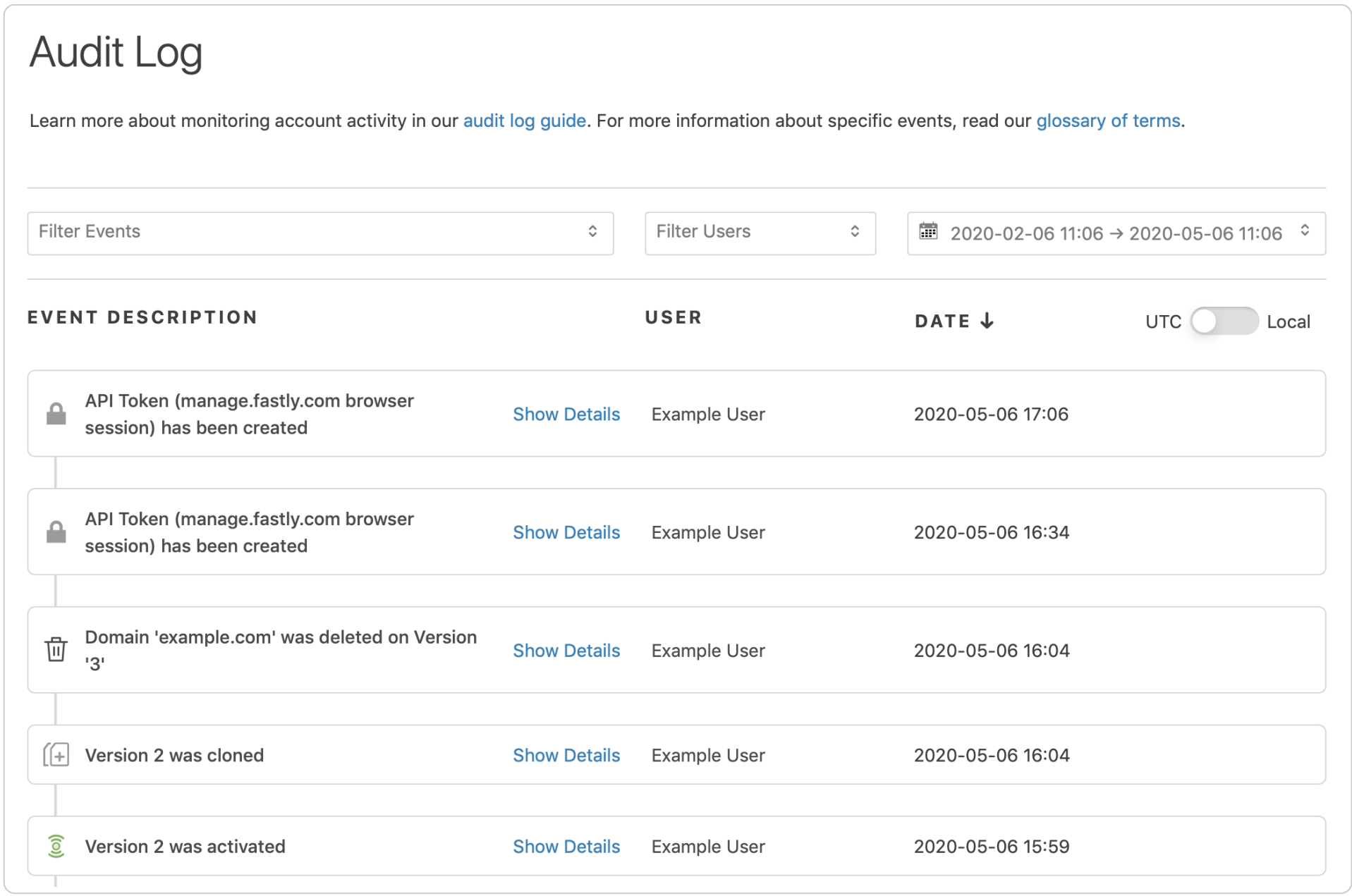
TIP

For information on monitoring events related to a service, see our guide on the event log. For information on monitoring the data that passes through Fastly, see our guide on Fastly's real-time log streaming features.

Accessing the audit log via the web interface

You must be assigned the role of superuser to view the audit log. Follow these instructions to access the audit log for your account:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Audit log** link. The Audit log page appears.



NOTE

As of June 22, 2021, audit log data is retained for a period of one year (365 days).

Accessing the audit log via the API

The `/events` API endpoint can be used to retrieve an account's audit log. You can filter these events by `user_id`, `service_id`, `customer_id`, and `event_type`. For example, you could make the following API call in a terminal application to view all recent events:

```
$ curl -g -H "Fastly-Key: FASTLY_API_TOKEN" "https://api.fastly.com/events?filter[customer_id]=x4xCwxxJxGCx123Rx5xTx&page[number]=1&page[size]=1"
```

The response will look like this:

```
{
  "data": [
      {
          "attributes": {
              "admin": false,
              "created_at": "2016-06-06T20:05:10Z",
              "customer_id": "x4xCwxxJxGCx123Rx5xTx",
              "description": "Version 2 was activated",
              "event_type": "version.activate",
              "ip": "127.0.0.0",
              "metadata": {
                  "version_number": 2
              },
              "service_id": "SU1Z0isxPaozGVKXdv0eY",
              "user_id": "4Pp0BW3UkBEJhG3N0kovLP"
          },
          "id": "5IH1QmNSV1Qi7jXc4oIZlZ",
          "type": "event"
      }
  ],
  "links": {
    "last": "https://api.fastly.com/events?filter[customer_id]=x4xCwxxJxGCx123Rx5xTx&page[number]=1&page[size]=1"
  }
}
```


See the API documentation for more information.

Reviewing service activity with the event log

Last updated: 2021-11-08

https://docs.fastly.com/en/guides/reviewing-service-activity-with-the-event-log

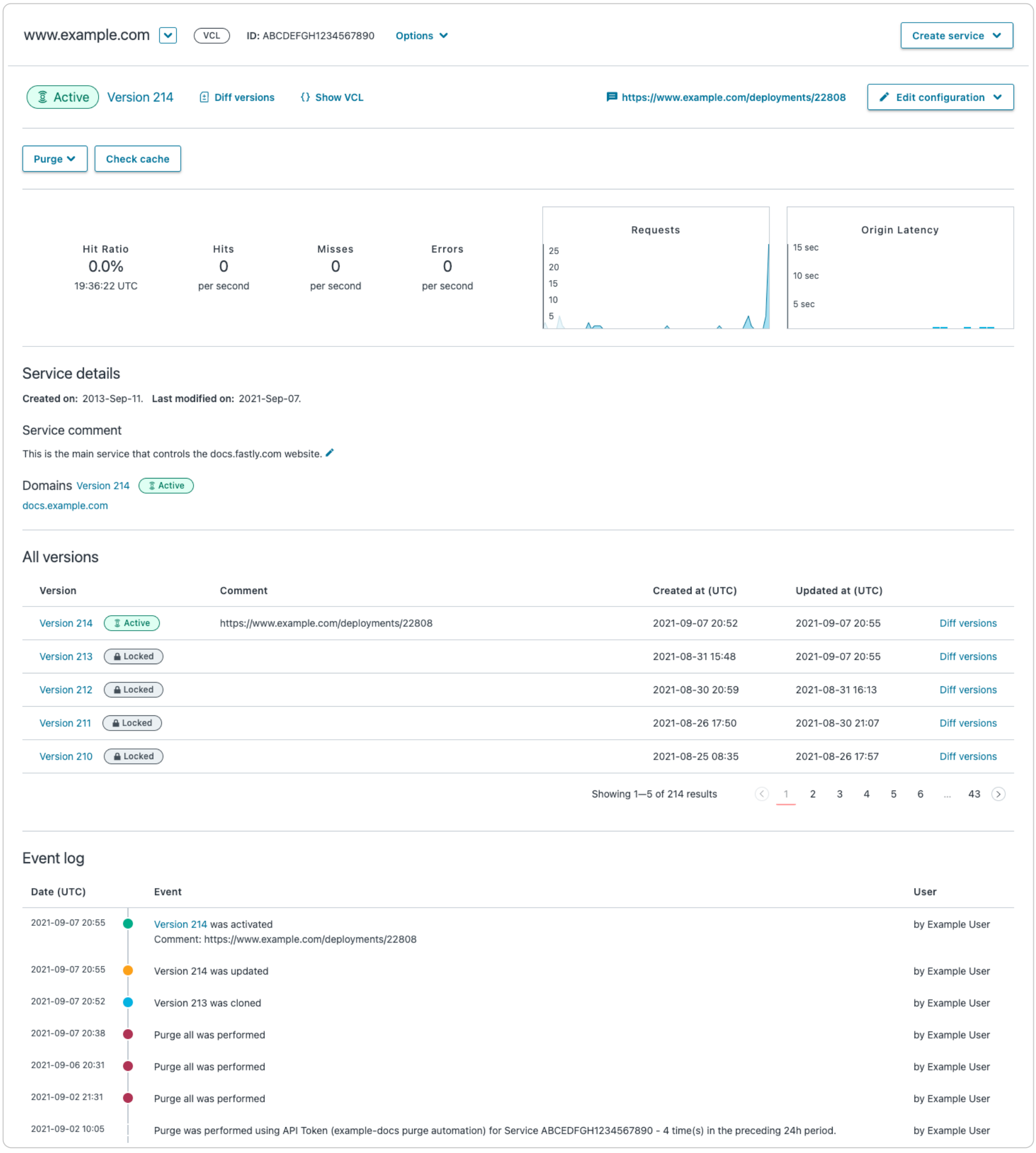
Event logs keep track of events related to a service. You can use event logs to determine which service-level changes were made and by whom. For example, you can use them to see who activated the most recent version of a service and view recent service configuration setting changes.

TIP

For information on monitoring account activity, see our guide on the audit log. For information on monitoring the data that passes through Fastly, see our guide on Fastly's real-time log streaming features.

Follow these instructions to access the event logs for a service:

1. Log in to the Fastly web interface and click either the **Deliver** link or the **Compute** link depending on your service type.
2. The most recent service-related events are displayed near the bottom of the page, in the Event log area.



The web interface displays the last 20 service-related events for the selected service.

> **NOTE**
>
> As of June 22, 2021, event log data is retained for a period of one year (365 days).

Using API tokens

Last updated: 2021-03-10

https://docs.fastly.com/en/guides/using-api-tokens

API tokens are unique authentication credentials assigned to individual users. You need to create an API token to use the Fastly API. You can use API tokens to grant applications restricted access to your Fastly account and services. For example, an engineer user could limit a token to only have access to a single service, and restrict the scope to only allow that token to purge by URL.

Every Fastly user can create up to 100 API tokens.

There are two places in the web interface where tokens are managed, depending on your user role. The Personal API tokens page allows you to create, view, and delete API tokens associated with your personal profile. The Account API tokens page allows superusers to view and delete any of the API tokens associated with the organization's Fastly account.

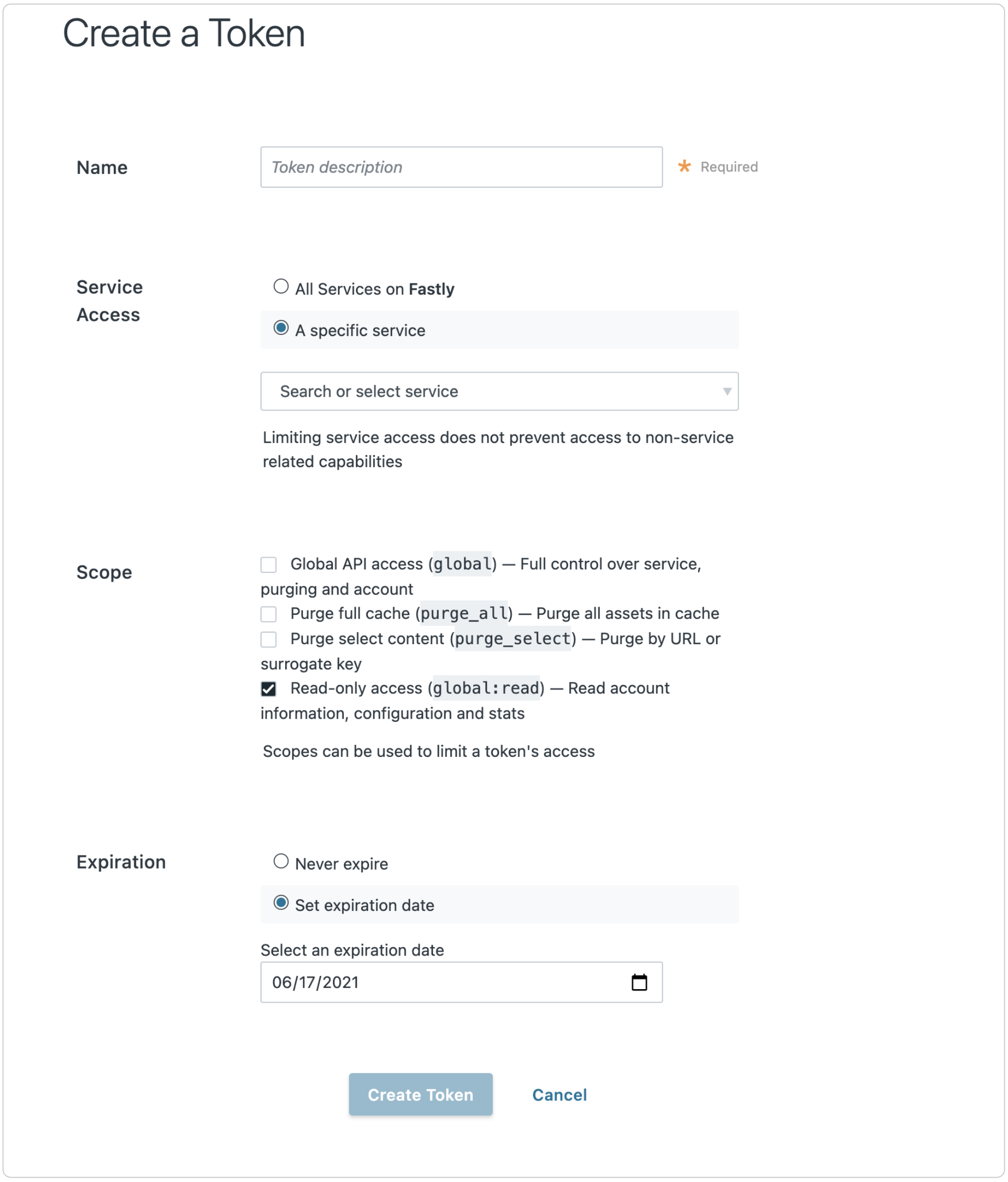
Best practices

Limiting an API token's service access and setting an expiration date restricts a credential's access, which can minimize the risk of damage if a credential is compromised. For more information, review the principle of least privilege.

Creating API tokens

To create an API token, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Personal API tokens** link. The Personal API Tokens page appears.
3. Click the **Create token** button. The Create a Token page appears.



NOTE

If prompted, be sure to re-authenticate your login.

4. Fill out the **Create a Token** fields as follows:
   - In the **Name** field, enter a descriptive name for the API token that indicates how or where you will to use the token.
   - In the **Apply to** area, select a service to restrict the service-level access of the token to one service or optionally switch to **All Services** to grant the API token access to all services.
   - In the **Set a scope** area, select one or more checkboxes to set a token's scope:
     - **Global API access (global):** Allows access to all endpoints, including purging.
     - **Purge select content (purge_select):** Allows purging with Surrogate-Key and URL. Does not include the ability to purge all cache.

- **Purge full cache (purge_all):** Allows purging an entire service via purge_all API request.
- **Read-only access (global:read):** Allows read-only access to account information, configuration, and stats.

- In the **Set a token expiration** area, optionally set the API token to never expire. API tokens are set to expire after 90 days by default. After a token expires, using it for any request will return an HTTP 401 response.

5. Click the **Create** button to create the new API token. The string that comprises the token appears.

This is the credential you'll use to authenticate via the Fastly API. Copy this string to a secure location — it will never be visible again. You may use the same token for multiple applications.

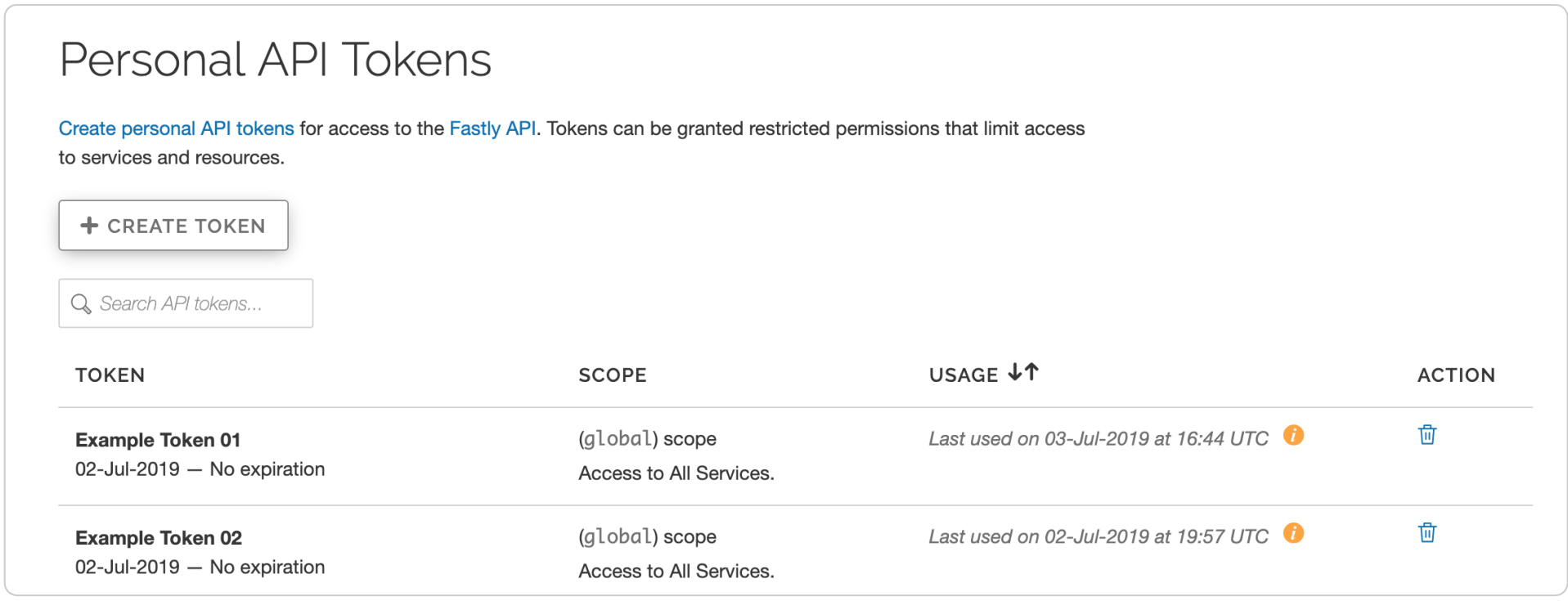
Viewing API tokens

You can view two types of API tokens for your account depending on your assigned role.

Viewing personal API tokens

To view personal API tokens, follow these steps:

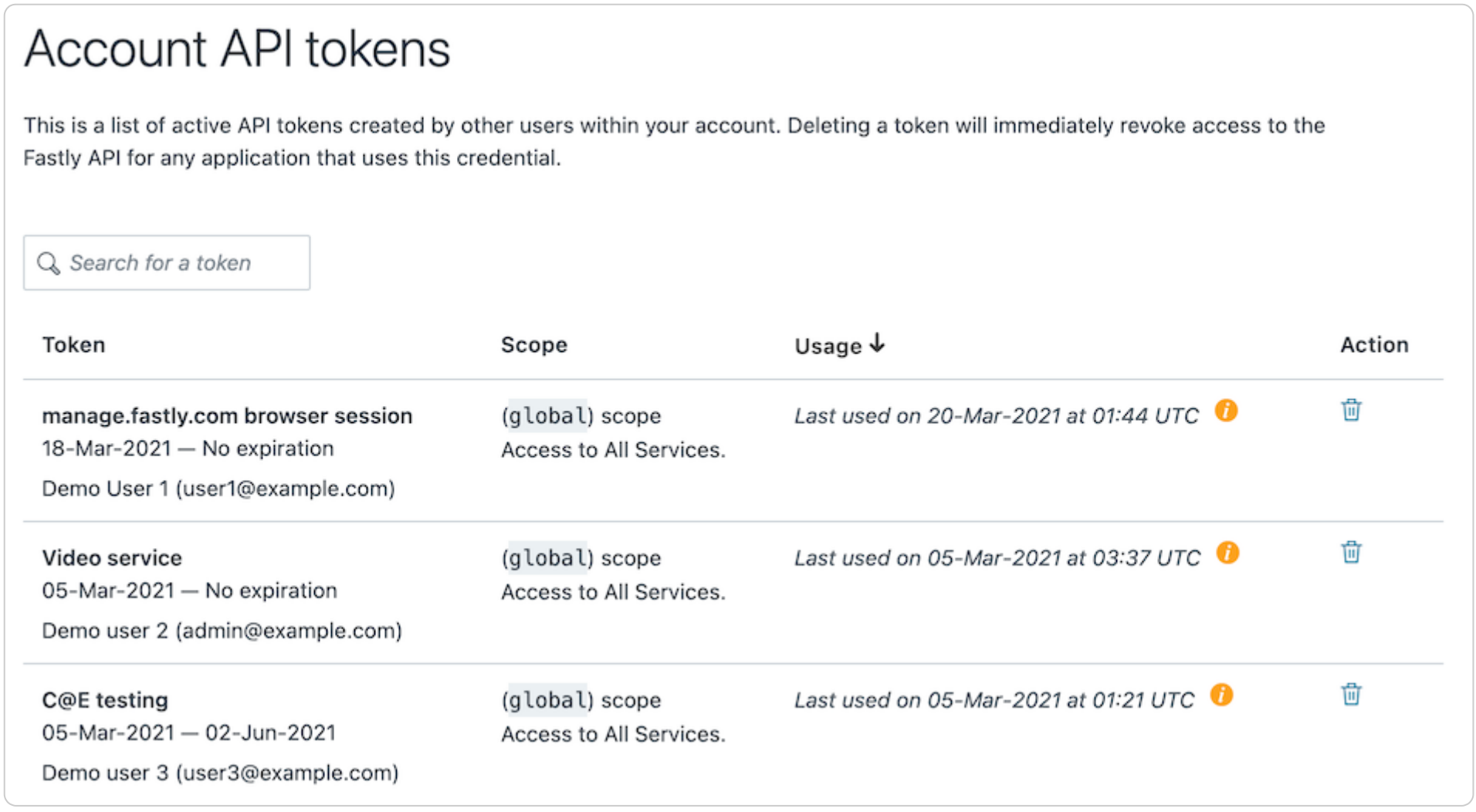
1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Personal API tokens** link. The Personal API tokens page appears with a list of your personal tokens.



Viewing account API tokens

To view account API tokens as a superuser, follow these steps:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Account API tokens** link. The Account API Tokens page appears with a list of tokens associated with your organization's Fastly account.



Deleting API tokens

WARNING

Deleting an API token will break any integration actively using that credential. Verify you have changed the API token for your integrations before proceeding.

Deleting personal API tokens

To delete a personal API token, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Personal API tokens** link. The Personal API Tokens page appears with a list of your personal tokens.
3. Find the API token you want to delete and click the trash icon. A warning message appears.
4. Click the **Delete** button to permanently delete the API token.

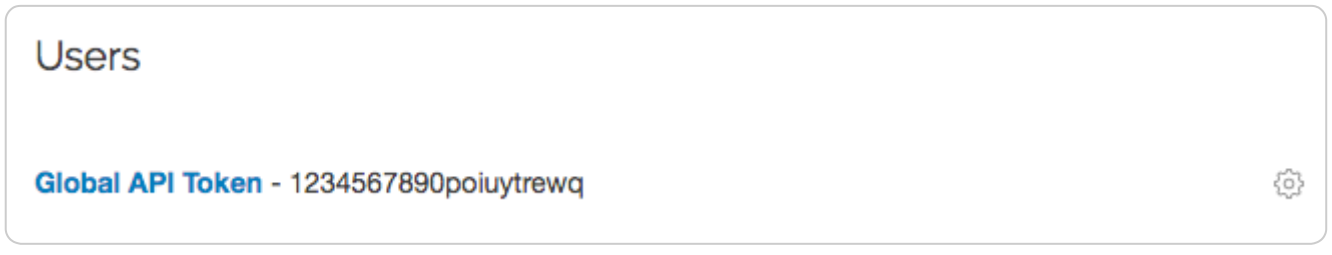
Deleting account API tokens

To delete an account API token or to revoke another user's API token as a superuser, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Account API tokens** link. The Account API Tokens page appears with a list of tokens associated with your organization's Fastly account.
3. Find the API token you want to delete and click the trash icon. A warning message appears.
4. Click the **Delete** button to permanently delete the API token.

Legacy API keys

If you created a Fastly account before May 15th, 2017, you may have used an API key (or multiple API keys) to authenticate API requests. This account-level credential was migrated to a personal API token with a `global` scope and access to all of your services. Because all tokens need to be owned by a user, this credential was assigned to a newly created, synthetic user with the name `Global API Token`.



§

These articles describe Fastly's billing and payment plans and how to make adjustments to your billing information.

https://docs.fastly.com/en/guides/account-info#_billing

Accounts and pricing plans

Last updated: 2021-10-29

https://docs.fastly.com/en/guides/accounts-and-pricing-plans

Fastly offers a variety of account types, which we detail below.

Free trial accounts

We provide several ways to give Fastly a try free of charge. Use a free trial to get hands-on experience exploring the capabilities of our platforms.

Free trials for delivery services

We offer a development free trial that allows you to test our delivery services free of charge by simply signing up. We allow you to test up to $50 of traffic per month for free to ensure everything fits your requirements. You can pay as you go from there by switching to a paid account directly in the web interface. Keep in mind that some add-on options (our TLS certificate options, for example) require you to switch your account to a paid account before that functionality becomes available to you.

Free trials for Compute@Edge services Limited Availability

We offer a free trial of Compute@Edge that allows you to gain experience working with Fastly's serverless edge platform. We allow you to test up to $50 of Compute charges. Once you've signed up for a Fastly account, start your free trial by navigating to the Compute tab in the web interface and then following the on-screen prompts. Use the free trial to test how serverless application concepts created in your non-production environment can be deployed to the Fastly Edge.

Limitations on free trials

Keep in mind the limitations that apply to all free trials:

- Free trial accounts are not designed for use with production traffic or workloads.
- Free trials do not include customer support for Compute@Edge issues, nor are the service availability SLAs applicable. Only paid accounts include this support and the associated SLAs.
- Specific resource allocation limitations apply during your Compute@Edge free trial. Compute@Edge has a lower resource allocation that can only be increased or removed if you upgrade to a paid account, at which point normal limitations and constraints apply. Contact sales@fastly.com for more information.

Paid accounts

You can easily sign up for a paid Fastly account via our web interface or by contacting us at sales@fastly.com.

Signing up for a paid account online

You can use Fastly's delivery services on a month-to-month basis across our global regions. Start by signing up for a free trial, explore a bit, and then, when you're ready to start pushing production traffic our way, upgrade your account directly in the web interface.

Once you switch to a paid account, the developer account trial option disappears and we'll begin billing you automatically at the end of every month using your credit card information. You can estimate your monthly delivery charges by using the pricing estimator on our pricing page. For more detailed information on those monthly charges, including how we measure your usage and when we charge you for it, see our guide to how we calculate your bill.

Contact us at sales@fastly.com for more information if you plan to push at least 2TB of data per month for delivery and require one of our TLS service options, if you plan to push a minimum of 4TB of data per month, or if you want to purchase our Gold or Enterprise support offerings. We also offer solutions targeted to the needs of specific industries.

Other paid accounts

If you signed up for a paid Fastly account via a method other than our web interface, refer to your contract with Fastly for specific details about your services.

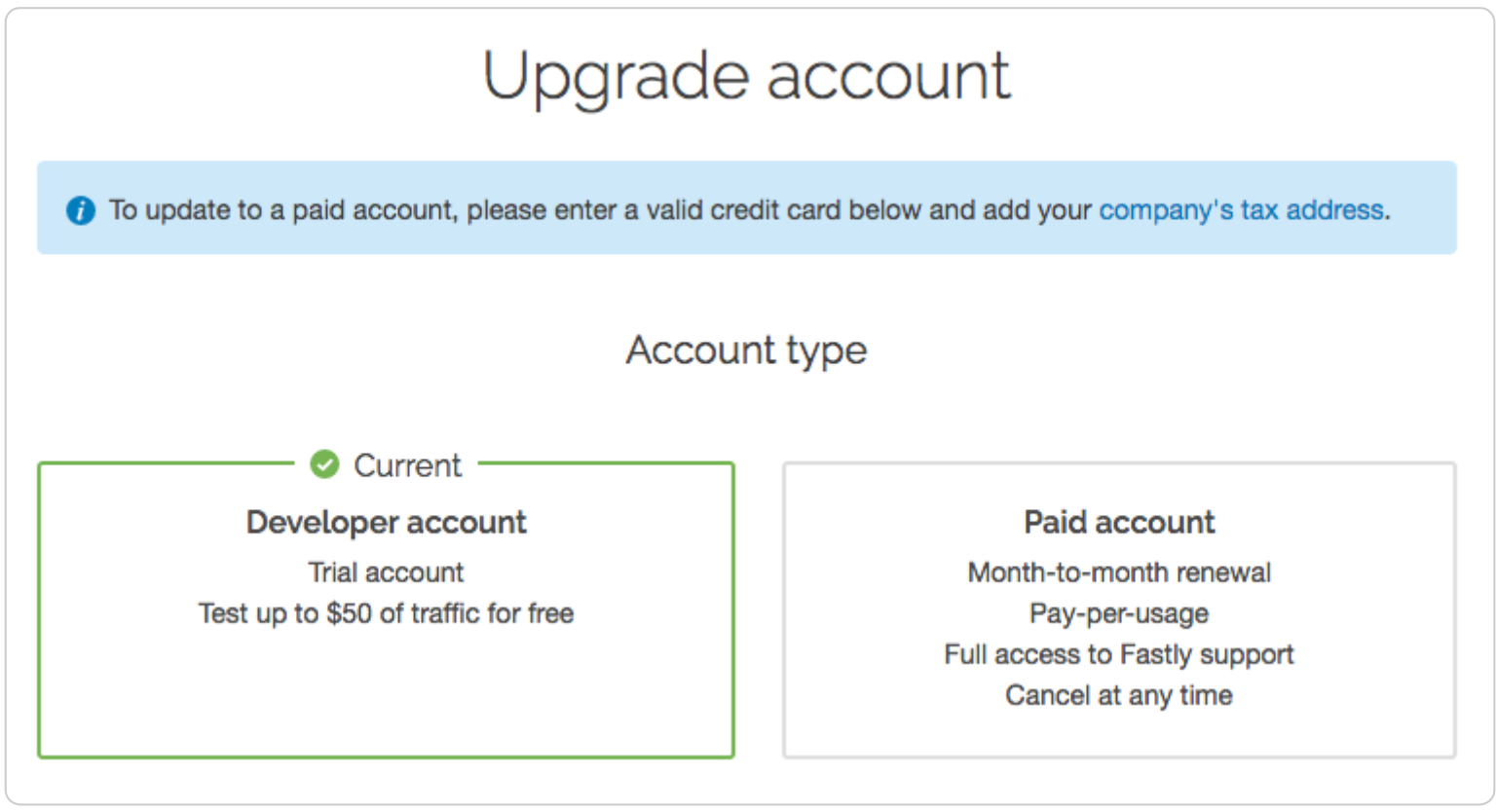
Fastly's open source and non-profit program

We provide free services to members of our open source and nonprofit program. If you are interested in participating in the program, contact us at community@fastly.com to get started.

Upgrading your account

To upgrade your account for Compute@Edge services, contact sales@fastly.com. You can upgrade your account for delivery services directly in the web interface by following these steps:

1. Log in to the Fastly web interface.
2. From the account menu, select **Billing**. Your account's billing information appears.
3. Click the **Upgrade account** link. Information about your plan's current account type appears.



4. Click the **Paid account plan** option.
5. Agree to Fastly's Terms of Service by selecting the **I agree to the terms of service** checkbox.
6. Click the **Upgrade Account** button. The development trial option disappears.

Canceling your account

You can cancel your account at any time. Have your account owner follow these steps:

1. Deactivate and then delete all services on your account.

2. If you've purchased one of Fastly's hosted or managed certificate options, contact support@fastly.com to begin the process of deleting your certificates.
3. From the user menu, click **Account**. Your account information appears.
4. In the **Company settings** area, click the **Cancel Account** button. A confirmation window appears.
5. In the **Your password** field of the confirmation window, enter the password associated with your account and click **Confirm and Cancel**.

After your account is canceled, you'll be billed for any outstanding charges accrued through the day you canceled. For questions about your final billing statement, contact our billing team for assistance. If you decide at a later date to reactivate your account, contact support@fastly.com and request reactivation.

How we calculate your bill

Last updated: 2022-02-16

https://docs.fastly.com/en/guides/how-we-calculate-your-bill

We bill you monthly according to that month's use of Fastly's services. Your bill is affected by a combination of things including the actual traffic Fastly has served on your behalf, the products you've purchased, the features you've enabled, and the specific configuration settings you've chosen (like enabling shielding or compression).

We charge for egress traffic from our POPs, including traffic served to end users and, if shielding is enabled, traffic served from the shield POP to other POPs. Specifically, we charge for each response and for the size of the response (which includes the header and body). Each response is billed as a single request, and the response size in bytes is billed as bandwidth. We charge for bandwidth and requests for content delivered to clients from the CDN and for bandwidth for traffic sent from the CDN to our customers' origins.

> **NOTE**
>
> If you're using Anycast IP addresses, these IPs use our global network and will route a request to the nearest POP located in a billing region that may charge a higher rate. Our billing regions can be found on the Fastly Pricing page. We announce new billing regions regularly via our network status page.

Charges for any options you've chosen are applied in addition to the bandwidth and request usage we charge for normal content delivery and streaming. In accordance with local laws, Fastly may also collect sales tax based on your shipping or billing address on file.

About the measurements and calculations we use

We measure months according to Coordinated Universal Time (UTC). For usage-based charges, bandwidth is recorded in bytes and presented in gigabytes (GB), and requests are recorded individually and presented in units of 10,000.

Fastly uses The International System of Units (SI Units) to measure bandwidth. In our calculations, 1 gigabyte (GB) = 10^9 (1,000,000,000) bytes, 1 terabyte (TB) = 10^{12} bytes (or 1,000 GB), and 1 petabyte (PB) = 10^{15} bytes (or 1,000 TB). Your invoice shows your usage and that matches the usage shown on the Stats page.

About the monthly minimum charges

We bill a minimum of $50 per month so we can fully support all of our customers. This is the minimum price you'll pay in any month once you've completed your testing trials.

For example, say that you're done testing Fastly's services and you've begun to push production-level traffic through Fastly. If most of your site's traffic for the current month is in North America and Europe and your site uses 10GB of traffic over 10 million requests, the combined bandwidth and request charges would be $8.70 for the month. Because this amount falls below the $50 monthly minimum, we would charge you $50 for that month, not $8.70.

Bandwidth and request prices for some billing regions are slightly higher. If most of your site's traffic were in these other regions instead, then at the above traffic levels your bandwidth and request usage charges would still fall below the monthly minimum and we would charge you $50 for that month.

NOTE

If you're using Fastly for content delivery via Heroku's cloud development services, see Fastly's Heroku add-ons pricing plan for additional details.

When we charge you for Fastly services

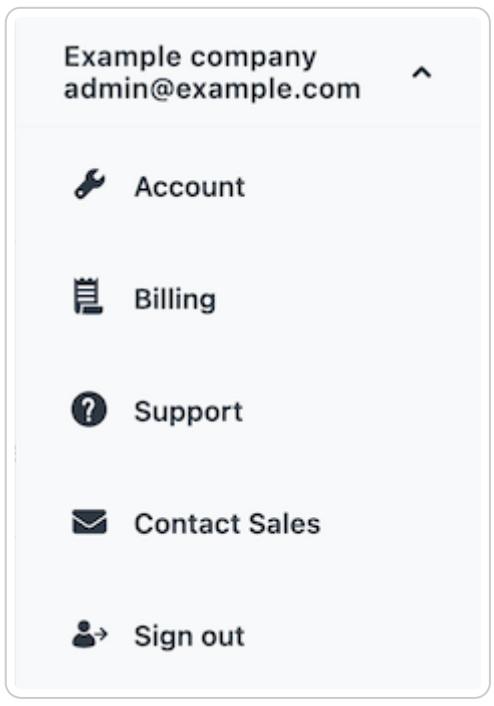
Fastly bills in arrears, not in advance, meaning that we bill you for services after you've used them, not before. For example, if you sign up for and start using Fastly services in January, the bill you receive in February reflects January's charges and services, your March bill reflects February's charges and services, and so forth.

How account cancellation affects your bill

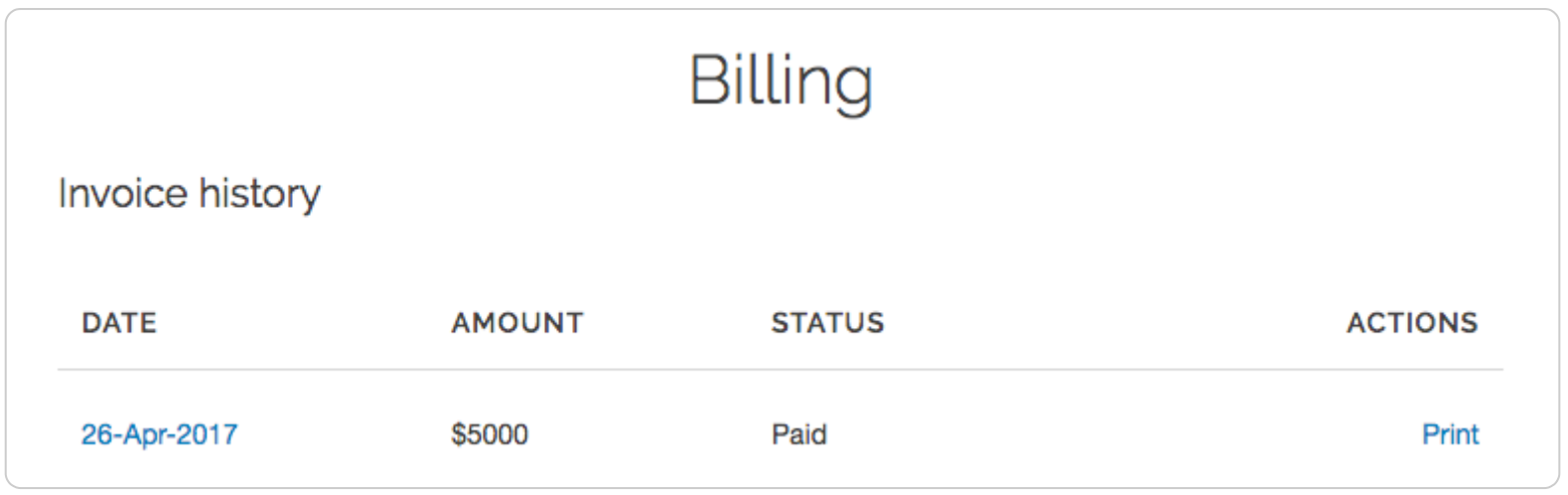
If you ever cancel your account, you'll be billed for any outstanding charges accrued through the day you canceled, or at least the monthly minimum, whichever amount is greater.

Reviewing the charges to your account

If you've been assigned a superuser or billing role, you can review your account use and the associated charges via the Billing page in the Fastly web interface. Access billing information by selecting Billing from the user menu at the top right of any page.



By default, the current balance for your account appears, followed by the invoice history.



Clicking on the linked date of any invoice displays a summary of charges for that month.

August 2019

Summary		
Bandwidth	20,000.10 GB	$3,100.01
Requests	49,532.65	$415.80
Wildcard TLS Certificate		$275
Customer Certificate Hosting Service		$600
Professional Services		$1000
Grand total		**$5,390**

The billing invoice summary includes the overall bandwidth you used and the associated charges, followed by the charges you incurred for requests. The bottom of the summary displays the grand total dollar amount owed for the dated month.

Below the month's summary on the invoice, we include regional bandwidth and request details.

United States

Bandwidth

TIER	PRICE	UNITS	AMOUNT
North America Bandwidth (10,000 gigabytes @ $.12)	$0.12 / GB	10,000.0	**$1,200.00**

Requests

TIER	PRICE	UNITS	AMOUNT
North America Requests (10,000 units @ $.0075)	$0.0075 / 10K	10,000.0	**$75.00**
North America Requests (10,000 units @ $.0075)	$0.0075 / 10K	10,000.0	**$75.00**
Region total			**$1,350**

The bottom of each regional details section includes the total charge for bandwidth and requests for that region alone for the dated month.

> **TIP**
>
> A breakdown of billing charges per service is not available at this time. Our historical stats API, however, provides data on unrated request and bandwidth used by a service, aggregated by billing region.

Printing account use details

You can print account use details for any month by finding that month in the invoice history and clicking **Print** in the **Actions** column for that month.

Estimating your month-to-date bill

You can estimate your month-to-date (MTD) bill via the web interface or via the API.

Via the web interface

To view an estimated report of account usage for the current partial month, use any standard web browser to log in to your Fastly account and navigate to:

https://manage.fastly.com/account/invoices/month-to-date

> **NOTE**
>
> A small number of billing plans cannot be calculated month-to-date and only include an end-of-month generated invoice. If you have one of these billing plans, the web interface will clearly tell you that you can't see the report due to your account's status.

Via the API

As part of our API, a billing endpoint exists to generate a report of your usage for the current partial month (known as month-to-date, or MTD). Full details of this endpoint's output format can be found in our Billing API documentation. Generating a report via API usually takes only a few seconds, but can potentially take up to 60 seconds. During this time, the API call will return an HTTP `202 Accepted` response.

```
{
  "data" : {
     "attributes" : {
        "status" : "Pending: waiting for another process"
     },
     "id" : "MTD_2i0wWA8Zvo6uUpmATZYuQi",
     "type" : "mtd-invoice-pending"
  }
}
```

Paying your bill

Last updated: 2020-07-24

https://docs.fastly.com/en/guides/paying-your-bill

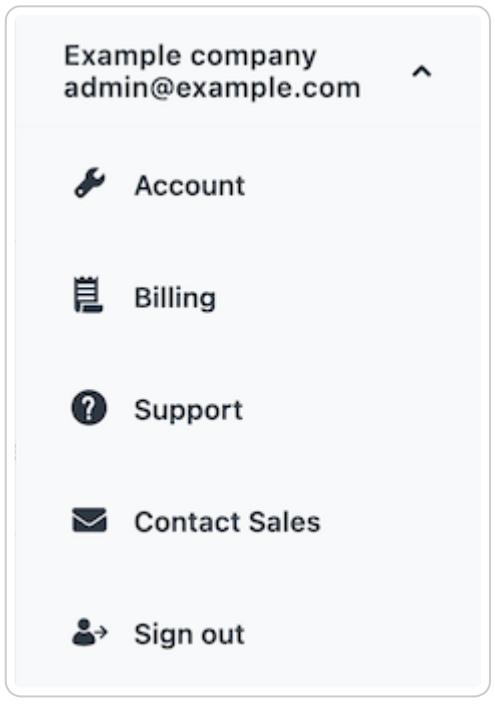
At the end of each month, your account's billing contact will be sent an email summarizing your current usage levels and the charges your account incurred for the month. The email contains a link to an online copy of the related invoice.

You'll need both a valid credit card and current billing address when you switch to a paid, month-to-month account. Once your invoice gets generated, your credit card is automatically charged for the full, outstanding balance.

Changing your credit card information

To change the information for the credit card we use for automatic billing, follow the steps below:

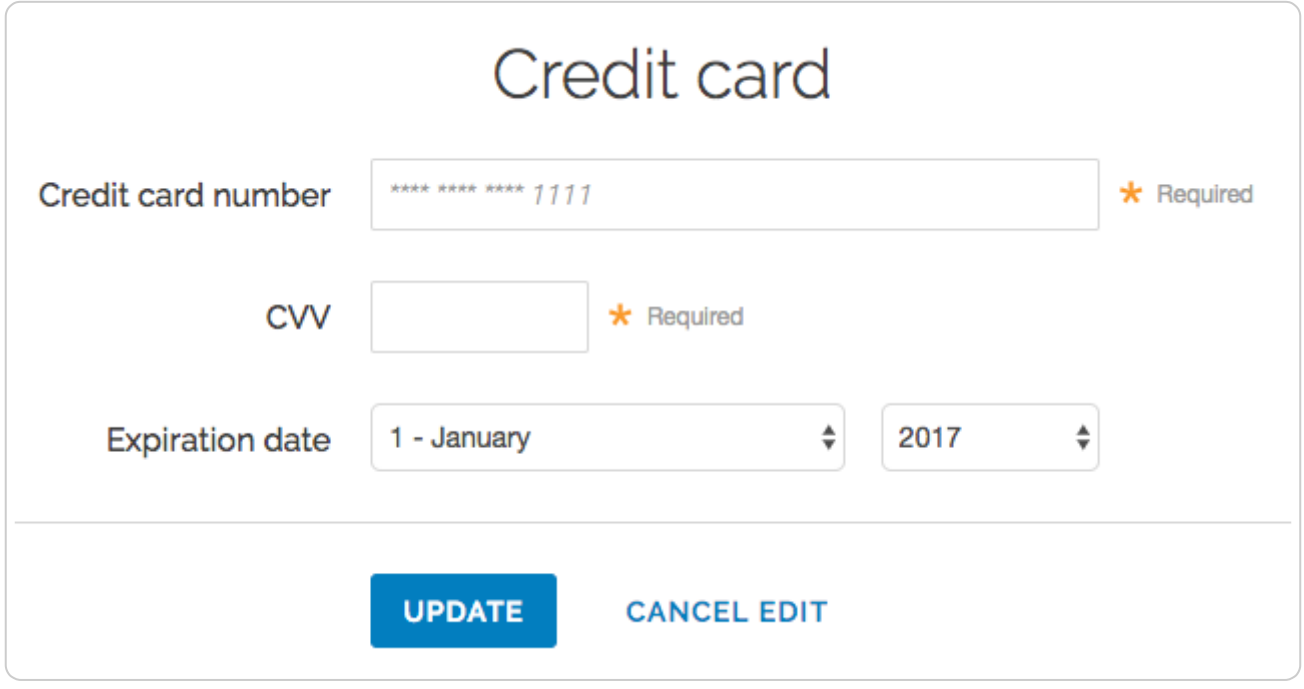
1. Log in to the Fastly web interface.
2. From the user menu, select **Billing**.



3. Click the **Credit card** link. The Credit card page appears.

4. Click **Edit**. Details appear for the credit card you have on file with Fastly.

5. Make any necessary changes to the credit card information in the fields provided.



6. Click **Update** to save your credit card information.

> **TIP**
>
> Fastly never sees your credit card number. All transactions are handled by our fully PCI compliant payment gateway and their privacy policy can be found at https://www.fisglobal.com/en/privacy.

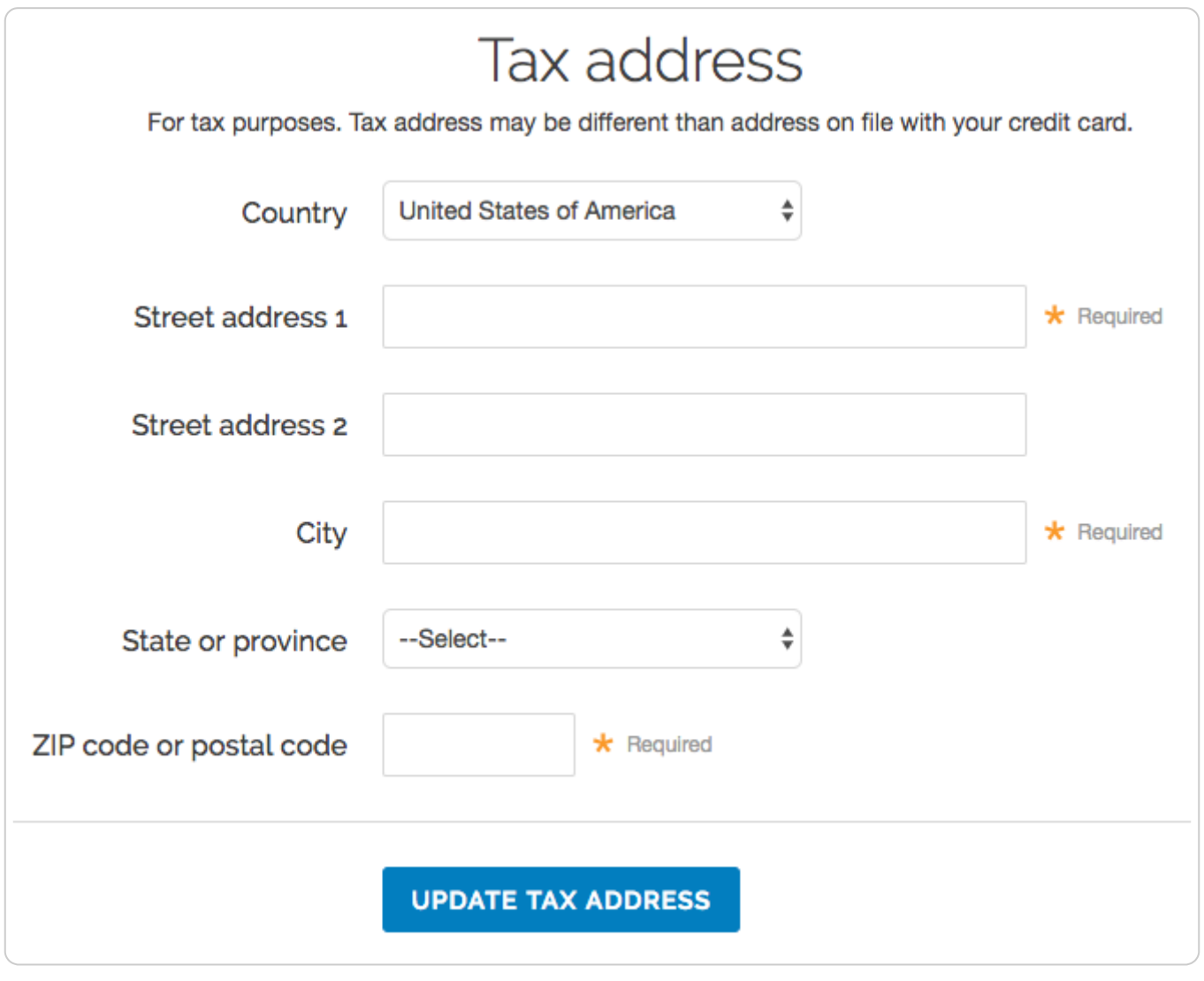
Changing your tax or billing address

To change your tax or billing address, follow the steps below:

1. Log in to the Fastly web interface.

2. From the user menu, select **Billing**.

3. Click the **Tax address** link and enter the tax address information you use in the fields provided.



4. Click the **Update Tax Address** button to save the tax address information.

Changing who receives your bill

By default, your account owner is considered your billing contact and will receive your bill for Fastly services. To change who receives your bill or to add multiple email addresses for several billing contacts, contact billing@fastly.com with the addresses you'd like to send invoices to.

IMPORTANT

Invoices are only sent to the email addresses of the account owner or the billing contact. Invoices are not sent to every user assigned a billing role.

---

§

These articles describe how to manage users with permission to access to your account.

https://docs.fastly.com/en/guides/account-info#_user-access-and-control

Adding and deleting user accounts

Last updated: 2021-09-01

https://docs.fastly.com/en/guides/adding-and-deleting-user-accounts

Fastly allows you to add users to an account via invitation, assigning them different roles and permissions as appropriate. You can delete user accounts when you no longer want someone to have access.

IMPORTANT

You must be assigned the role of superuser to add users to or delete users from an account.

Adding account users

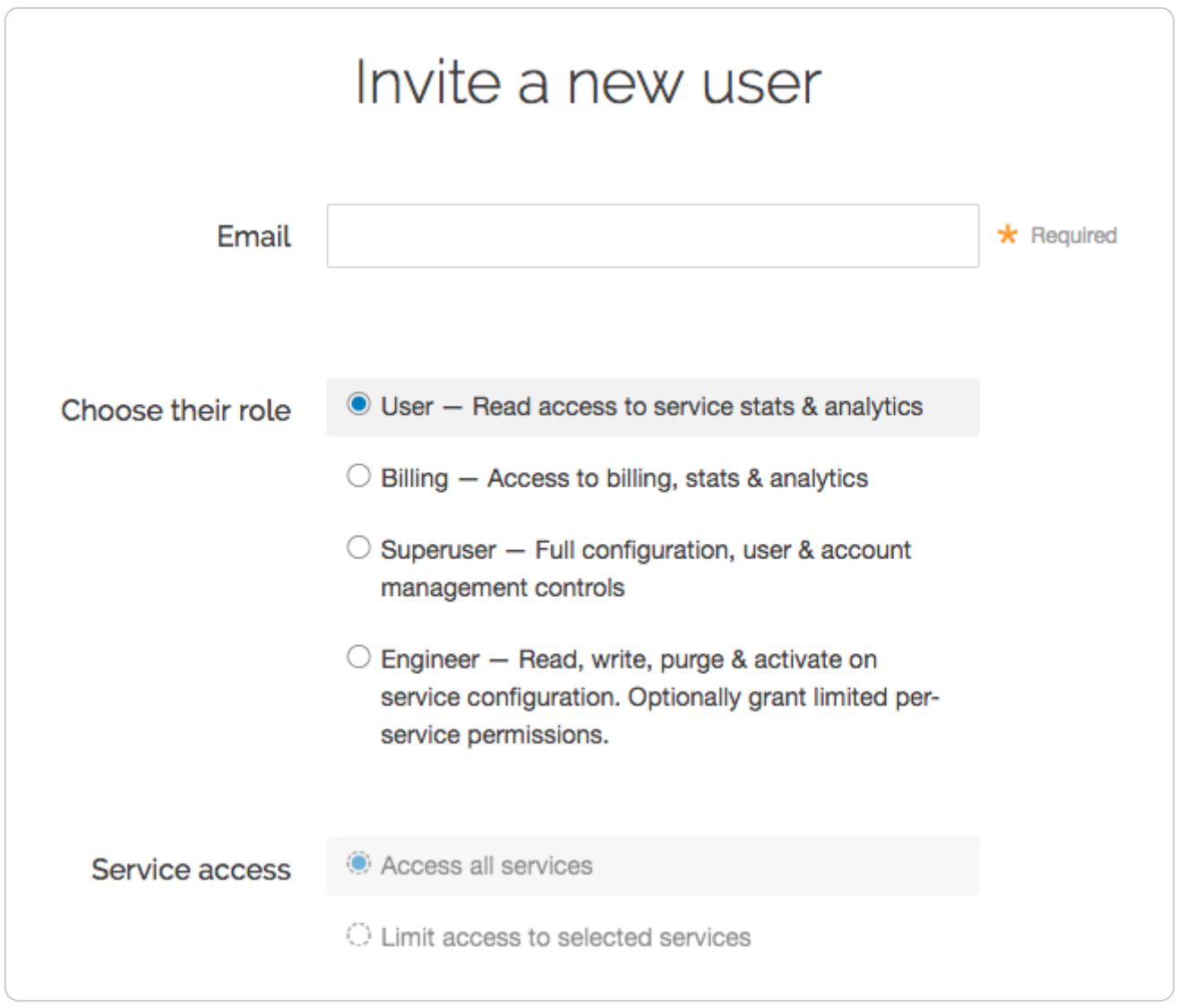
TIP

Adding a new user to make them the billing contact for an account? Follow our billing contact instructions instead.

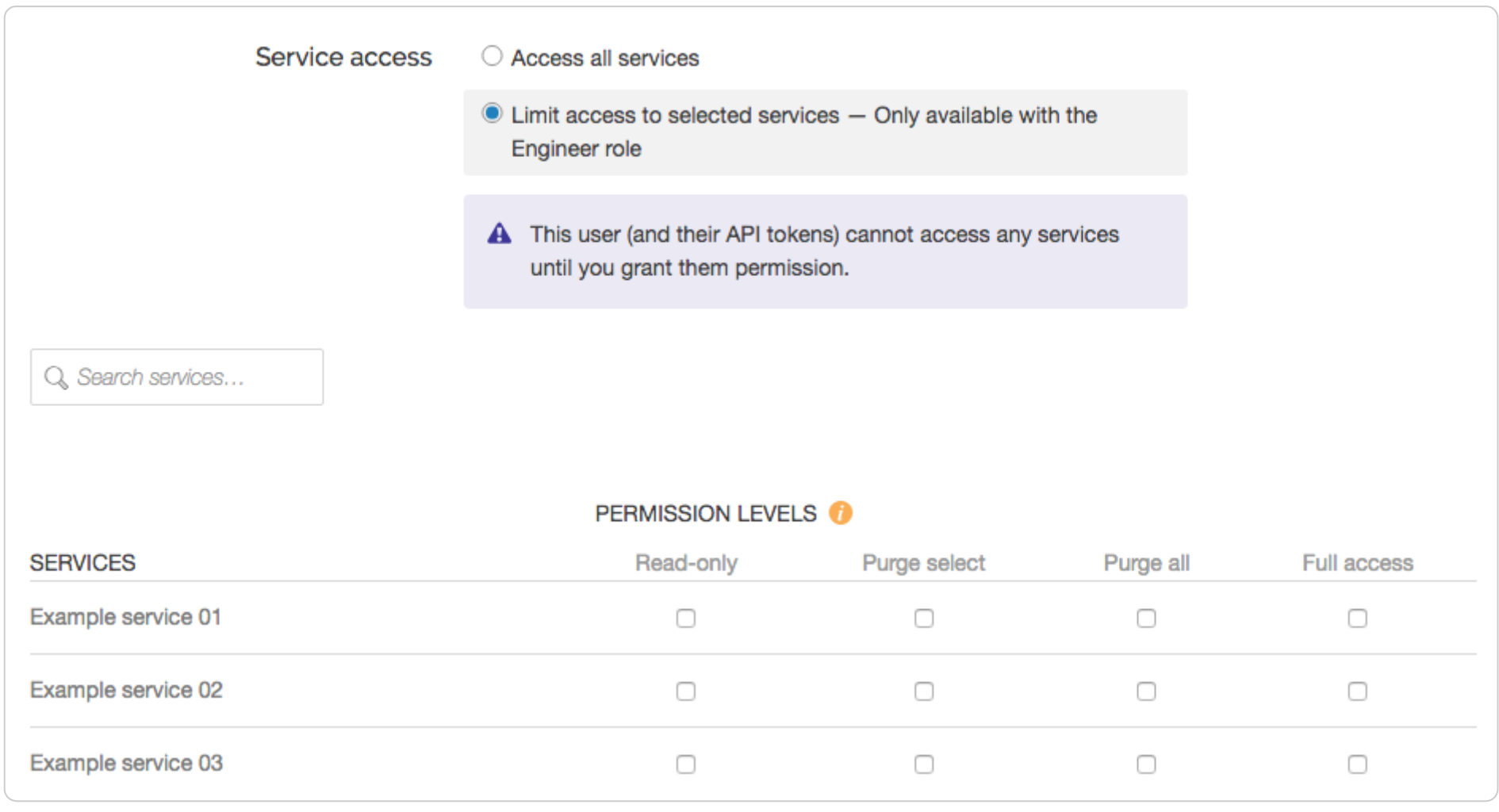
Adding a user to an account

To add a Fastly user to an account, send them an invitation to join following the steps below:

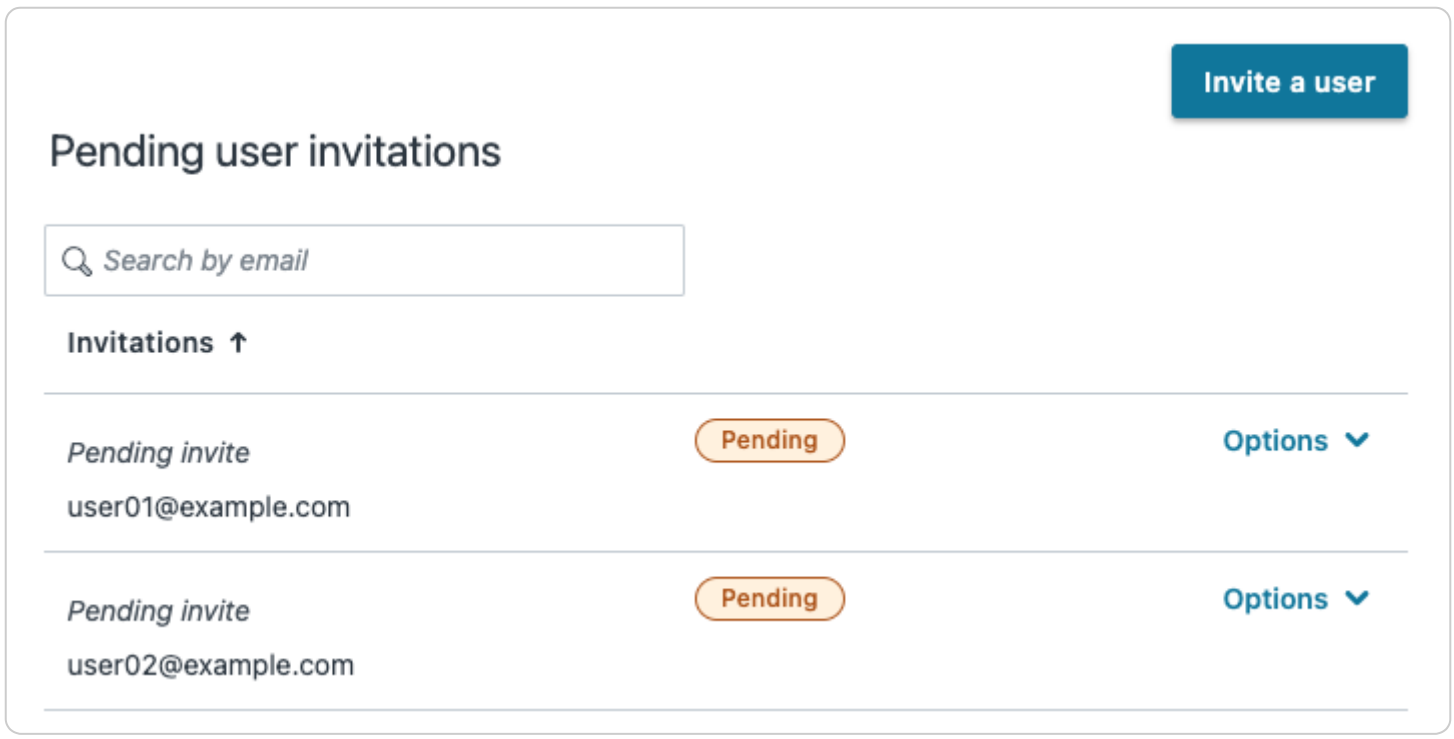
1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link.
3. In the **Pending user invitations** area, click the **Invite a user** button. The Invite a new user page appears.



4. In the **Email** field, enter the email address of the user to invite.
5. From the **Choose their role** options, select the role to assign the user once they accept the invitation.
6. From the **Service access** controls, optionally select **Limit access to selected services** to limit access to selected services for users assigned the role of engineer.



7. If you've chosen to limit access to selected services for a user assigned the role of engineer, select the specific permission levels for each service associated with the account.
8. Click the **Invite** button to send an invitation to the email you specified. The email address of the user you invited appears in the Pending user invitations area and remains there until the invitation is accepted or you delete the invitation.



Resending an account invitation

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link.
3. In the **Pending user invitations** area, find the email address associated with the original invitation.
4. From the **Options** menu, select **Resend email**.

Deleting an account invitation

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link.
3. In the **Pending user invitations** area, find the email address associated with the original invitation.
4. From the **Options** menu, select **Delete invitation**.

Deleting account users

> **TIP**
>
> Deleting the owner of the account? Be sure to transfer ownership first.

To delete a user from an account, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link.
3. In the **Active users** area, find the name or email of the user to delete.
4. Click the **Options** menu to the right of the user to be deleted, then select **Delete user** from the menu that appears. A confirmation window appears.
5. If the user has active API tokens associated with their account, click the **Review this user's API tokens** link to manually review and revoke them. Alternatively, select the checkbox to automatically revoke all of the user's API tokens and delete the user.

   > **WARNING**
   >
   > Deleting an API token will break any integration actively using that credential. Verify you have changed the API token for your integrations before proceeding.

6. Click **Confirm and delete**.

Configuring user roles and permissions

Last updated: 2022-02-09

https://docs.fastly.com/en/guides/configuring-user-roles-and-permissions

Your Fastly account can be managed by multiple users through the role-based access controls in the web interface. These controls allow you to manage the scope of a user's service access and their specific permission levels for that service access, all based on the role assigned to them.

TIP

The roles, service access, and permission levels you assign to users do not affect their ability to submit requests to Fastly Customer Support.

User roles and what they can do

Fastly allows you to assign one of four different roles to each user allowed access to your account. In general, the abilities granted to each role are as follows:

- **User.** View stats and analytics for all services on an account.
- **Billing.** View billing information about an account. View stats and analytics information for all services on an account.
- **Engineer.** View configuration details, issue purge requests, and make configuration changes, including activating new service versions. Some of these abilities may be restricted on a per service basis.
- **Superuser.** Full account access, including service configuration, user access and control, and account management capabilities for an account. Superusers cannot close or cancel an account unless they are also the account owner.

Abilities granted to user roles are selective, not additive. Specifically, each role has access to the following functionality, if at all:

	User	Billing	Engineer	Superuser
Stats dashboards				
View historical stats	Full access	Full access	Full access	Full access
View real-time service stats	Full access	Full access	Full access	Full access
Configure				
View service configurations	Potential restrictions	Full access	Full access	Full access
Create services	No access	No access	Full access	Full access
Delete services	No access	No access	Potential restrictions	Full access
Configure services	No access	No access	Potential restrictions	Full access
Compare service versions	No access	No access	Potential restrictions	Full access
Deactivate services	No access	No access	Potential restrictions	Full access
Purge	No access	No access	Potential restrictions	Full access

View and download generated VCL	No access	No access	Potential restrictions	Full access
Customize VCL	No access	No access	Potential restrictions	Full access
TLS management	Potential restrictions	Potential restrictions	Potential restrictions	Full access
Account & Organization				
Update personal profile settings	Full access	Full access	Full access	Full access
Update company settings	No access	No access	No access	Full access
Company contacts	No access	No access	Read only	Full access
Invite all new user roles	No access	No access	No access	Full access
Invite new engineer and user roles (API only)	No access	No access	Full access	No access
Assign and change roles and permissions	No access	No access	No access	Full access
Issue password reset emails	No access	No access	No access	Full access
Delete account users	No access	No access	No access	Full access
Enable and disable personal 2FA	Full access	Full access	Full access	Full access
Enable and disable company-wide 2FA	No access	No access	No access	Full access
Manage personal API tokens	Full access	Full access	Full access	Full access
Revoke account API tokens	No access	No access	No access	Full access
Billing				
View invoices	No access	Full access	No access	Full access
View billing history	No access	Full access	No access	Full access
Pay bills	No access	Full access	No access	Full access
Update credit card info	No access	Full access	No access	Full access
Change account type	No access	Full access	No access	Full access

Service access and permission levels

All user roles grant access by default to every service on an account now and in the future. The engineer role is unique, however, in that you can change that default. Superusers can limit an engineer's access to specific services and can control the level of permissions on each of those services as follows:

- **Read-only.** Allows an engineer to view a specific service's configuration but does not allow them to issue purge requests for that service nor make changes to its configuration.
- **Purge select.** Allows an engineer to view a specific service's configuration and also allows them to issue purge requests for that service via URL or surrogate key. They cannot use the purge all function on the service, nor can they make configuration changes to that service.
- **Purge all.** Allows an engineer to view a specific service's configuration and issue purge requests via URL, surrogate key, or the purge all function. They cannot, however, make configuration changes to that service.
- **Full access.** Allows an engineer full access to a specific service, including permission to issue purge requests via any method on that service. They can make configuration changes to that service and can activate new versions of it at will.

Permission levels are additive. Each level includes the previous level's permissions. When new services are added to an account by a superuser, engineers with limited access to services will not be granted permissions to those services until a superuser specifically grants those permission levels manually.

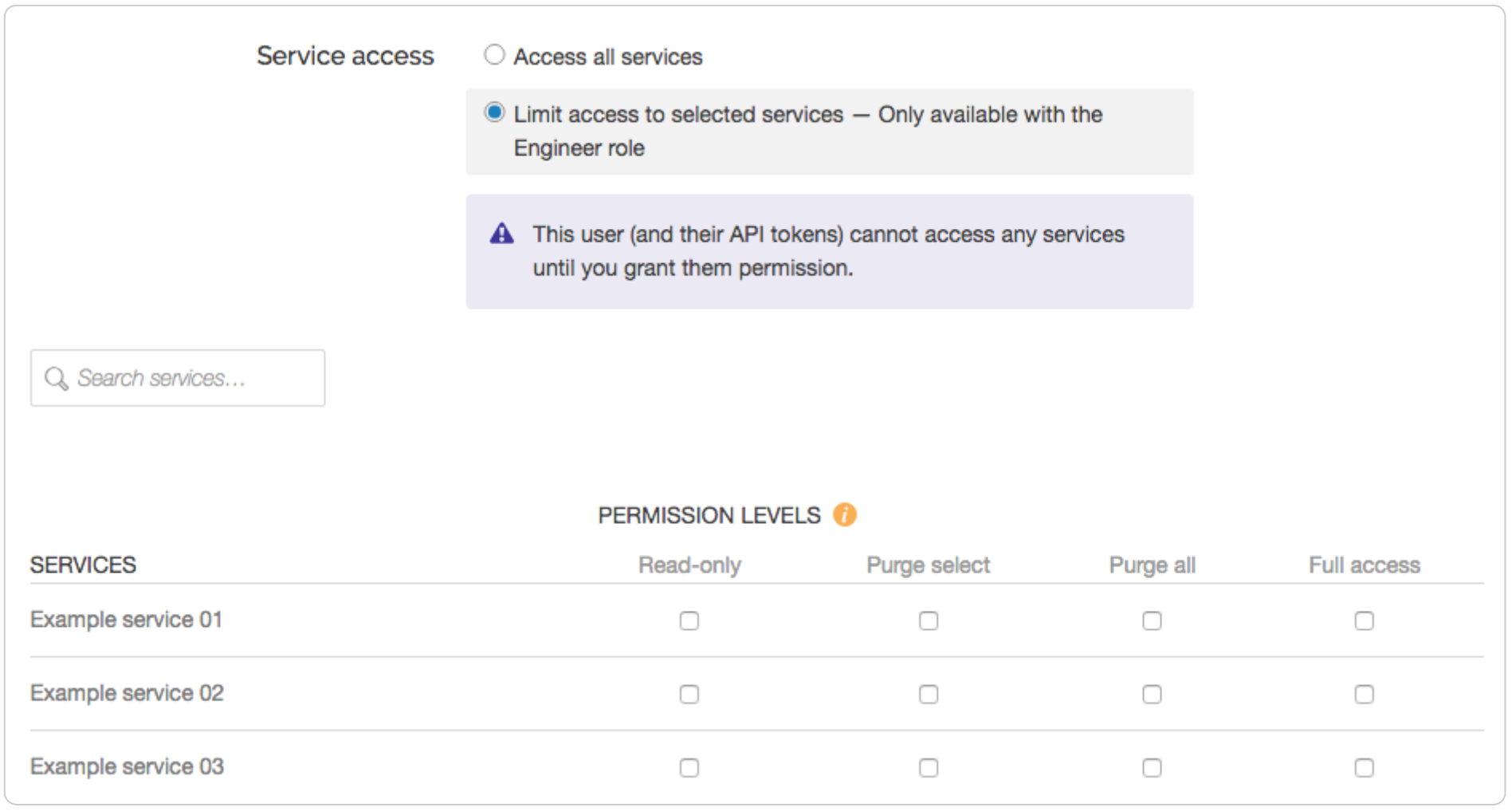
Users assigned the role of `engineer` can create new services (this is especially useful for learning about configuration options without affecting production services). By default, an engineer will automatically have full access to any service they create until their permission levels on that service are modified by an account superuser.

Changing user roles and access permissions for existing users

Users assigned the superuser role can change the role, service access, or permission levels for any existing user on your account. Plan your changes carefully.

> **WARNING**
>
> Role, service access, and permission level changes for existing users apply instantly and get saved automatically.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link. The User management page appears.
3. In the **Active users** area, click the **Options** menu next to a user name and then select **Access controls** from the menu that appears. The Edit access control page appears for the selected user.
4. From the **Choose their role** choices, optionally select a new role for the user.
5. Optionally, check the **TLS management** box to grant TLS configuration access to a user. Users with the role of superuser have this permission by default.
6. From the **Service access** controls, optionally select **Limit access to selected services** to limit access to selected services for users assigned the role of engineer.



7. If you've limited access to selected services for a user assigned the role of engineer, select the specific permission levels for each service associated with the account.

8. Click **Update**. The user's role and permission levels will be changed accordingly.

Account ownership

We assign the special role of *owner* to the first user who signs up for an account for your organization and we automatically assign that owner the superuser role. Any superuser on your account can change the permissions on an owner role or transfer ownership via the Company settings, which are accessible from the Account controls of the web interface.

Account owners typically serve as the primary point of contact for billing purposes. Invoices are sent to them, but if a specific billing contact has been defined for an account, invoices go to that contact instead. In addition, accounts can only be canceled by owners.

Name and password changes

Last updated: 2022-02-25

https://docs.fastly.com/en/guides/email-and-password-changes

The Fastly web interface allows you to change the name and password currently associated with your account.

Changing your name

Follow these instructions to change the name currently associated with your account:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.

2. Click the **Your profile** link. The Your profile page appears.

3. In the **Name** field, enter your name.

4. Click **Update Profile** to save the changes.

Changing your password

Follow these instructions to change the password currently associated with your account:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.

2. Click the **Change password** link. The Change password page appears.
3. Fill out the page as follows:
   - In the **Current password** field, enter your existing password.
   - In the **New password** field, enter the new password.
   - In the **Confirm password** field, enter the new password a second time.
4. Click **Change Password** to save the changes.

Changing someone else's name

If you've been assigned the role of superuser, you can change the name of any user currently associated with your company.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link. The User management page appears.
3. In the **Users** area, find the appropriate user and then click the gear icon next to their name.
4. From the menu that appears, select **Edit user**. The Edit user window appears.
5. In the **Name** field, enter the updated name.
6. Click **Update** to save the changes.

Resetting someone else's password

If you've been assigned the role of superuser, you can send a password reset email to any user currently associated with your company.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link. The User management page appears.
3. In the **Users** area, find the appropriate user and then click the **Options** menu next to their name.
4. From the Options menu, select **Send password reset**. The Send password reset window appears.
5. Verify the email address of the user, and then click **Confirm and send** to save the changes. Password reset instructions are emailed to the user.

Password requirements

When choosing a password keep in mind that it must:

- be at least 7 characters long
- contain at least one letter and one number

In addition, passwords cannot solely contain:

- sequences of letters or numbers (e.g., `12345678`, `abcdefg`)
- repeated characters (e.g., `222222`, `aaaaaa`)
- adjacent key placements on a standard keyboard (e.g., `QWERTY`)

The system will prevent you from choosing a password that:

- matches any of your four previous passwords
- matches commonly used passwords (e.g., `password123`, `changeme`)
- uses popular dictionary words in passwords less than 16 characters (e.g., `batterystaple`)
- matches your user name or your email address

Managing multiple accounts

Last updated: 2020-11-09

https://docs.fastly.com/en/guides/managing-multiple-accounts

Fastly's multi account user access feature allows a single user to manage multiple customer accounts. If you've been invited to more than one customer account, you can quickly switch between accounts without logging out of the Fastly web application. The multi account user access feature works with single sign-on (SSO) and two-factor authentication.

NOTE

Because API tokens are not scoped across customer accounts, you'll need to create a separate API token for each customer account. See the using API tokens section for more information.

Managing users on your account

Users who have been assigned the role of superuser can invite existing Fastly users to an account.

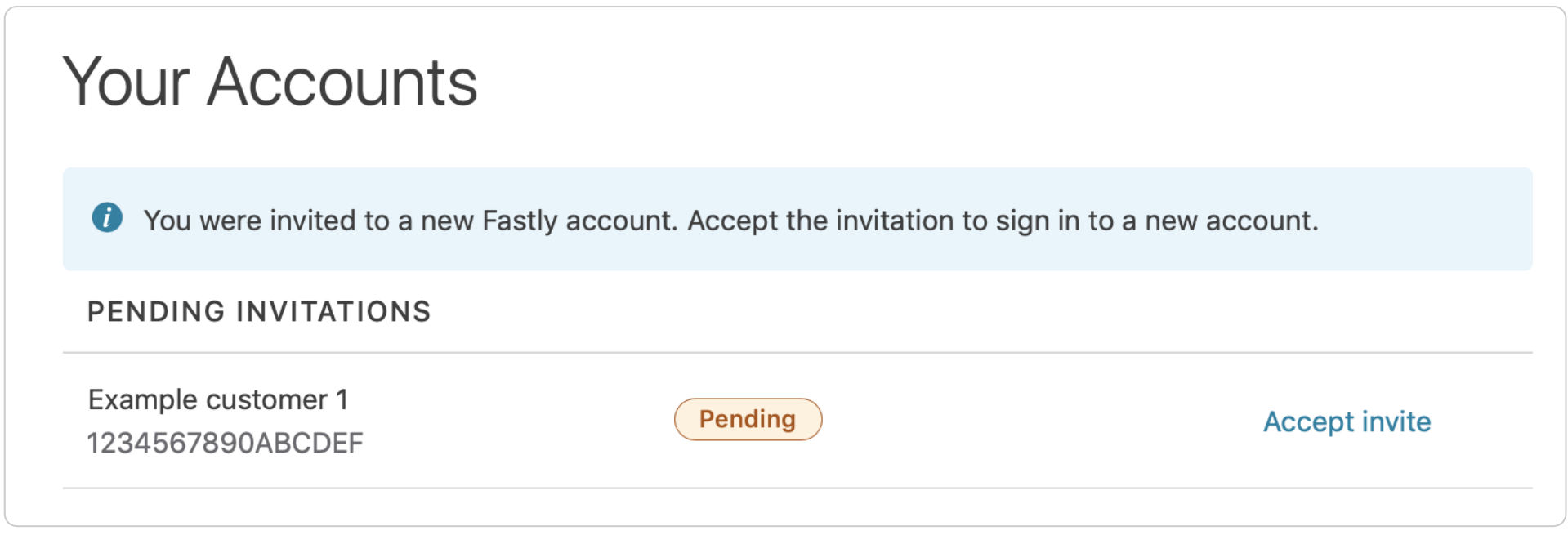
- To send a new or existing Fastly user an invitation to join your account, follow our instructions for adding a new user to your account.
- To delete a user from your account, follow our instructions for deleting account users.

Accepting an invitation to join an account

If you've been invited to a customer account and you don't have a Fastly user account, follow the instructions sent via email to create a user account and accept the invitation.

If you're an existing user who already has a Fastly user account, follow the steps below:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Your Accounts** link. The Your Accounts page appears.

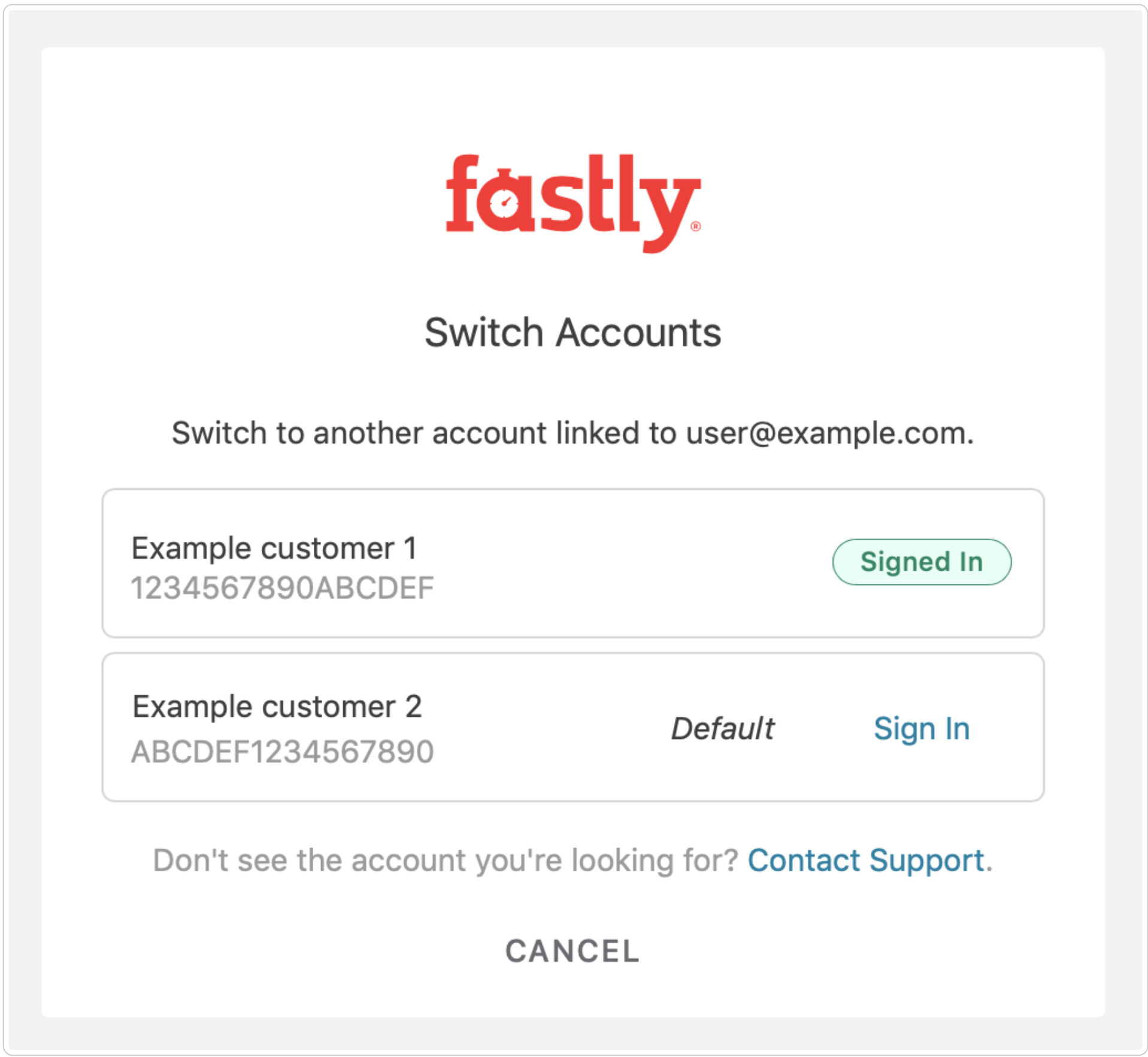


3. Find the invitation in the list and click the **Accept invite** link.

Switching accounts

Follow the steps below to switch between the customer accounts you can access:

1. Log in to the Fastly web interface and click the **Switch Accounts** link from the user menu. The Switch Accounts page appears.



2. Click the **Sign in** link next to the service you want to log in to.

3. If prompted, enter your password.

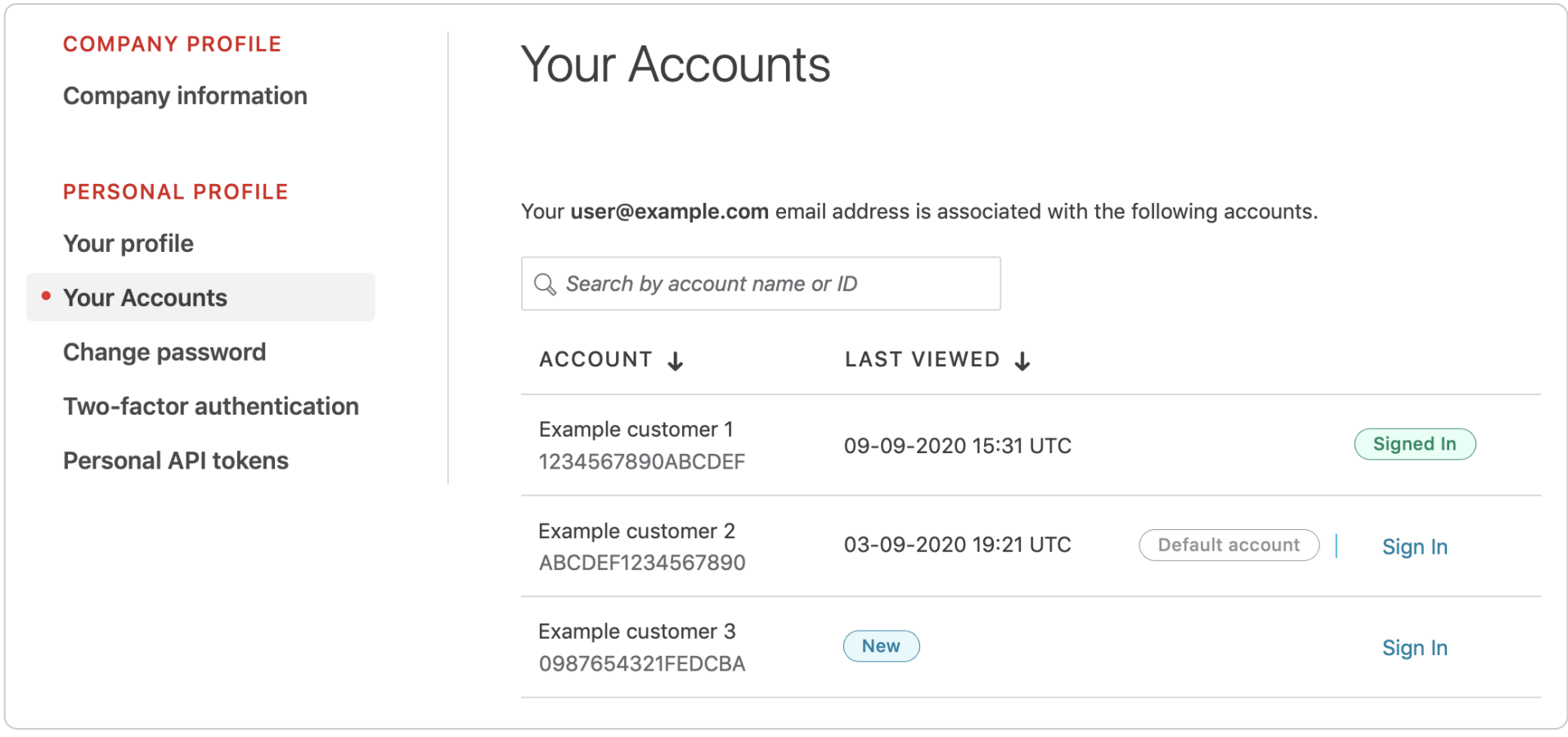
4. If you have two-factor authentication enabled for your account, enter the time-based authentication code from your mobile device, then click the **Sign in** button.

Viewing the accounts you can access

Follow the steps below to view the customer accounts you can access:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.

2. Click the **Your Accounts** link. The Your Accounts page appears with a list of the customer accounts you can access.



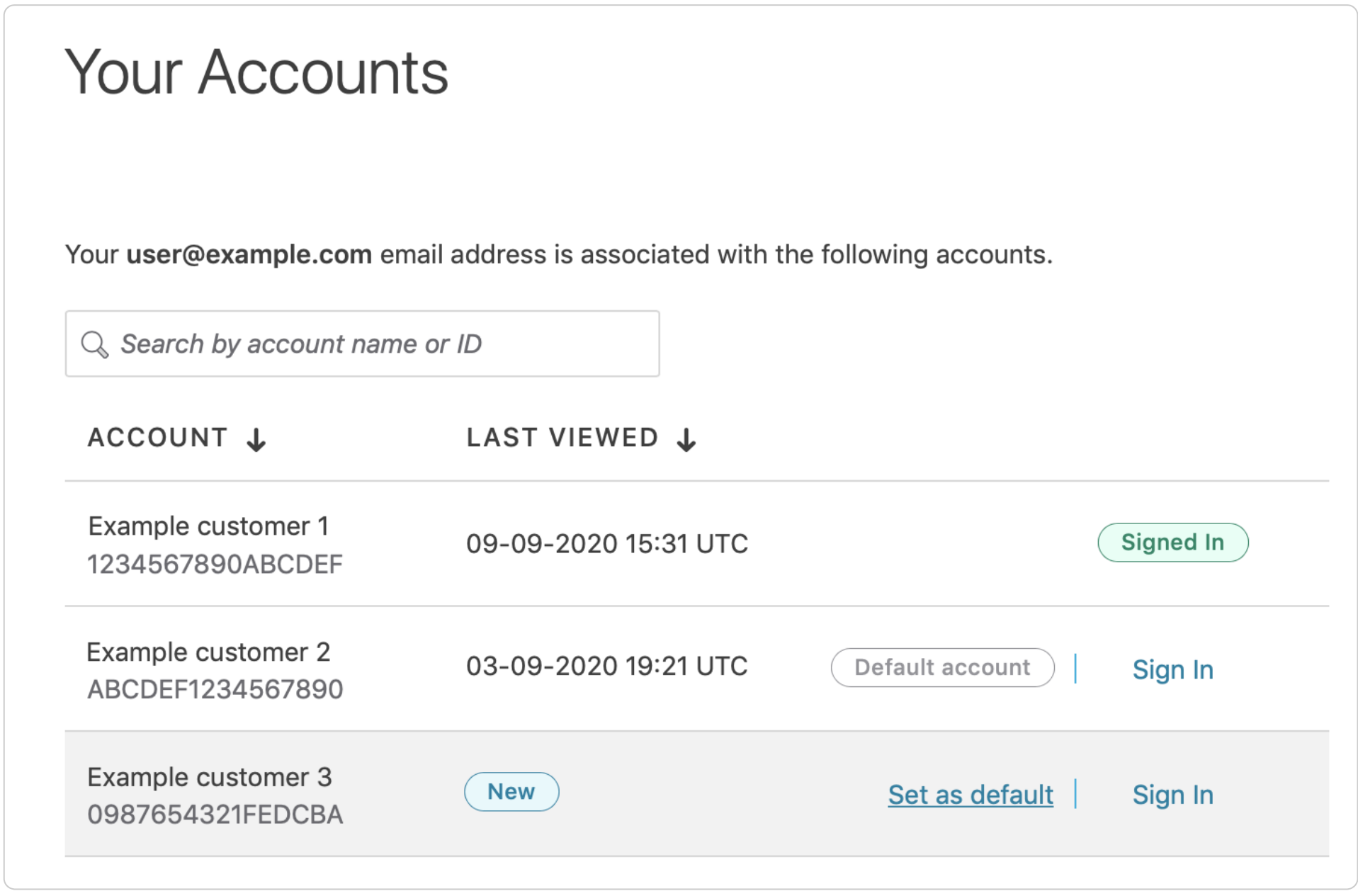
Setting a default account

You can set a default customer account that appears automatically when you log in to the Fastly web application.

> **IMPORTANT**
>
> Customer accounts configured to use single sign-on (SSO) can't be set as the default account.

Follow the steps below to set a default account:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Your Accounts** link. The Your Accounts page appears.



3. Hover your cursor over a service, then click the **Set as default** link that appears.

When you log in with your email address and password, the customer account you set as the default will automatically appear.

Using API tokens

Since an API token is scoped to a specific customer account, you'll need to create separate API tokens for each customer account. To create a new API token for a customer account, switch to the account, then follow our instructions for creating API tokens.

Setting up single sign-on (SSO)

Last updated: 2020-12-02

https://docs.fastly.com/en/guides/setting-up-single-sign-on-sso

If your company uses an identity provider (IdP) like Okta or OneLogin to manage user authentication, you can enable Fastly's single sign-on (SSO) feature to either allow or require your organization's users to sign in to the Fastly web interface using the IdP instead of an email address and password.

Prerequisites

To enable SSO or require that it be applied to all of your organization's users when they log in to the Fastly web interface, you must:

- be assigned the role of superuser for your Fastly account
- have access to your IdP's administration console

In addition, your IdP must support:

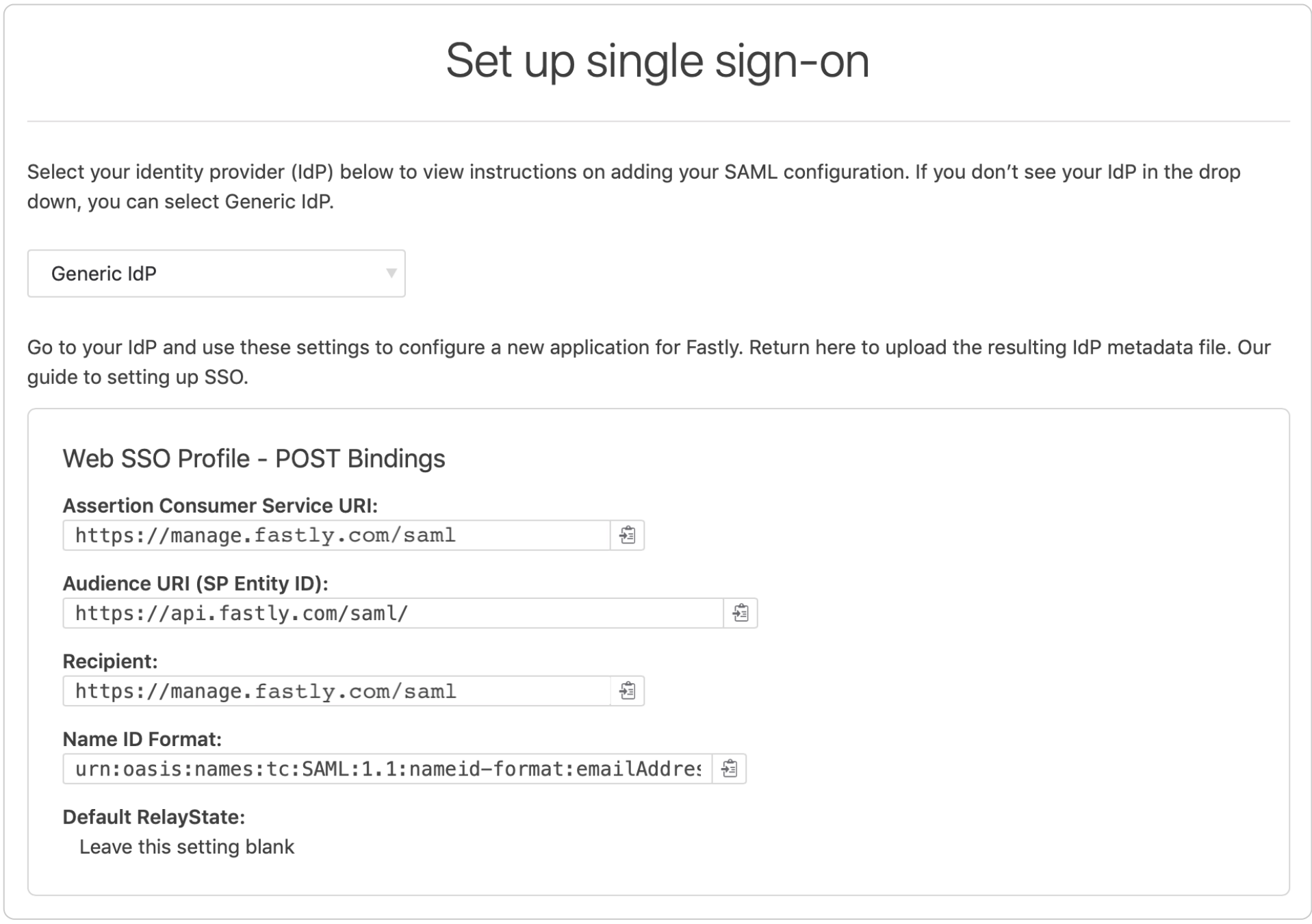
- Security Assertion Markup Language 2.0 (SAML 2.0)
- IdP-initiated SSO

You should also review this feature's limitations before enabling SSO.

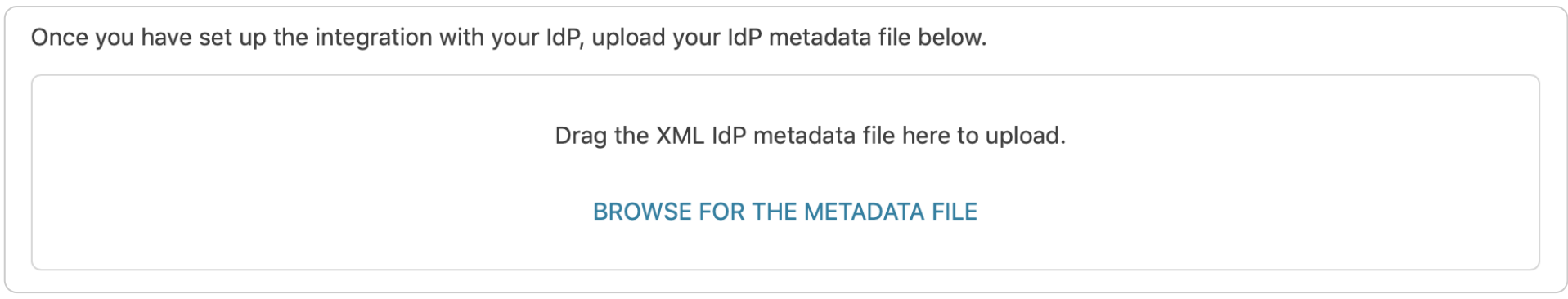
Enabling SSO

Start by selecting an IdP and configure that provider's settings keeping in mind the prerequisites. You'll need to retrieve a metadata file containing your IdP's SAML configurations for use in the Fastly web interface:

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Single sign-on** link. The Single sign-on page appears.
3. Click the **Add SAML Configuration** button. The Set up single sign-on page appears.
4. From the menu, select your organization's IdP.



5. Using the configuration details that appear in the Fastly web interface, create a new SAML 2.0 application in your IdP's administration console and assign the application to new and existing users. Refer to your IdP's documentation for more information.
6. After creating the SAML 2.0 application in your IdP, download the XML metadata file with your application's SAML configuration. The XML file includes a public certificate used to verify the signature of SAML assertions.
7. Upload your IdP metadata file. You can do this by dragging and dropping the file into the area provided or by browsing for the file and uploading it.

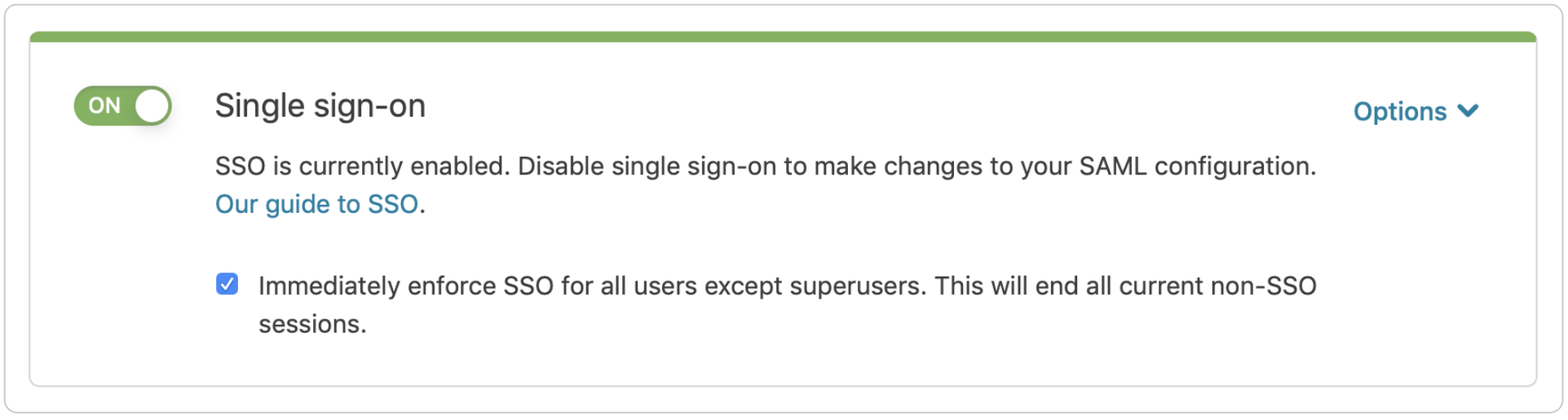


8. Click the **Save and Enable SSO** button. Your metadata will be saved and the SSO controls will indicate that SSO is enabled.

Requiring SSO for your organization

To require SSO for everyone in your organization except superusers, follow these instructions.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Single sign-on** link. The Single sign-on page appears.



3. Select the **Immediately enforce SSO** checkbox. Currently open non-SSO sessions for users assigned the role of user, billing, or engineer will be logged out and they will need to re-authenticate using SSO via your IdP.

> **NOTE**
>
> Users who have been assigned the role of superuser can always log in with their email address and password, whether or not **Single sign-on** is enabled.

Performing account tasks differently with SSO enabled

If your organization has enabled SSO, you may notice different feature availability in the Fastly web interface. This section describes the differences.

- **Changing your email address and password.** Because SSO requires user email addresses in Fastly to match those in the IdP, you won't be able to change your email address while logged in using SSO. You also won't be able to modify your password or enable two-factor authentication.
- **Creating an API token.** To create an API token while logged in to the Fastly web interface using SSO, you'll need to reauthenticate with your IdP. Follow the instructions for creating an API token and click the **Re-Authenticate** button on the Create a Token page.

  > **NOTE**
  >
  > You can't create API tokens when using G Suite for authentication.

- **Managing sessions.** Sessions created by logging in to the Fastly web interface using SSO expire after 12 hours of inactivity. Sessions created by logging in with a username and password expire after 48 hours.

Changing SSO providers

To change SSO providers, follow these instructions.

> **WARNING**
>
> Disabling the SSO feature for your organization will expire all active SSO sessions, including your own. Users will automatically be logged out of the Fastly web interface.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Single sign-on** link. The Single Sign On page appears.
3. From the **Options** menu, select **Change your SAML configuration**. A confirmation message appears.
4. In the confirmation window, click the **Confirm, Disable SSO and Delete** button.
5. Follow the instructions in the enabling SSO section.

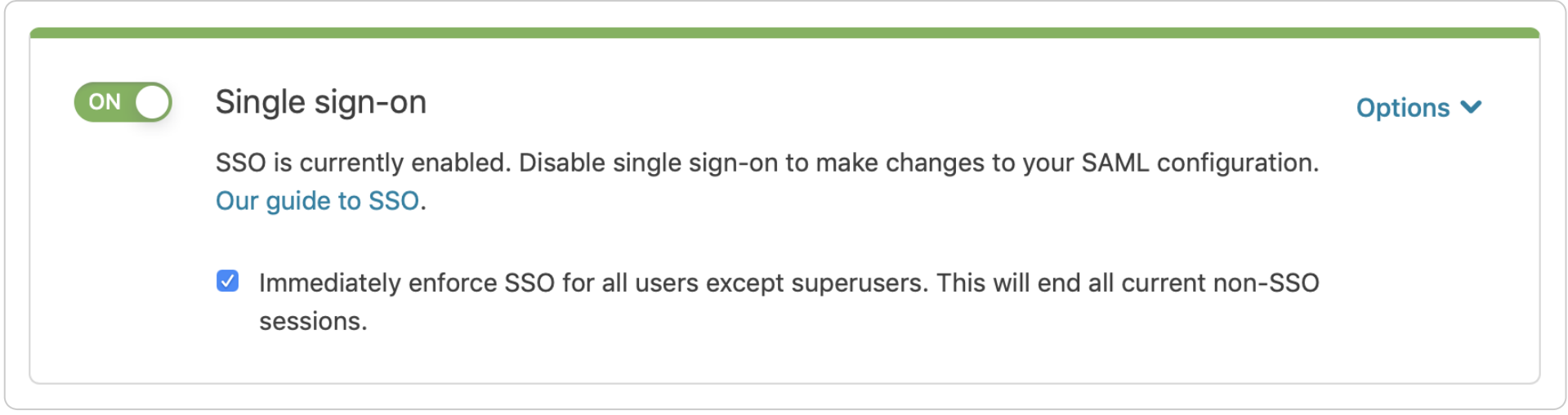
Disabling SSO

To disable SSO for your organization, follow these instructions. Disabling SSO won't delete your SSO settings. You can re-enable SSO at any point using the same IdP configuration metadata you uploaded when you first enabled SSO.

WARNING

Disabling the SSO feature for your organization will expire all active SSO sessions, including your own. Users will automatically be logged out of the Fastly web interface.

1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Single sign-on** link. The Single Sign On page appears.



3. Click the **Single sign-on** switch to disable SSO for your organization. A confirmation message appears.
4. In the confirmation window, click the **Disable SSO** button. SSO will be disabled and will not be required for your organization's users. All active SSO sessions will expire, including your own, and users will automatically be logged out of the Fastly web interface.

Temporarily disabling SSO

If your SSO provider experiences an outage, you may need to temporarily disable SSO for your organization. If you've been assigned the role of superuser, log in using an email address and password, then follow the instructions to disable SSO.

Limitations

Fastly's SSO feature has the following limitations:

- Users cannot create API tokens from the Fastly web interface when using G Suite SSO for a session's authentication.
- Fastly does not currently support Service Provider Initiated (SP-initiated) SSO.
- Fastly does not currently support automatic provisioning and de-provisioning of SSO.

Transferring services to other accounts

Last updated: 2021-11-10

https://docs.fastly.com/en/guides/transferring-services-to-other-accounts

If several employees at your company independently create testing accounts when learning about Fastly products and services, you can have the services that were created on the testing accounts transferred to another account by emailing the Customer Support team at support@fastly.com with the following information:

- the Customer IDs of the accounts
- the names of the services to be transferred
- which account should be considered the primary account

After you contact us, we'll reach out to verify the ownership of each account. If we can confirm ownership, we'll initiate the transfer.

NOTE

[Account API tokens](#) can't be transferred and don't work across multiple accounts. If you use account API tokens for purging or service automation on the service that you are transferring, an [engineer or superuser](#) will need to recreate those API tokens in the account that is receiving the service.

TIP

Fastly's [multi account user access feature](#) allows a single user to manage multiple customer accounts. If you've been invited to more than one customer account, you can quickly switch between accounts without logging out of the Fastly web application.

[Unsubscribing from Fastly marketing email](#)

Last updated: 2021-12-09

https://docs.fastly.com/en/guides/unsubscribing-from-fastly-marketing-email

If you receive what appears to be a legitimate marketing communication or promotion from Fastly, you may opt-out of these emails at any time by clicking the unsubscribe link provided in the email or by forwarding it to abuse@fastly.com so that we can address it. You will still receive emails from support@fastly.com.

Dealing with other unsolicited emails

You may receive unsolicited email messages that are neither sent by Fastly nor associated with Fastly, even if you've never [created a Fastly account](#). These unwanted emails use [email spoofing](#) to give the false impression that they were sent by Fastly. They may contain links to Fastly websites to give them the appearance of legitimacy.

If an email was not sent by Fastly's mail servers, there is nothing we can do to stop additional spam emails from being sent to your address. To block these messages entirely, you will need to use the spam protections made available by your email provider.

Compute@Edge

These articles describe how to configure Compute@Edge services.

https://docs.fastly.com/en/guides/compute-at-edge

§

These articles describe Fastly's support for protocols that allow you to stream logs to a variety of locations, including third-party services, for storage and analysis.

https://docs.fastly.com/en/guides/compute-at-edge#_cate-logging-endpoints

[Compute@Edge log streaming: Amazon Kinesis Data Streams](#)

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-amazon-kinesis-data-streams

Fastly's [Real-Time Log Streaming](#) feature for Compute@Edge services can send log files to [Amazon Kinesis Data Streams](#). Amazon Kinesis Data Streams (KDS) is a real-time data streaming service that can continuously capture data from a variety of sources.

IMPORTANT

This information is part of a limited availability release. For additional details, read our [product and feature lifecycle](#) descriptions.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

How Amazon Kinesis Data Streams work with Fastly log streaming

Amazon KDS sends data records to a *stream*. Each stream comprises one or more shards. A shard represents a fixed amount of processing capacity and the total processing capacity of a stream is determined by the number of shards. The number of shards may be increased or decreased over the lifetime of a stream. This is important because the Fastly Kinesis logging endpoint monitors the number of shards and attempts to uniformly distribute the log data records across the available shards. When the number of shards for a stream changes, the Fastly Kinesis logging endpoint automatically adjusts in response. The goal is to make the best use of the throughput capability of the stream while minimizing the configuration overhead required for our customers.

If the log volume exceeds the throughput capacity of the stream, Amazon KDS will return errors to our system that indicate that the stream is being throttled and that may prevent some logs from being delivered. AWS CloudWatch provides a metric for Kinesis Data Streams, `WriteProvisionedThroughputExceeded`, that can be used to monitor this so that adjustments to the stream capacity can be made as necessary.

> **TIP**
>
> For more information about working with Amazon KDS and understanding the capacity limits, refer to the Kinesis Developer Guide.

Prerequisites

Before adding Amazon KDS as a logging endpoint for Fastly Compute@Edge services, we recommend creating Identity and Access Management (IAM) credentials in your AWS account specifically for Fastly. Our recommended way for doing this is by creating an AWS IAM role, which lets you grant temporary credentials. For more information, see Creating an AWS IAM Role for Fastly Logging. Alternatively, create an IAM user and grant the user `kinesis:PutRecords` and `kinesis:ListShards` permissions for the logging stream. For more information, see Amazon's guidance on understanding and getting your AWS credentials.

Adding Amazon Kinesis as a logging endpoint

After you've registered for an AWS account and created an IAM user in Amazon Kinesis, follow these instructions to add Amazon KDS as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   > **TIP**
   >
   > Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Amazon Kinesis Data Streams **Create endpoint** button. The Create an Amazon Kinesis Data Streams endpoint page appears.

3. Fill out the **Create an Amazon Kinesis Data Streams endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Access method** field, select either **User Credentials** or **IAM Role**.
   - If you select **User Credentials**, enter the access key and secret key associated with the IAM user you created in your AWS account specifically for Fastly. See Amazon's documentation on security credentials for more information.

     > **NOTE**
     >
     > Password management software may mistakenly treat the **Secret Key** field as a password field because of the way your web browser works. As such, that software may try to auto-fill this field with your Fastly account password. If this happens to you, the AWS integration with Fastly services won't work and you will need to enter **Secret Key** manually instead.

- If you select **IAM Role**, enter the Amazon Resource Name (ARN) for the IAM role granting Fastly access to KDS. For more information, see Creating an AWS IAM Role for Fastly Logging.
- In the **Stream name** field, enter the name of the Kinesis stream to which log data will be sent.
- From the **Region** menu, select the region to stream logs to. This must match the region where you created your Kinesis stream.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Amazon S3

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-amazon-s3

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Amazon Simple Storage Service (Amazon S3). Amazon S3 is a static file storage service used by developers and IT teams. You can also use the instructions in this guide to configure log streaming to another S3-compatible service.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Amazon S3 as a logging endpoint for Fastly Compute@Edge services, we recommend creating Identity and Access Management (IAM) credentials in your AWS account specifically for Fastly. Our recommended way for doing this is by creating an AWS IAM role, which lets you grant temporary credentials. For more information, see Creating an AWS IAM Role for Fastly Logging. Alternatively, create an IAM user and grant the user `s3:PutObject` permissions for the logging stream. For more information, see Amazon's guidance on understanding and getting your AWS credentials.

Adding Amazon S3 as a logging endpoint

After you've registered for an Amazon S3 account and created an IAM user in Amazon S3, follow these instructions to add Amazon S3 as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Amazon Web Services S3 **Create endpoint** button. The Create an Amazon S3 endpoint page appears.
3. Fill out the **Create an Amazon S3 endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Bucket name** field, enter the name of the Amazon S3 bucket in which to store the logs.
   - In the **Access method** field, select either **User Credentials** or **IAM Role**.

- If you select **User Credentials**, enter the access key and secret key associated with the IAM user you created in your AWS account specifically for Fastly. See Amazon's documentation on security credentials for more information.

  > **NOTE**
  >
  > Password management software may mistakenly treat the **Secret Key** field as a password field because of the way your web browser works. As such, that software may try to auto-fill this field with your Fastly account password. If this happens to you, the AWS integration with Fastly services won't work and you will need to enter **Secret Key** manually instead.

- If you select **IAM Role**, enter the Amazon Resource Name (ARN) for the IAM role granting Fastly access to S3. For more information, see Creating an AWS IAM Role for Fastly Logging.

- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a new S3 endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create an Amazon S3 endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **Domain** field, optionally enter the domain of the Amazon S3 endpoint. If your Amazon S3 bucket was not created in the US Standard region, you must set the domain to match the appropriate endpoint URL. Use the table in the S3 section of the Regions and Endpoints Amazon S3 documentation page. To use an S3-compatible storage system (such as DreamHost's DreamObjects), set the domain to match the domain name for that service (for example, in the case of DreamObjects, the domain name would be `objects.dreamhost.com`).
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
   - From the **Redundancy level** menu, select a setting. This value defaults to **Standard**. Amazon's Using Reduced Redundancy Storage Guide provides more information on using reduced redundancy storage.
   - From the **ACL** menu, optionally select an access control header. See Amazon's Access Control List (ACL) Specific Request Headers for more information.
   - In the **Server side encryption** area, optionally select an encryption method to protect files that Fastly writes to your Amazon S3 bucket. Valid values are **None**, **AES-256**, and **AWS Key Management Service**. If you select **AWS Key Management Service**, you'll have to provide an AWS KMS Key ID. See Amazon's guide on protecting data using server-side encryption for more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> Although Fastly continuously streams logs into Amazon S3, the Amazon S3 website and API do not make files available for access until after their upload is complete.

Compute@Edge log streaming: Microsoft Azure Blob Storage

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-azure-blob-storage

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Microsoft Azure Blob Storage (Blob Storage). Blob Storage is a static file storage service used to control arbitrarily large amounts of unstructured data and serve them to users over HTTP and HTTPS.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Blob Storage as a logging endpoint for Fastly Compute@Edge services, create an Azure storage account in the Azure portal. For help creating the account, see Microsoft's account creation documentation.

We recommend creating a Shared Access Signature (SAS) user specifically for Fastly. For more information, see Microsoft's shared access signatures (SAS) documentation, paying specific attention to the Account SAS URI examples.

Here is an example of a SAS token that provides write permissions to a blob:

```
sv=2018-04-05&ss=b&st=2018-04-29T22%3A18%3A26Z&sr=b&se=2020-04-30T02%3A23%3A26Z&sp=w&sig=Z%2FRHIX5Xcg0Mq2rqI3OlWTjEg2tYkboXr1P9ZUXDtkk%3D
```

The table breaks down each part of the token to understand how it contributes to the SAS:

Element	Example	Description
`sv`	`sv=2018-04-05`	Storage services version.
`ss`	`ss=b`	The `signedservice` field. This is required and should be `b` for "blob storage."
`st`	`st=2018-04-29T22%3A18%3A26Z`	The start time of the token, specified in UTC.
`sr`	`sr=b`	Store resources for which this token has access. We require blob (`b`).
`se`	`se=2020-04-30T02%3A23%3A26Z`	The expiry time of the token, specified in UTC. Ensure you update your token before it expires or the logging functionality will not work.
`sp`	`sp=w`	The permissions granted by the SAS token. We require write (`w`).
`sig`	`sig=Z%2FRHIX5Xcg0Mq2...`	The signature to authorize access to the blob.

Adding Blob Storage as a logging endpoint

After you've registered for an Azure account and created a SAS token, follow these instructions to add Blob Storage as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Azure Blob Storage **Create endpoint** button. The Create a Microsoft Azure Blob Storage endpoint page appears.

3. Fill out the **Create a Microsoft Azure Blob Storage endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.

- In the **Storage account name** field, enter the unique Azure namespace in which your data objects will be stored.
- In the **Container** field, enter the name of the Blob Storage container to store logs in. See Microsoft's Blob storage page for more information.
- In the **SAS token** field, enter the token associated with the container.

> **TIP**
>
> Ensure you update your token before it expires otherwise the logging functionality will not work.

- In the **Maximum bytes** field, optionally enter the maximum file size in bytes.
- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
- In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.

4. Click the **Advanced options** link of the **Create a Microsoft Azure Blob Storage endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Microsoft Azure Blob Storage endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

> **NOTE**
>
> Although Fastly continuously streams logs into Azure Blob Storage, the storage portal and API do not make files available for access until after their upload is complete.

Ingesting data for Azure Data Explorer

Azure Data Explorer is a data exploration service for log and telemetry data. To ingest your data correctly, Data Explorer requires your logs to be formatted as comma-separated values (CSVs). When creating your logging endpoint:

- Set the **Log format** to a CSV string (`%H,%{time.start.sec}V,%{regsub(req.http.User-Agent, \{"""\}, \{"""\})}V`).
- Specify **blank** when you **Select a log line format** in the **Advanced options**.

Our guide on changing log line formats provides more information.

Compute@Edge log streaming: Cloud Files

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-cloudfiles

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log file to Cloud Files. Operated by Rackspace, Cloud Files is a file storage service used by developers and IT teams.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

If you don't already have a Rackspace Cloud account, you'll need to register for one. Follow the instructions on Rackspace's website.

Creating a Cloud Files user and container

Start by creating a Cloud Files user with restricted permissions via Rackspace's cloud control panel.

1. Log in to Rackspace's cloud control panel.
2. From the user account menu, select **User Management**.
3. Click **Create User** and fill in all appropriate details.
4. In the **Product Access** section, set **User Role** to **Custom**.
5. Review the **Product Access** list. For all items in the **Product** column, set **Role** to **No Access** except the **Files** item.
6. Set the **Files** item **Role** to **Admin**. This will allow you to create the files to store the logs in, but not access any other services.

Next, find the API key for your Cloud Files account. You'll use this later to authenticate using the Cloud Files API.

1. From the user account menu, select **Account Settings**.
2. Show the API key in the **Login details** and make a note of it.

Now that you've created the Cloud Files user and found the API key, you can set up a Cloud Files container.

1. From the **Storage** menu, select **Files**.
2. Click **Create Container**.
3. Assign the container a meaningful name like `Fastly logs - my service`.
4. Choose a region to keep the files in and make sure the container is private.
5. Click **Create Container**.

Adding a Cloud Files logging endpoint

Once you have created the Cloud Files user and container, follow these instructions to add Cloud Files as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Rackspace Cloud Files **Create endpoint** button. The Create a Cloud Files endpoint page appears.
3. Fill out the **Create a Cloud Files endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.

- In the **Bucket name** field, enter the name of the Cloud Files container in which to store the logs.
- In the **User** field, enter the username of the Cloud Files user you created above.
- In the **Access key** field, enter the API key of your Cloud Files account.
- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a Cloud Files endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create a Cloud Files endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - From the **Region** menu, select the region to stream logs to.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Datadog

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-datadog

Fastly's Real-Time Log Streaming feature for Compute@Edge services can be configured to send logs in a format readable by Datadog. Datadog is a cloud-based monitoring and analytics solution that allows you to see inside applications within your stack and aggregate the results.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

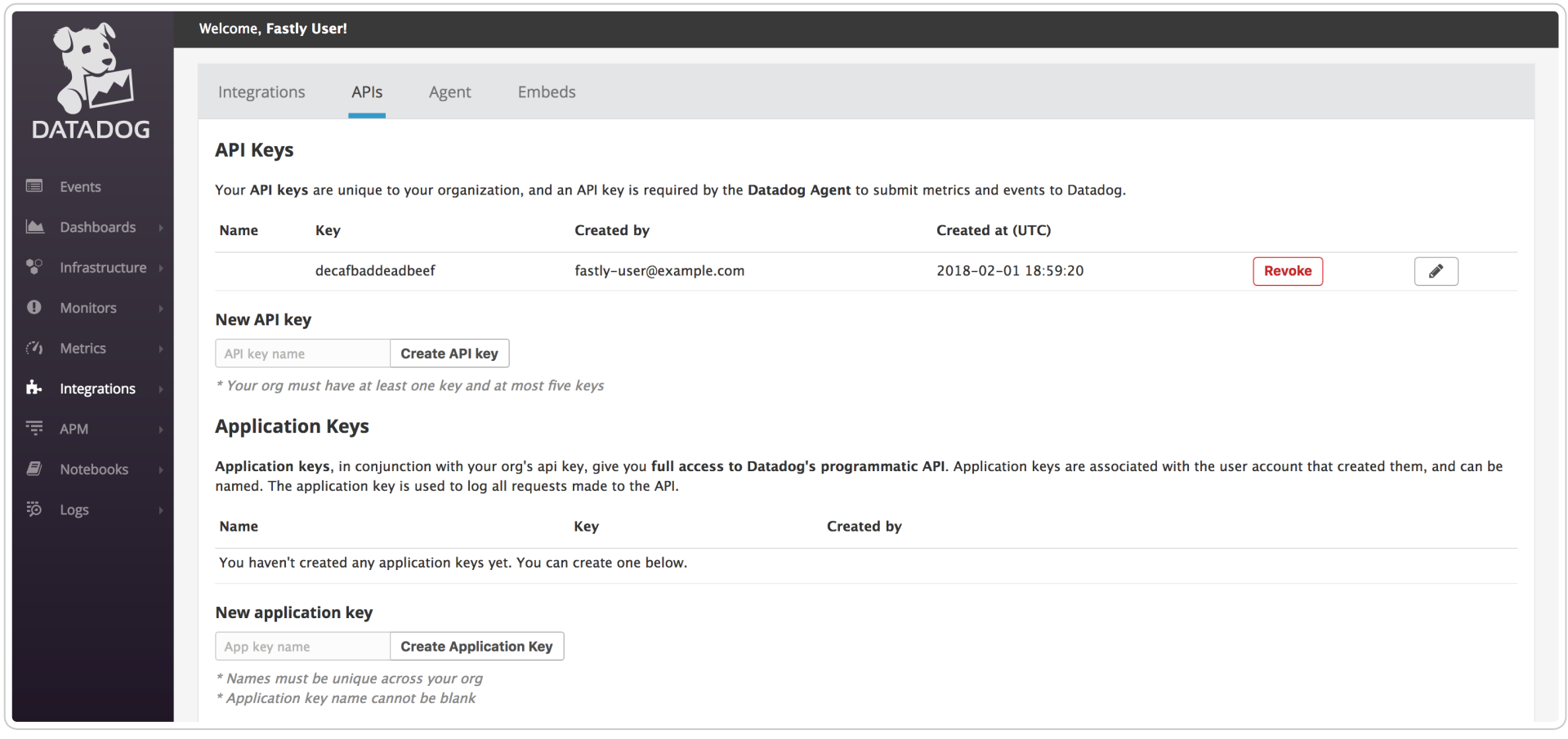
NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Datadog as a logging endpoint for Fastly Compute@Edge services, you will need to:

- **Register for a Datadog account.** You can sign up for a Datadog account on their site. A free plan exists that has some restrictions or you can upgrade for more features. Where you register your Datadog setup, either in the United States (US) or the European Union (EU), will affect which commands you use during logging endpoint setup at Fastly.
- **Get your Datadog API key from your settings page on Datadog.** In the Datadog interface, navigate to "Integrations → APIs" where you'll be able to create or retrieve an API key.



This example displays the key `decafbaddeadbeef`. Your API key will be different. Make a note of this key somewhere.

Adding Datadog as a logging endpoint

After you've created a Datadog account and noted your Datadog API key, follow the steps below to add Datadog as a logging endpoint for Fastly Compute@Edge services.

1. Review the information in our Setting Up Remote Log Streaming guide.

> **TIP**
>
> Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Datadog **Create endpoint** button. The Create a Datadog endpoint page appears.
3. Fill out the **Create a Datadog endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - From the **Region** menu, select the region to stream logs to.
   - In the **API key** field, enter the API key of your Datadog account.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Logs should begin appearing in your Datadog account a few seconds after you've created the endpoint and deployed your service changes. These logs can then be accessed via the Datadog Log Explorer on your Datadog account.

Compute@Edge log streaming: DigitalOcean Spaces

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-digitalocean-spaces

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to DigitalOcean Spaces. DigitalOcean Spaces is an Amazon S3-compatible static file storage service used by developers and IT teams.

> **IMPORTANT**

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding DigitalOcean Spaces as a logging endpoint for Fastly Compute@Edge services, you'll need to create a DigitalOcean account if you don't already have one. Then you'll need to create a space with private access permissions on DigitalOcean's website, generate a secret key and an access key, and make a note of the endpoint.

Adding DigitalOcean Spaces as a logging endpoint

After you've created a DigitalOcean Space, follow these instructions to add DigitalOcean Spaces as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Spaces by DigitalOcean **Create endpoint** button. The Create a DigitalOcean endpoint page appears.

3. Fill out the **Create a DigitalOcean endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Space name** field, enter the name of the DigitalOcean Space in which to store the logs.
   - In the **Access key** field, enter the access key associated with the DigitalOcean Space. See the DigitalOcean Spaces Authentication Guide for more information.
   - In the **Secret key** field, enter the secret key associated with the DigitalOcean Space.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create a DigitalOcean endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create a DigitalOcean endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **Domain** field, enter the region-specific endpoint for your domain. In most cases, this should be `nyc3.digitaloceanspaces.com`. If the DigitalOcean Space was not created in the `nyc3` region, refer to DigitalOcean's documentation to find the correct domain.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Elasticsearch

Last updated: 2022-03-18

https://docs.fastly.com/en/guides/compute-log-streaming-elasticsearch

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Elasticsearch. Elasticsearch is a distributed, RESTful search and analytics engine.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Elasticsearch as a logging endpoint for Fastly Compute@Edge services, ensure Elasticsearch is running on a remote server. You'll need to know the endpoint URL that includes a port to which logs should be sent (make sure it can receive traffic from Fastly) and also the name of the index to send logs to. For more information on setting up Elasticsearch, see the Elasticsearch setup documentation.

This logging endpoint works with all actively supported versions of Elasticsearch as well as some versions that have already reached their end-of-life. We also work with OpenSearch server integration. Other distributions that are API-compatible with Elasticsearch may also work but have not been explicitly tested and are not guaranteed.

Required privileges

We send data using the Bulk API via the `index` action. When using basic authentication, ensure that the required index privileges to use the `index` action are granted to the user role.

We also require access to the root path API of the Elasticsearch server. This API returns metadata about the server, such as the version number, that allows our integration to make the best choice about which bulk data API to use for each customer's server. Access to this API allows us to properly work with the wide range of Elasticsearch versions used by our customers as well as other Elasticsearch-compatible distributions.

Adding Elasticsearch as a logging endpoint

Follow these instructions to add Elasticsearch as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Elasticsearch **Create endpoint** button. The Create an Elasticsearch endpoint page appears.

3. Fill out the **Create an Elasticsearch endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **URL** field, enter the Elasticsearch endpoint URL that includes a port to which logs should be sent. The URL must be sent using HTTPS on a port that can receive incoming TCP traffic from Fastly.

- In the **Index** field, enter the name of the Elasticsearch index to send logs to. The index must follow the Elasticsearch index format rules. We support strftime interpolated variables inside braces prefixed with a pound symbol. For example, `#{%F}` will interpolate as YYYY-MM-DD with today's date.
- In the **Pipeline** field, optionally enter the ID of the Elasticsearch ingest pipeline to apply pre-process transformations to before indexing (for example, `my_pipeline_id`).
- In the **Maximum logs** field, optionally enter the maximum number of logs to append to a batch, if non-zero.
- In the **Maximum bytes** field, optionally enter the maximum size of the log batch.
- In the **BasicAuth user** field, optionally enter your basic authentication username.
- In the **BasicAuth password** field, optionally enter your basic authentication password.
- In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported.
- In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority.
- In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection.
- In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: FTP

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-ftp

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to password-protected and anonymous FTP servers.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding FTP as a logging endpoint

Follow these instructions to add FTP as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the FTP **Create endpoint** button. The Create a File Transfer Protocol (FTP) endpoint page appears.

3. Fill out the **Create a File Transfer Protocol (FTP) endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Address** field, enter the hostname or IP address of the FTP server. In the port field, enter the port number you're using for FTP (the default is `21`).
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **User** field, enter the username used to authenticate to the FTP server. For anonymous access, use the username `anonymous`.
   - In the **Password** field, enter the password used to authenticate to the FTP server. For anonymous access, use an email address as the password.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
4. Click the **Advanced options** link of the **Create a File Transfer Protocol (FTP) endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a File Transfer Protocol (FTP) endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Google BigQuery

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-google-bigquery

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to BigQuery, Google's managed enterprise data warehouse.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding BigQuery as a logging endpoint for Fastly Compute@Edge services, you will need to:

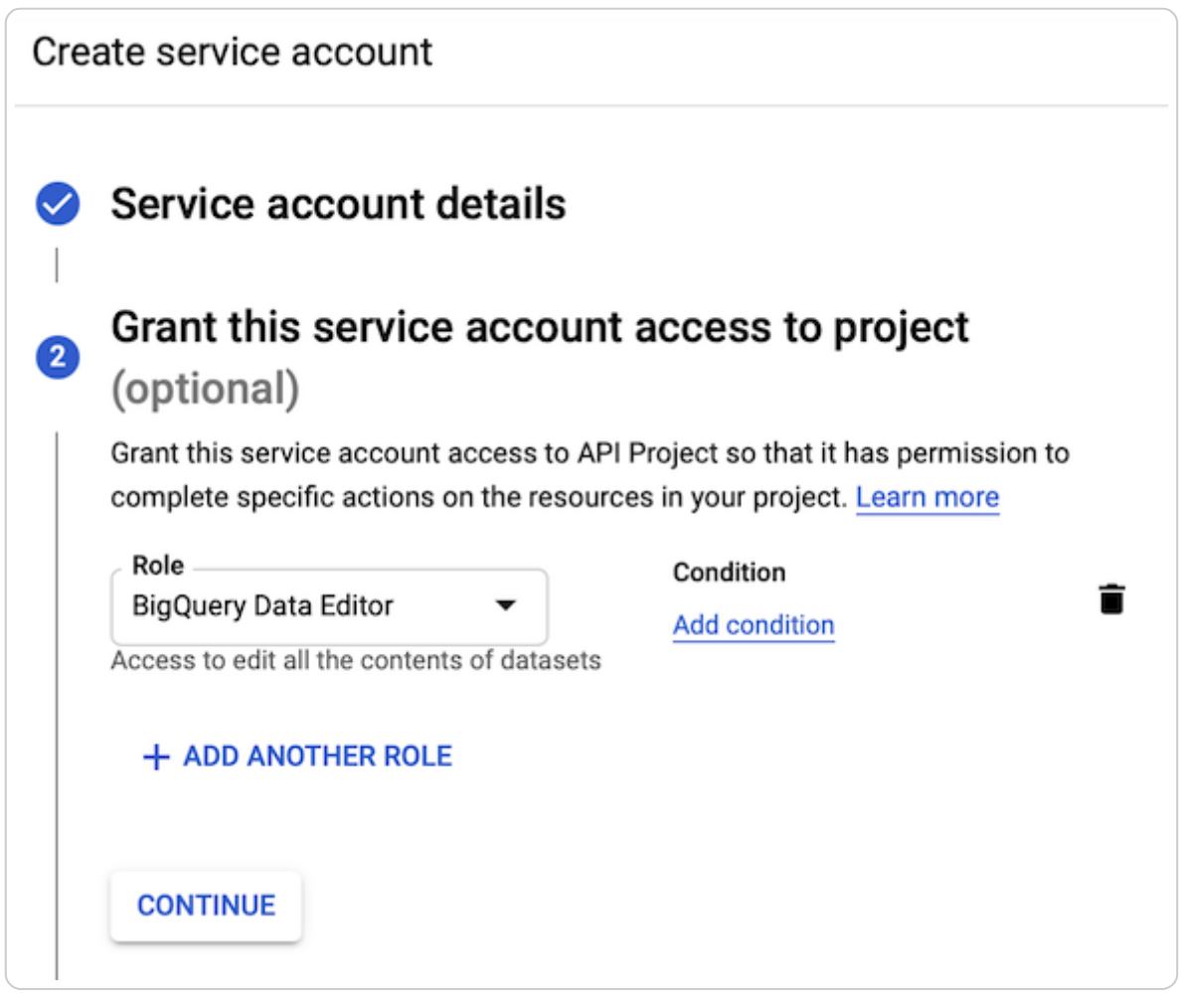
- Register for a Google Cloud Platform (GCP) account.

- Create a service account on Google's website.
- Obtain the `private_key` and `client_email` from the JSON file associated with the service account.
- Enable the BigQuery API.
- Create a BigQuery dataset.
- Add a BigQuery table.

Creating a service account

BigQuery uses service accounts for third-party application authentication. To create a new service account, follow the instructions in the Google Cloud documentation. Keep the following in mind when creating the service account:

- The service account must be assigned the **Big Query Data Editor** role to write to the table you use for Fastly logging. See BigQuery Roles for details about the default permissions assigned to the Big Query Data Editor role.



- Set the key type to **JSON** when creating the service's private key pair.

Obtaining the private key and client email

When you create the BigQuery service account, a JSON file automatically downloads to your computer. This file contains the credentials for your BigQuery service account. Open the file and make a note of the values of the `private_key` and `client_email` fields.

Enabling the BigQuery API

To send your Fastly logs to your BigQuery table, you'll need to enable the BigQuery API in the Google Cloud Platform API Manager.

Creating the BigQuery dataset

After you've enabled the BigQuery API, follow these instructions to create a BigQuery dataset:

1. Open the BigQuery page in the Cloud Console.
2. In the **Explorer** panel, select the project where you want to create the dataset.
3. In the details panel, click **Create dataset**.
4. In the **Dataset ID** field, enter a name for the dataset (e.g., `fastly_bigquery`).
5. Click the **Create dataset** button.

Adding a BigQuery table

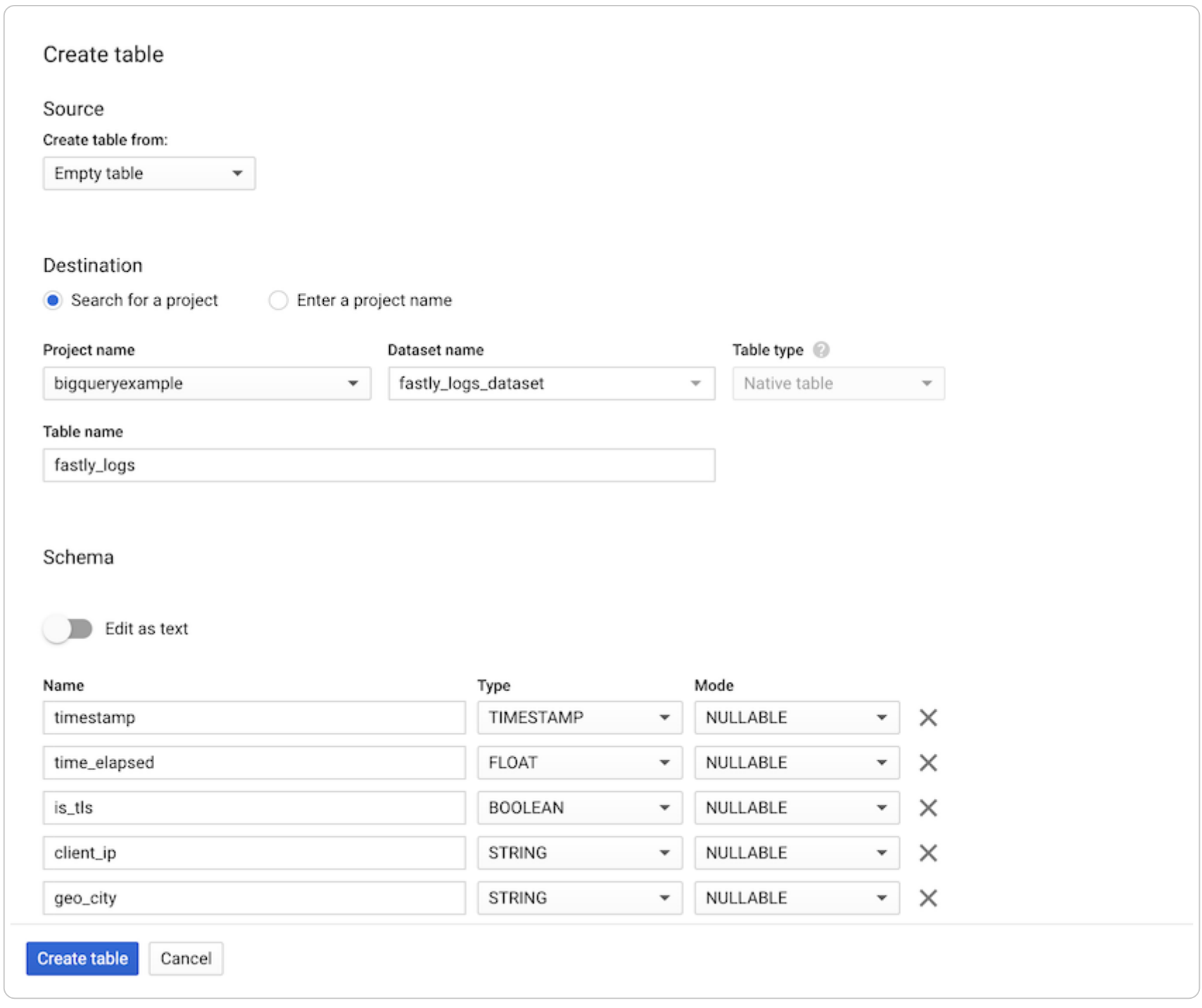
After you've created the BigQuery dataset, you'll need to add a BigQuery table. There are four ways of creating the schema for the table:

- Edit the schema using the BigQuery web interface.
- Edit the schema using the text field in the BigQuery web interface.
- Use an existing table.
- Set the table to automatically detect the schema.

> **NOTE**
>
> Setting the table to automatically detect the schema may give unpredictable results.

Follow these instructions to add a BigQuery table:

1. Open the BigQuery page in the Cloud Console.
2. In the Explorer panel, expand your project and select the BigQuery dataset you created previously.
3. In the **Source** section, select **Empty Table** from the **Create table from:** menu. The **Create table** dialog appears.



4. In the **Table name** field, enter a name for the table (e.g., `logs`).
5. In the **Schema** section of the BigQuery website, use the interface to add fields and complete the schema. See the example schema section for details.
6. Click the **Create Table** button.

Adding BigQuery as a logging endpoint

Follow these instructions to add BigQuery as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

> **TIP**
>
> Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Google BigQuery **Create endpoint** button. The Create a BigQuery endpoint page appears.

3. Fill out the **Create a BigQuery endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Email** field, enter the `client_email` address associated with the BigQuery service account.
   - In the **Secret key** field, enter the value of the `private_key` associated with your BigQuery service account.
   - In the **Project ID** field, enter the ID of your Google Cloud Platform project.
   - In the **Dataset** field, enter the name of your BigQuery dataset.
   - In the **Table** field, enter the name of your BigQuery table.
   - In the **Template** field, optionally enter an `strftime` compatible string to use as the template suffix for your table.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Example schema

The BigQuery schema would look something like this:

```
timestamp:TIMESTAMP,client_ip:STRING,geo_country:STRING,geo_city:STRING,url:STRING,request_referer:STRING,request_user_agen
t:STRING,fastly_is_edge:BOOLEAN,response_state:STRING,response_status:STRING,response_reason:STRING,response_body_size:STRIN
G,request_method:STRING,request_protocol:STRING,fastly_server:STRING,host:STRING
```

Compute@Edge log streaming: Google Cloud Pub/Sub

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-google-cloud-pubsub

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Cloud Pub/Sub, Google's global messaging and event data ingestion product.

> **IMPORTANT**
>
> This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Cloud Pub/Sub as a logging endpoint for Fastly Compute@Edge services, you will need to register for a Google Cloud Platform (GCP) account and then:

- Create a service account on Google's website.
- Navigate to the Pub/Sub section of the Google Cloud console. Follow the prompts to enable the API.

- Create a Pub/Sub topic.
- Obtain the private key from the JSON file associated with the service account configured for your Pub/Sub topic.

> **NOTE**
>
> Read more about Cloud Pub/Sub in Google's documentation.

Adding Cloud Pub/Sub as a logging endpoint

Follow these instructions to add Cloud Pub/Sub as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   > **TIP**
   >
   > Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Google Cloud Pub/Sub **Create endpoint** button. The Create a Google Cloud Pub/Sub endpoint page appears.
3. Fill out the **Create a Google Cloud Pub/Sub endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Project ID** field, enter the ID of your Google Cloud Platform project.
   - In the **Email** field, enter the email address of the service account configured for your Pub/Sub topic.
   - In the **Topic** field, enter the Pub/Sub topic to which logs should be sent.
   - In the **Secret Key** field, enter the exact value of the private key associated with the service account configured for your Pub/Sub topic.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Google Cloud Storage

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-google-cloud-storage

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Google Cloud Storage (GCS). GCS is an online file storage service used for storing and accessing data on Google's infrastructure. One advantage of using GCS is that you can use Google BigQuery to analyze the log files.

> **IMPORTANT**
>
> This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding GCS as a logging endpoint for Fastly Compute@Edge services, you will need to:

- Register for a GCS account.
- Create a bucket and service account on Google's website.

- Obtain the `private_key` and `client_email` from the JSON file associated with the service account.
- Enable the Google Cloud Storage JSON API.

Creating a GCS bucket

You can create a new GCS bucket to hold the logs, or you can use an existing bucket. Be sure to note the name of the bucket as you will need it later. To learn how to create a GCS bucket, see Google's guide on creating a bucket.

Creating a service account

GCS uses service accounts for third-party application authentication. You will need to create a new service account on Google's website with the role of `Storage Object Creator` and make sure you've added it as a member of the GCS bucket you created. To learn how to create a service account, see Google's guide on generating a service account credential. When you create the service account, be sure to set the **Key Type** to `JSON`.

Obtaining the private key and client email

After you create the service account, a JSON file will be downloaded to your computer. This file contains the credentials for the GCS service account you just created. Open the file with a text editor and make a note of the `private_key` and `client_email`.

Enabling the Google Cloud Storage JSON API

To ensure the Fastly logs are sent to your GCS bucket, you need to enable the Google Cloud Storage JSON API. For more information, see Google's instructions for activating the API.

Adding GCS as a logging endpoint

Follow these instructions to add GCS as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   > **TIP**
   >
   > Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Google Cloud Services **Create endpoint** button. The Create a Google Cloud Storage (GCS) endpoint page appears.
3. Fill out the **Create a Google Cloud Storage (GCS) endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Email** field, enter the `client_email` address listed in the JSON file associated with the service account you created on Google's website.
   - In the **Bucket name** field, enter the name of the GCS bucket in which to store the logs.
   - In the **Secret key** field, enter the `private_key` value listed in the JSON file associated with the service account you created on Google's website. We strip out the JSON newline escape characters for you so don't worry about removing them.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
4. Click the **Advanced options** link of the **Create a Google Cloud Storage (GCS) endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create a Google Cloud Storage (GCS) endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Honeycomb

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-honeycomb

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send logs in JSON format to Honeycomb. Honeycomb is a tool that allows developers to explore the operations of complex systems, microservices, and databases.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Honeycomb as a logging endpoint for Fastly Compute@Edge services, you'll need to perform the following steps:

- Sign up for a Honeycomb account if you don't already have one.
- Obtain the Write Key for your team on the Honeycomb Account page.
- Choose a Dataset name. If you plan to collect data from multiple environments (like production, development, staging), Honeycomb recommends creating a Dataset for each environment and naming your Datasets accordingly (e.g., `prod.queries`, `dev.queries`, and `staging.queries`). If a Dataset doesn't exist, Honeycomb will create one automatically.

Adding Honeycomb as a logging endpoint

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Honeycomb **Create endpoint** button. The Create a Honeycomb endpoint page appears.
3. Fill out the **Create a Honeycomb endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Write Key** field, enter the write key for your Honeycomb team. This is available on the Honeycomb Account page.
   - In the **Dataset** field, enter the name of the Honeycomb Dataset (e.g., `myDataset`).

4. Click the **Create** button to create the new logging endpoint.

5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: HTTPS

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-https

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to an HTTPS endpoint.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

When sending logs to a HTTPS endpoint, Fastly requires proof that you control the domain name specified in the **URL** field by using a HTTP challenge on a well-known path. If, for example, your **URL** field is `foo.example.com/some/log/path`, then the following challenge path must send a 200 response:

`foo.example.com/.well-known/fastly/logging/challenge`

Responses must include the hex representation of the SHA-256 of your Fastly service ID and it must appear on its own line in the response. For example:

```
$ sha256sum <SERVICEID>

ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c
```

If multiple service IDs are used, multiple hex(sha256) lines can be added to that challenge body. In addition, an asterisk (`*`) can be used on a line to allow any service to post to the HTTP endpoint. For example:

```
ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c
06ae6402e02a9dad74edc71aa69c77c5747e553b0840bfc56feb7e65b23f0f61
*
```

Adding HTTPS as a logging endpoint

Follow these instructions to add HTTPS as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the HTTPS **Create endpoint** button. The Create an HTTPS endpoint page appears.

3. Fill out the **Create an HTTPS endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **URL** field, enter the URL to which log data will be sent (e.g., `https://logs.example.com/`).

- In the **Maximum logs** field, optionally enter the maximum number of logs to send as a batch.
- In the **Maximum bytes** field, optionally enter the maximum size of a log batch.

4. Click the **Advanced options** link of the **Create an HTTPS endpoint** page. The Advanced options appear.
5. Fill out the **Advanced options** of the **Create an HTTPS endpoint** page as follows:
   - In the **Content type** field, optionally enter the content type to use when sending logs (e.g., `application/json`).
   - In the **Custom header name** field, optionally enter a custom header to use when sending logs (e.g., `Authorization`).
   - In the **Custom header value** field, optionally enter a custom header value to use when sending logs (e.g., `Bearer <token>`).
   - In the **Method** area, optionally select the appropriate HTTP method to use.
   - In the **JSON log entry format** area, select the appropriate log entry format to use. The JSON log entry format enforces valid JSON formatting. Selecting **Array of JSON** wraps JSON log batches in an array. Selecting **Newline delimited** places each JSON log entry onto a new line in a batch.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
6. Fill out the **Using your own certificate authority (CA)** section of the **Advanced options** area as follows:
   - In the **TLS Hostname** field, optionally enter the hostname used to verify the server's certificate. This can be either the Common Name (CN) or Subject Alternative Name (SAN). If the hostname is not specified, the hostname of the first broker in the Brokers field will be used. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.
7. Click the **Create** button to create the new logging endpoint.
8. Click the **Activate** button to deploy your configuration changes.

Firewall considerations

Your HTTPS endpoint may have limited security features. For this reason, it's best to create a firewall for your HTTP endpoint server and only accept TCP traffic on your configured port from our address blocks. Our list of IP address blocks is dynamic, so we recommend programmatically obtaining the list whenever possible.

Compute@Edge log streaming: Kafka

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-kafka

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Apache Kafka. Kafka is an open-source, high-throughput, low-latency platform for handling real-time data feeds.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding Apache Kafka as a logging endpoint for Fastly Compute@Edge services, ensure Kafka is running on a remote server. You'll need to know the hostname or IP address of one or more servers (Brokers) and the category or feed name to which messages will be stored (Topic). For more information on setting up Kafka see the Apache Kafka Quickstart guide.

Adding Kafka as a logging endpoint

Follow these instructions to add Kafka as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   > **TIP**
   >
   > Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Apache Kafka **Create endpoint** button. The Create an Apache Kafka endpoint page appears.

3. Fill out the **Create an Apache Kafka endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Brokers** field, enter the hostname or IP address of one or more servers (Kafka brokers). Specify multiple servers using a comma-separated string.
   - In the **Topic** field, enter the name of the topic to send logs to.
   - In the **Maximum bytes** field, optionally enter the maximum size of a log batch.
   - From the **Parse key-values** controls, optionally select whether or not to parse any key-value pairs within the log format.
   - In the **Write acknowledgement** area, optionally select the appropriate write acknowledgement a leader must receive before a write is considered successful.
   - In the **Compression codec** area, optionally select the appropriate codec to use for compression of your logs.
   - From the **Use SASL** controls, optionally select whether or not to enable SASL authentication. SASL authentication can be enabled concurrently with TLS encryption. When you select Yes, additional SASL authentication fields appear.
   - From the **SASL authentication mechanism** menu, select the appropriate challenge-response mechanism to use for authenticating the SASL client authentication username and password.
   - In the **User** field, enter the SASL client authentication username.
   - In the **Password** field, enter the SASL client authentication password.
   - From the **Use TLS** controls, optionally select whether or not to enable TLS encryption for the Kafka endpoint. TLS encryption can be enabled concurrently with SASL authentication. When you select Yes, additional TLS fields appear.
   - In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported. If the hostname is not specified, the hostname of the first broker in the Brokers field will be used. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS

client certificate allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.

- In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.

4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Log Shuttle

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-log-shuttle

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Log Shuttle. Log Shuttle is an open source application designed to provide simpler encrypted and authenticated log delivery.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding Log Shuttle as a logging endpoint

Follow these instructions to add Log Shuttle as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Log Shuttle **Create endpoint** button. The Create a Log Shuttle endpoint page appears.
3. Fill out the **Create a Log Shuttle endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Token** field, enter the data authentication token. This is required for some endpoints like Heroku's Log Integration.
   - In the **URL** field, enter the URL to which log data will be sent (e.g., `https://logs.example.com/`).
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: LogDNA

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-logdna

Fastly's Real-Time Log Streaming feature for Compute@Edge services can be configured to send logs in a format that is readable by LogDNA. LogDNA is a cloud-based log management system that aggregates system and application logs into a single location.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding LogDNA as a logging endpoint for Fastly Compute@Edge services, you'll need to perform the following steps:

- Sign up for a LogDNA account if you don't already have one. You can sign up for a free (but restricted plan) or upgrade a LogDNA plan to include more features.
- Set up a new LogDNA syslog source via the LogDNA web application by following their account-tailored log source instructions. Be sure to make note of the port number displayed at the end of the syslog URL when you complete set up. This is the port number you'll enter when setting up LogDNA as a logging endpoint for Fastly.

Adding LogDNA as a logging endpoint

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the LogDNA (via Syslog) **Create endpoint** button. The Create a Syslog endpoint page appears.
3. Fill out the **Create a Syslog endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Syslog address** field, enter `syslog-a.logdna.com`. In the port field after the colon, enter the LogDNA port number you noted during your LogDNA account setup.
   - From the **TLS** menu, select **Yes** to enable encryption for the syslog endpoint. The TLS Hostname and TLS CA Certificate fields will both appear.
   - In the **TLS Hostname** field, enter `syslog-a.logdna.com`. This is the hostname Fastly will use to verify the syslog server's certificate.
4. Click the **Advanced options** link of the **Create a Syslog endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Syslog endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Logs should begin appearing in your LogDNA account a few seconds after you've created the endpoint and deployed your service.

Compute@Edge log streaming: Logentries

Last updated: 2021-11-01

https://docs.fastly.com/en/guides/compute-log-streaming-logentries

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Logentries. Logentries is a real-time log management and analytics system that you can use to monitor your Fastly logs.

> **IMPORTANT**
>
> This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding Logentries as a logging endpoint

If you already have a Logentries account, or if you sign up for a Logentries account on the Logentries website, you can add Logentries as a logging endpoint in the Fastly web interface.

Prerequisites

1. Register for a Logentries account.
2. Create a new log in the Logentries application by following the instructions on the Logentries website.
3. During new log creation, select **Manual Configuration** and **Token TCP**.
4. Make a note of the token provided in the Logentries configuration panel. We recommend you use this token when you create the Logentries logging endpoint for Fastly services.

Creating the logging endpoint in the web interface

After you've created a new log in Logentries and found the token, follow these instructions to add Logentries as a logging endpoint for Fastly Compute@Edge services:

1. Review the information in our Setting Up Remote Log Streaming guide.

   > **TIP**
   >
   > Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Logentries by Rapid7 **Create endpoint** button. The Create a Logentries endpoint page appears.
3. Fill out the **Create a Logentries endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Token** field, enter the token provided in the Logentries configuration panel.
   - From the **Region** menu, select the region to stream logs to. For older Logentries accounts, where the log view URL starts with `https://logentries.com/`, select `EU`. For InsightOps accounts, select the region based on which data storage region you chose on signup for your Rapid7 account. For example, if your log view URL is `https://us2.ops.insight.rapid7.com/`, then your selected region would be `US-2`.
4. Click the **Advanced options** link of the **Create a Logentries endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Logentries endpoint** page as follows:
   - From the **TLS** menu, optionally select **Yes**.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Next steps

Logentries maintains the Fastly Community Pack that leverages custom VCL to provide advanced User-Agent statistics, regional statistics, error tracking, and more.

Compute@Edge log streaming: Loggly

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-loggly

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Loggly. Loggly is an agent-less log collection and management tool.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

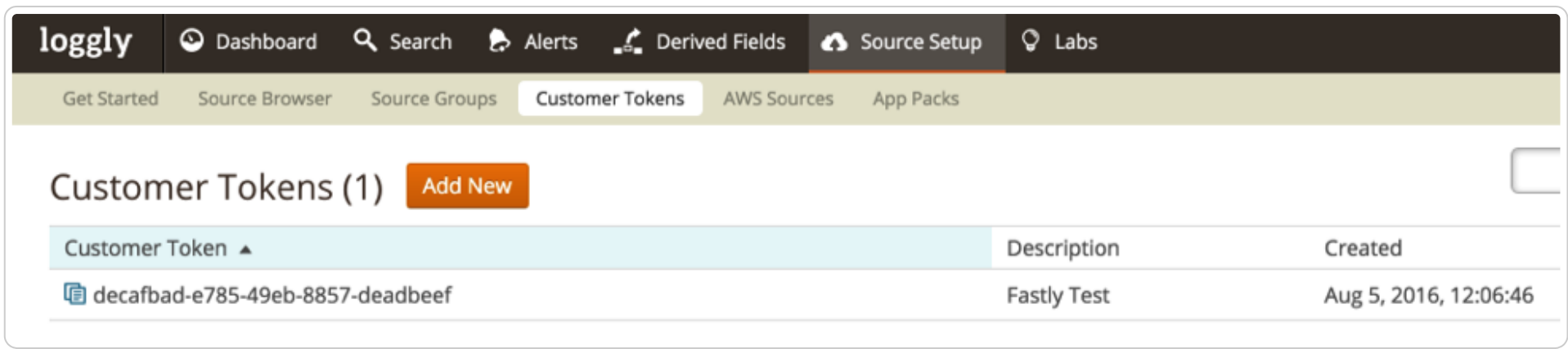
Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

If you don't already have a Loggly account, you'll need to register for one. Follow the signup instructions on the Loggly website.

Follow the steps below to find your Loggly customer token:

1. Navigate to the Customer Tokens area in the **Source Setup** on your Loggly dashboard.



2. Make note of your Loggly customer token. Loggly uses this to associate data you send them with your account.

Adding Loggly as a logging endpoint

After you've created a Loggly account and obtained your customer token, follow these instructions to add Loggly as a logging endpoint for Fastly Compute@Edge services:

1. Review the information in our Setting Up Remote Log Streaming guide.

TIP

Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Loggly **Create endpoint** button. The Create a Loggly endpoint page appears.

3. Fill out the **Create a Loggly endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Token** field, enter your Loggly customer token.

4. Click the **Create** button to create the new logging endpoint.

5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Heroku's Logplex

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-logplex

As part of our Real-Time Log Streaming feature for Compute@Edge services, if you use our Heroku add-on, you can send log files directly to Heroku's Logplex system. Logplex is Heroku's distributed syslog router that collates and distributes log entries from a variety of sources into a single channel.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before continuing, you will need the token from your Heroku Logplex account. If you don't have a Heroku Logplex account, now is the time to set one up by signing up for Heroku.

Once enabled, your Fastly logs will be available in exactly the same way as your regular app and hosted service logs. You can view them using the Heroku command line log viewer or send them to a logging add-on.

Adding Heroku Logplex as a logging endpoint

Follow these instructions to add Heroku Logplex as a logging endpoint for Fastly Compute@Edge services:

1. Review the information in our Setting Up Remote Log Streaming guide.

TIP

Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Heroku Logplex **Create endpoint** button. The Create a Heroku Logplex endpoint page appears.
3. Fill out the **Create a Heroku Logplex endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Token** field, enter your Heroku Logplex token.
   - In the **URL** field, enter `https://1.us.logplex.io/logs` unless otherwise instructed by our support staff.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: New Relic Logs

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-newrelic-logs

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to New Relic Logs.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

Before adding New Relic Logs as a logging endpoint for Fastly Compute@Edge services, you will need to:

- Register for a New Relic account.
- Obtain your license key or optionally create an Insert API key.

Adding New Relic Logs as a logging endpoint

Follow these instructions to add New Relic Logs as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the New Relic Logs **Create endpoint** button. The Create a New Relic Logs endpoint page appears.
3. Fill out the **Create a New Relic Logs endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **License key / Insert key** field, enter your New Relic license key or Insert API key.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: OpenStack

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-openstack

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to OpenStack. OpenStack is an open-source platform for cloud-computing that many companies deploy as an infrastructure-as-a-service.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding OpenStack as a logging endpoint

Follow these instructions to add OpenStack as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

> **TIP**
>
> Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the OpenStack **Create endpoint** button. The Create an OpenStack endpoint page appears.

3. Fill out the **Create an OpenStack endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Auth URL** field, enter the URL used for OpenStack authentication (e.g., `https://auth.api.rackspacecloud.com/v1.0`).
   - In the **Bucket name** field, enter the name of the OpenStack bucket in which to store the logs.
   - In the **User** field, enter your OpenStack username.
   - In the **Access Key** field, enter your OpenStack access key.
   - In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
4. Click the **Advanced options** link of the **Create a new OpenStack endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create an OpenStack endpoint** page as follows:
   - In the **Path** field, optionally enter the path within the bucket to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the bucket's root path. Our guide on changing where log files are written provides more information.
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Papertrail

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-papertrail

> **IMPORTANT**
>
> This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Papertrail. Papertrail is a web-based log aggregation application used by developers and IT teams. Instructions for setting up remote log streaming via Papertrail are detailed in the Papertrail setup and configuration documentation.

> **TIP**
>
> Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Compute@Edge log streaming: Scalyr

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-scalyr

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Scalyr. Scalyr pulls all your server logs and metrics into a centralized, searchable system in real time.

> **IMPORTANT**
>
> This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

> **NOTE**
>
> Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

If you don't already have a Scalyr account, you'll need to register for one. Follow the signup instructions on the Scalyr website.

Once you've signed up, navigate to the **API Keys** area in the **Settings** on your Scalyr dashboard and make note of your Scalyr Write Token. Scaylr uses this to associate data you send them with your account. You'll need this token when you set up your endpoint with Fastly.

If you're adding the Scalyr endpoint via the command line, instead of the web interface, you should also have your Fastly API token and the service ID and version number of the Fastly service for which you'll be enabling Scalyr logging.

Adding Scalyr as a logging endpoint

Follow these instructions to add Scalyr as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   > **TIP**
   >
   > Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Scalyr **Create endpoint** button. The Create a Scalyr endpoint page appears.

3. Fill out the **Create a Scalyr endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Logfile** field, optionally specify the log file name under which your logs will appear on Scalyr's Overview page. Defaults to `logplex`.
   - In the **Token** field, enter the Scalyr Write Token provided in the Scalyr dashboard.
   - From the **Region** menu, select the region to stream logs to.

4. Click the **Create** button to create the new logging endpoint.

5. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: SFTP

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-sftp

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to SFTP, a secure file transfer subsystem for the Secure Shell (SSH) protocol. Our SFTP endpoint supports both password-based authentication and SSH public-key authentication, with SSH public-key authentication being preferred. To learn more about SSH public-key authentication, or to learn how to generate public and private key pairs, see this guide.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding SFTP as a logging endpoint

Follow these instructions to add SFTP as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

   TIP

   Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the SFTP **Create endpoint** button. The Create an SSH File Transfer Protocol (SFTP) endpoint page appears.

3. Fill out the **Create an SSH File Transfer Protocol (SFTP) endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Timestamp format** field, optionally enter a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written provides more information.
   - In the **Address** field, enter the hostname or IP address of the SFTP server. In the port field after the colon, enter the port number you're using for SFTP (the default is `22`).
   - In the **Path** field, enter the path to use for storing log files. Leaving the default `/` in this field means the files will be saved in the root path. We describe this variable in more detail in our guide on changing where log files are written.

     TIP

     If you save logs on the SFTP server, make sure the directory already exists.

   - In the **User** field, enter the username used to authenticate to the SFTP server.
   - In the **Known hosts** field, enter a Host key for each Host you can connect to over SFTP. Each Host key you enter must be on its own line. Known hosts entries should match what's stored in your known_hosts file located in your home directory (or the local account settings if you're working with a Mac or Windows operating system). A known hosts entry looks like this:

     ```
     1.2.3.4 ecdsa-sha2-nistp256 aBc123xYz…
     ```

     where the `1.2.3.4` is the SFTP IP address, `ecdsa-sha2-nistp256` is your Host key algorithm, and `aBc123xYz…` is your public key.

- In the **Secret key** field, enter the SSH secret key used to connect to the server. If both Secret key and Password are entered, the Secret key will be used in preference.
- In the **Password** field, enter the password used to authenticate to the SFTP server. If both Password and Secret key are entered, the Secret key will be used in preference.
- In the **Period** field, optionally enter an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.

4. Click the **Advanced options** link of the **Create an SSH File Transfer Protocol (SFTP) endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create an SSH File Transfer Protocol (SFTP) endpoint** as follows:
   - In the **PGP public key** field, optionally enter a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You will only be able to read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. See our guide on log encryption for more information.
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.
   - In the **Compression** field, optionally select the compression format you want applied to the log files. Our guide on changing log compression options provides more information.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Compute@Edge log streaming: Splunk

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-splunk

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Splunk. Splunk is a web-based log analytics platform used by developers and IT teams.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Prerequisites

To use Splunk as a logging endpoint, you'll need to enable the HTTP Event Collector (HEC), create a token, and enable it. Follow the instructions on Splunk's website:

1. Enable HEC.
2. Create an HEC token.
3. Enable the HEC token.
4. Disable indexer acknowledgment for tokens used by Fastly to stream logs.

You'll need to remember the HEC token and find the URL for your collector. The URL structure depends on the type of Splunk instance you're using. Use the table below to find the URL structure for your Splunk instance.

Type	URL
Self hosted	`https://<hostname>:8088/services/collector/event`

Type	URL
Self-service Splunk Cloud plans	`https://input-<hostname>:8088/services/collector/event`
All other Splunk Cloud plans	`https://http-inputs-<hostname>:8088/services/collector/event`

While logged in to Splunk, you can find the hostname for the URL in your web browser's address bar.

Adding Splunk as a logging endpoint

After you've created a Splunk account and obtained your customer token, follow these instructions to add Splunk as a logging endpoint for Fastly Compute@Edge services:

1. Review the information in our Setting Up Remote Log Streaming guide.

> **TIP**
>
> Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Splunk **Create endpoint** button. The Create a Splunk endpoint page appears.
3. Fill out the **Create a Splunk endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **URL** field, enter the URL to send data to (e.g., `https://<splunk_host>:8088/services/collector/event/1.0`).
   - In the **Token** field, enter the token for the HEC.
   - From the **Use TLS** controls, optionally select whether or not to enable TLS. When you select Yes, additional TLS fields appear.
   - In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported. If you are using Splunk Enterprise see the Splunk Enterprise section below for more information.
   - In the **TLS CA certificate** field, enter the CA certificate used to verify that the origin's certificate is valid. It must be in PEM format. This is not required if your origin-side TLS certificate is signed by a well-known CA. See the using TLS CA certificates section for more information.
   - In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection.
   - In the **Maximum logs** field, optionally enter the maximum number of logs to append to a batch, if non-zero.
   - In the **Maximum bytes** field, optionally enter the maximum size of the log batch, if non-zero.
4. Click the **Create** button to create the new logging endpoint.
5. Click the **Activate** button to deploy your configuration changes.

Using TLS CA certificates

If you've installed your own TLS certificate in Splunk Enterprise or Splunk Cloud, you'll need to provide the corresponding CA certificate.

Splunk Cloud

For Splunk Cloud, the default set up has the following CA certificate:

```
-----BEGIN CERTIFICATE-----
MIIB/DCCAaGgAwIBAgIBADAKBggqhkjOPQQDAjB+MSswKQYDVQQDEyJTcGx1bmsg
Q2xvdWQgQ2VydGlmaWNhdGUgQXV0aG9yaXR5MRYwFAYDVQQHEw1TYW4gRnJhbmNp
c2NvMRMwEQYDVQQKEwpTcGx1bmsgSW5jMQswCQYDVQQIEwJDQTEVMBMGA1UECxMM
U3BsdW5rIENsb3VkMB4XDTE0MTExMDA3MDAxOFoXDTM0MTEwNTA3MDAxOFowfjEr
MCkGA1UEAxMiU3BsdW5rIENsb3VkIENlcnRpZmljYXRlIEF1dGhvcml0eTEWMBQG
A1UEBxMNU2FuIEZyYW5jaXNjbzETMBEGA1UEChMKU3BsdW5rIEluYzELMAkGA1UE
CBMCQ0ExFTATBgNVBAsTDFNwbHVuayBDbG91ZDBZMBMGByqGSM49AgEGCCqGSM49
AwEHA0IABPRRy9i3yQcxgMpvCSsI7Qe6YZMimUHOecPZWaGz5jEfB4+p5wT7dF3e
QrgjDWshVJZvK6KGO7nDh97GnbVXrTCjEDAOMAwGA1UdEwQFMAMBAf8wCgYIKoZI
zj0EAwIDSQAwRgIhALMUgLYPtICN9ci/ZOoXeZxUhn3i4wIo2mPKEWX0IcfpAiEA
8Jid6bzwUqAdDZPSOtaEBXV9uRIrNua0Qxl1S55TlWY=
-----END CERTIFICATE-----
```

Splunk Enterprise

Splunk Enterprise provides a set of default certificates, but we strongly recommend you configure your own certificates for your Fastly logging endpoint rather than relying on the default certificates. The certificates provided by Splunk Enterprise only specify a Common Name (CN), which cannot be used to properly verify the identity of the Splunk host presenting the certificate. Additionally, these certificates are less secure because the same root certificate is available in every Splunk Enterprise download. We encourage you to maintain the best possible security posture by configuring your own certificates rather than relying on the default certificates. The Splunk documentation provides a guide for configuring your own certificates.

Compute@Edge log streaming: Sumo Logic

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-sumologic

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to Sumo Logic. Sumo Logic is a web-based log analytics platform used by developers and IT teams.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

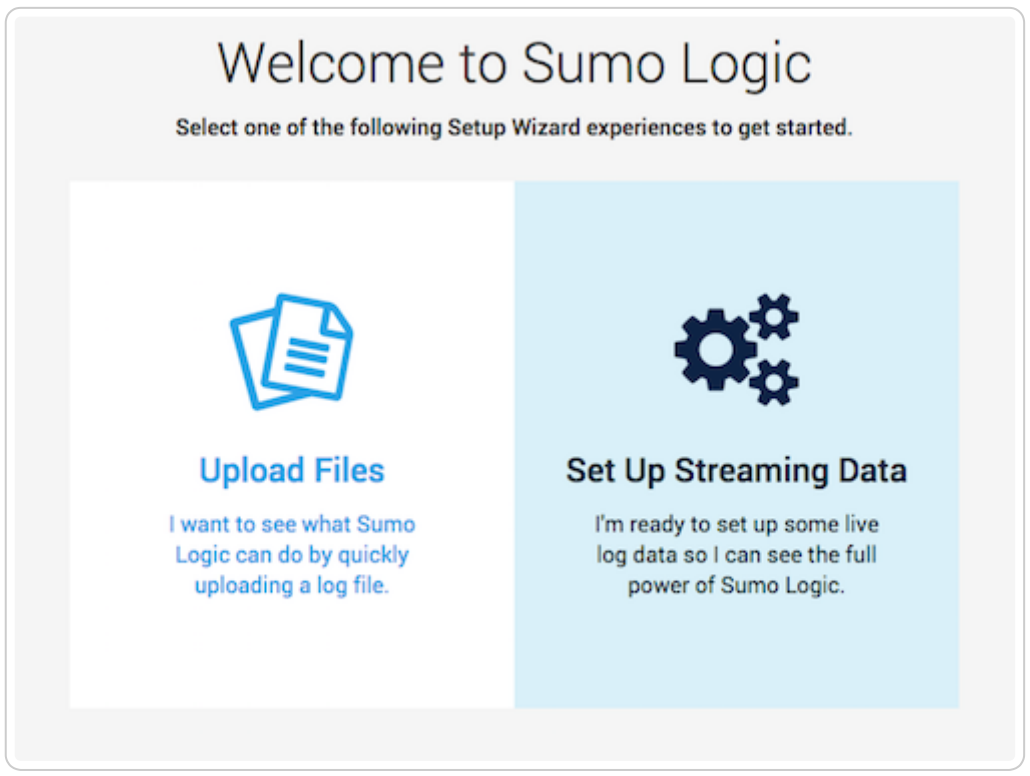
NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

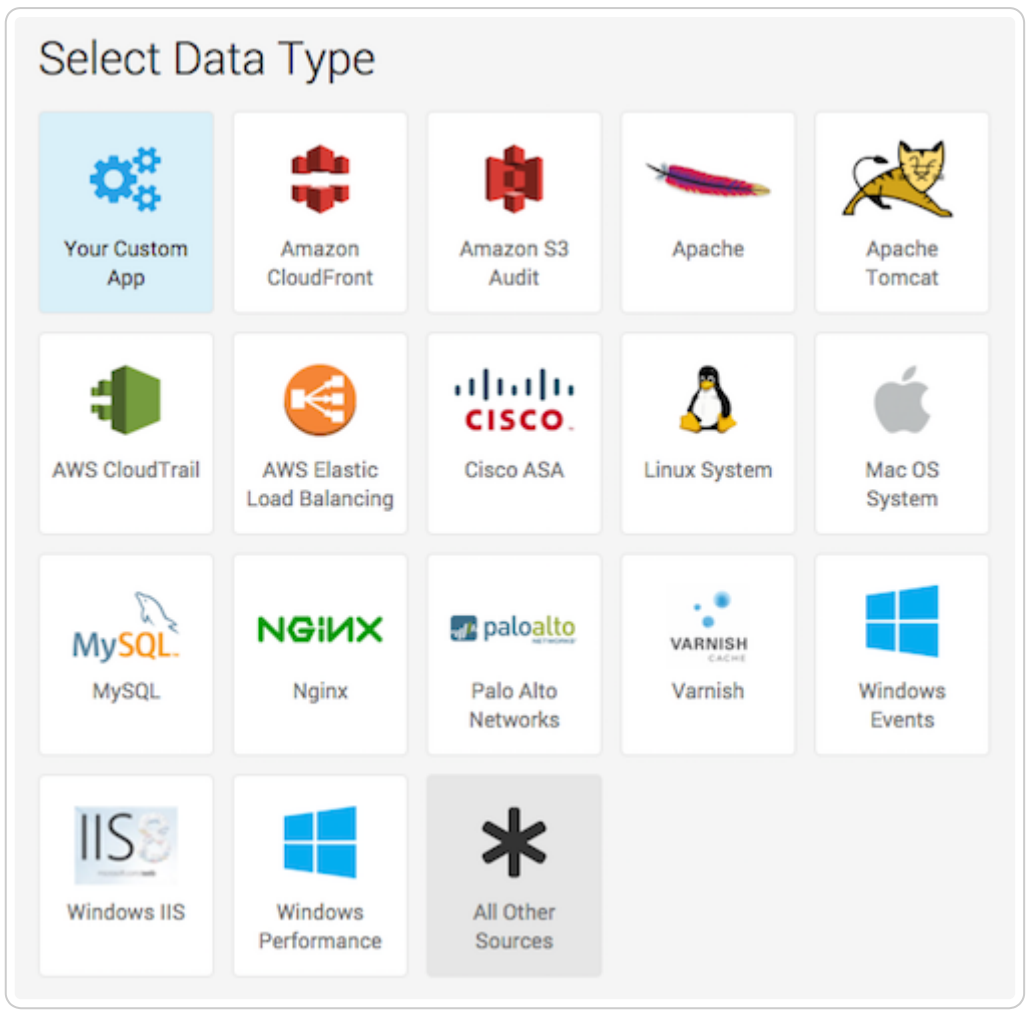
Setting up Sumo Logic

To use Sumo Logic as a logging endpoint, you'll need to create a Sumo Logic account, add a new source, and save the HTTP Source URL. Follow these instructions to add a new source in the Sumo Logic website:

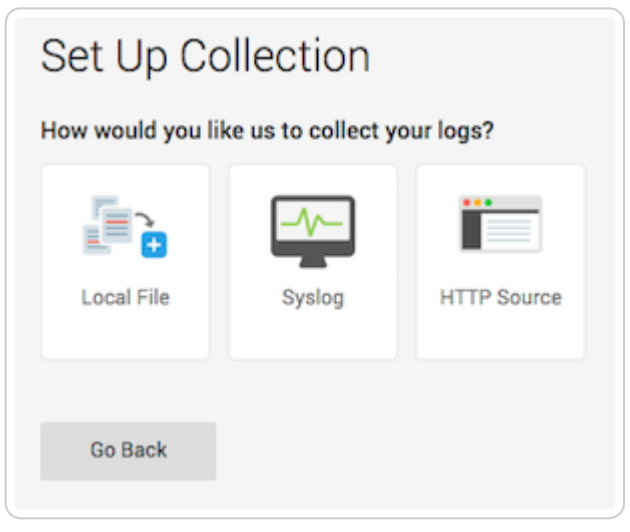
1. The process starts with the Sumo Logic Setup Wizard, which appears immediately after you create your Sumo Logic account. If you already have an account, you can access the wizard by selecting **Setup Wizard** from the **Manage** menu at the top of the Sumo Logic application.



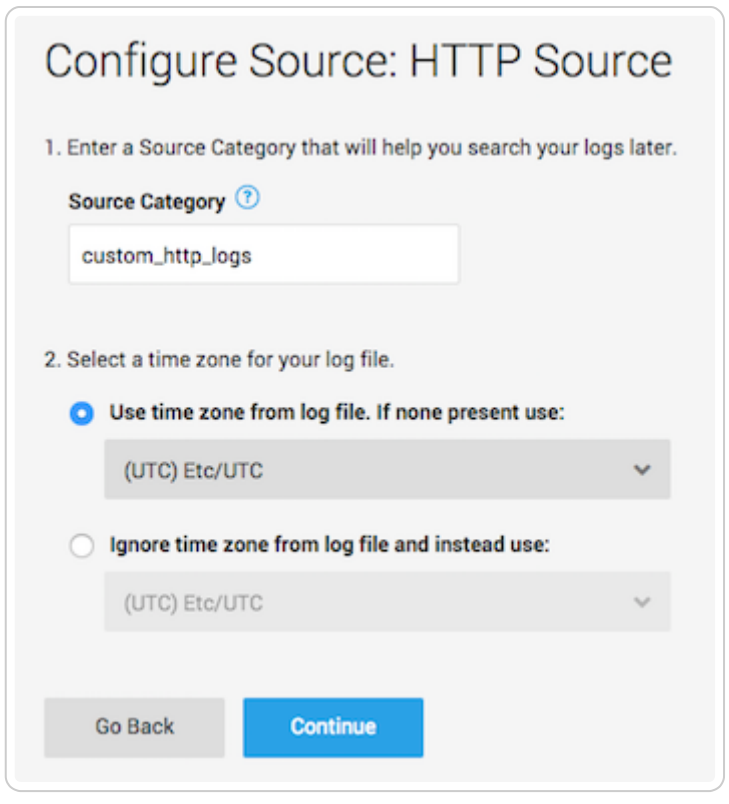
2. Click **Set Up Streaming Data**. The Select Data Type window appears.



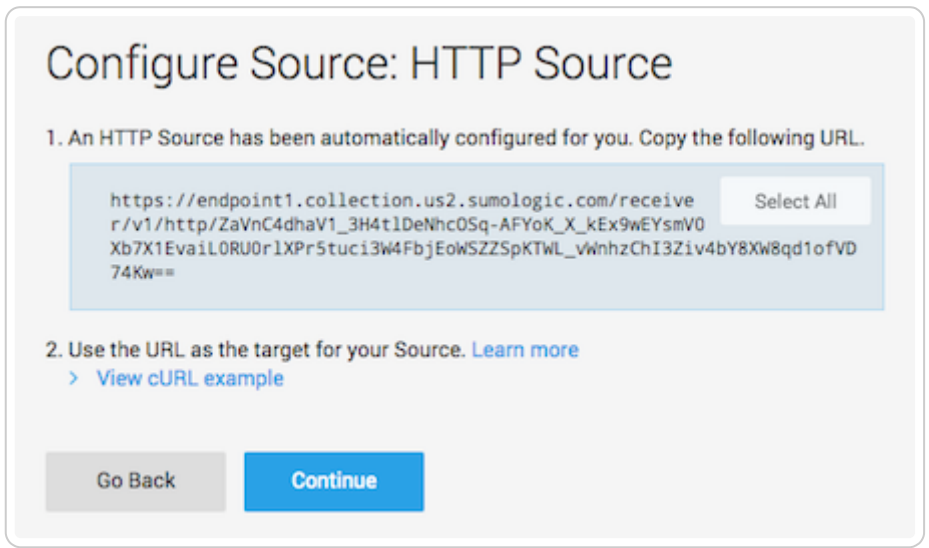
3. Click **All Other Sources**. The Set Up Collection window appears.



4. Click **HTTP Source**. The Configure Source: HTTP Source window appears.



5. In the **Source Category** field, enter a human-readable name for the category (e.g., `fastly_cdn`) and select a time zone for your log file.
6. Click **Continue**. The HTTP Source URL appears.



7. Copy the HTTP Source URL. You will enter this value in the Fastly web interface.
8. Click **Continue**. Sumo Logic will add the new source.

Adding Sumo Logic as a logging endpoint

After you've created a Sumo Logic account and obtained the HTTP Source URL, follow these instructions to add Sumo Logic as a logging endpoint for Fastly services:

1. Review the information in our Setting Up Remote Log Streaming guide.

> **TIP**
>
> Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Sumo Logic **Create endpoint** button. The Create a Sumo Logic endpoint page appears.
3. Fill out the **Create a Sumo Logic endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Collector URL** field, enter the address of the HTTP Source URL you found in the Sumo Logic website.
4. Click the **Advanced options** link of the **Create a Sumo Logic endpoint** page and decide which of the optional fields to change, if any.
5. Fill out the **Advanced options** of the **Create a Sumo Logic endpoint** page as follows:

- In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

Troubleshooting

The Sumo Logic logging endpoint is designed for services with sustained levels of traffic. If you aren't seeing any logs in Sumo Logic, try waiting a bit.

Compute@Edge log streaming: Syslog

Last updated: 2021-09-09

https://docs.fastly.com/en/guides/compute-log-streaming-syslog

Fastly's Real-Time Log Streaming feature for Compute@Edge services can send log files to syslog-based logging software. Syslog is a widely used standard for message logging.

IMPORTANT

This information is part of a limited availability release. For additional details, read our product and feature lifecycle descriptions.

NOTE

Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

Adding syslog as a logging endpoint

Follow these instructions to add syslog as a logging endpoint:

1. Review the information in our Setting Up Remote Log Streaming guide.

TIP

Our developer documentation provides more information about logging with Compute@Edge code written in Rust, AssemblyScript, and JavaScript.

2. Click the Syslog **Create endpoint** button. The Create a Syslog endpoint page appears.

3. Fill out the **Create a Syslog endpoint** fields as follows:
   - In the **Name** field, enter the name you specified in your Compute@Edge code. For example, in our Rust code example, the name is `my_endpoint_name`.
   - In the **Syslog address** field, enter the domain name or IP address and port to which logs should be sent. Be sure this port can receive incoming TCP traffic from Fastly. See the firewall considerations section for more information.
   - In the **Token** field, optionally enter a string prefix (line prefix) to send in front of each log line.
   - From the **TLS** menu, select **No** to disable encryption for the syslog endpoint, or **Yes** to enable it. When you select Yes, additional TLS fields appear.
   - In the **TLS hostname** field, optionally enter a hostname to verify the server's certificate. This should be one of the Subject Alternative Name (SAN) fields for the certificate. Common Names (CN) are not supported. This field only appears when you select Yes from the Use TLS menu.
   - In the **TLS CA certificate** field, optionally copy and paste the certification authority (CA) certificate used to verify that the origin server's certificate is valid. The certificate you upload must be in PEM format. Consider uploading the certificate if it's not signed by a well-known certification authority. This value is not required if your TLS certificate is signed by a well-known authority. This field only appears when you select Yes from the Use TLS menu.

- In the **TLS client certificate** field, optionally copy and paste the TLS client certificate used to authenticate to the origin server. The TLS client certificate you upload must be in PEM format and must be accompanied by a client certificate. A TLS client certificate allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.

- In the **TLS client key** field, optionally copy and paste the TLS client key used to authenticate to the backend server. The TLS client key you upload must be in PEM format and must be accompanied by a TLS client certificate. A TLS client key allows your server to authenticate that Fastly is performing the connection. This field only appears when you select Yes from the Use TLS menu.

4. Click the **Advanced options** link of the **Create a Syslog endpoint** page and decide which of the optional fields to change, if any.

5. Fill out the **Advanced options** of the **Create a Syslog endpoint** page as follows:
   - In the **Select a log line format** area, select the log line format for your log messages. Our guide on changing log line formats provides more information.

6. Click the **Create** button to create the new logging endpoint.

7. Click the **Activate** button to deploy your configuration changes.

Adding separators or static strings

To insert a separator or other arbitrary string into the syslog endpoint format:

1. Create a new header with the following fields:
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, enter any suitable header name (for example, `http.X-Separator`).
   - In the **Source** field, enter any special character or string you want (for example, `"|"`).
2. Reference the new header variable in the log format box for your specific provider (for example, `req.http.X-Separator`).

Syslog facility and severity

The syslog output includes the following facility and severity values:

```
facility: local0
severity: info
```

Firewall considerations

Syslog has limited security features. For this reason, it's best to create a firewall for your syslog server and only accept TCP traffic on your configured port from our address blocks. Our list of IP address blocks is dynamic, so we recommend programmatically obtaining the list whenever possible.

Reference

These articles provide reference information about common Fastly terms and configuration settings.

https://docs.fastly.com/en/guides/reference

Resource limits

Last updated: 2022-03-28

https://docs.fastly.com/en/guides/resource-limits

This guide details Fastly resource limits and summarizes the implications of exceeding those limits.

Cache limits

The cache limits for your account depend on when you became a Fastly customer.

Account created on or after June 17, 2020

If you created your account on or after June 17, 2020, the following cache limits apply.

Item	Limit	Implications
Cache file size (with Segmented Caching enabled)	unlimited	None
Cache file size (without Segmented Caching enabled)	20MB	Exceeding this limit when trying to cache a file results in a `503 Response object too large` error.

Account created prior to June 17, 2020

If you created your account prior to June 17, 2020, the following cache limits apply.

Item	Limit	Implications
Cache file size (with Segmented Caching enabled)	unlimited	None
Cache file size (with streaming miss and without Segmented Caching enabled)	5GB	Exceeding this limit when trying to cache a file results in a `503 Response object too large` error unless Segmented Caching is enabled.
Cache file size (without streaming miss and without Segmented Caching enabled)	2GB	Exceeding this limit when trying to cache a file results in a `503 Response object too large` error unless Segmented Caching is enabled.

Rate and time limits

Item	Limit	Implications
API rate limit	1000 requests/hour	Exceeding this limit results in a `Too many requests` error. The limit is applied to the authenticated user making the request. See API rate limiting for more info.
TLS connections limit	10 minutes	Exceeding this limit results in a `502 gateway timeout` error.

Request and response limits

Item	Limit	Implications
URL size	8KB	Exceeding the limit results in a `414 URI Too Long` error.
Cookie size	32KB	Exceeding the limit results in Fastly stripping the cookie and setting `req.http.Fastly-Cookie-Overflow = "1"`.
Request header size	69KB	Depending on the circumstances, exceeding the limit can result in Fastly closing the client connection abruptly, or it can result in the client either receiving a `502 Gateway Error` response with an `I/O error` in the body, or receiving a `503 Service Unavailable` response with a `Header overflow` error in the body.
Response header size	69KB	Exceeding the limit results in a `503 backend read error`. See Common 503 errors for more info.

Item	Limit	Implications
Request header count	96	Exceeding the limit results in a `Header overflow` error. A small portion of this limit is reserved for internal Fastly use, making the practical limit closer to 85.
Response header count	96	Exceeding the limit results in a `Header overflow` error. A small portion of this limit is reserved for internal Fastly use, making the practical limit closer to 85.
`req.body` size	8KB	Exceeding the limit results in the `req.body` variable being blank. Request body payload is available in `req.body` only for payloads smaller than 8KB. `req.postbody` is an alias for `req.body`.
Surrogate key size	1KB	Exceeding the limit results in purging API failures stating "surrogate key too long, must be less than 1024 bytes." Any keys that exceed the limit will be dropped instead of truncated.
Surrogate key header size	16KB	Exceeding the limit results in no error and any keys past the one that exceeds the limit will be dropped.

Service, domain, and origin limits

Item	Limit	Implications
Services total per account	10	Exceeding this limit results in an `Exceeding max_total_services` error. Contact support@fastly.com to discuss raising this limit.
Origins per service	5	Exceeding this limit results in an `Exceeding max_backends` error. Contact support@fastly.com to discuss raising this limit.
Domains per service	20	Exceeding this limit results in an `Exceeding max number of domains` error. Contact support@fastly.com to discuss raising this limit.
Connections per service	200	Exceeding this limit results in an `Error 503 backend.max_conn reached` error. You can increase this limit as high as 1000 by updating the backend connection setting to limit the connections a single Fastly cache server will make to a specific origin server.

VCL and configuration limits

Item	Limit	Implications
Custom VCL file size	1MB	Exceeding the limit results in a `Content too long` error.
Maximum VCL file size	3MB	Exceeding the limit results in a `VCL is too long` error.
Varnish restart limit	3 restarts	Exceeding the limit results in a `Service Unavailable` error. This limit exists to prevent infinite loops.
ACL container entries count	1000	Exceeding the limit results in an `Exceeding max ACL entries` error. Contact support@fastly.com to discuss raising this limit.
Edge dictionary items count	1000	Exceeding the limit results in an `Exceeding max dictionary items` error. Contact support@fastly.com to discuss raising this limit.

Item	Limit	Implications
Edge dictionary item key length	256 characters	Exceeding the limit results in an `Item key is too long` error.
Edge dictionary item value length	8000 characters	Exceeding the limit results in an `Item value cannot be greater than` error.
Synthetic response characters	No character limit	Synthetic responses have no character limit, but large responses may trigger an error for the custom VCL file size limit.
Vary objects count	200 soft, 400 hard	Exceeding the soft limit results in no error. Newer variants displace the oldest. Active fetches from backends are limited to 400 variants. Exceeding this hard limit results in a `Too many variants` response. Once fetches complete, objects will be removed until the soft limit is reached.

Fastly 101

Fastly 101 is a step-by-step tutorial that shows you how to use Fastly with an example website and domain name. It guides you through the steps of caching and delivering a static website using the Jekyll static site generator, Amazon AWS, and the Fastly CDN.

https://docs.fastly.com/en/guides/fastly-101

1. Introduction

Last updated: 2022-02-03

https://docs.fastly.com/en/fundamentals/1-introduction

If you own a website or application, you need to find the best way to deliver it to users. Using a content delivery network (CDN) can speed up your website or application by caching it closer to users. Fastly's Deliver@Edge CDN provides powerful tools that allow you to customize caching and delivery rules for a fast user experience, all while increasing availability and lowering hosting costs.

To help you get started with Fastly, we've created this step-by-step tutorial. It shows you how to use Fastly with an example website and domain name, and it guides you through the steps of caching and delivering a static website using the Jekyll static site generator, Amazon AWS, and the Fastly CDN. We'll spice things up by using Taco Labs, a fully functional taco recipe website, as an example. By the end of this tutorial, you'll understand how to cache and deliver websites using the Fastly CDN.

Prerequisites

This tutorial assumes that you're familiar with the following technologies and concepts:

- **Static site generators:** We'll use the Jekyll static site generator to output our Markdown source as HTML files.
- **Version control:** We'll use git for version control and GitHub for remote storage and continuous integration.
- **Amazon Web Services (AWS):** We'll use S3 to store our HTML files and serve them as a website, and we'll use Route 53 for DNS.
- **DNS records:** We'll update DNS records for our domain to point to Fastly.
- **Command line interface:** We'll use curl to examine HTTP headers.

How to follow along

Trying things yourself is the best way to learn! To show you how things work, we use a specific domain name (`www.tacolabs.com`) as an example throughout this tutorial. However, to make the most of your experience, we encourage you to fork the example website and follow along by using your own domain name, DNS records, and web hosting provider. Use your own domain names wherever you see our example domain names.

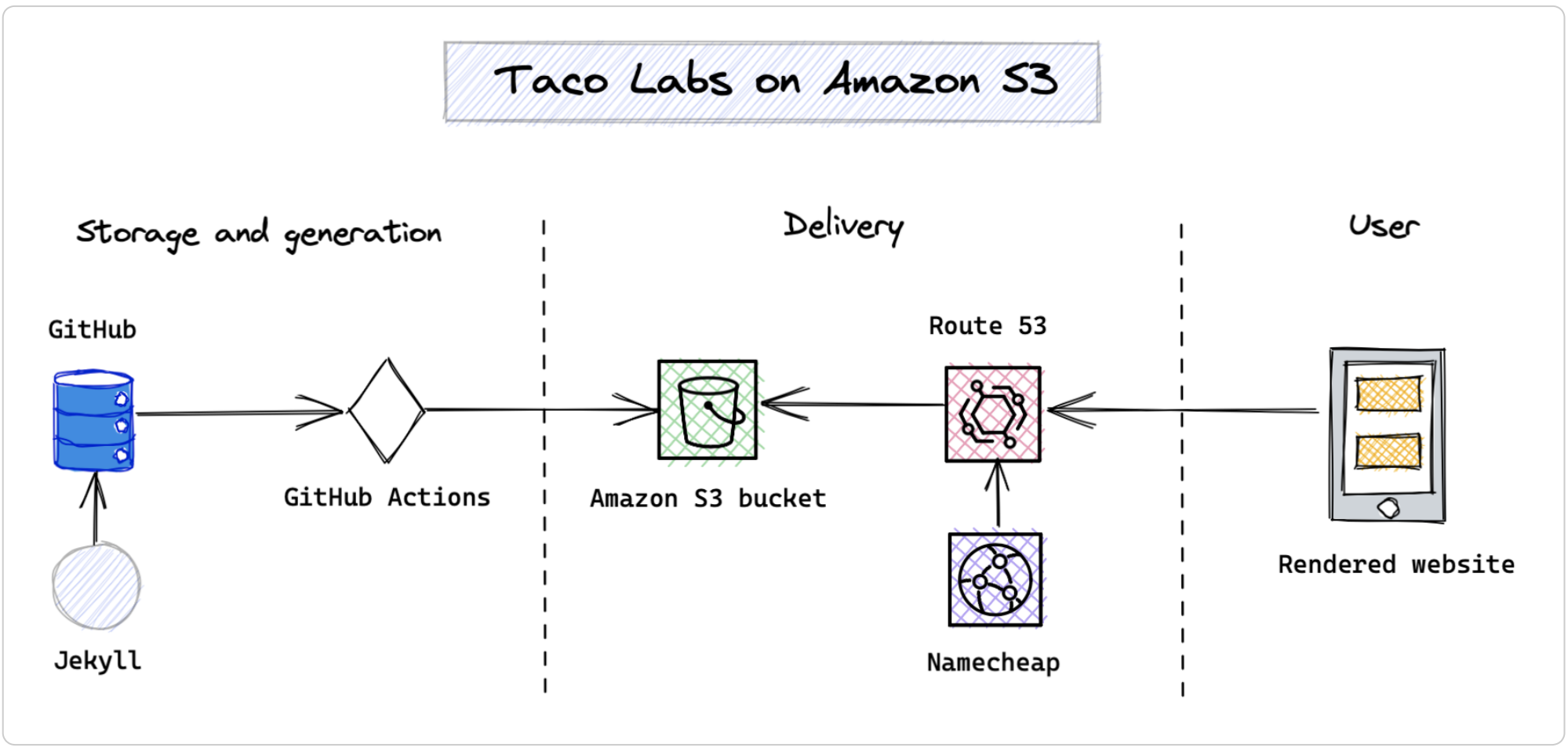
> **NOTE**
>
> We don't recommend using the Taco Labs domain name to follow along. If you use it, you might see errors in the web interface and unexpected output in your Terminal application.

Deciding how to host a static website

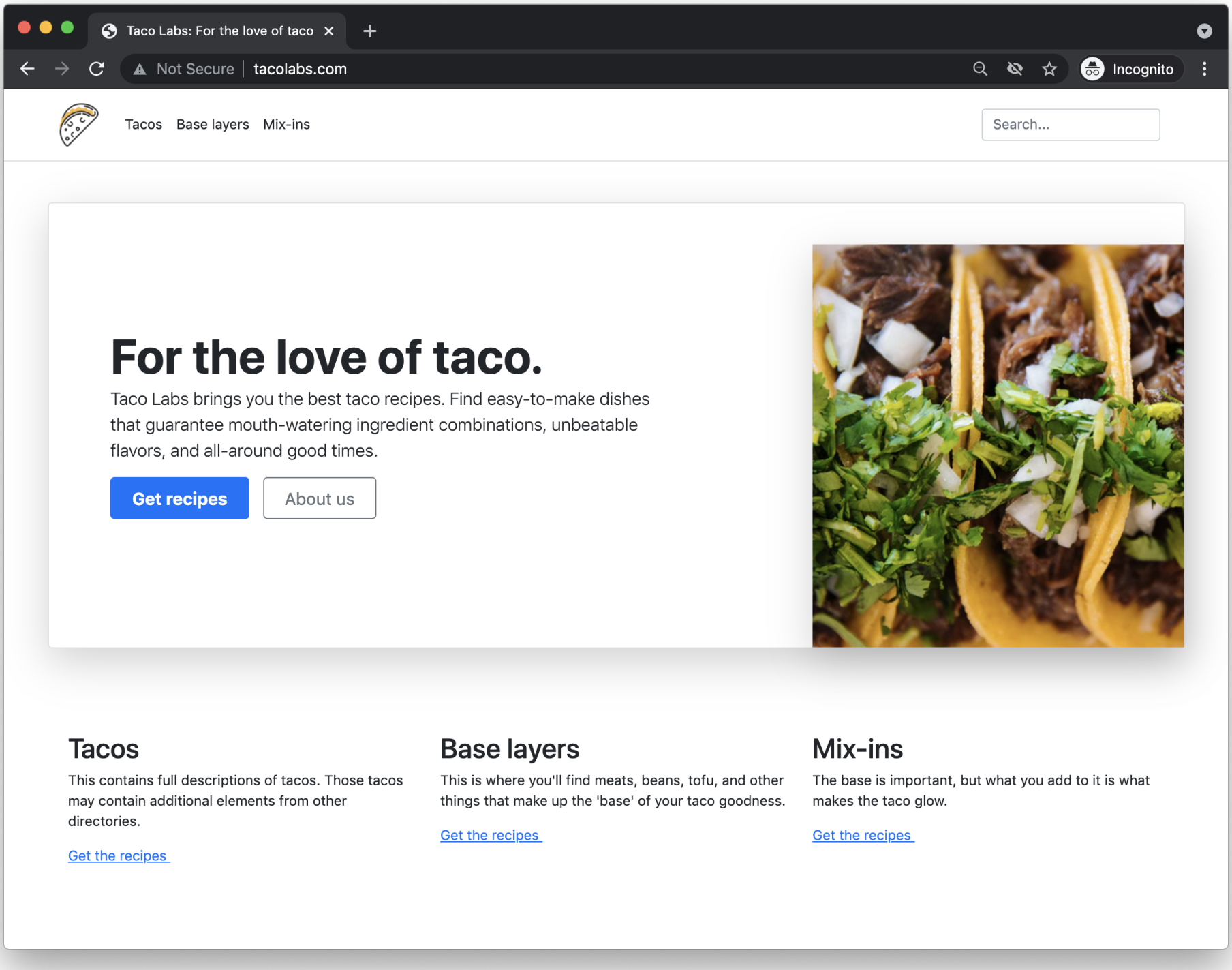
In a real world scenario, we'd already have our website or application developed and hosted somewhere before thinking about using a CDN. For the purposes of this tutorial, it's worth exploring some of the architectural decision making that goes on behind the scenes.

Before we can use Fastly, we need to decide how to host our static website. These days, hosting a static website is an inexpensive and trivial matter thanks to a wide selection of modern services. We could host Taco Labs practically anywhere. However, a closer inspection of hosting services reveals that each has its own unique advantages and disadvantages, some of which are deal breakers.

For example, hosting a static site on a virtual server would provide us with the freedom to use any software and configuration we like, but we'd be saddled with never-ending maintenance tasks. On the other end of the spectrum, GitHub Pages is easy to use, but it's limited to Jekyll and it doesn't allow the use of custom Jekyll plugins. In the end, we decided to host Taco Labs on Amazon S3 using the static website hosting feature.



The diagram above provides an overview of how Taco Labs is generated and delivered to users before we start using Fastly. We store content in Markdown files, use git for version control, and rely on Jekyll to build the website. When we `git push` the main branch to GitHub, a GitHub action automatically builds the site and moves the generated HTML files to our Amazon S3 bucket.



The name servers for `tacolabs.com` at our domain name registrar are pointed at Amazon's Route 53 DNS service. The single A DNS record hosted at Route 53 points `www.tacolabs.com` at our S3 bucket, as shown above.

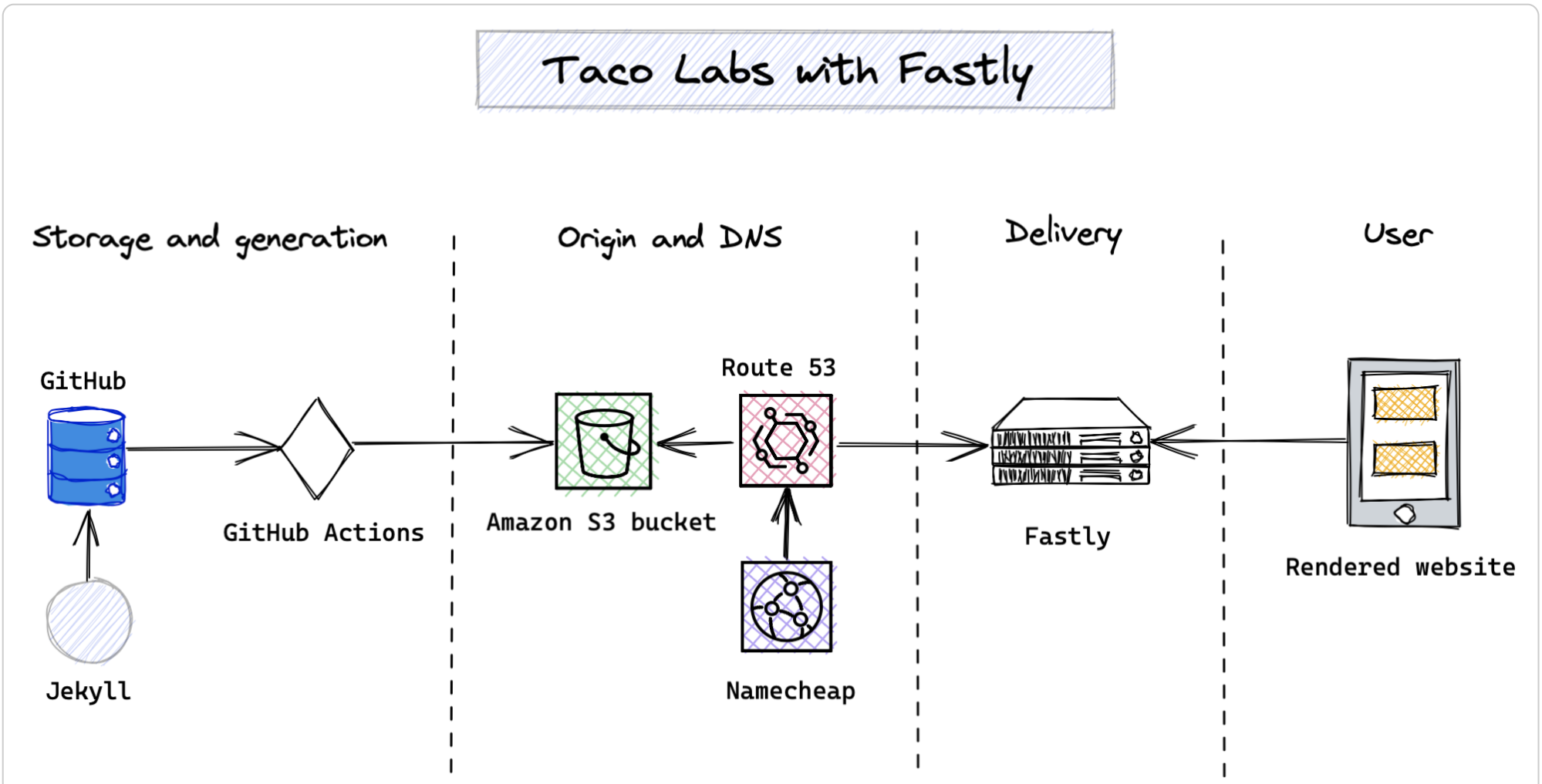
Problems and pitfalls

This setup is simple and reliable, but it also has several disadvantages. The most obvious drawback is the inability to use a Transport Layer Security (TLS) certificate with Amazon S3 — search engines and web browsers will penalize us for that. Another potential issue is the cost associated with traffic spikes. Amazon S3 provides cheap storage, but transfer costs are separate and could become unwieldy if we get a massive influx of visitors or hit with a DDoS attack.

One of the biggest limitations is the S3 bucket, or what we refer to as the *origin*. When we created our S3 bucket, we had to locate it in an AWS region — a single datacenter in one geographic location (in this case, the `us-east-2` region located in the state of Ohio in the USA). All visitors to Taco Labs, regardless of physical location, will need to request our website assets from that datacenter. The users located furthest away from our origin will have the worst experience since it will take more time to transmit the assets over that physical distance. We can do better!

Why use Fastly?

Fastly can make our website's existing implementation better by improving its delivery. Fastly takes our website and *caches* it at data centers all over the world, where it's closer to users. (We refer to these data centers as *points of presence*, or POPs.) Our website will essentially be mirrored at various geographic locations around the world.



The diagram above shows how things will work after we start using Fastly. When a user visits our website, the request will be routed to the nearest Fastly POP instead of the AWS region. That'll result in faster loading times for users.

But wait — there's more! Since Fastly sits between our users and our origin, at what we call the *edge*, we can take advantage of Fastly's numerous other features and its serverless environment. For example, we can use Fastly to secure our website with TLS certificates, manage redirects, log traffic, configure responses, and monitor real-time traffic statistics, all while protecting our origin from DDoS attacks and traffic spikes. And that really just scratches the surface of what you can do with Fastly.

Our initial configuration

Before we move on and start using Fastly to deliver our website, let's take a snapshot of our current configuration:

- **Domain:** `www.tacolabs.com`
- **S3 bucket name:** `www.tacolabs.com`
- **Origin hostname:** `www.tacolabs.com.s3-website.us-east-2.amazonaws.com`
- **An alias DNS record:** `www.tacolabs.com` to `s3-website.us-east-2.amazonaws.com`

---

2. Getting started with Fastly

Last updated: 2022-03-01

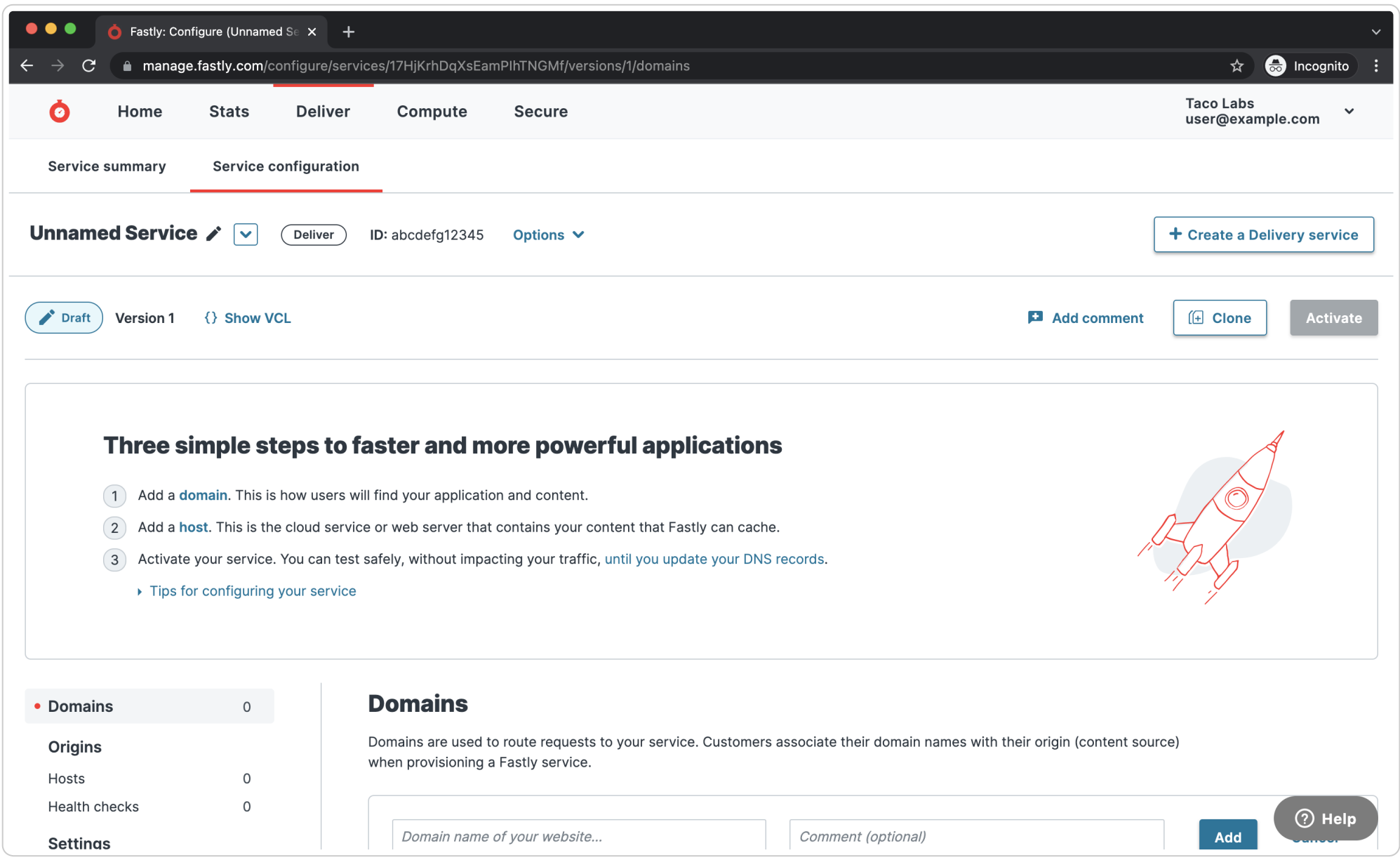
https://docs.fastly.com/en/fundamentals/2-getting-started-with-fastly

This page is part of Fastly 101, a step-by-step tutorial that shows you how to use Fastly with an example website and domain name. It guides you through the steps of caching and delivering a static website using the Jekyll static site generator, Amazon AWS, and the Fastly CDN. For more information, see the introduction.

Let's get started by creating a Fastly account and logging in to the Fastly web interface. We can sign up for a free trial — no credit card information is required. After we sign up and verify our account, we'll see the screen shown below.

NOTE

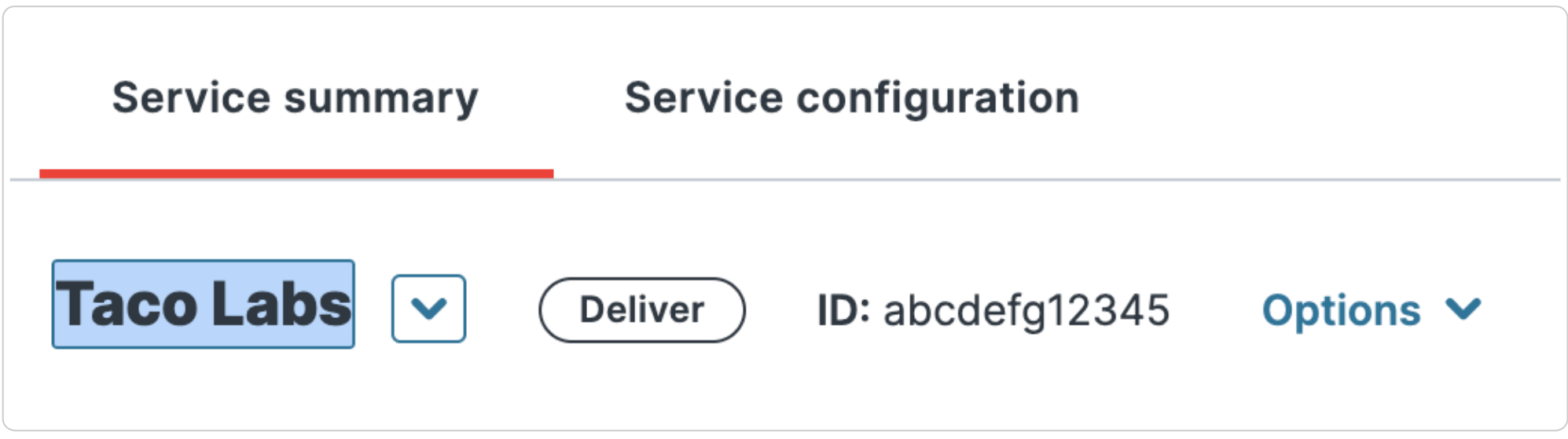
Creating a Fastly account and adding configuration settings won't change anything related to our production website. Traffic will continue flowing to our S3 bucket until we update our DNS records.



Changing the service name

Fastly created a *service* for us when we signed up for our account. We're looking at the *service configuration* in the screenshot above. We can have multiple services in our Fastly account, each of which can correspond to a different website or application. The service configuration holds all of the settings for our service — things like domain names, origins, headers, cache settings, and more.

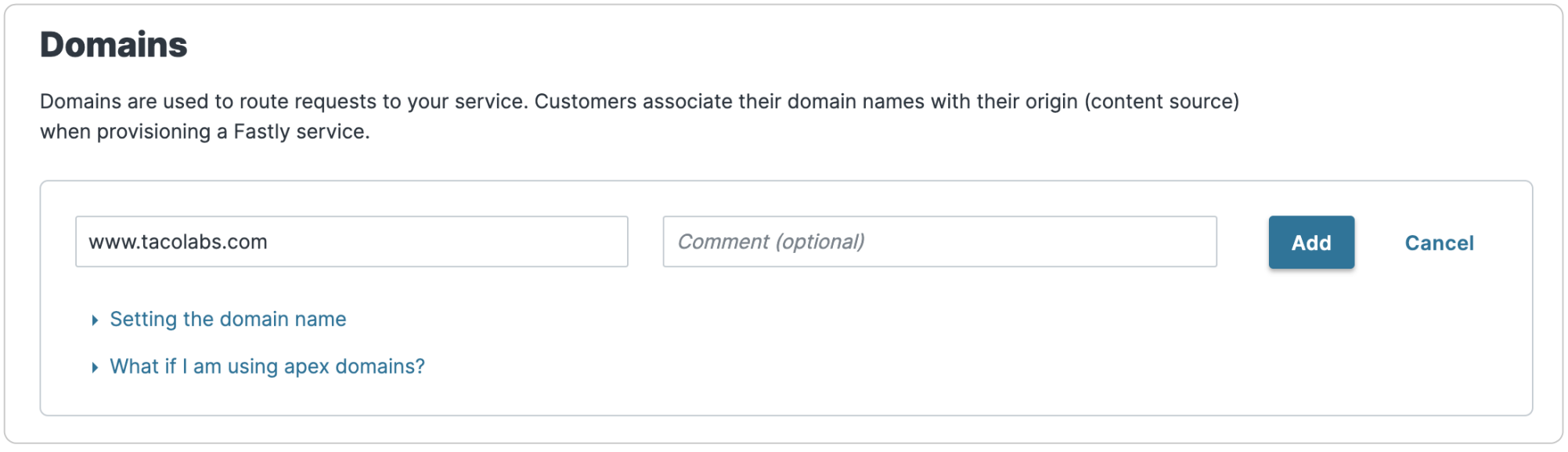
Let's rename the service by clicking on its current name (`Unnamed Service`) and giving the service a memorable name, like `Taco Labs`, as shown below.



We have only one service right now, but we might have more in the future. Giving our service a memorable name can help us distinguish it from our other services, all of which will be displayed together on the home (All services) page.

Adding the domain

Now we're ready to add the domain to our service configuration. The domain is the public URL we want users to visit. Setting this lets Fastly know where traffic to our service will originate from. Type `www.tacolabs.com` in the domain field and click **Add** to save the domain to our service configuration, as shown below.

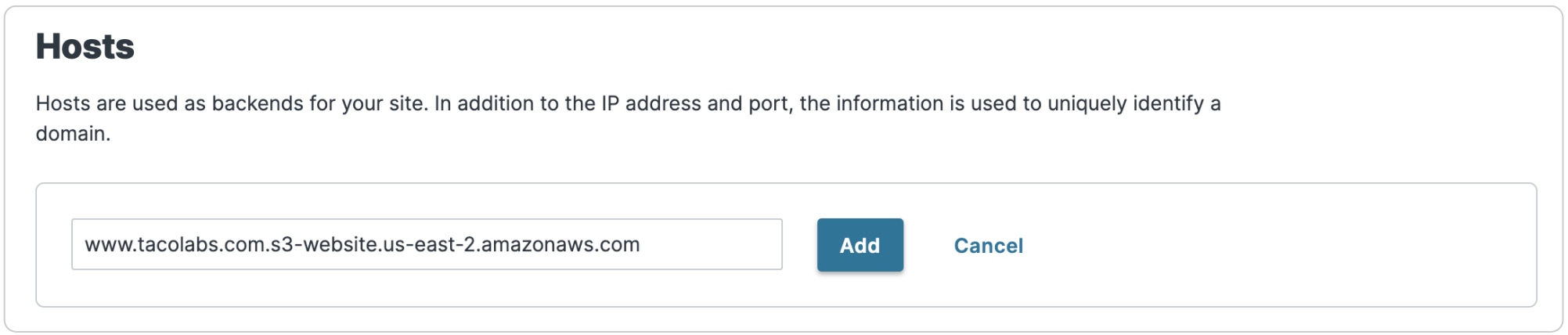


NOTE

You won't be able to add `www.tacolabs.com` as a domain in your service since we've already added it. Fastly doesn't allow multiple services to use the same domain.

Adding the origin hostname

Next, we'll add our origin's hostname to our service configuration so Fastly knows where to pull content from. We'll click the **Hosts** link on the sidebar, and then enter the public URL of our S3 static website (`www.tacolabs.com.s3-website.us-east-2.amazonaws.com`), as shown below.

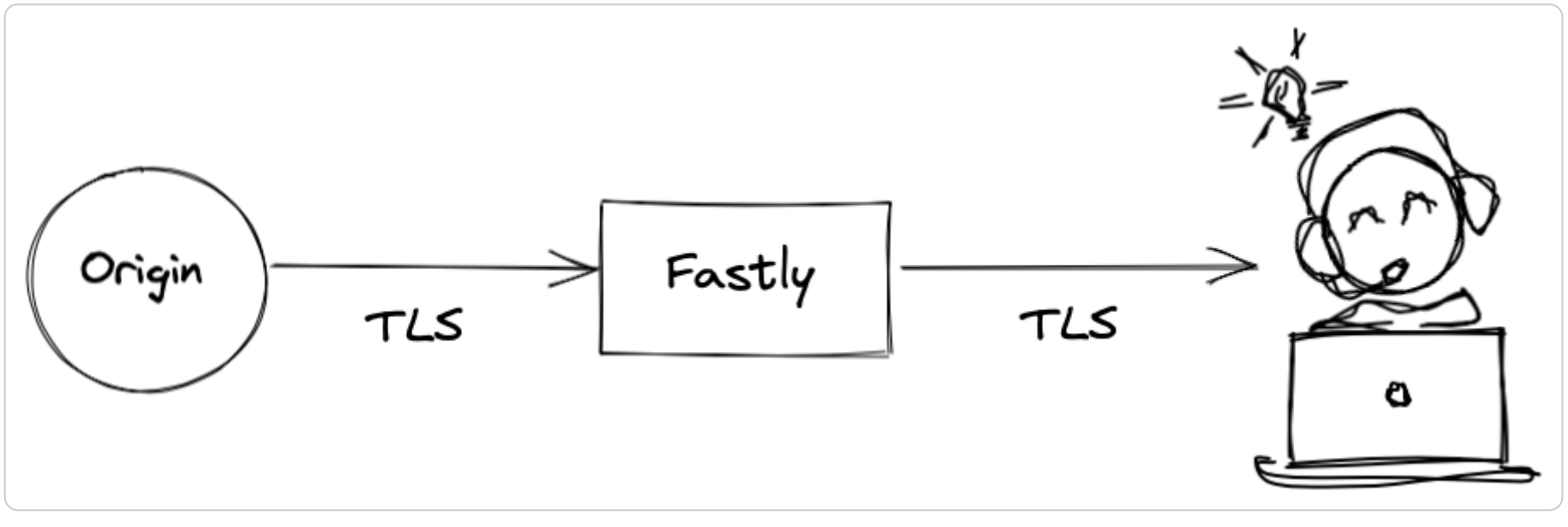


TIP

Finding your origin's hostname can take a bit of sleuthing. If you're using a cloud service, such as AWS or Google Cloud Platform, you can sometimes find the hostname in the admin console, but not always. In this case, because we're using AWS with the S3 static website hosting feature, our origin hostname is the public URL for our S3 website (`www.tacolabs.com.s3-website.us-east-2.amazonaws.com`). If we were using a stock S3 bucket without the static website hosting feature, our origin hostname would follow the format described in our Amazon S3 integration guide (`www.tacolabs.com.s3.us-east-2.amazonaws.com`). And if our origin was a server, the origin hostname would probably be an IP address.

Disabling TLS for the origin connection

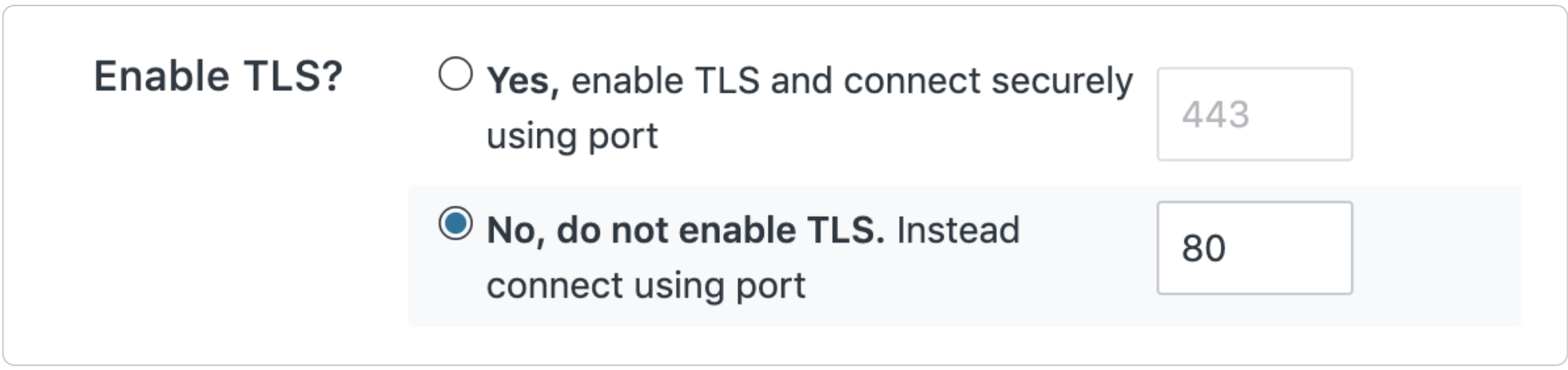
There are two connections we can secure with TLS: The connection between our origin and Fastly, and the connection between Fastly and our users. You can see the difference between the two connections in the diagram shown below.



Fastly automatically enabled TLS between our origin and Fastly when we entered a hostname in the web interface. (If we had entered an IP address, TLS would have been automatically disabled.) Since S3's static hosting feature doesn't support TLS, we need to adjust our origin settings to disable TLS between our S3 bucket and Fastly. We'll set up TLS between Fastly and our users later.

> **TIP**
>
> Encrypting connections is always a good idea! In a real-world situation, you'll want to encrypt connections between Fastly and your origin whenever possible. Depending on your origin, you may even be able to install a free certificate from Let's Encrypt.

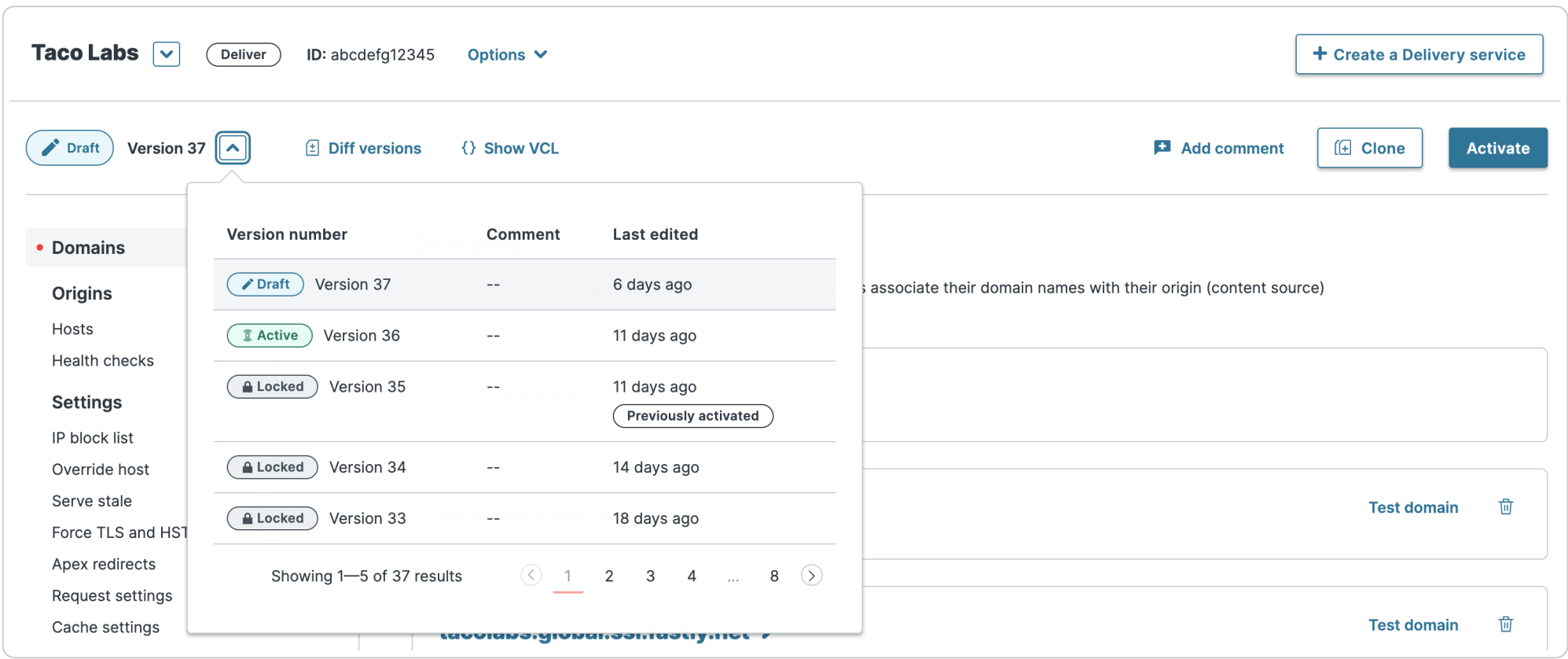
Click the pencil icon to edit the origin settings and select **No, do not enable TLS**, as shown below. Then click the Update button to save our settings. All set!



Understanding service versioning

We're ready to activate our service configuration, but before we do, let's talk about how Fastly manages changes to service configurations. This is something that confuses a lot of new users, so it's worth exploring how it works now, before we go any further.

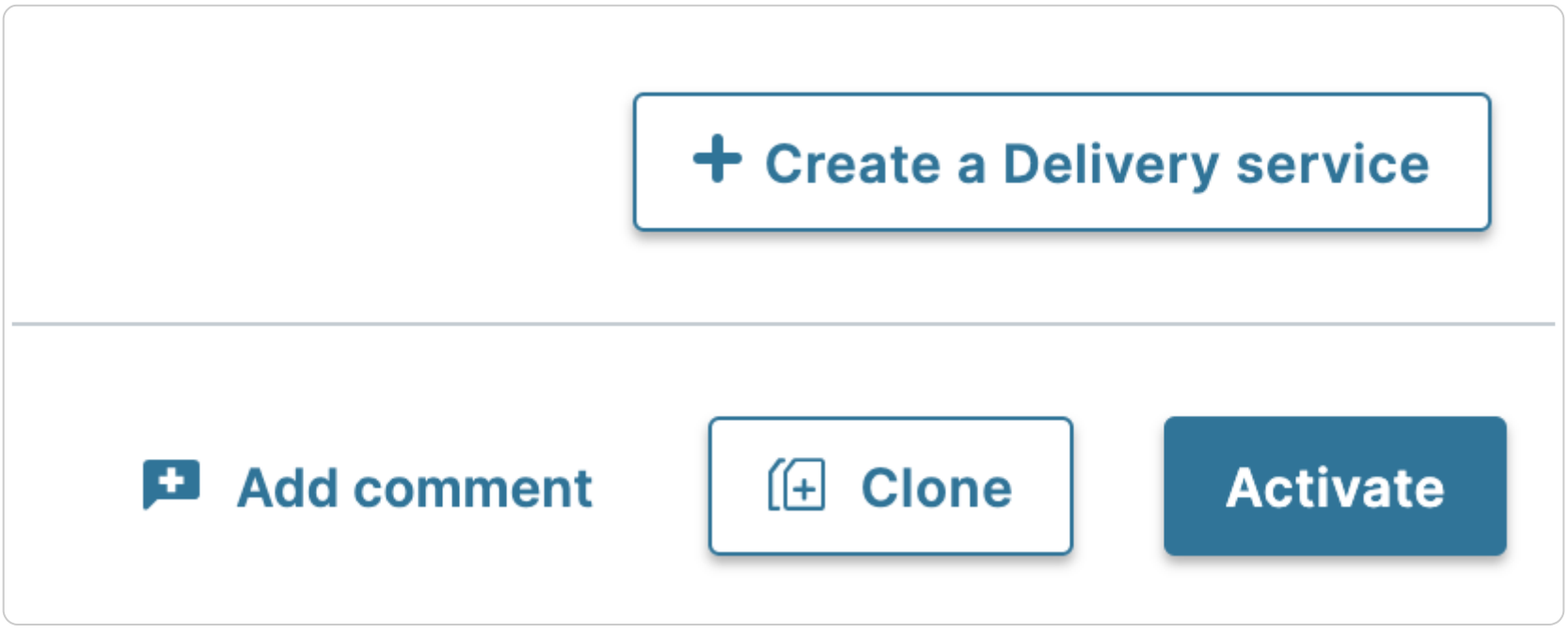
Fastly versions service configurations. It's a bit like having version control principles applied to our Fastly service settings. So far in this tutorial, we've been working on a *draft* service configuration. Once we activate our draft service configuration, Fastly will push the settings to production and lock our service configuration. To make changes, we'll need to *clone* the active version and edit a new version.



Each version of a service configuration is assigned a version number, as shown above. You can't edit previous and active service configurations, but you can clone a previous version to change it in a new version. You can use the web interface to add comments to a version, show differences between two different versions, and roll back to a previous version of a service configuration. To learn more about services and how they work, see our documentation on working with services.

Activating a service configuration

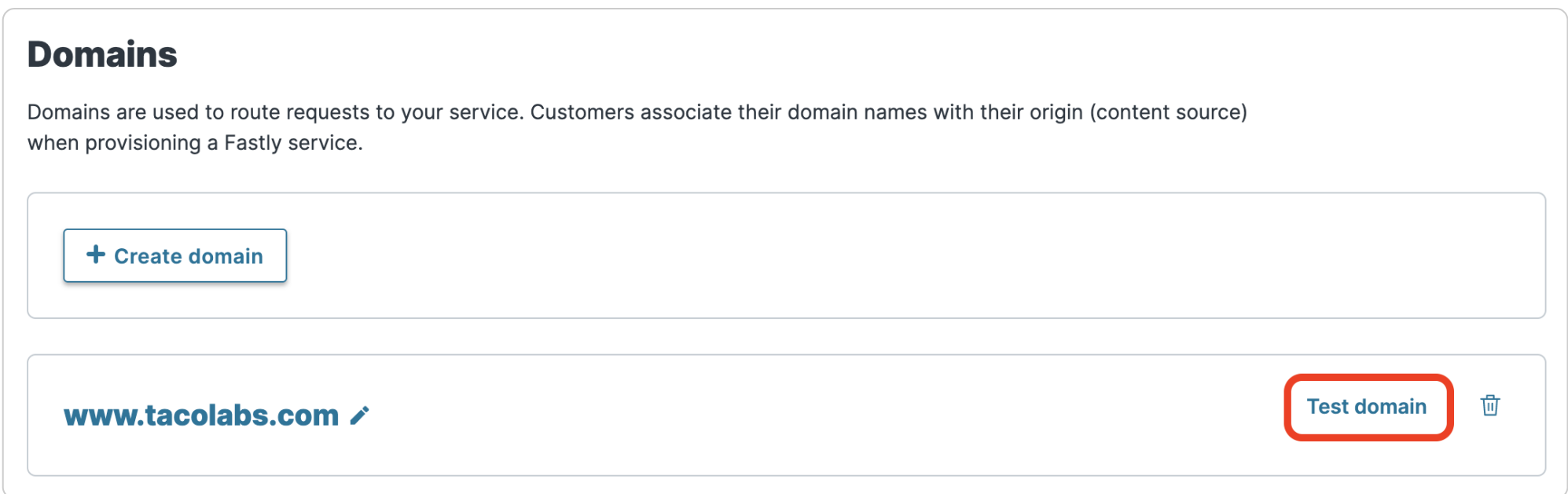
Let's activate our service configuration so we can preview how Fastly caches and delivers our website. Click the **Activate** button in the top-right corner of the screen, as shown below. Remember, this won't impact our production website in any way — we're not changing our DNS records yet.



Fastly will lock our service configuration and push the settings to production.

Previewing our website

It can take a couple of minutes for Fastly to propagate the service configuration changes. Once the changes are live, we can preview how Fastly caches and delivers our website. Fastly provides a test domain name for every service. We can find a link to this on the Domains page, as shown below.



Let's click the link to open it in our web browser. Boom! There's our website at `http://www.tacolabs.com.global.prod.fastly.net/`. Fastly pulled our website from our S3 static site, cached it on a POP server, and delivered it through the test domain.

Checking cache

Even at this early stage, it's a good idea to start checking the cache and inspecting the HTTP headers to see how Fastly is delivering our website. There are two ways of doing this. We can use the web interface to check the cache status of an object, or we can use a command line utility called curl.

First, let's try using the command line interface. Curl is installed by default on most Unix and Linux-based systems. Open a terminal application and enter the following command:

```
$ curl -svo /dev/null -H "Fastly-Debug:1" http://www.tacolabs.com.global.prod.fastly.net
```

We'll see the following in the output:

```
< Age: 585551
< X-Served-By: cache-sna10742-LGB
< X-Cache: HIT
< X-Cache-Hits: 1
```

The `X-Cache: HIT` tells us the object is in Fastly's cache. (If we ran the command before visiting the website, we'd see `X-Cache: MISS` — that would indicate that our website was not yet in Fastly's cache.) We can also see the current age of the object in cache and the cache node that served the content to our computer.

> **TIP**
>
> Astute readers will notice that we added a header to the Curl command. The `Fastly-Debug` header is proprietary to Fastly. When this header is present, it prompts Fastly cache servers to output additional response headers.

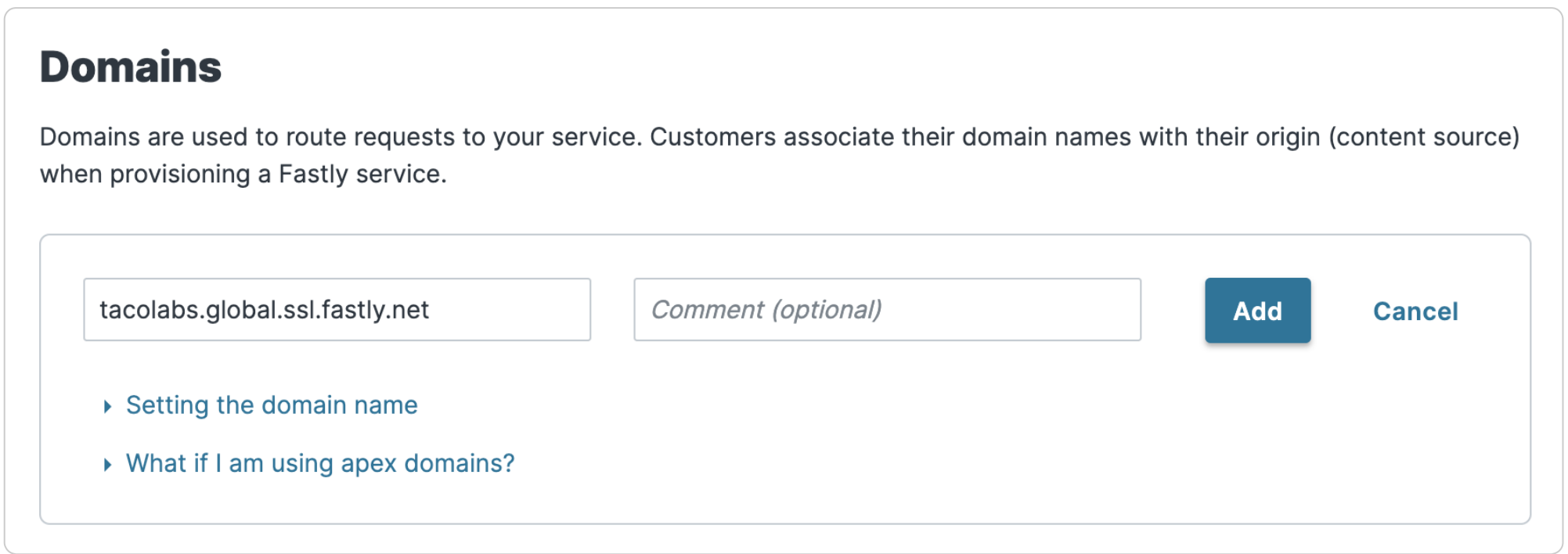
We'll continue checking cache and examining headers throughout this tutorial. If you're interested in learning more, see our guide on checking cache and our HTTP header reference documentation.

Securing our site with free TLS

You might have noticed that the URL of our preview site, like that of the URL of our S3 static site, isn't protected by TLS. We can fix that by using Fastly's free TLS option. That will give us another preview URL for our service, one that's designed to use TLS by default.

Before we can add the new TLS domain, we need to clone the active version of our service configuration. Click **Clone** to create a new draft version. Then, on the Domains page, click the **Create Domain** button and add the new domain, as shown below. We'll use Fastly's shared TLS domain (`<name>.global.ssl.fastly.net`) and put `tacolabs` at the beginning, so the full URL will be `tacolabs.global.ssl.fastly.net`.

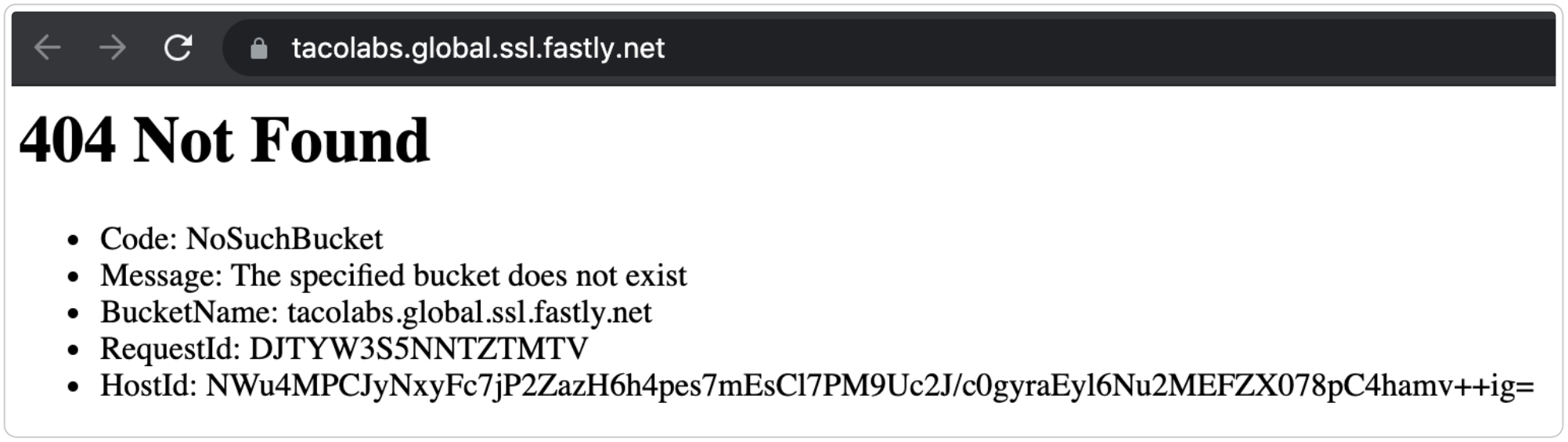
> **TIP**
>
> If you're following these instructions with your own domain name, you can substitute any word for `tacolabs` as long as it's unique and isn't a dot-separated name (e.g., `www.example.org.global.ssl.fastly.net` wouldn't work).



Let's click the **Add** button to add the domain, and then click the **Activate** button to activate the new version of the service configuration. Fastly will deploy our changes and then we'll be able to see our website at https://tacolabs.global.ssl.fastly.net. The connection will finally be encrypted! Or will it?

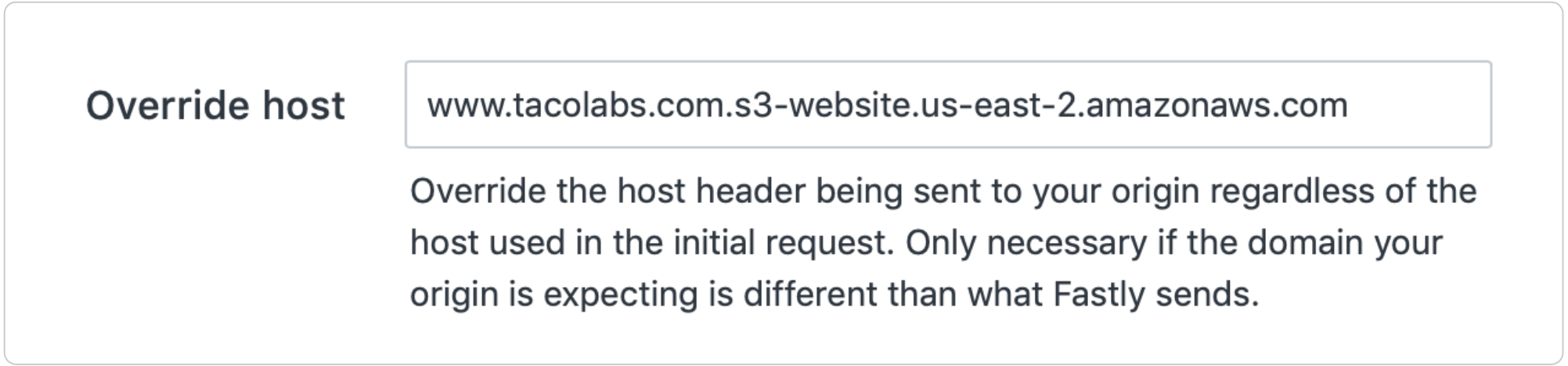
Overriding the Host header for the origin

As it turns out, there's a slight problem with the version of the service configuration we just activated. When we visit https://tacolabs.global.ssl.fastly.net, we see the page shown below. What's wrong?



We talked briefly about HTTP headers earlier. Now we need to set one to get Amazon to route our request correctly. In this case, we'll specify an *override host* by setting an override Host header at the origin level. We need to do this because Amazon is expecting a different host header than the one Fastly is sending.

Let's click the **Edit configuration** button to clone our service and create a new draft version. Then, on the Origins page, click the pencil icon to edit the origin settings. Just above the Advanced options section, we'll enter our origin hostname (`www.tacolabs.com.s3-website.us-east-2.amazonaws.com`) in the **Override host** field, as shown below, and then click the **Update** button.



Click the **Activate** button to activate the new version of the service configuration. Fastly will deploy our changes. This might take a few minutes. Then, we'll be able to see our website at https://tacolabs.global.ssl.fastly.net.

---

3. Adding niceties

Last updated: 2022-02-04

https://docs.fastly.com/en/fundamentals/3-adding-niceties

This page is part of Fastly 101, a step-by-step tutorial that shows you how to use Fastly with an example website and domain name. It guides you through the steps of caching and delivering a static website using the Jekyll static site generator, Amazon AWS, and the Fastly CDN. For more information, see the introduction.

In the previous section, we created a Fastly service, added our domain and origin hostname, activated a Fastly service configuration, used the curl command line utility to verify that Fastly is caching our website, and used free TLS to create a secure preview domain name.

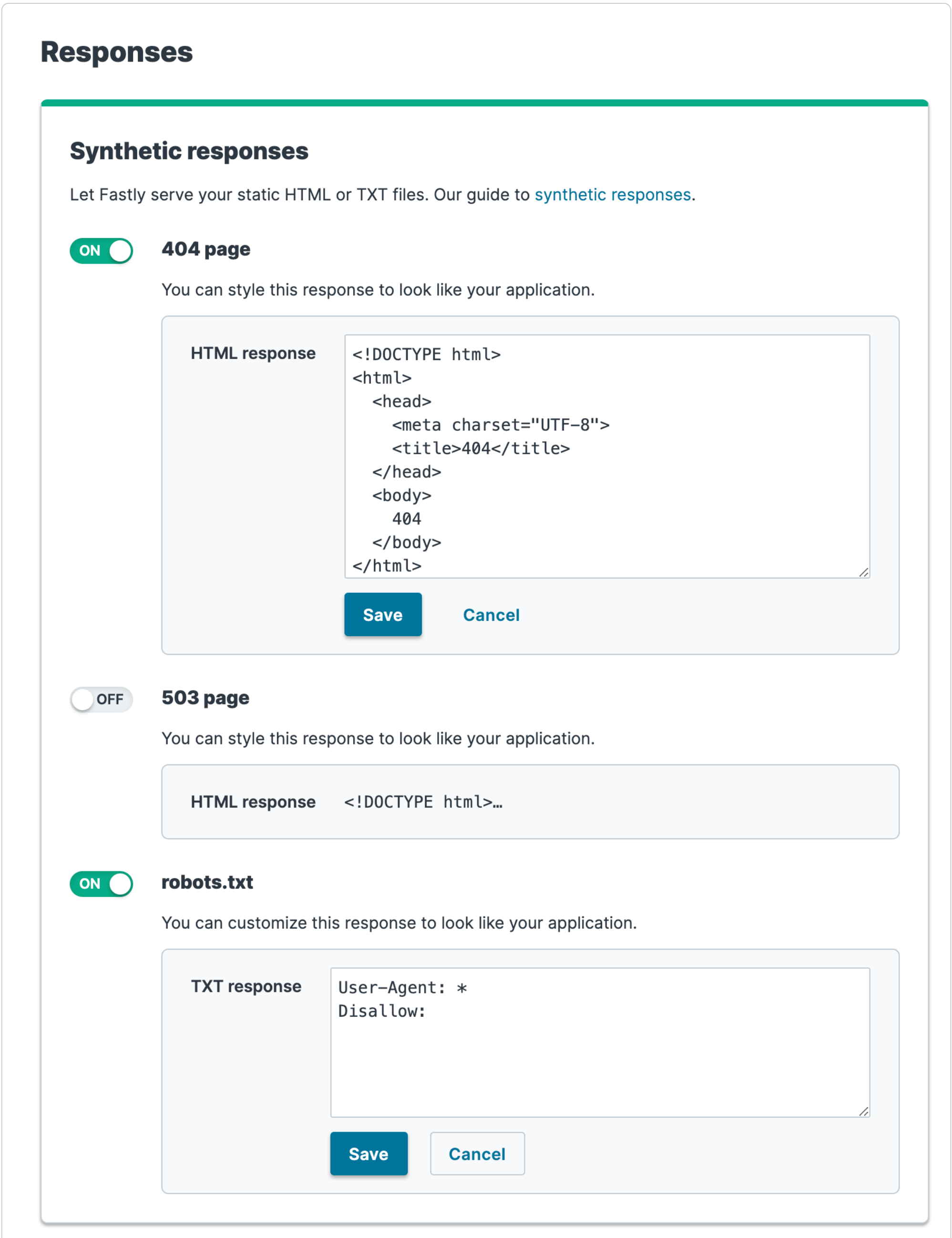
We could probably stop here if the domain didn't make a difference. But there's so much more to explore! Fastly can do a lot more than just cache our website.

In this section, we'll explore some of the edge features available to our Fastly service. For example, we'll configure Fastly to serve synthetic responses to certain types of requests, enable compression of our assets, and configure a streaming logging endpoint so we can see who's visiting our website.

Configuring synthetic responses

We can configure Fastly to serve synthetic responses when Fastly receives an error code from our origin or when the request matches a condition. This comes in especially handy for things like 404 pages and robots.txt files. Fastly provides several quick configurations for common responses, but you can also configure custom responses based on HTTP status codes or conditions. See our responses tutorial for more information. We'll discuss conditions later in this tutorial.

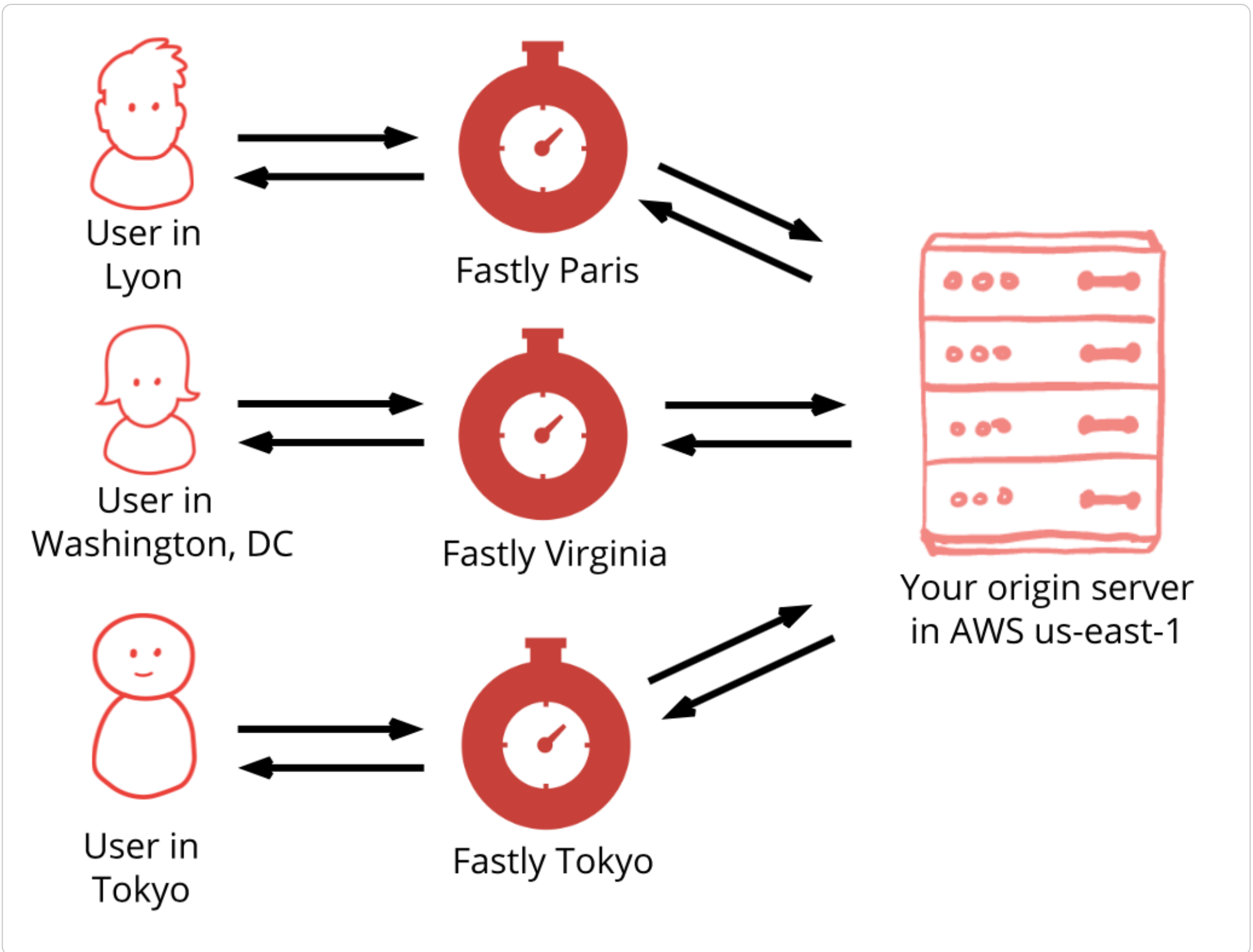
Let's configure synthetic responses for our 404 page and our robots.txt page. Click the **Edit configuration** button to clone the service and create a new draft version. Then, on the Content page, click the **On** switches next to **404 page** and **robots.txt**, as shown below. You can edit the default responses before clicking the **Save** buttons.



Click the **Activate** button to activate the new version of the service configuration. Now, when we visit https://tacolabs.global.ssl.fastly.net/robots.txt, we see the content of our robots.txt synthetic response.

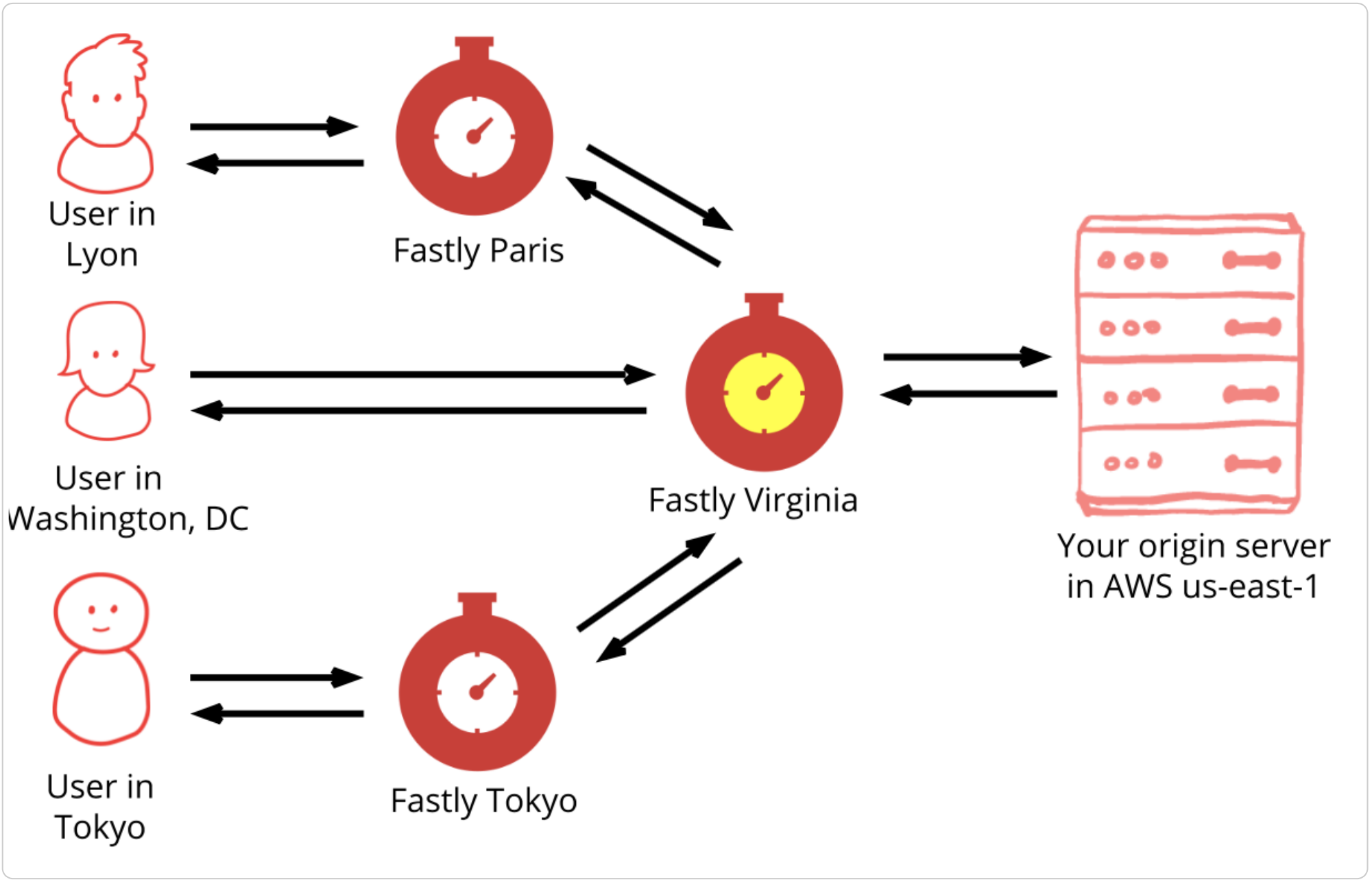
Enabling shielding

By default, Fastly POPs pull resources directly from your origin, as shown below. It can take a while for the cache to populate since every POP has to individually fetch the resources from your origin. The spike in traffic to your origin is temporary and will only last as long as it takes to populate the cache.

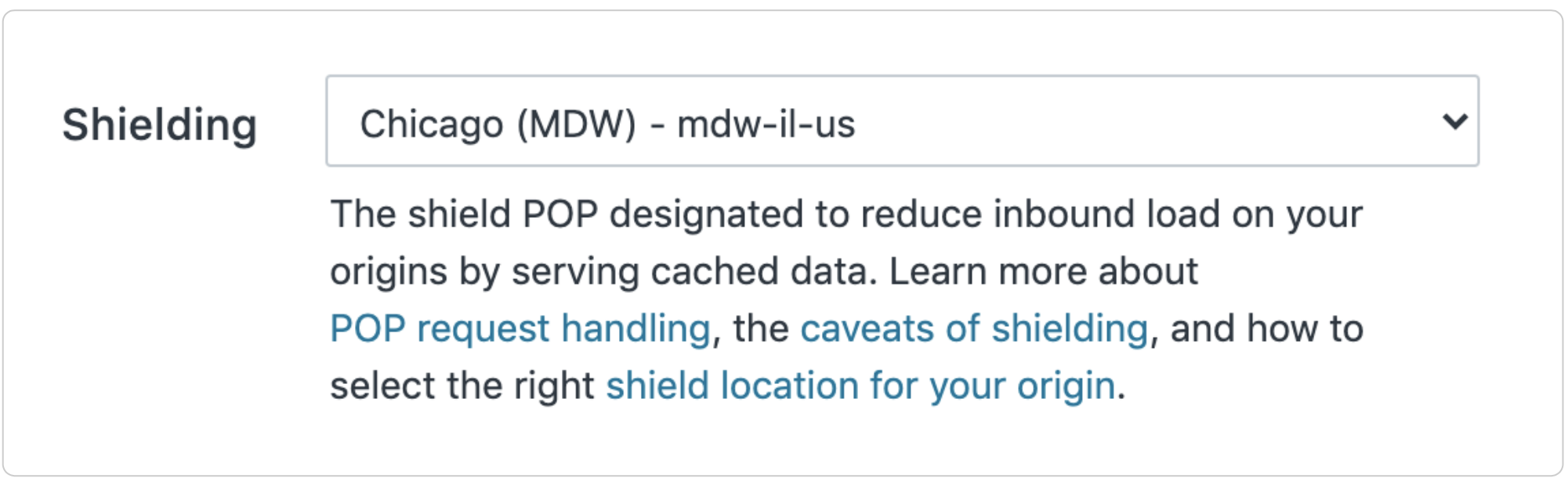


That method works, but Fastly provides a more efficient way of populating the cache. The *shielding* feature allows you to designate a single POP to handle all of the requests to your origin. Enabling shielding can reduce the load on your origin and reduce your web hosting expenses by ensuring that requests to your origin come from the shield POP, as shown below.

> **TIP**
>
> Be sure to read our shielding documentation before enabling this feature. Shielding can affect traffic, hit ratios, and performance.



Let's enable shielding for Taco Labs. Click the **Edit configuration** button to clone our service and create a new draft version. Then, on the Origins page, click the name of our origin to edit the origin's settings. On the Edit this host page, select a shield POP, as shown below. In this case, we'll select **Chicago (MDW)** since this is the closest POP to our AWS region, `us-east-2`. Click the **Update** button to save the changes.



TIP

Generally speaking, we recommend selecting a POP close to your origin. This allows faster content delivery because Fastly optimizes requests between the shield POP and the edge POP (the one close to the user making the request). To help you choose a POP location close to your origin, we provide several lists for AWS and Google Cloud Services (GCS). See our documentation on choosing a shield location for more information.

Now we can activate! Click the **Activate** button to activate the new version of the service configuration.

Checking the headers

We've enabled shielding for our service, but how can we tell if it's working? Let's check the HTTP headers again. Open a terminal application and enter the following command:

```
$ curl -svo /dev/null -H "Fastly-Debug:1" https://tacolabs.global.ssl.fastly.net
```

We'll see the following in the output:

```
< X-Served-By: cache-mdw17347-MDW, cache-bur17582-BUR
< X-Cache: MISS, MISS
< X-Cache-Hits: 0, 0
```

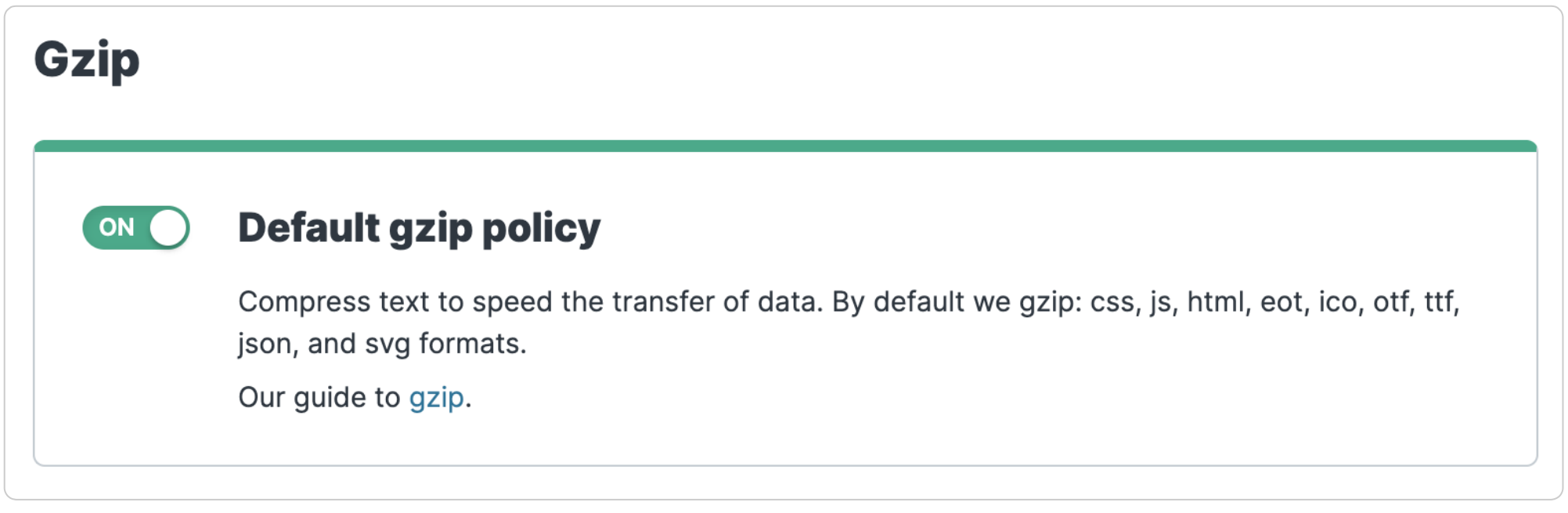
We know shielding is enabled because two cache nodes are displayed in the `X-Served-By` header. The `X-Cache` header (`MISS, MISS`) tells us that the object was not in cache at the shield or the edge node, so Fastly fetched the website from our origin. The next time we run the curl command or load the website, Fastly will load the website from cache.

> **TIP**
>
> For more information about the possible values of the `X-Cache` header, see our shielding debugging documentation.

Using compression

Taco Labs is a static website, but its performance could still benefit from *compression*. Using gzip compression at Varnish can reduce the size of assets by compressing them before they're sent to the visitor's web browser, effectively speeding up the delivery of the website. When we enable this feature in the Fastly web interface, Fastly will dynamically fetch content from our origin, compress it, and then cache it.

Let's click the **Edit configuration** button to clone our service and create a new draft version. Then, on the Content page, click the **On** switch next to **Default gzip policy**, as shown below.



> **TIP**
>
> The default setting is good enough for most websites since it compresses most common file types. If you have specific file types you need compressed, you can set up an advanced gzip policy.

Now we can click the **Activate** button to activate our service configuration.

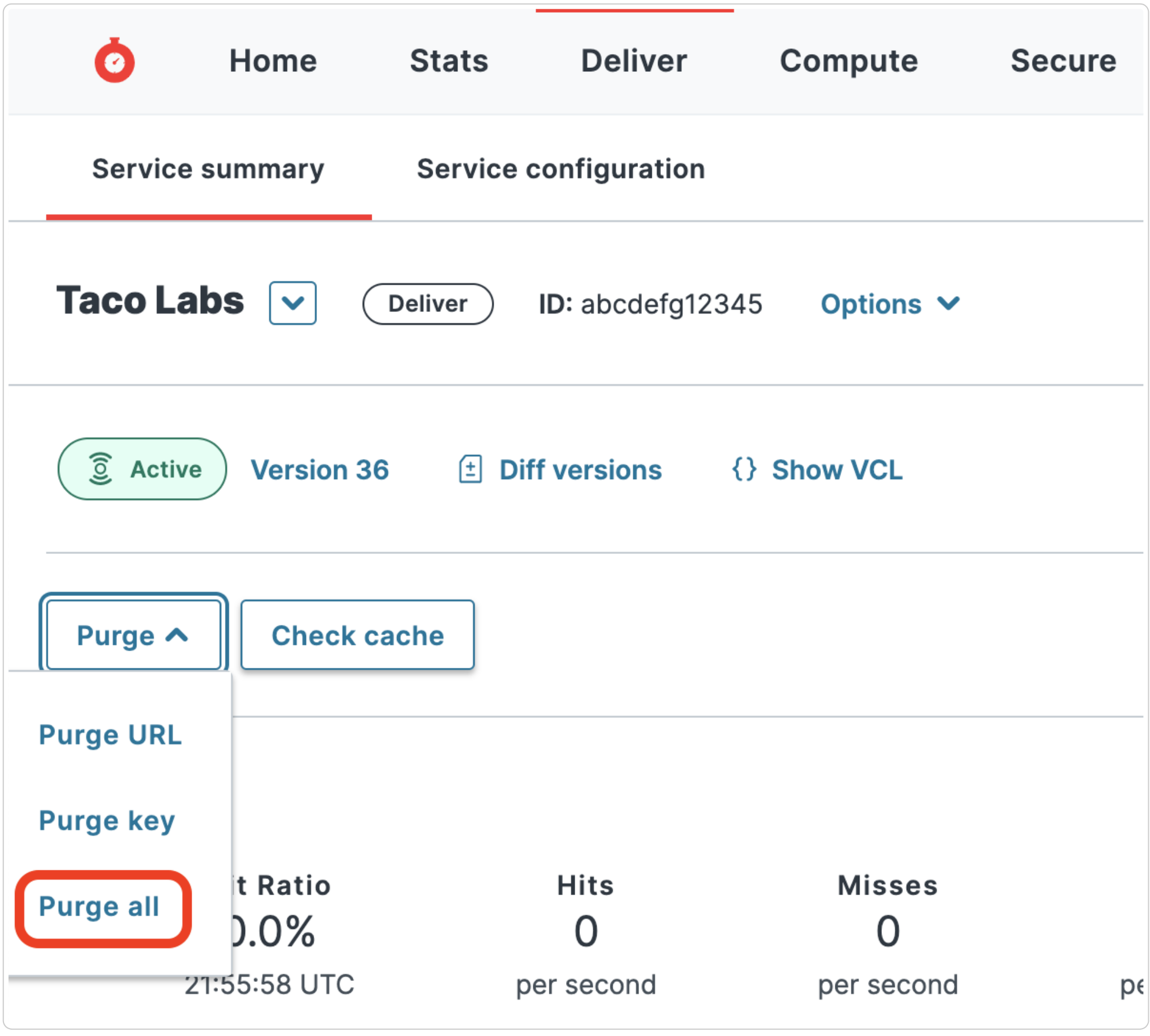
Purging the website

We've enabled gzip compression, but we can't yet test our website to see if it's being compressed. To understand why, recall that we recently enabled shielding. Since the shield POP has cached the uncompressed version of our website, the newly compressed version of our website won't be live yet. How can we tell the shield POP and all of the other cache nodes to remove the cached version of our website and fetch the new, compressed version of our website?

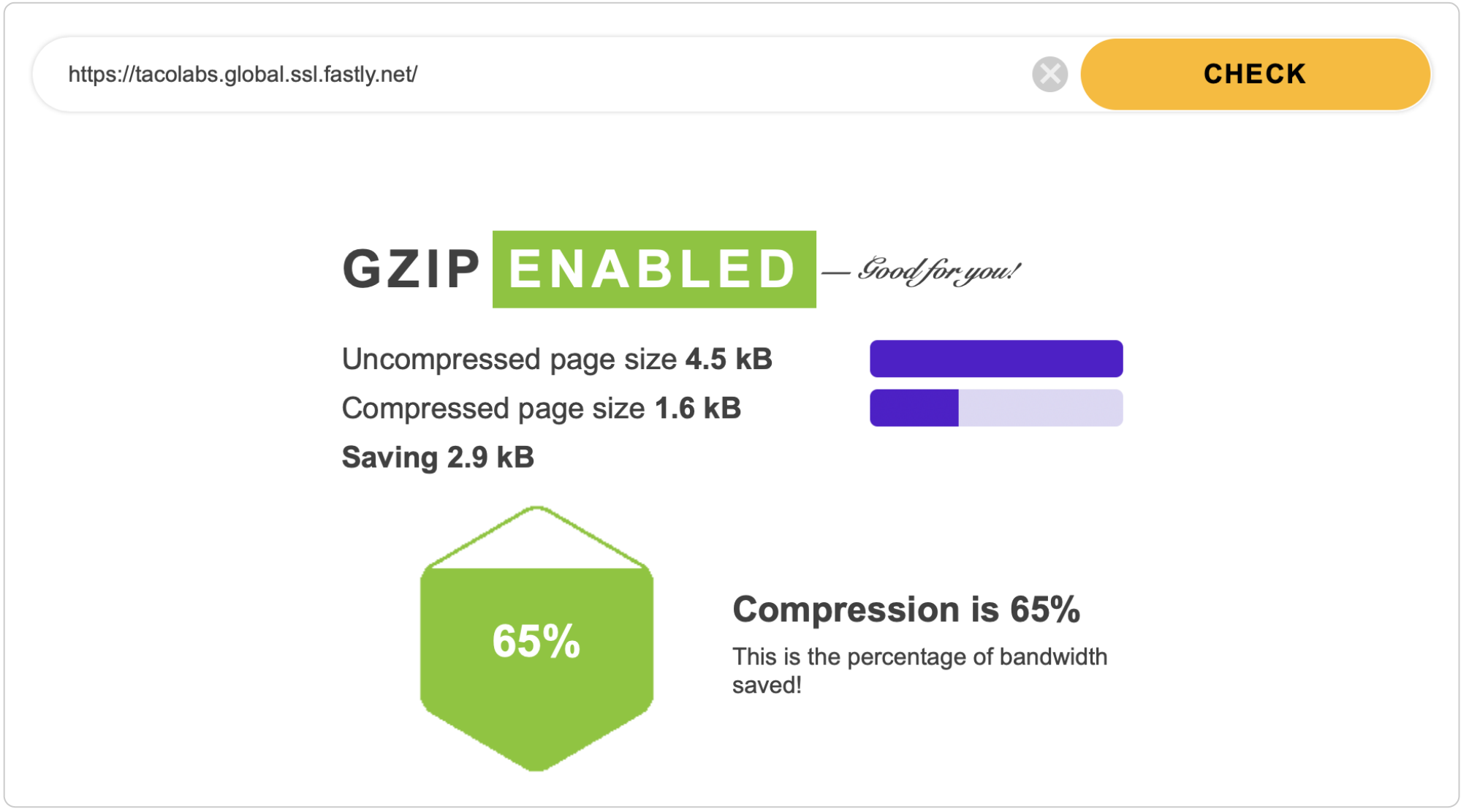
In a word, *purging*. We can tell Fastly to purge some or all of our website's content from cache by using one of the provided purging methods. Purging is something we'll need to do anytime we change content on our website. For example, if we updated the title of a blog post, we'd need to purge the URL of the blog post to see the new title. And if we updated the logo shown on every page of our website, we'd need to purge the URL of the image.

Purging can be expensive, so it's best to purge only the content that has changed and nothing else. We'll discuss more targeted purging options later in this tutorial. For now, since we've implemented a big change that potentially impacts all of our assets, we'll use *purge all*. This method will purge all of our website's content from Fastly's cache nodes.

Let's purge all of our content now. Click the **Deliver** link at the top of the screen, and then the **Service summary** link. Then select **Purge all** from the **Purge** menu, as shown below.



Now that the previous uncompressed version of our website has been purged, we can test to see if our website is effectively using gzip compression. There are a number of free websites and tools that can check the status of gzip compression. The results are displayed below. Mission accomplished!



Enabling logging

Logging events and traffic is an important part of monitoring your website or application. One side effect of using a CDN is that you'll need to change how you log traffic. After you start using Fastly, virtually all of the traffic to your origin will originate from Fastly. To collect visitor logs going forward, you'll need to use Fastly's real-time log streaming feature to set up a log streaming endpoint.

Fastly's real-time log streaming feature provides integrations for a number of third-party services. Our logging documentation provides a list of supported services. You can connect your Fastly service to one or more logging endpoints.

Understanding the two types of logging

For the purposes of this tutorial, we'll set up two logging endpoints: Amazon S3 and Papertrail. These services each work a bit differently. For Amazon S3, Fastly batches the logs and writes them hourly to files in the Amazon S3 bucket. It's a cost-effective way to store log files that aren't needed for real-time analysis.

Papertrail is built specifically for streaming log files — when paired with Fastly's real-time log streaming feature, it's like tailing a file that updates in real time. You can see the logs scrolling across the screen and use search features to find specific log entries.

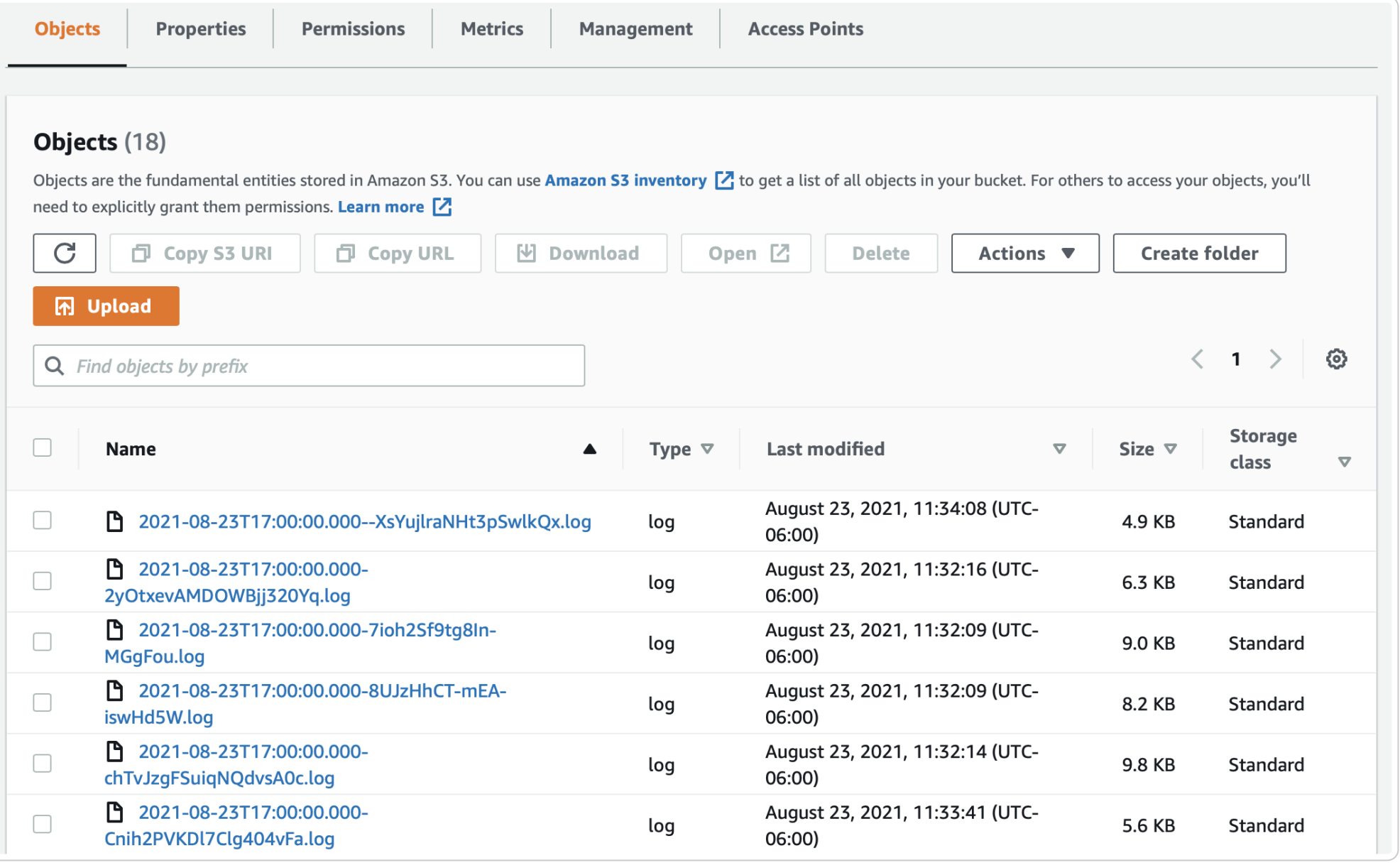
Setting up Amazon S3 log streaming and troubleshooting issues

To set up an Amazon S3 log streaming endpoint, we'll create a new S3 bucket for our logs and follow the instructions in the Fastly documentation for setting up an Amazon S3 endpoint. It's as easy as filling out a form and activating the new service configuration.

Fastly provides several tools that you can use to troubleshoot issues with logging endpoints. First, log in to the Fastly web interface. If Fastly tried and failed to send logs to a logging endpoint, an error message will appear in the web interface, as shown below.

In this case, we forgot to change the default value in the Domain field (under Advanced options). The exact hostname will vary depending on the AWS region your S3 bucket is located in. Since we're using the `us-east-2` region, we need to update the default hostname (`s3.amazonaws.com`) to `s3.us-east-2.amazonaws.com`. You can find the full list of AWS region codes on the AWS website.

Fastly writes the logs to our S3 bucket every hour, so it could take some time before the logs appear in the S3 bucket. Eventually, we'll see several new log files, as shown below. See our documentation on changing where log files are written to learn how to change the file names and control where the files are written.

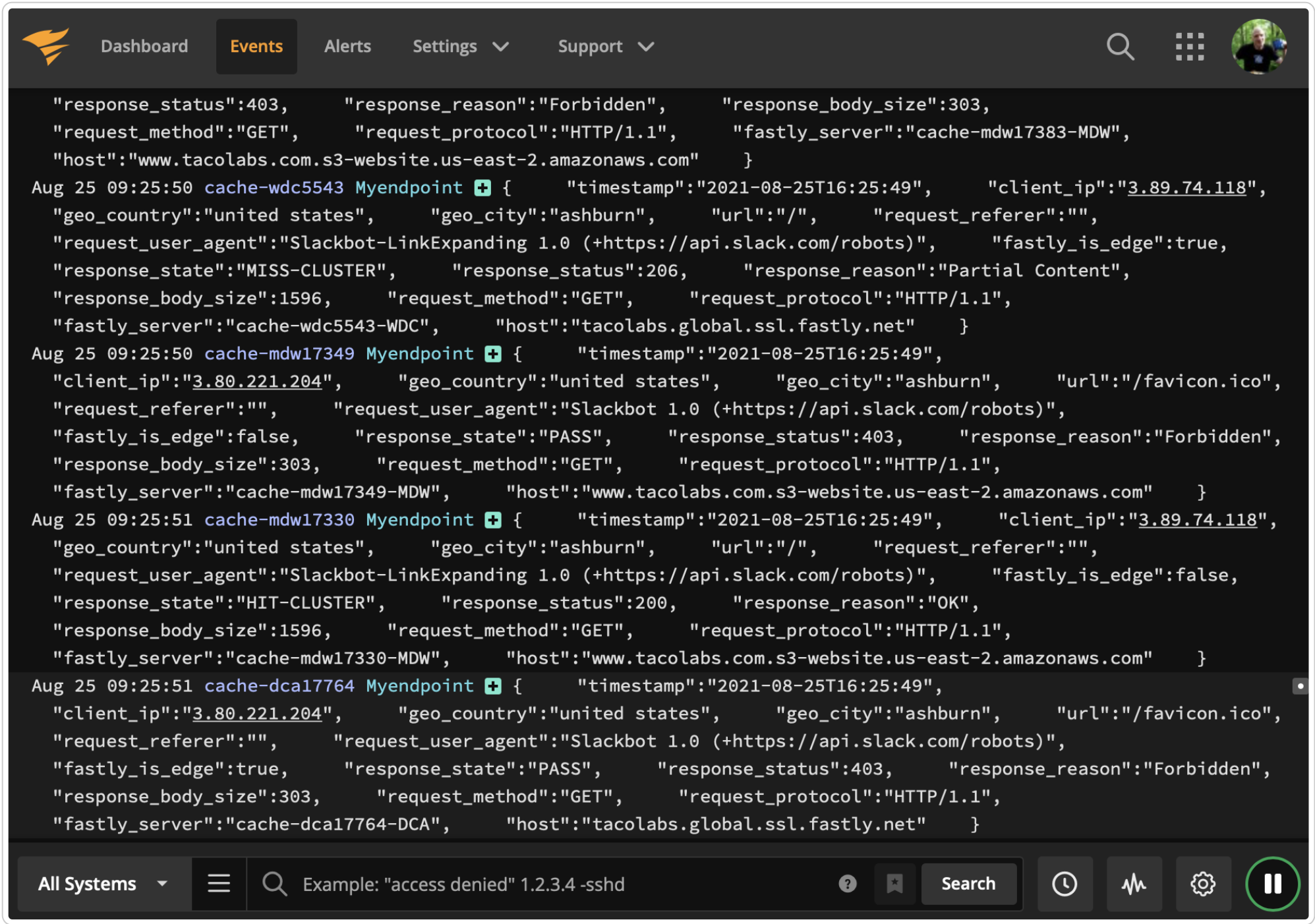


We can use the AWS web interface to open the files. An excerpt of one of the files is shown below. Our documentation on streaming logs provides more information about the log formats and variables.


```
{    "timestamp":"2021-08-23T16:32:15",    "client_ip":"73.127.172.124",    "geo_country":"united states",    "geo_city":"albuquerque",    "url":"/tacos",    "request_referer":"https://tacolabs.global.ssl.fastly.net/base-layers/",    "request_user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.107 Safari/537.36",    "fastly_is_edge":true,    "response_state":"HIT",    "response_status":302,    "response_reason":"Moved Temporarily",    "response_body_size":313,    "request_method":"GET",    "request_protocol":"HTTP/1.1",    "fastly_server":"cache-phx12422-PHX",    "host":"tacolabs.global.ssl.fastly.net"    }
{    "timestamp":"2021-08-23T16:32:15",    "client_ip":"73.127.172.124",    "geo_country":"united states",    "geo_city":"albuquerque",    "url":"/favicon.ico",    "request_referer":"https://tacolabs.global.ssl.fastly.net/tacos/",    "request_user_agent":"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.107 Safari/537.36",    "fastly_is_edge":true,    "response_state":"PASS",    "response_status":403,    "response_reason":"Forbidden",    "response_body_size":303,    "request_method":"GET",    "request_protocol":"HTTP/1.1",    "fastly_server":"cache-phx12422-PHX",    "host":"tacolabs.global.ssl.fastly.net"    }
```


Setting up Papertrail log streaming

Papertrail is a useful all-in-one service for people who are new to logging. Unlike S3, which holds your log files but doesn't do anything else with them, Papertrail is built exclusively for real-time logging and analysis. After you create a Papertrail account and set up a logging endpoint for Papertrail in the Fastly web interface, can you watch real-time log events scroll by in the Papertrail interface, as shown below.



Papertrail has features that make it easier to read logs. For example, you use Papertrail to group types of log entries, search through logs, and configure alerts for certain types of events.

You can also configure Papertrail to filter logs. By dropping the lines for successful requests for things like static assets (e.g., CSS files and images), you can reduce the chatter in the logs and focus on log lines for requests with 4xx and 5xx statuses. This makes it much easier to find issues.

---

4. Configuring caching

Last updated: 2022-02-03

https://docs.fastly.com/en/fundamentals/4-configuring-caching

This page is part of Fastly 101, a step-by-step tutorial that shows you how to use Fastly with an example website and domain name. It guides you through the steps of caching and delivering a static website using the Jekyll static site generator, Amazon AWS, and the Fastly CDN. For more information, see the introduction.

When it comes to caching your website or web application's content, Fastly provides a wide variety of options. Fastly caches most content by default, but most people will want or need to customize their caching settings. You have complete control over what content is cached, and for how long that content is cached.

There's no one-size-fits-all approach to caching — everyone's needs can be slightly different. In a real world scenario, your caching requirements will depend on a variety of factors. For example, if you used Django to build your web application, you might want to create different caching rules for static and dynamic content.

Since Taco Labs is a static website, our caching needs are relatively simple. Our recipe website primarily contains HTML and images. The goal is to cache everything for a long time. Then, when we make a change, we'll purge the impacted URL or just do a *purge all*.

> **TIP**
>
> It probably goes without saying that caching is an enormous topic that can't be comprehensively covered in this tutorial. To learn more about caching, see our caching documentation.

How caching and purging works with our workflow

Let's consider how and when we'll need to purge content from cache. There are three possible scenarios:

- Adding new content
- Revising content
- Deleting content

The revising and deleting content scenarios should be obvious. If we revise a single recipe, we can soft purge the URLs of the page and any associated images. Deleting content is a bit more complicated. Obviously, we'll need to purge the URL of the deleted page and any associated images. We'll also need to purge any of the index pages that link to the deleted page.

New content isn't cached, of course. But even adding new content isn't a straightforward matter. New recipes will appear on the category index pages, and since those pages are cached, they'll need to be purged when the new content is added.

To keep things simple, we'll update our GitHub Action in the next section to use the Fastly API to automatically purge all content after we build and deploy our website to Amazon S3. Purge all is not the most efficient way of purging content — you definitely wouldn't want to do this if you were, say, managing a content management system that receives hundreds of thousands of visitors a day. But for our static site, this setup reduces complexity and makes it easier for team members to handle deployments.

Setting caching headers

We can set HTTP headers that control caching for both Fastly's servers and end users' web browsers. There are several relevant headers available — for a full list, see our documentation on cache freshness. We'll use the `Surrogate-Control` and `Cache-Control` headers for Taco Labs:

```
Surrogate-Control: max-age=31557600
Cache-Control: no-store, max-age=0
```

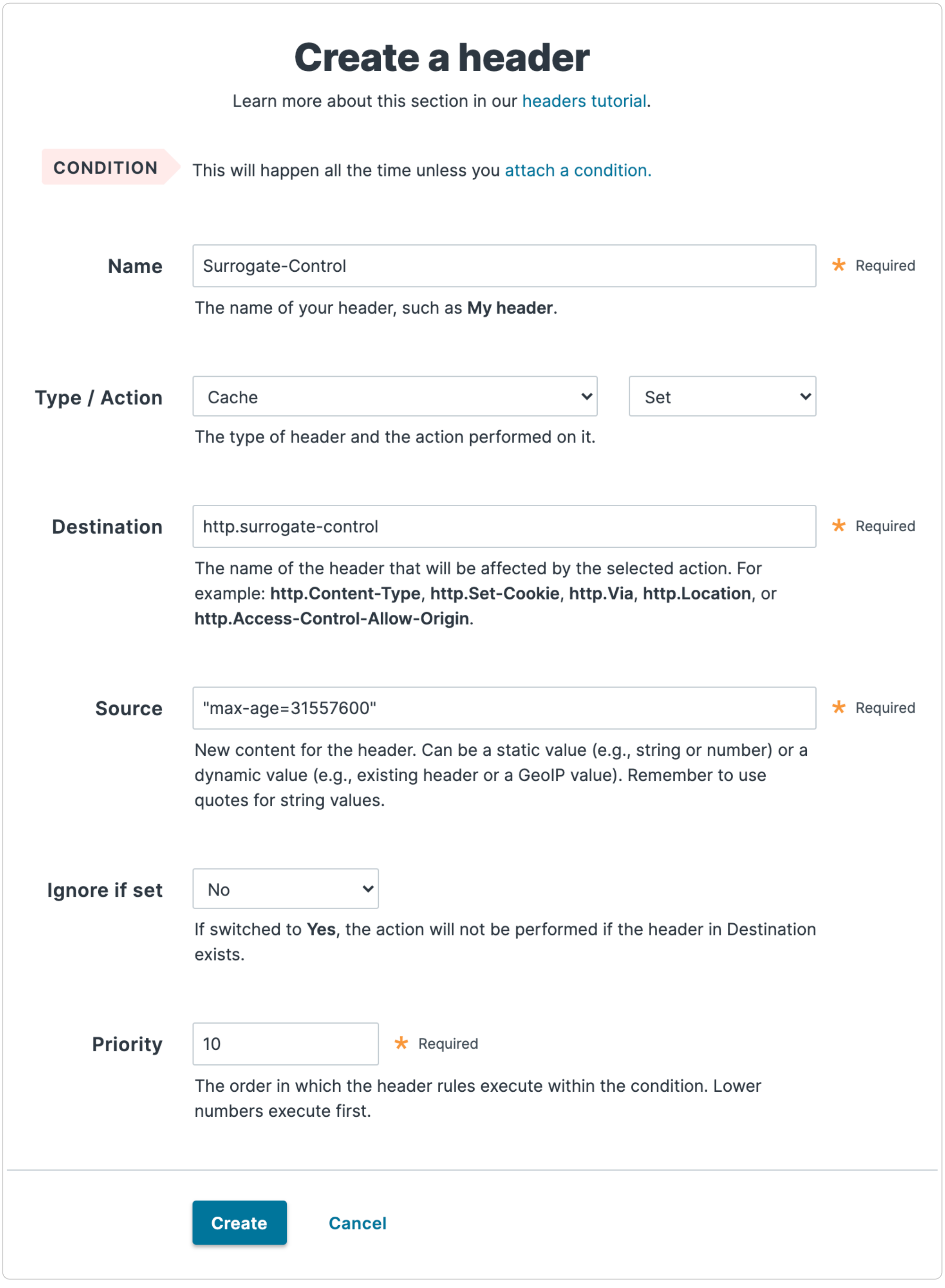
The `Surrogate-Control` header is proprietary to Fastly. Here, it's telling Fastly to cache objects for a maximum of 31557600 seconds (one year). We might go weeks or months without updating the existing content on Taco Labs — this header ensures that our content will stay in Fastly's cache for a long time.

On the other hand, we don't really care about caching in the end user's web browser. In fact, we'd prefer not to since Fastly can't invalidate an end user's web browser cache. Purge requests clear Fastly's cache, but they don't touch web browsers. That's where the `Cache-Control` header comes in. We can use it to tell the user's web browser to not cache the object.

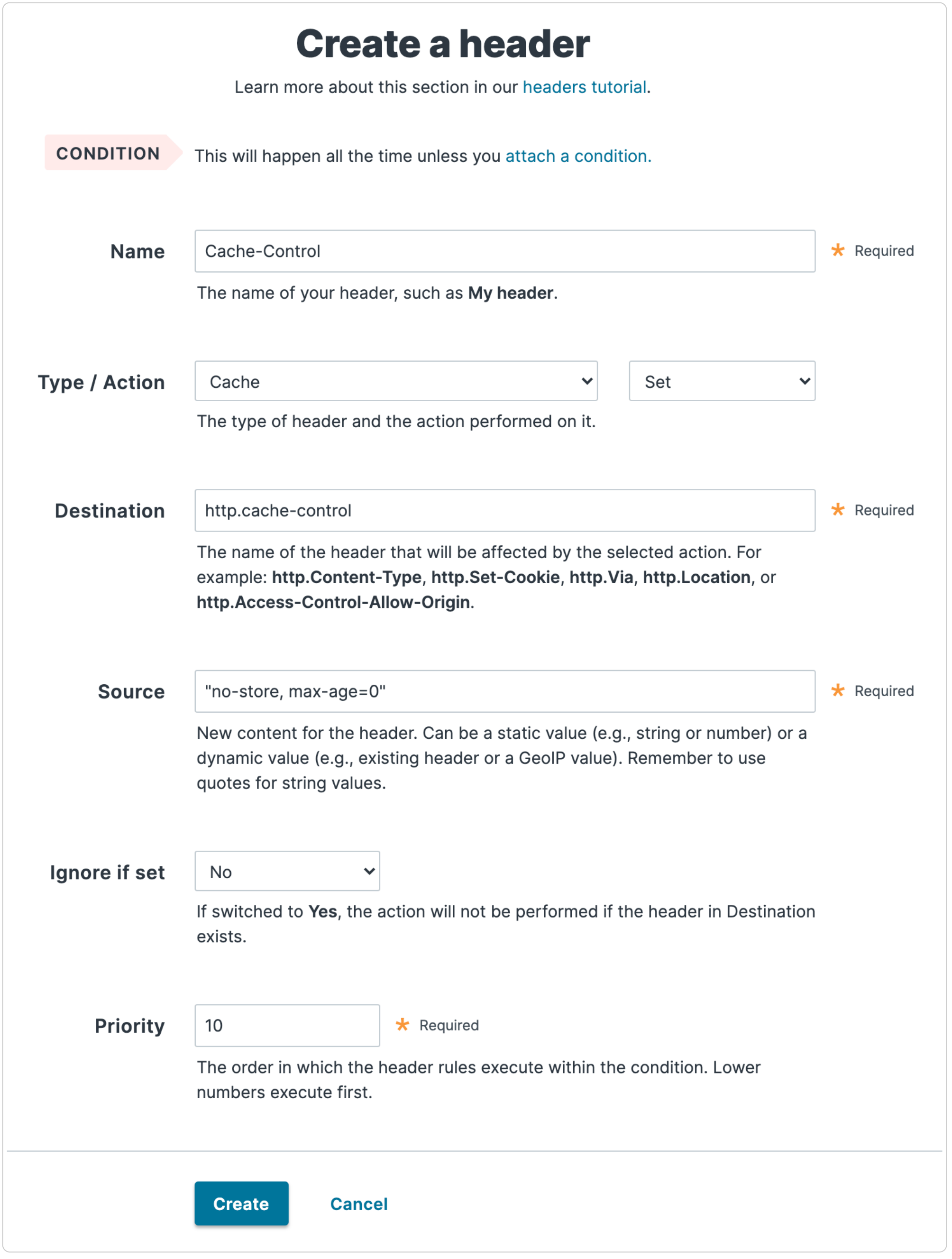
Acting together, these headers tell Fastly's servers to cache objects for a year and end users' web browsers to not cache objects at all.

> **NOTE**
>
> If we wanted to leverage web browser cache in this example, we could set the `Cache-Control` header to something like `max-age=86400`. That way, the end user's web browser would cache objects for up to a day.

We can set the headers in the Fastly web interface. Let's start with the `Surrogate-Control` header. Click the **Edit configuration** button to clone our service and create a new draft version. Then, on the Content page, click the **Create a header** button. We'll fill out the form fields as shown below.



Let's click the **Create** button to save the header. Next, we'll click the **Create a header** button to create the `Cache-Control` header. We'll fill out the form fields as shown below.



We'll click the **Create** button to save the header, and then click the **Activate** button to activate our service configuration.

After performing another purge all (follow the instructions from [the previous section]), our new headers should be live. We can check by using the following curl command:

```
$ curl -svo /dev/null -H "Fastly-Debug:1" https://tacolabs.global.ssl.fastly.net
```

The output should contain the following:

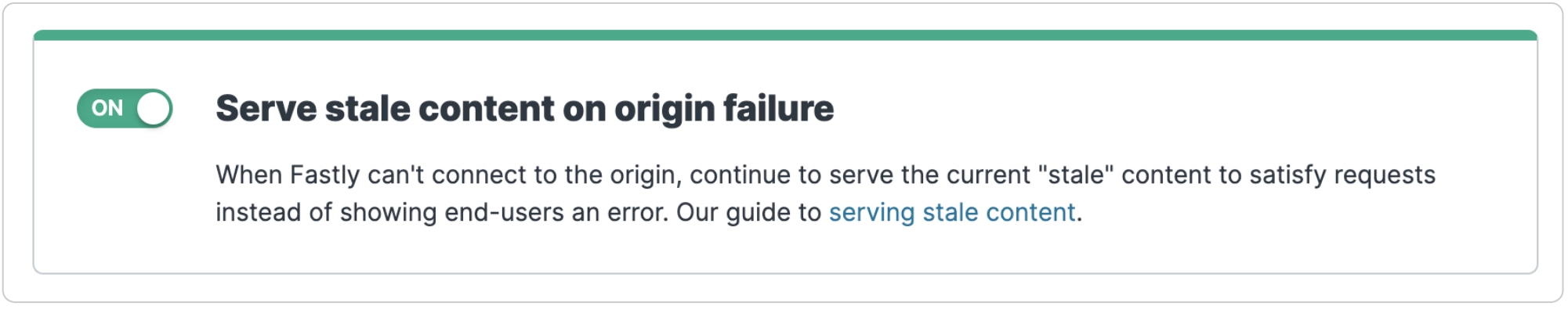
```
< HTTP/2 200
< x-amz-id-2: i0l9PThu7fvjt4FVKxtZYOVHEKAJIPh/rp9bjojQI+WXPFUKPW6WJZ7a4V0Pq3pyaR0VAvE/g5E=
< x-amz-request-id: 2XK0VGXYP9TWDW3J
< last-modified: Fri, 23 Jul 2021 21:54:38 GMT
< etag: "dcf9e4efa41b023a4280c8305070a1cf"
< content-type: text/html
< server: AmazonS3
< via: 1.1 varnish, 1.1 varnish
< cache-control: no-store, max-age=0
< accept-ranges: bytes
< date: Tue, 26 Oct 2021 21:21:11 GMT
< age: 85457
< x-served-by: cache-mdw17340-MDW, cache-phx12433-PHX
< x-cache: HIT, MISS
< x-cache-hits: 2, 0
< x-timer: S1635283271.482635,VS0,VE46
< vary: Accept-Encoding
< content-length: 4469
```

We can see our `Cache-Control` header there in the output. It's working! Note that we don't see the `Surrogate-Control` header in the output because Fastly removes that header before a response is sent to an end user.

Enabling serve stale

Fastly can serve stale content when there's a problem with our origin server or if it's taking a long time to fetch new content from our origin server. For example, in the unlikely event that Amazon S3 has a service disruption, Fastly can continue to serve cached content from Taco Labs. This can help mitigate origin server outages — the only catch is that we have to enable it *before* our origin server goes down.

Let's click the **Edit configuration** button to clone our service and create a new draft version. Then, on the Settings page, click the **On** switch next to **Serve stale content on origin failure**, as shown below.



Now we can click the **Activate** button to activate our service configuration. We're all set! Now if Amazon S3 ever goes offline, Fastly will continue serving Taco Labs to visitors.

If you're interested in customizing the settings for this feature, be sure to read through the serve stale content documentation. There's a lot more you can do.

Preventing a page from caching

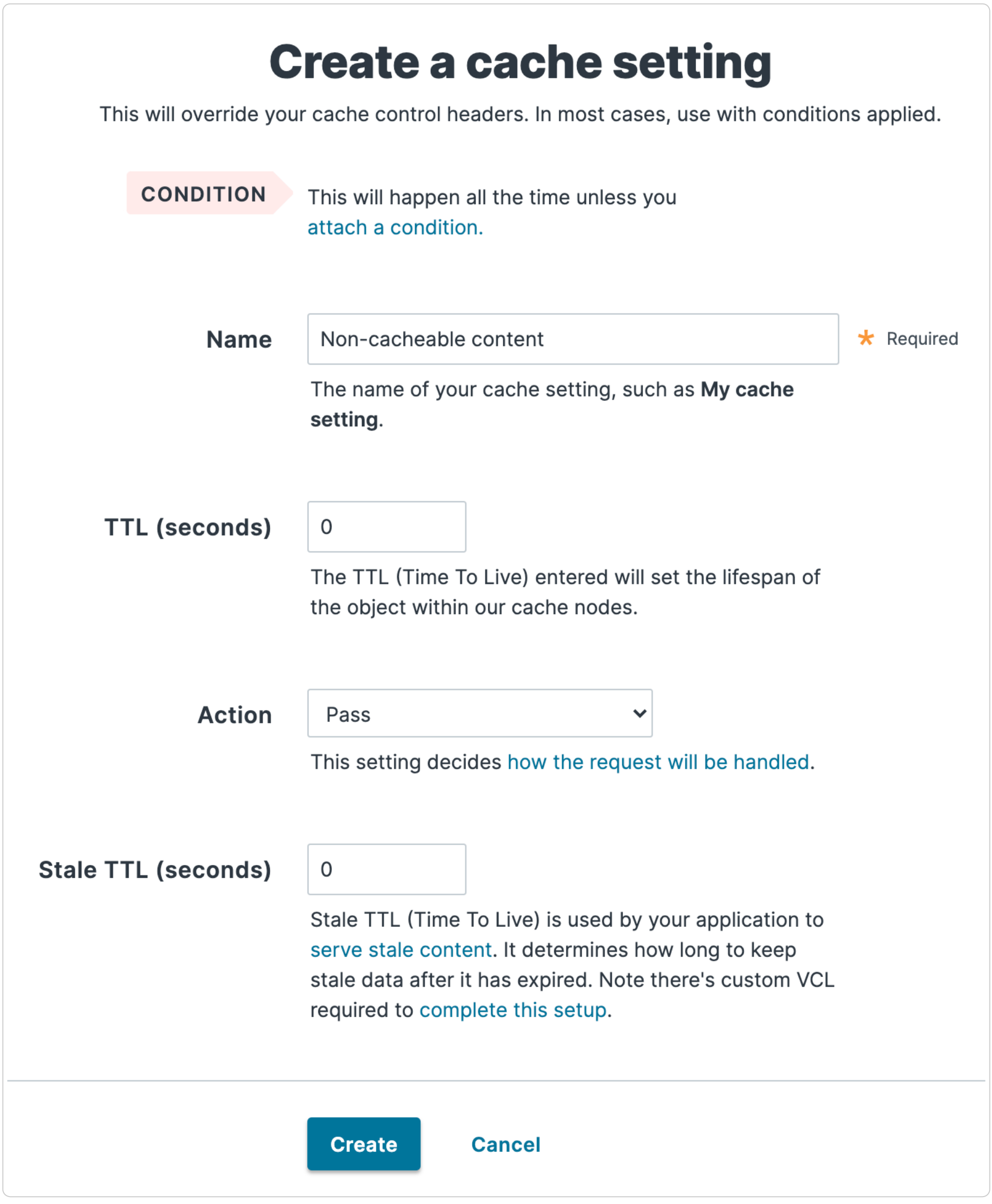
By default, Fastly will cache all of the objects on the Taco Labs website, but we can create a *condition* to prevent certain objects from being cached. There are situations where this could be useful. For example, maybe the images in your content management system are constantly being updated and you'd rather not cache them. Or maybe you want to add real-time interactive elements to a single page. Whatever the case, it's nice knowing that you can update your service configuration to prevent things from being cached.

> **TIP**
>
> Conditions logically control how requests are processed. They're really powerful! We encourage you to read more about conditions.

In this particular example, we'll prevent the caching of one of our recipe index pages (`http://www.tacolabs.com/tacos/`). Let's pretend that we're interested in adding some real-time information to this page in the form of two widgets: One that shows how many times our taco recipes have been viewed, and another that shows how many tacos have been eaten at a certain restaurant in Albuquerque. By conditionally preventing the page from being cached, we can ensure that visitors will always see the most up-to-date information.

We can set this up in the web interface. First, we'll create a new cache setting that tells Fastly which page not to cache. Then, we'll create a new condition that tells Fastly when to use the cache setting.

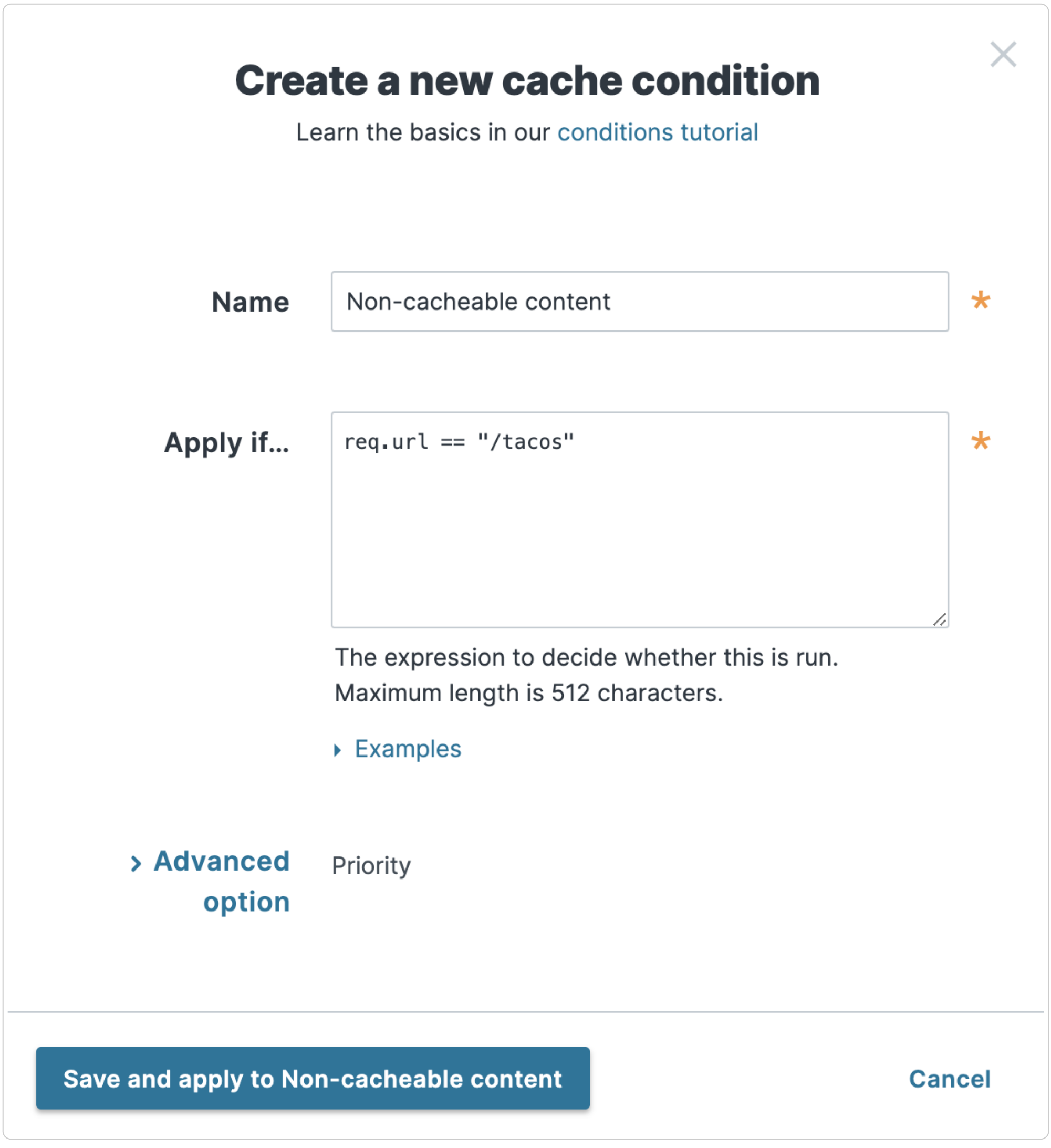
Let's click the **Edit configuration** button to clone our service and create a new draft version. Then, on the Settings page, click the **Create cache setting** button. Fill out the fields as shown below. Make sure that you set **Action** to **Pass**.



Click the **Create** button to create the cache setting, and then click the **Attach a condition** link. Fill out the fields as shown below. The condition works by looking at a VCL variable called `req.url`. If the variable is set to the path of the tacos recipe index page, the condition will apply the cache setting and *pass* the request without caching it.

TIP

You're about to use your first VCL variable — a big milestone. But what is VCL, and what are variables? Let's back up! The Fastly CDN is based on the open source Varnish software. Everything a Fastly service does is powered by Fastly's modified version of the Varnish Configuration Language (VCL). Believe it or not, throughout this tutorial, we've been using the Fastly web interface to modify the VCL code for our service. The web interface hides the complexity, but behind the scenes, it's been taking our settings and using them to generate custom VCL code that is used every time we activate our service configuration. That VCL code tells Fastly when and how to cache objects on our website. We're at the point in this tutorial where we need to start writing bits of custom VCL code to implement advanced logic in our service configuration. You can learn more about VCL in our documentation and try it out using Fastly Fiddle.



Click the **Save and apply to Non-cacheable content** button. Then we'll click the **Activate** button to activate our service configuration. After performing another purge all (follow the instructions from the previous section), our new cache setting and condition should be live. We can check by using the following curl command:

```
$ curl -svo /dev/null -H "Fastly-Debug:1" https://tacolabs.global.ssl.fastly.net/tacos/
```

The output should look like this:

```
< HTTP/2 200
< x-amz-id-2: lSWEBpWjnJqWvUpaCYRGHcJcnau/qMIlnu1ranRqgRgGB+O6Tj+EPURc6mx6Uxf7/n1YFt7bNJc=
< x-amz-request-id: F46TZ9RP94DM9SFW
< last-modified: Fri, 23 Jul 2021 21:54:38 GMT
< etag: "9f61cab2ab8c1347259ac814ff3068c9"
< content-type: text/html
< server: AmazonS3
< accept-ranges: bytes
< via: 1.1 varnish, 1.1 varnish
< cache-control: no-store, max-age=0
< date: Fri, 29 Oct 2021 20:54:33 GMT
< x-served-by: cache-mdw17349-MDW, cache-bur17526-BUR
< x-cache: MISS, MISS
< x-cache-hits: 0, 0
< x-timer: S1635540873.002019,VS0,VE154
< vary: Accept-Encoding
< strict-transport-security: max-age=300
< content-length: 2807
```

It looks about the same as the previous output, right? But watch the `x-cache` and `x-cache-hits` headers — those don't change on repeated executions of the curl command. If the page was being cached, we'd see a HIT in the `x-cache` header.

---

5. Go live

Last updated: 2022-02-03

https://docs.fastly.com/en/fundamentals/5-go-live

This page is part of Fastly 101, a step-by-step tutorial that shows you how to use Fastly with an example website and domain name. It guides you through the steps of caching and delivering a static website using the Jekyll static site generator, Amazon AWS, and the Fastly CDN. For more information, see the introduction.

We've done a lot of work to prepare our Fastly service for the Taco Labs website! Now it's time to put the finishing touches on things so we can go live and start using Fastly for our production website.

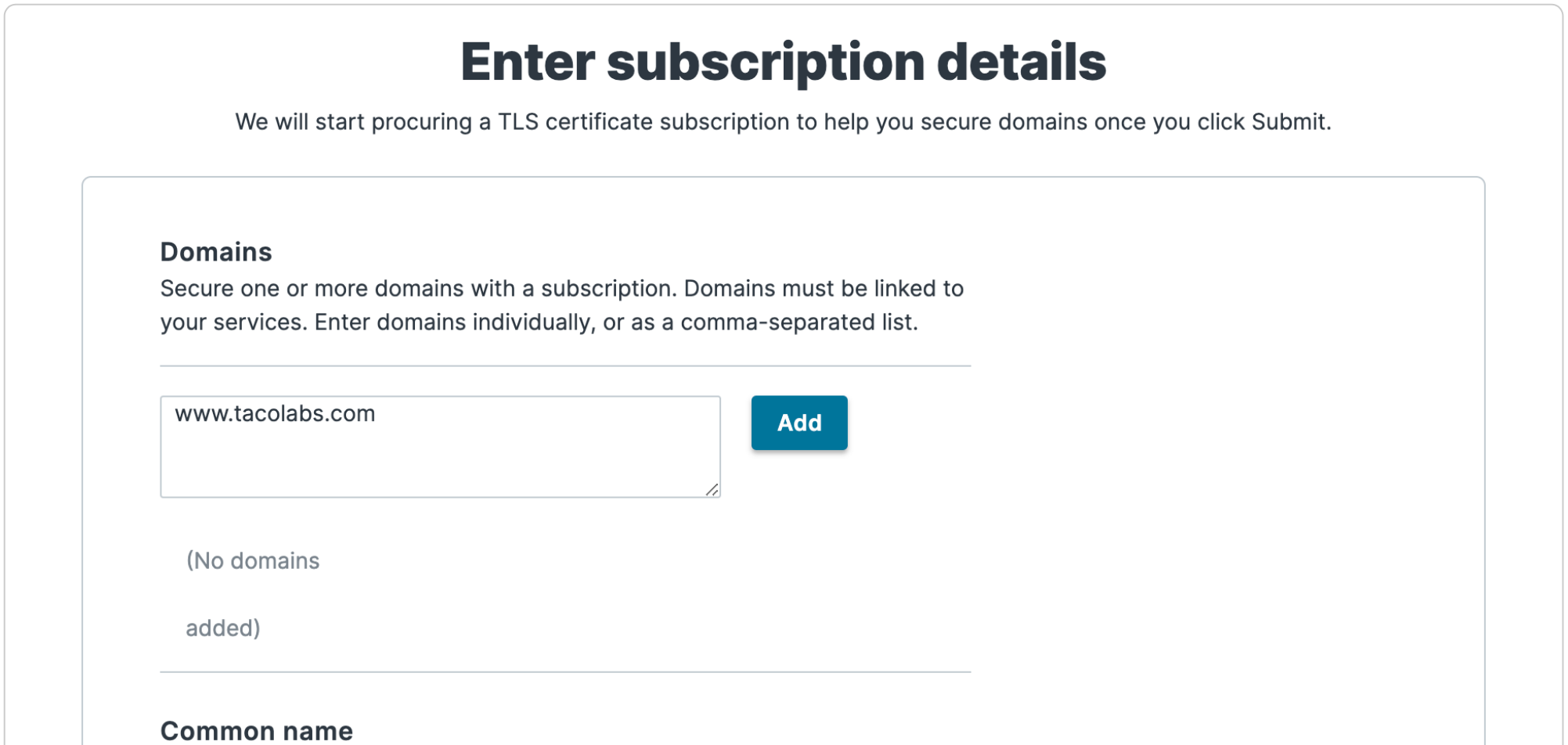
Enabling Fastly TLS

It probably goes without saying that encryption is a hard requirement for virtually all websites and web applications these days. Taco Labs is a static website that lacks many interactive features common to web applications, such as user authentication. We still want to use TLS though, because search engines and web browsers will penalize us if we don't!

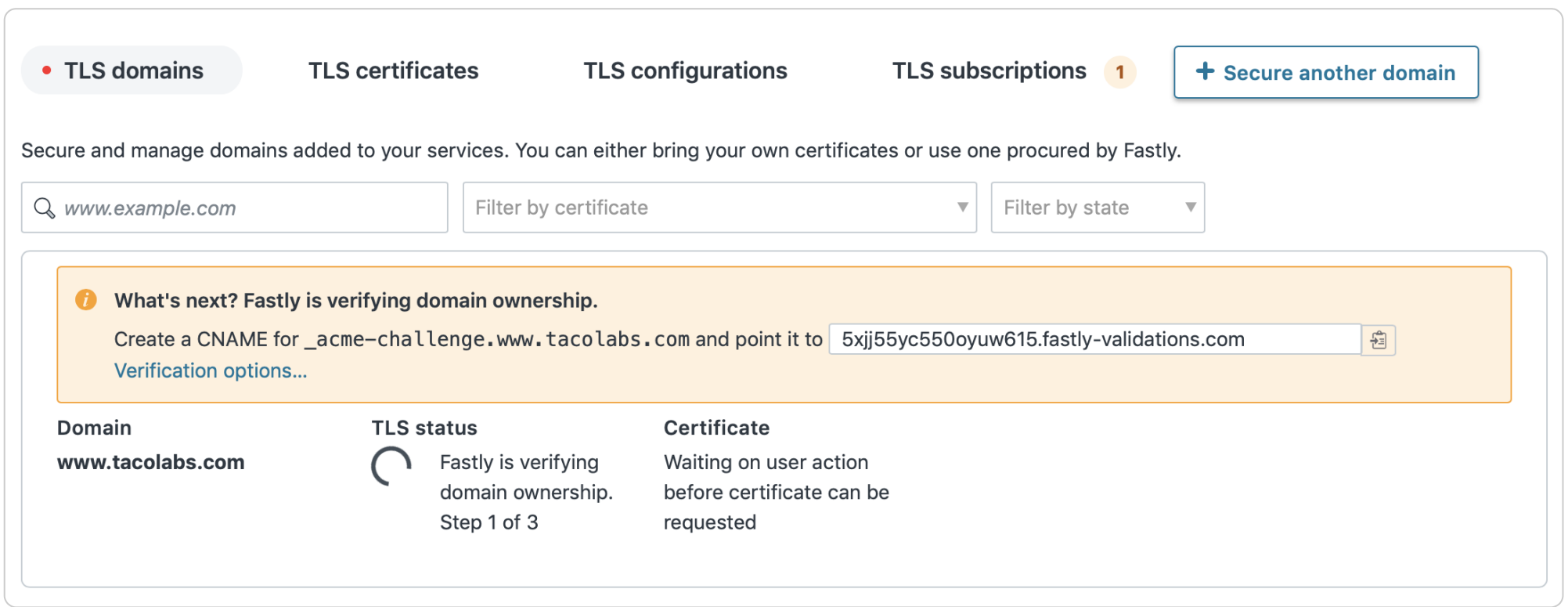
Up to this point, we've been using Fastly's Free TLS option with a shared certificate to secure traffic between Fastly and client web browsers. We could continue to use that URL (`https://tacolabs.global.ssl.fastly.net`) if it wasn't exposed to customers. For example, if we were using Fastly to serve assets from within an iOS application, we could probably continue using Free TLS. But we can't use Free TLS for custom domains, like `tacolabs.com`, because visitors will receive a certificate error.

Since we need TLS for a custom domain, we can use Fastly TLS to generate a certificate for our domain. After we add a credit card number to upgrade to a paid account, we can use Fastly TLS to generate and manage TLS certificates for up to five domains at no additional cost. We'll use the free Let's Encrypt option, which is automatically managed and renewed by Fastly. Alternatively, we could upload and manage our own certificate.

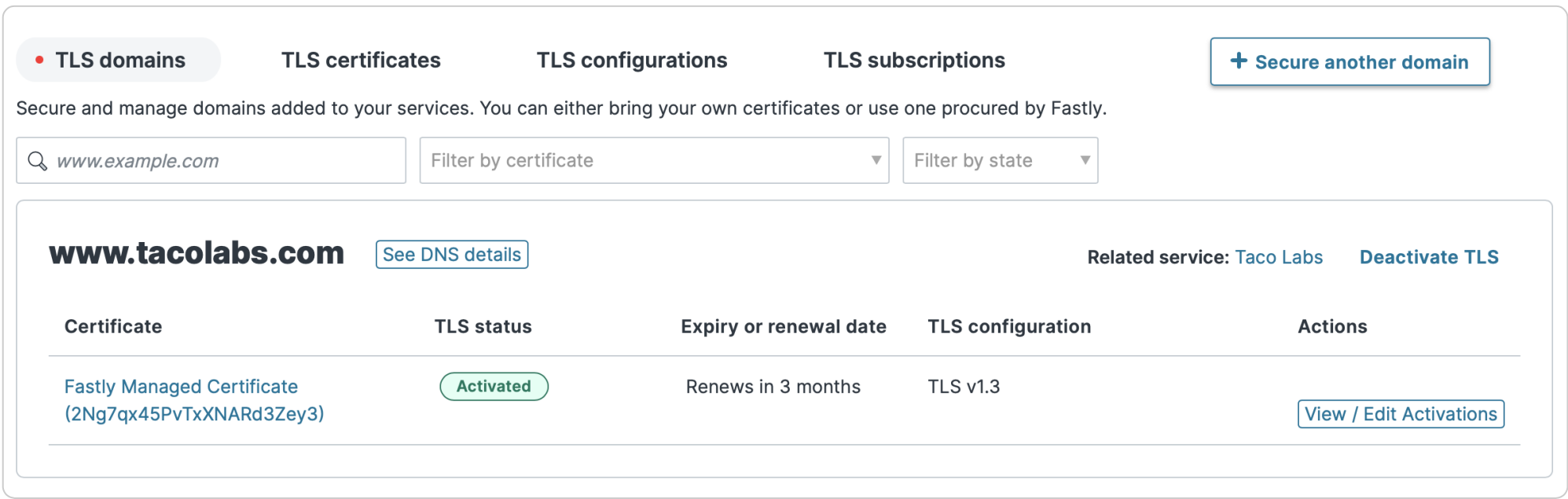
Click the **Secure** link at the top of the screen, and then click the **TLS management** link. Then click the **Get started** button. The window shown below appears.



After we fill out the form, add our domain, and click the **Submit** button, we'll see instructions directing us to add a CNAME record, as shown below. Fastly uses this information to verify that we own the `tacolabs.com` domain name.



To add the CNAME record, we'll head back over to AWS Route 53, which is handling our DNS service. It can take several minutes for the DNS record to propagate. When Fastly discovers the DNS record and verifies it, TLS will be enabled for `www.tacolabs.com`, as shown below.

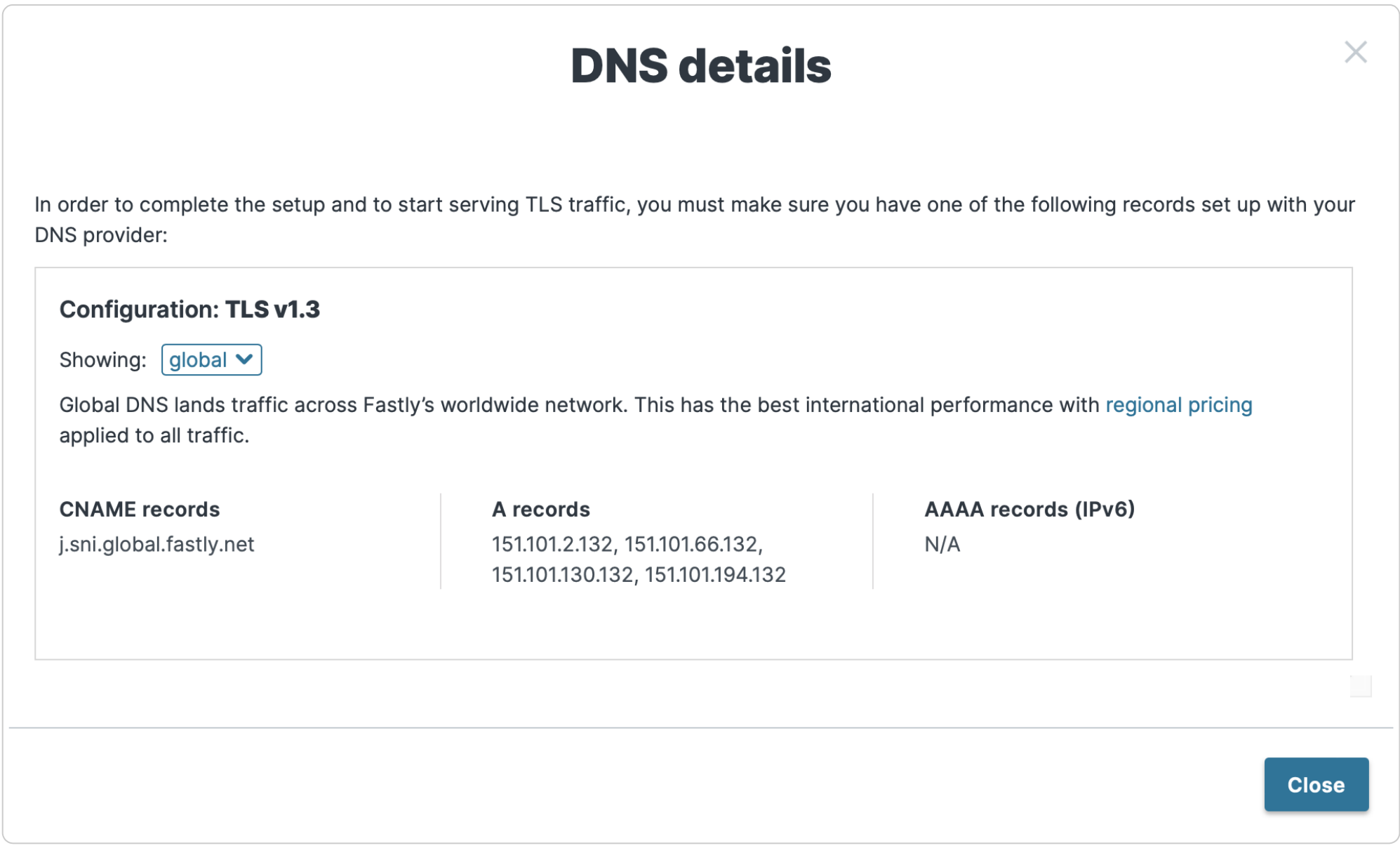


Updating DNS records

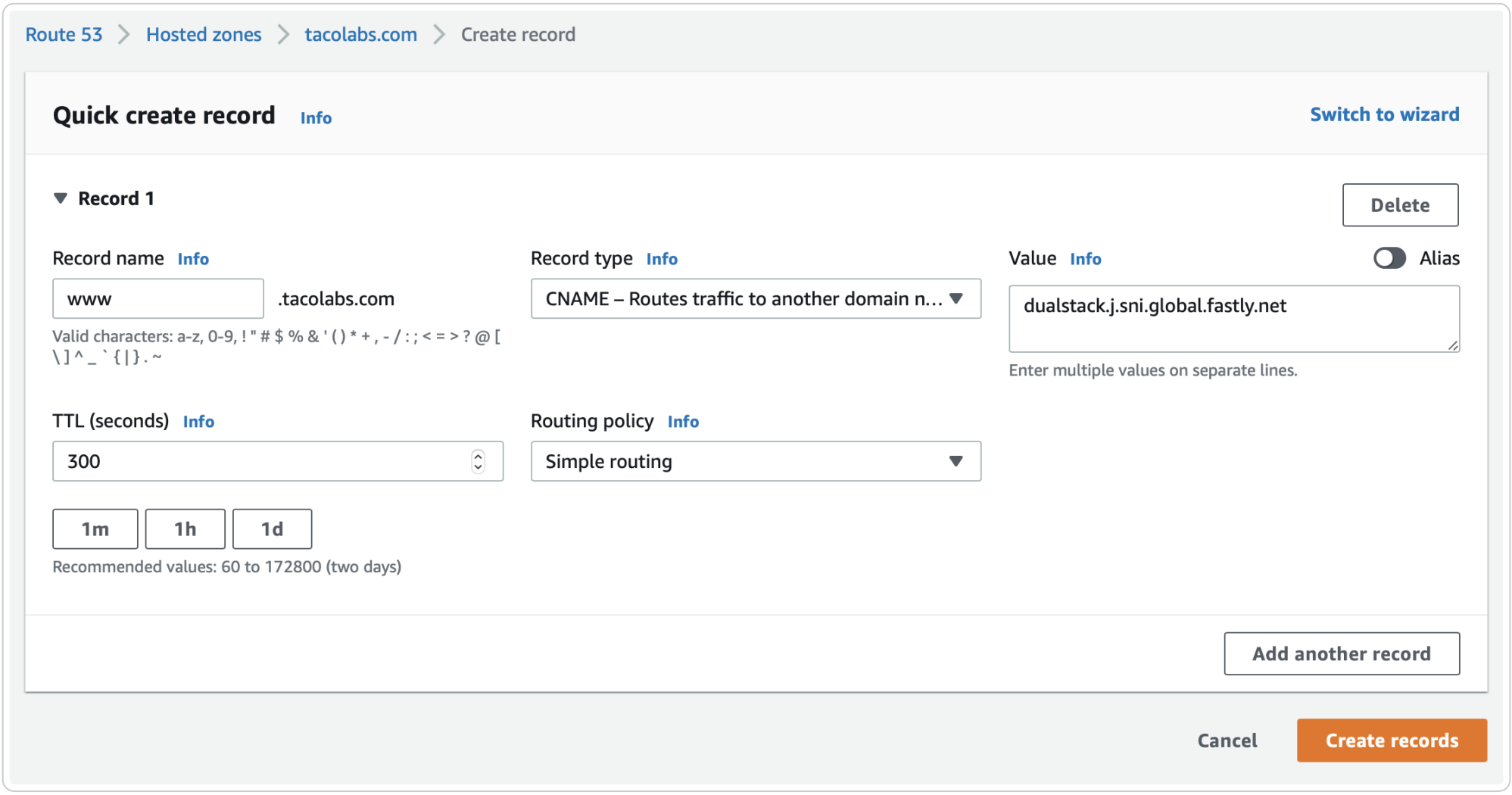
We've enabled TLS for our domain and now we're ready for the final step: Updating our domain's DNS records to point to Fastly. After we do this, all production traffic will start flowing through Fastly. It's a good idea to review the settings one more time to make sure everything's in order.

The first order of business is taking stock of any existing DNS records for your website or web application. Taco Labs is a new website — we didn't have any DNS records to start with. But in a real-world scenario, you'll probably be working with an existing website that already has DNS records. Before you add the new CNAME record for Fastly, you may need to remove existing DNS records for your domain.

You can find the value for the CNAME record for your domain by clicking the **See DNS details** button on the TLS domains page, as shown below. This value doesn't support IPv6 addresses, but you can preface `dualstack` to enable IPv6 support.



In this case, since the Fastly web interface says that the value for our CNAME record is `j.sni.global.fastly.net`, we're going to use `dualstack.j.sni.global.fastly.net` as the value to enable IPv6 support. Let's add the CNAME record to AWS Route 53. The entry is shown below.



DNS records can take some time to propagate, but we can use the following command to check the status:

```
$ dig www.tacolabs.com +short
dualstack.j.sni.global.fastly.net.
151.101.198.132
```

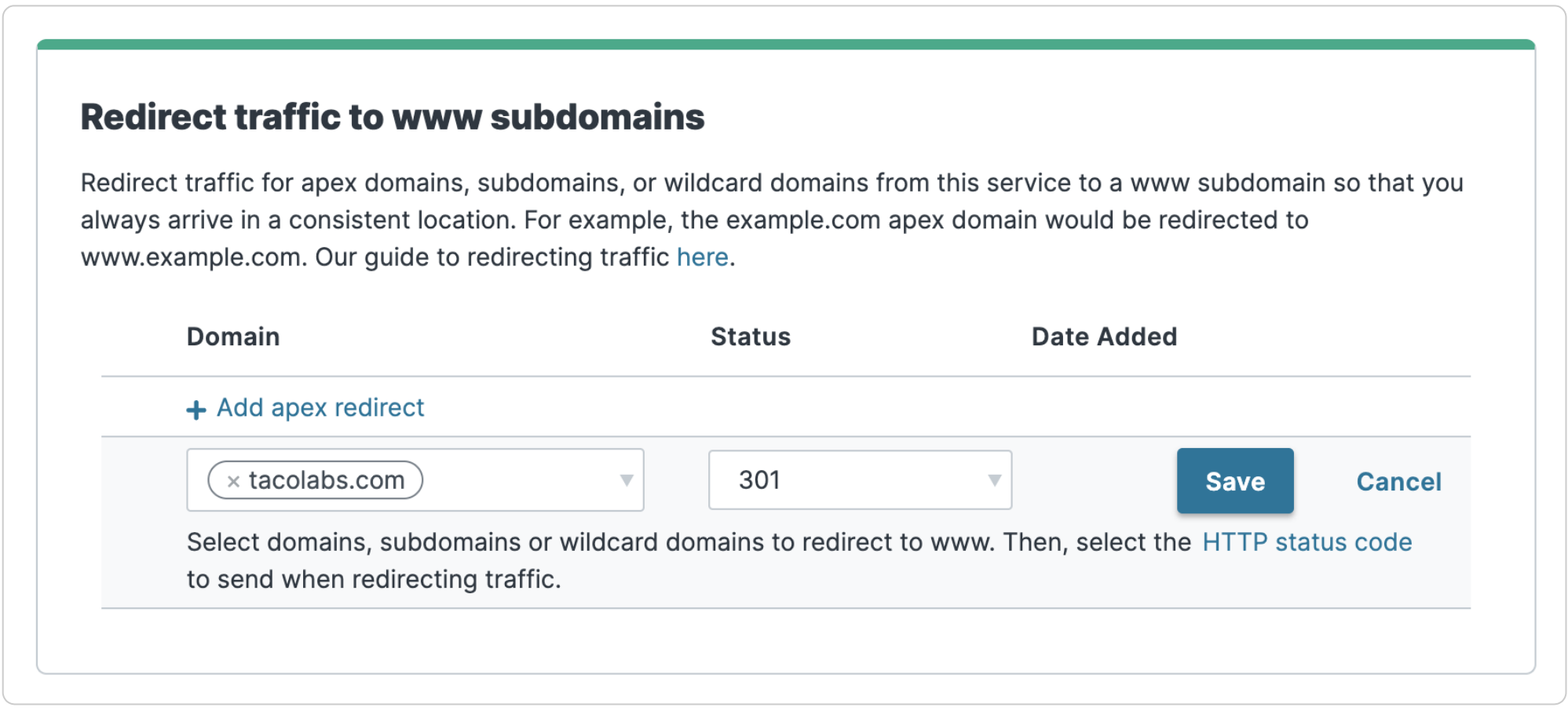
Looks like we're all set!

Redirecting the apex domain to www

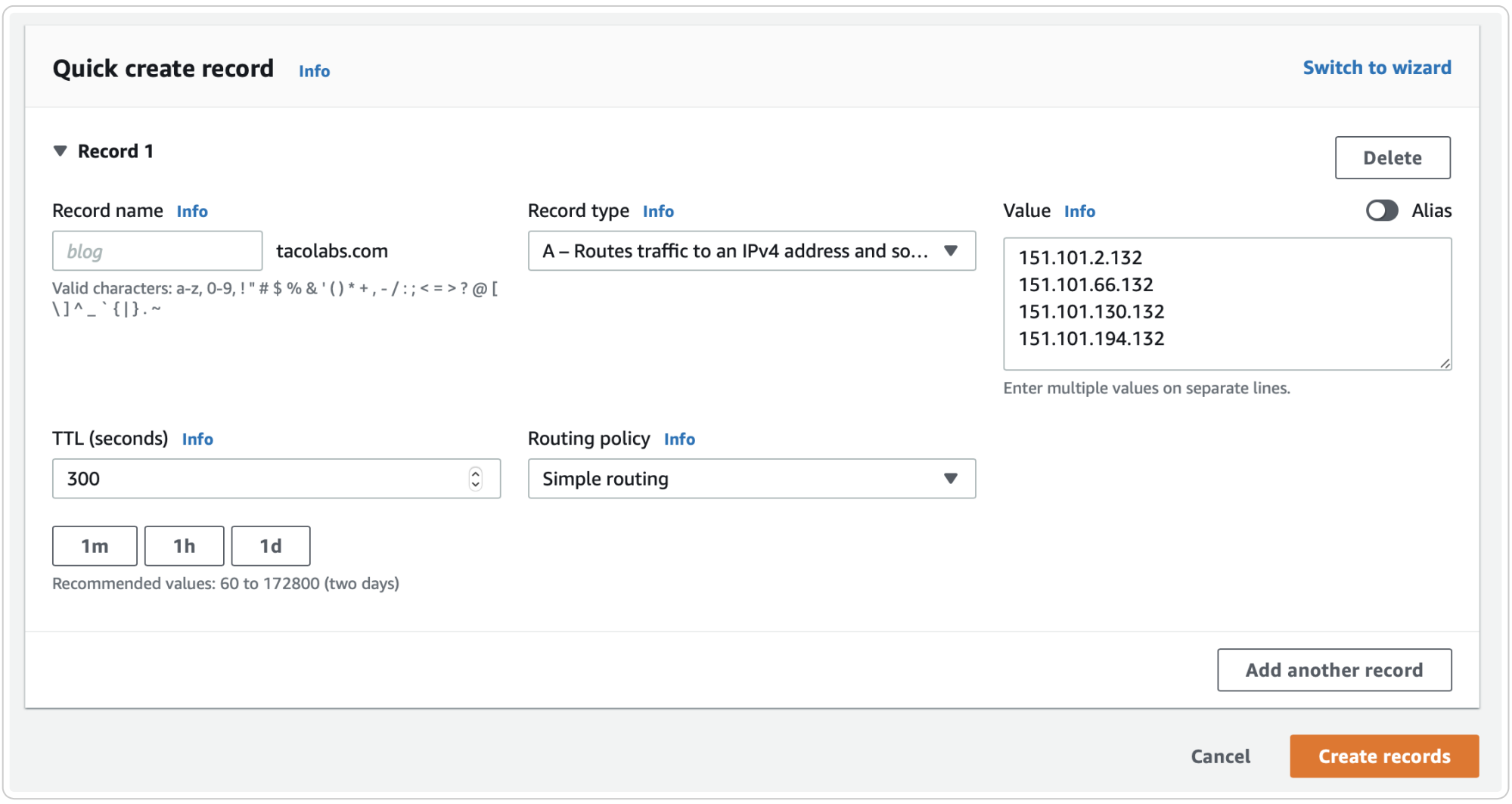
One thing we need to address is our apex domain: `http://tacolabs.com`. We're using Fastly for `www.tacolabs.com`, but what if people type in our domain without the `www`? We need to add a redirect so that people who type `http://tacolabs.com` will be automatically redirected to `http://www.tacolabs.com`.

First, we need to add `tacolabs.com` as a domain in our Fastly service. Let's click the **Edit configuration** button to clone our service and create a new draft version. Then, on the Domains page, we'll click the **Create domain** button. In the **Domain** field, enter `tacolabs.com`, and then click the **Add** button.

Now we can add the redirect for the apex domain. On the Settings page, in the Redirect traffic to www subdomains section, we'll click the **Add apex redirect** link. We'll select **tacolabs.com** from the Domain menu, and then select **301** from the Status menu to use a 301 redirect, as shown below. Finally, we'll click the **Save** button, and then click the **Activate** button to activate the new version of the service configuration.



Our Fastly service is set up to handle the redirect. Now we need to add a DNS record for `tacolabs.com`. Recall that the TLS page had a details section that also displayed anycast IP addresses for DNS A records. We'll use those anycast IP addresses to create a new DNS A record in Route 53 for `tacolabs.com`, as shown below.



After the DNS record has propagated, we can use curl to test that our redirect is working. We should see that `tacolabs.com` is redirected to `www.tacolabs.com`, as shown below.

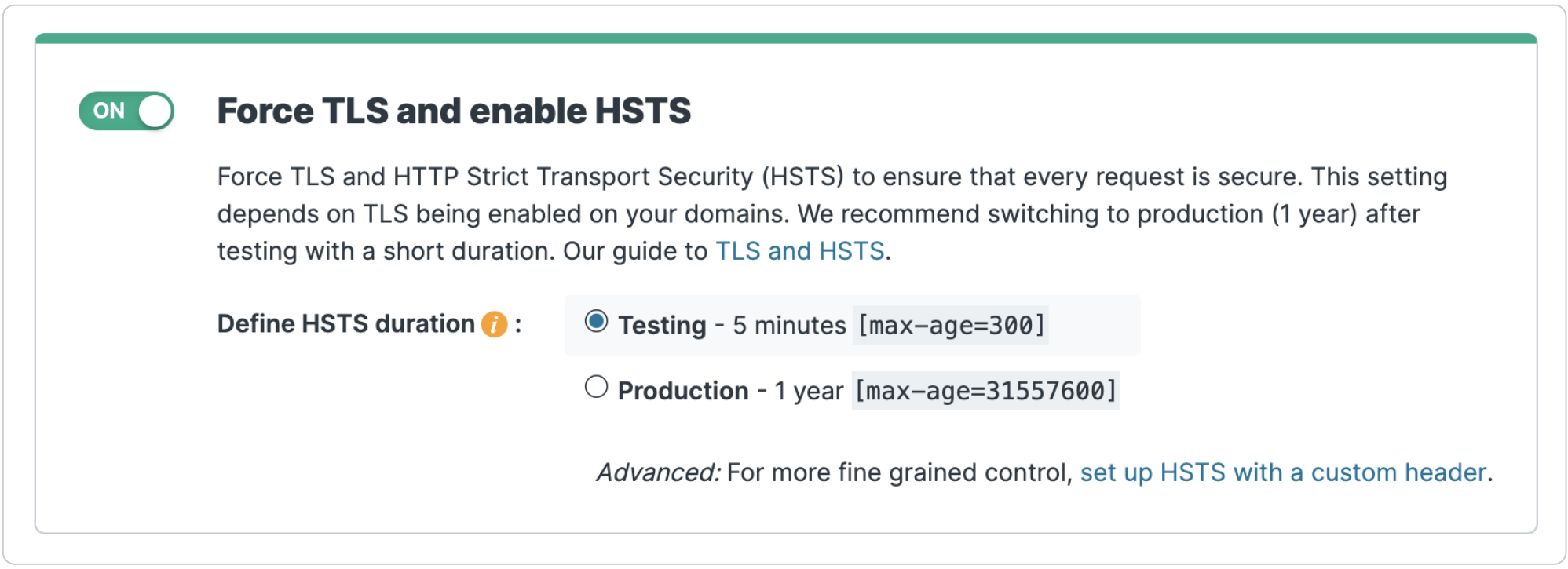
```
$ curl -sLD - http://tacolabs.com -o /dev/null -w '%{url_effective}'
HTTP/1.1 301 Moved Permanently
Server: Varnish
Retry-After: 0
cache-control: max-age=86400
Location: http://www.tacolabs.com/
Content-Length: 0
Accept-Ranges: bytes
Date: Mon, 18 Oct 2021 19:25:46 GMT
Via: 1.1 varnish
Connection: close
X-Served-By: cache-den8264-DEN
X-Cache: HIT
X-Cache-Hits: 0
X-Timer: S1634585146.929981,VS0,VE79

HTTP/1.1 200 OK
x-amz-id-2: NXLv0EZjIymQiCtAvpbeUyXlfiKjvMBbpSOqz84U5JNOfm5ysn1c43hvahA6oKH1/t+uiJS2Xrw=
x-amz-request-id: T6JEN0EXBQFXVRCF
cache-control: no-store, max-age=0
Last-Modified: Fri, 23 Jul 2021 21:54:38 GMT
ETag: "dcf9e4efa41b023a4280c8305070a1cf"
Content-Type: text/html
Server: AmazonS3
Via: 1.1 varnish, 1.1 varnish
Content-Length: 4469
Accept-Ranges: bytes
Date: Mon, 18 Oct 2021 19:25:46 GMT
Age: 0
Connection: keep-alive
X-Served-By: cache-mdw17341-MDW, cache-phx12427-PHX
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
X-Timer: S1634585146.116375,VS0,VE94
Vary: Accept-Encoding
```

Enabling HSTS and forcing TLS

We're almost finished, but as it currently stands, things still aren't quite working right. Visitors can still get to the unencrypted version of `www.tacolabs.com`. We need to force visitors to the encrypted version of the site at `https://www.tacolabs.com`.

We can do that by forcing TLS and enabling HTTP Strict Transport Security (HSTS). Let's click the **Edit configuration** button to clone our service and create a new draft version. On the Settings page, click the **Force TLS and enable HSTS** switch to the on position, as shown below. It's a good idea to leave the HSTS duration set to the testing interval, at least to start.



Let's click the **Activate** button to activate the new version of the service configuration. Now we can check our redirect again. This time, the output from the curl command shown below shows that `http://tacolabs.com` redirects to `https://www.tacolabs.com`.

```
$ curl -sLD - http://tacolabs.com -o /dev/null -w '%{url_effective}'
HTTP/1.1 301 Moved Permanently
Server: Varnish
Retry-After: 0
cache-control: no-store, max-age=0
Location: http://www.tacolabs.com/
Content-Length: 0
Accept-Ranges: bytes
Date: Mon, 18 Oct 2021 20:38:35 GMT
Via: 1.1 varnish
Connection: close
X-Served-By: cache-bur17572-BUR
X-Cache: HIT
X-Cache-Hits: 0
X-Timer: S1634589516.510082,VS0,VE58
Strict-Transport-Security: max-age=300

HTTP/1.1 301 Moved Permanently
Server: Varnish
Retry-After: 0
Location: https://www.tacolabs.com/
Content-Length: 0
Accept-Ranges: bytes
Date: Mon, 18 Oct 2021 20:38:35 GMT
Via: 1.1 varnish
Connection: close
X-Served-By: cache-phx12424-PHX
X-Cache: HIT
X-Cache-Hits: 0
X-Timer: S1634589516.811935,VS0,VE1
Strict-Transport-Security: max-age=300

https://www.tacolabs.com/
```

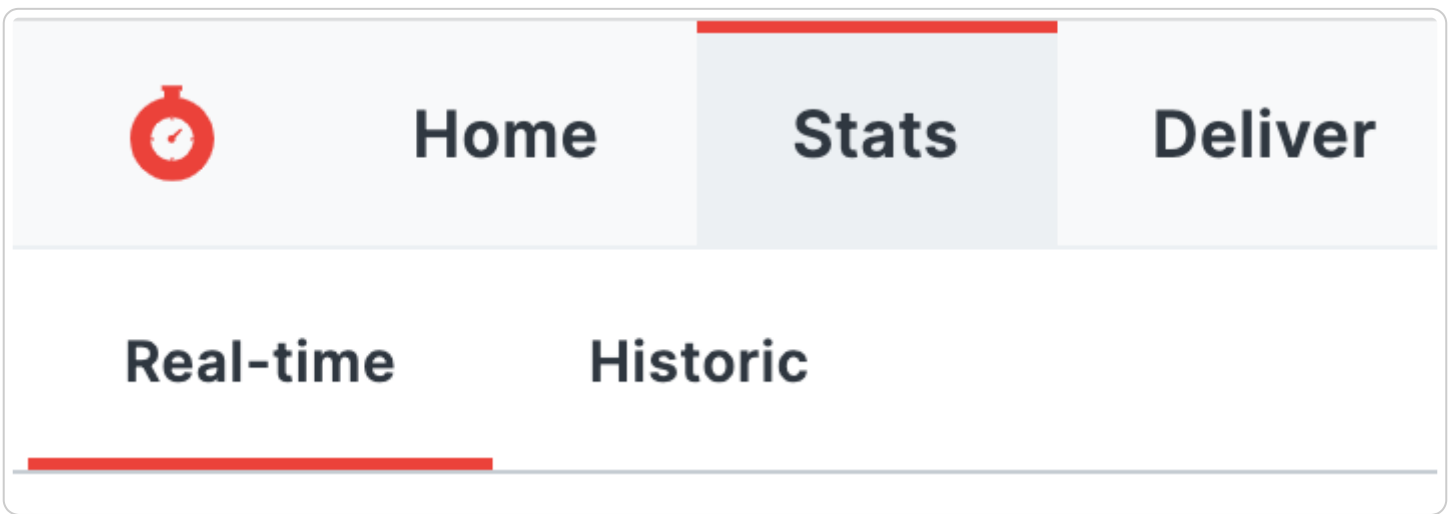
There's one other problem we need to address: The HTTPS redirect (`https://tacolabs.com` to `https://www.tacolabs.com`) returns a certificate mismatch error. To resolve that, we'll need to follow the instructions in the previous section and add another TLS certificate for the apex domain (`tacolabs.com`). After we've done that, all of the redirects will be working.

Watching stats

Everything is set up and working now! Go ahead and give yourself a pat on the back — you've successfully set up Fastly to pull your website or application from your origin server and deliver it to your customers in the most efficient way possible.

Now comes the fun part: Watching traffic in real time. Taco Labs is a fake website that hasn't hit the big time (yet), but we can still simulate traffic with a terminal command. That will allow us to get a feel for Fastly's real-time and historic stats features.

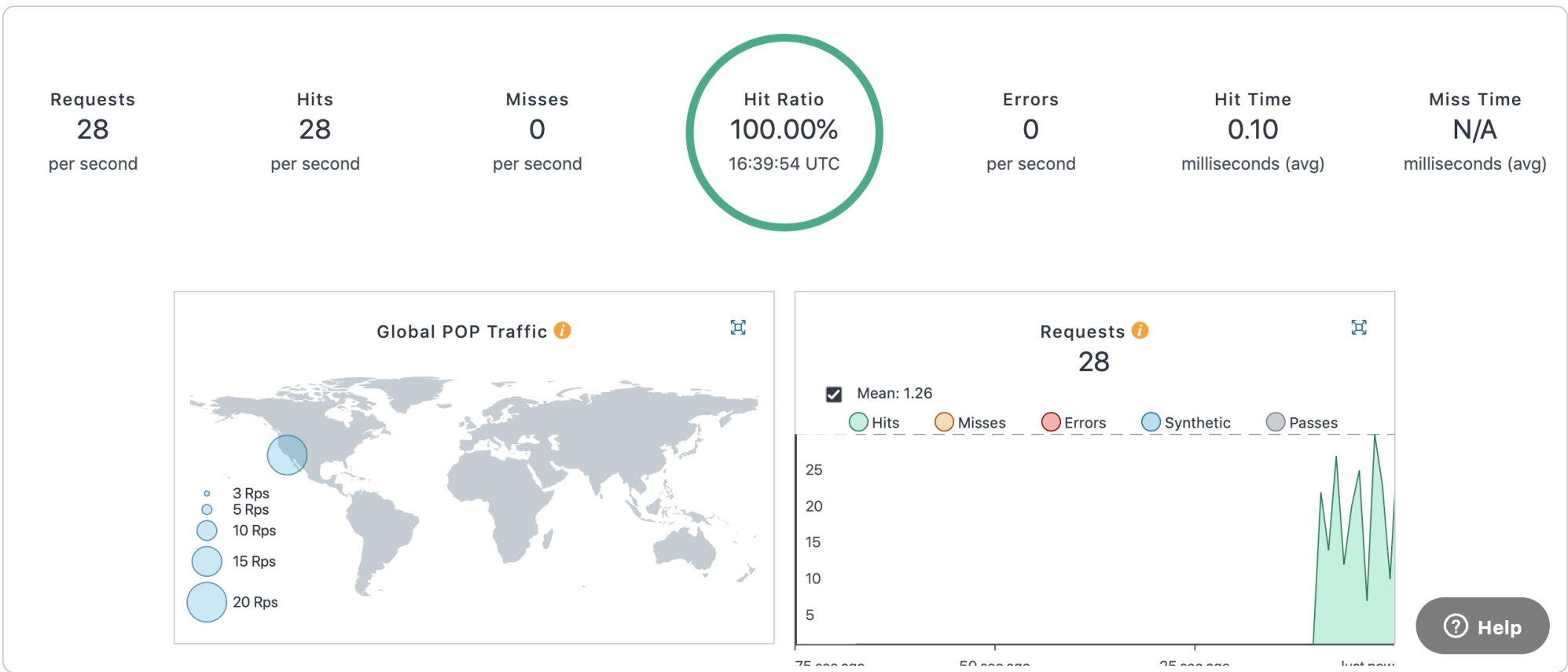
Up to this point, we've almost exclusively worked in the Deliver tab of the Fastly web interface. Now we're going to click **Stats** to head to another part of the web interface, as shown below. There are two pages under Stats: Real-time and Historic. As the name suggests, the real-time stats page displays real-time analytics and historical caching statistics.



There's nothing to see here yet, so let's send some traffic to our website. We can use the freely available Apache bench command line tool with the watch command to simulate visitors:

```
$ watch -n 1 ab -r -n 30 -c 30 -g out.data https://www.tacolabs.com/
```

With that command running in our terminal application, we start to see the real-time stats popping up on the Stats page, as shown below.

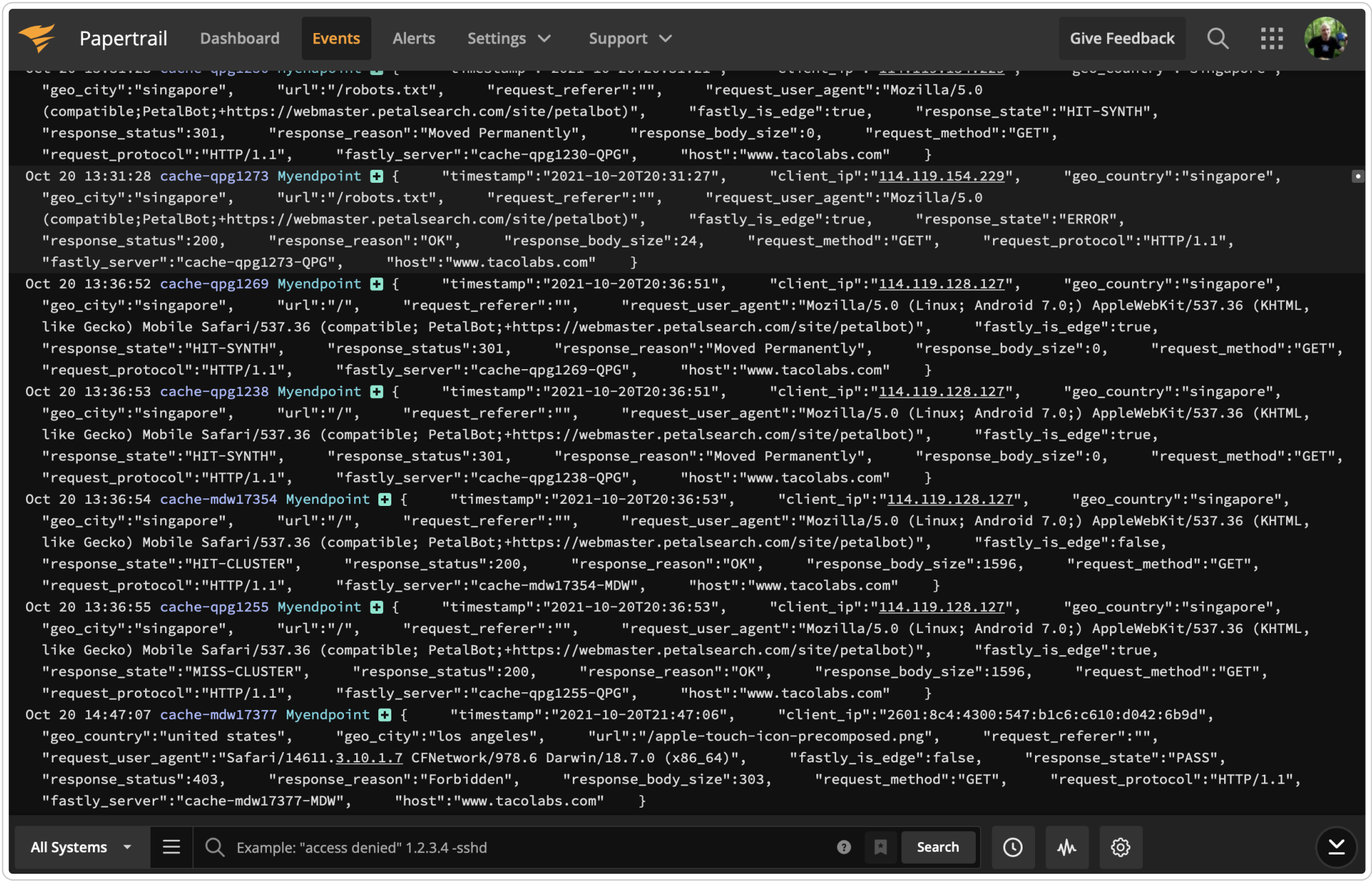


We won't waste time delving into the specifics, but know that the Stats page is a good way to get a bird's eye view of the traffic to your website or application. Our documentation provides more information about the Stats pages.

Checking logs

Earlier in this tutorial, we spent a lot of time setting up logging. That work is about to pay off. With production traffic ramping up on our Fastly service, we need to watch the logs to make sure visitors aren't experiencing any errors. In a real-world scenario, you'll want to look for error status codes, lower conversion rates, and any other issues specific to your website or application that might indicate there's a problem. Think of it as proactive monitoring. We're trying to head off small, unforeseen issues before they become catastrophic problems.

We've already set up two endpoints — one for Amazon S3 and another for Papertrail. Since the Papertrail endpoint is a real-time log streaming endpoint, that's the one we'll use here. The Amazon S3 is more of a historical record that can be used to examine old events. We can see from the Papertrail interface shown below that bots are already crawling Taco Labs, and there are some errors related to missing images.

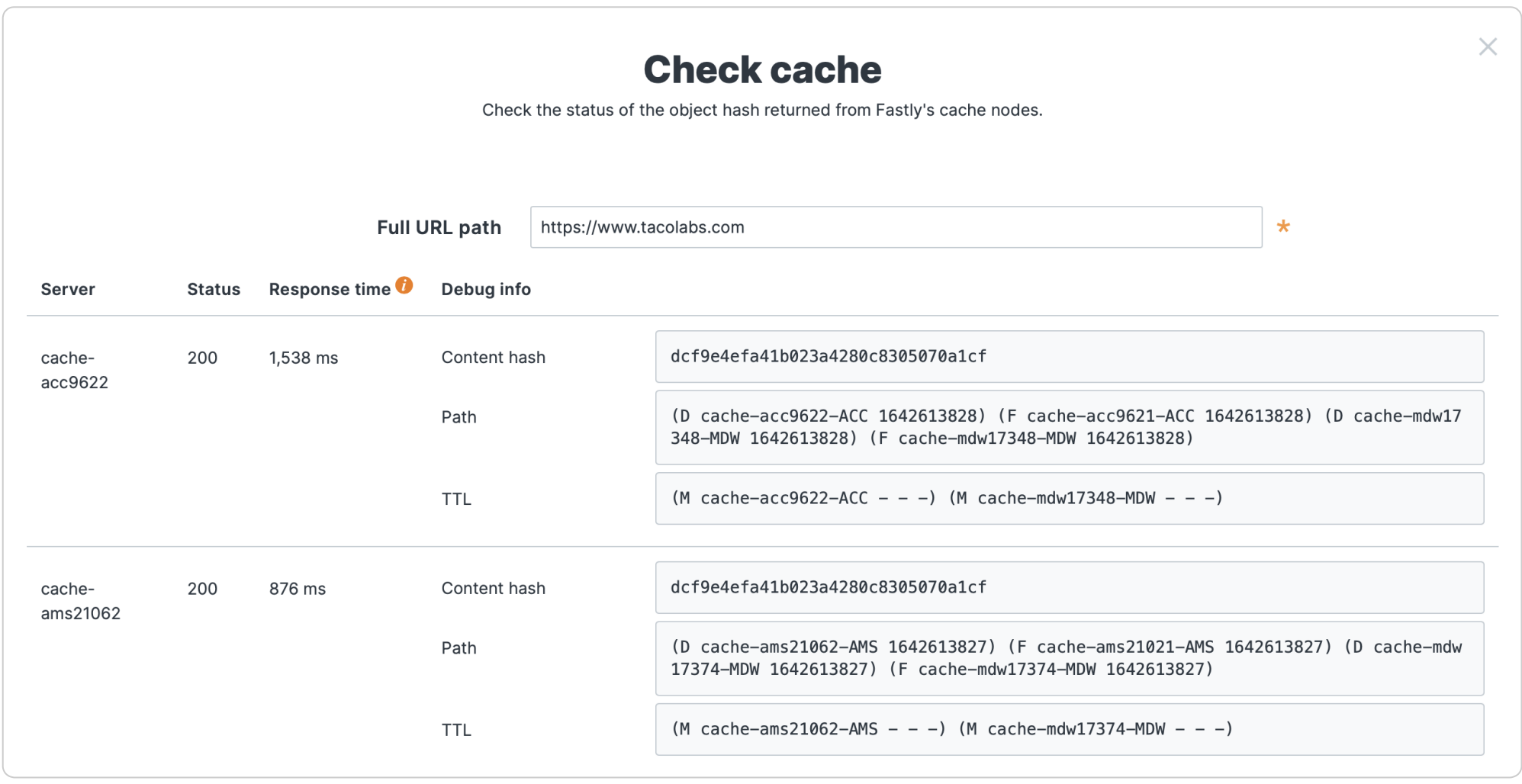


As discussed previously, it's a good idea to filter your logs to find the events you're really interested in. For example, you could filter out all of the entries with HTTP status code 200. That way, you'd only see the errors.

Checking cache using the web interface

We've been using curl throughout this tutorial to check how Fastly is caching our website. Now let's try using the web interface.

If we look under the Service summary tab, we'll see a **Check cache** button. We can click this button and then enter a URL that contains one of the domains we've specified in the web interface. The tool shows how an object is being cached throughout the Fastly network, as shown below.



As you can see, the cache checking tool in the web interface provides more information than curl. This should be one of the first tools you use when troubleshooting caching issues with your website.

6. Advanced configuration

Last updated: 2022-02-03

https://docs.fastly.com/en/fundamentals/6-advanced-configuration

This page is part of Fastly 101, a step-by-step tutorial that shows you how to use Fastly with an example website and domain name. It guides you through the steps of caching and delivering a static website using the Jekyll static site generator, Amazon AWS, and the Fastly CDN. For more information, see the introduction.

We could stop here! Fastly is caching our website. We're using TLS to secure connections and we're monitoring things with logs. Everything is working seamlessly. But, as you probably guessed, our work will never be entirely finished.

In this section, we'll provide some tips and tricks for dealing with those day-to-day website issues and headaches that come with the territory. Call it advanced configuration. We'll talk about using custom VCL to add redirects, using the API to purge the cache, and purging with surrogate keys.

Adding redirects using VCL snippets

When we remove content or rename a file, we should *redirect* the old URL to the new one. This helps inform search engines that the link is no longer available and that the new link should be indexed instead. Fastly makes it trivial to add redirects at the edge. By creating a couple of VLC snippets, we can add redirects to our Fastly service and activate them without ever touching our code or Amazon S3 bucket.

TIP

The instructions in this section come from our redirects tutorial. Refer to that documentation for a detailed explanation of the VCL code.

We'll create three VCL snippets to support our redirects. The first snippet will contain the redirect table — this is where we'll add URLs that we want to redirect. The other two snippets are "set it and forget it" — we won't need to touch these again after we create them. One matches inbound requests against our redirect table, and the other generates the redirection response.

The dataset of redirects that will be stored as a VCL table looks like this:

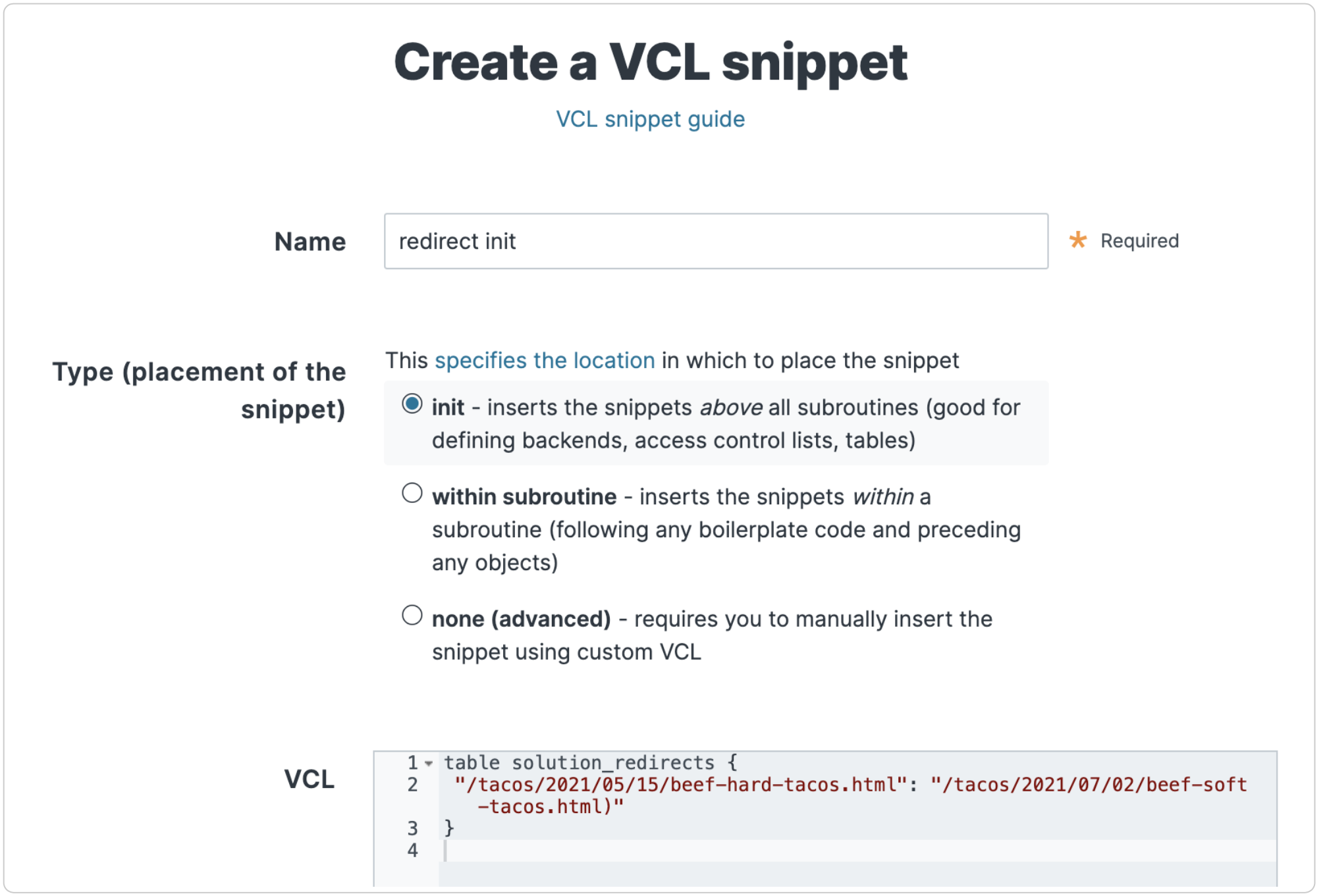
```
table solution_redirects {
 "/source1": "/dest1",
 "/source2": "/dest2"
}
```

Let's take a look at how this would work with Taco Labs. Say we had a recipe for beef hard tacos on our website that we needed to temporarily remove due to a critical national shortage of taco shells. We'll remove the file with that recipe, then take the old URL (`https://www.tacolabs.com/tacos/2021/05/15/beef-hard-tacos.html`) and redirect it to our recipe for beef soft tacos (`https://www.tacolabs.com/tacos/2021/07/02/beef-soft-tacos.html`) Our VCL table will look like this:

```
table solution_redirects {
 "/tacos/2021/05/15/beef-hard-tacos.html": "/tacos/2021/07/02/beef-soft-tacos.html"
}
```

Of course, we can continue adding URLs to this table in the future as we remove content and rename files.

We can use the Fastly web application to create all of the VCL snippets. Let's click the **Edit configuration** button to clone our service and create a new draft version. On the VCL snippets page, click the **Create snippet** button. We'll add a name for our snippet, select type **init**, and enter the VCL as shown below.



Click the **Create** button to save the VCL snippet.

Now we'll create another snippet to match inbound requests against the redirect table we just created. Using the web interface to create the snippet, set the type to **within subroutine** and **recv (vcl_recv)** and use the following VCL:

```
if (table.lookup(solution_redirects, req.url.path)) {
  error 618 "redirect";
}
```

Finally, we'll create the last VCL snippet to generate the redirection response. Using the web interface to create the snippet, set the type to **within subroutine** and **error (vcl_error)** and use the following VCL:

```
if (obj.status == 618 && obj.response == "redirect") {
  set obj.status = 308;
  set obj.http.Location = "https://" + req.http.host + table.lookup(solution_redirects, req.url.path) + if (std.strlen(req.url.qs) > 0, "?" req.url.qs, "");
  return (deliver);
}
```

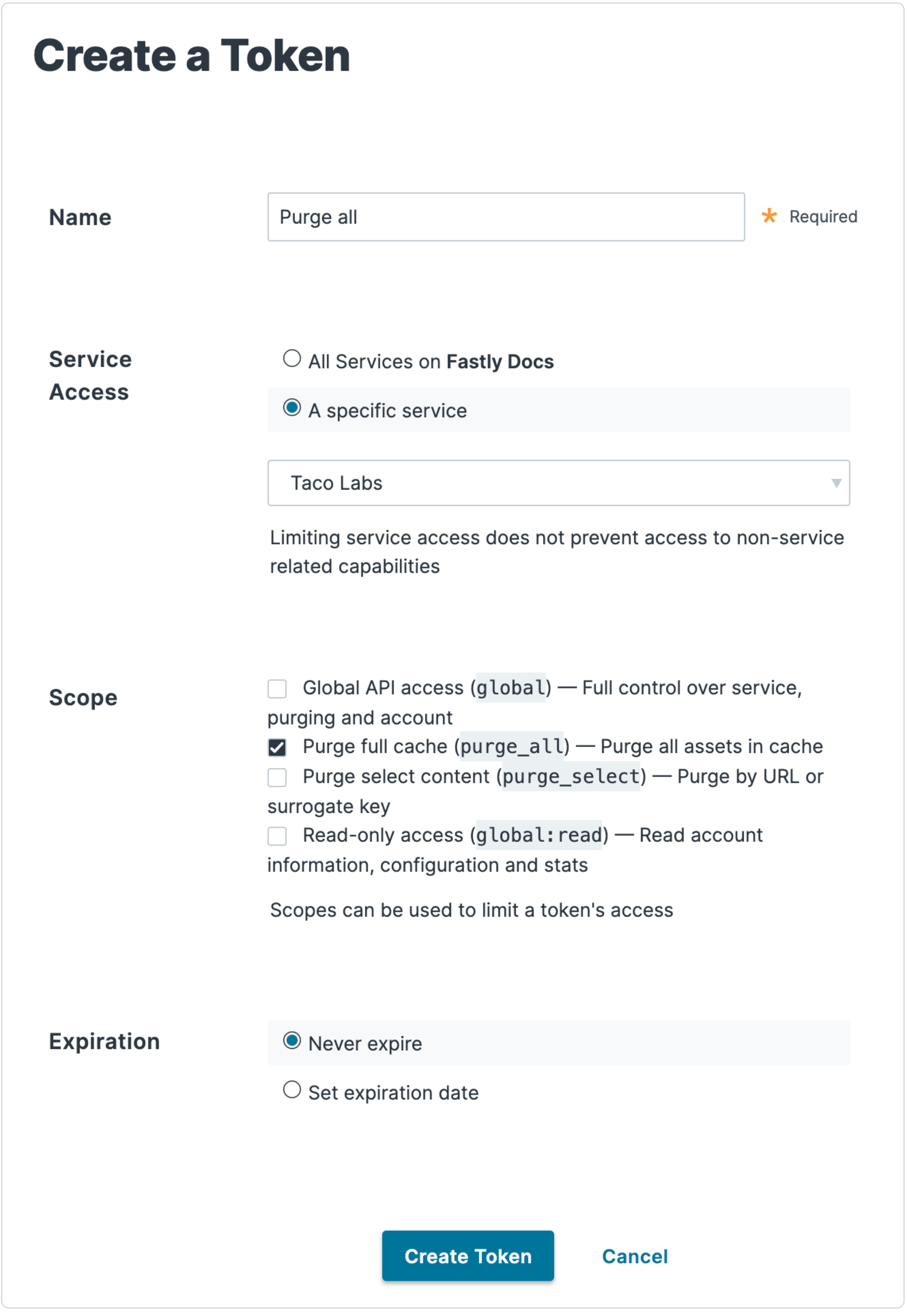
Let's click the **Activate** button to activate the new version of the service configuration. We can check that our redirect is working by pointing our web browser at `https://www.tacolabs.com/tacos/2021/05/15/beef-hard-tacos.html` — it should redirect us to `https://www.tacolabs.com/tacos/2021/07/02/beef-soft-tacos.html`. It's working!

Now that we've set everything up, it'll be easy to add new redirects in the future. We'll just clone our service, add the redirects to the VCL table, and activate our service. It's that easy.

Using the API to purge all cache

Recall that when we discussed our purging strategy earlier in this tutorial, we decided that using purge all was preferable to the other options. By using Fastly's API, we can use GitHub Actions to automatically perform the purge request as part of our deployment process.

First thing's first. Since purge all requests need to be authenticated, we'll use the web interface to create an account API token. From the user menu, select **Account**, and then select **Personal API tokens** from the sidebar. Click the **Create Token** button. Fill out the form fields as shown below.



Click the **Create Token** button to create the API token. Copy and paste the token to a safe location — you won't be able to see the token again after you browse away from the page.

Now we can test the purge all API endpoint using our Terminal application:

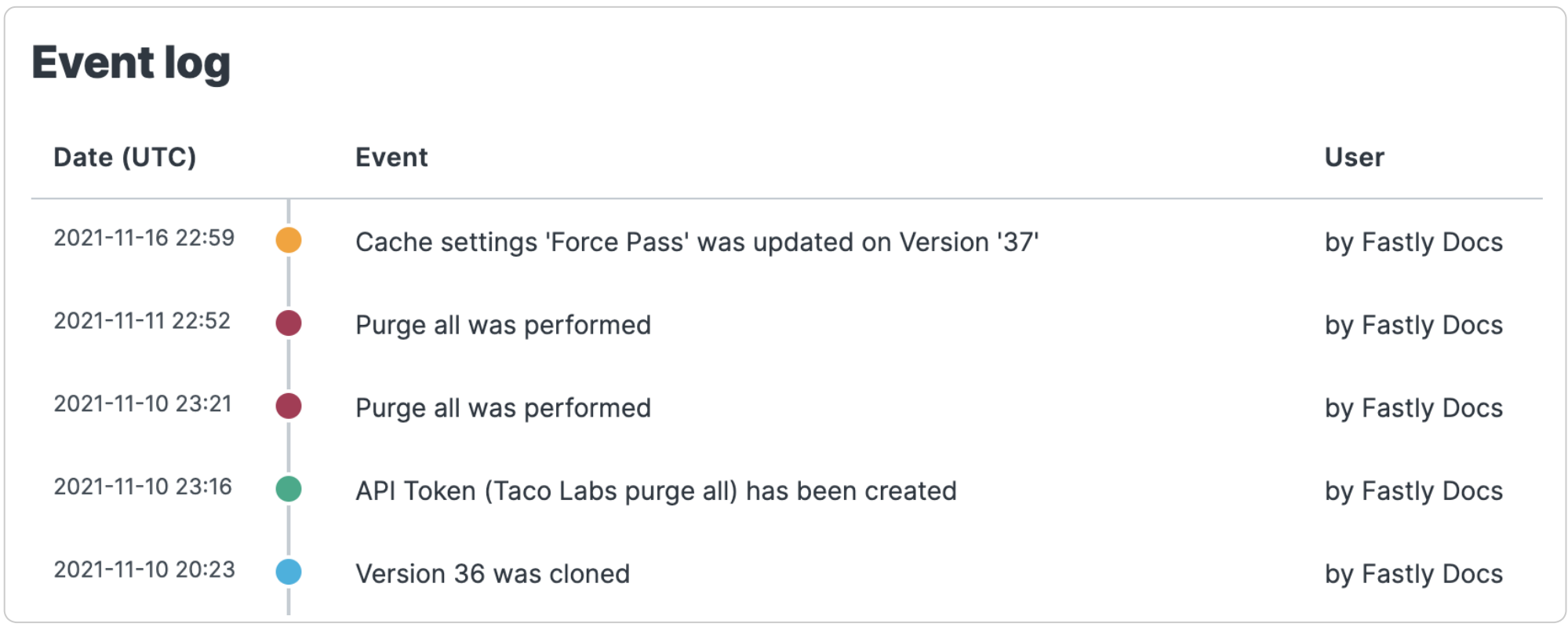
```
$ curl -X POST -H "Fastly-Key: YOUR_FASTLY_TOKEN" "https://api.fastly.com/service/SERVICE_ID/purge_all"
```

We should see the following response:

```
{"status":"ok"}
```

We can check that the purge request was successful by looking at the event log in the web interface, as shown below. It worked!

> **TIP**
>
> You can use roles and permissions to restrict a user's ability to purge using the Fastly web interface and API. For more information about the available user roles, see our documentation on configuring user roles and permissions.



Now we can add the curl command to the bottom of our GitHub Action:

```
    - name: "Purge Fastly cache"
      run: |
        curl -X POST -H "Fastly-Key: YOUR_FASTLY_TOKEN"
 "https://api.fastly.com/service/SERVICE_ID/purge_all"
```

That's it! Now our GitHub Action will automatically purge all of our Fastly cache after it deploys our website to Amazon S3.

Purging by surrogate key

We've committed to using purge all for our static site, but it's worth exploring — if only in theory — how we could use *surrogate keys* to purge a select group of content from cache. To understand how this could work, we'll take a subset of our content and use a header to tag those content items with a surrogate key. Then we can purge the cache of only the content items tagged with that surrogate key.

Let's say that we need a surrogate key for all of our mix-ins. Pretend that the prices of the ingredients change so often that we need to update and purge the mix-in content multiple times a day. We'll use a VCL snippet to create a `mixin` key that we'll use to tag all of our mix-ins content. We could also set the surrogate key at the origin or application level.

Let's click the **Edit configuration** button to clone our service and create a new draft version. On the VCL snippets page, click the **Create snippet** button. We'll add a name for our snippet, set the type to **within subroutine** and **fetch (vcl_fetch)** and use the following VCL:

```
if (req.url.path ~ "/mix-ins/") {
  set beresp.http.Surrogate-key = "mixin";
}
```

This will set the surrogate key for all content in the mix-ins path. Review the settings in the web interface, as shown below, and then click the **Create** button.

Create a VCL snippet

VCL snippet guide

Name mix-in surrogate key * Required

Type (placement of the snippet)

This specifies the location in which to place the snippet

- ○ **init** - inserts the snippets *above* all subroutines (good for defining backends, access control lists, tables)
- ◉ **within subroutine** - inserts the snippets *within* a subroutine (following any boilerplate code and preceding any objects)

  fetch (vcl_fetch)
- ○ **none (advanced)** - requires you to manually insert the snippet using custom VCL

VCL

```
1 if (req.url.path ~ "/mix-ins/") {
2   set beresp.http.Surrogate-key = "mixin";
3 }
```

Let's click the **Activate** button to activate the new version of the service configuration. After performing another purge all (follow the instructions from the previous section), we can check that our redirect is working by checking the headers of the mix-in index page:

```
$ curl -svo /dev/null -H "Fastly-Debug:1" "https://www.tacolabs.com/mix-ins/"
```

We can see the `surrogate-key` header in the output:

```
1  < HTTP/2 200
2  < x-amz-id-2: eDYmcGNu4Pk+AnTjHy5aO1kuOI802/tOdstMxa5FdZV0doL8Kjn7p/C+UBiE7u04Xg9bTmVooR8=
3  < x-amz-request-id: 6KF7YW10F6Q19T0Y
4  < last-modified: Thu, 04 Nov 2021 22:38:08 GMT
5  < etag: "d89bec2183579e4c842d2590bc45bf9f"
6  < content-type: text/html
7  < server: AmazonS3
8  < via: 1.1 varnish, 1.1 varnish
9  < surrogate-key: mixin
10 < cache-control: no-store, max-age=0
11 < surrogate-control: max-age=31557600
```

Now we can use the API to purge content tagged with the surrogate key:

```
$ curl -X POST -H "Fastly-Key: YOUR_FASTLY_TOKEN" "https://api.fastly.com/service/SERVICE_ID/purge/mixin"
```

---

7. Conclusion

Last updated: 2022-03-24

https://docs.fastly.com/en/fundamentals/7-conclusion

This page is part of Fastly 101, a step-by-step tutorial that shows you how to use Fastly with an example website and domain name. It guides you through the steps of caching and delivering a static website using the Jekyll static site generator, Amazon AWS, and the Fastly CDN. For more information, see the introduction.

That's a wrap! We've successfully started using Fastly to cache our static website. Go ahead, visit https://www.tacolabs.com — or whatever domain name you've been using to follow along with — and marvel at how fast the site loads. And give yourself a pat on the back for doing all of the legwork!

We've covered a lot of ground in this tutorial. Just think. You've learned the basics of content delivery networks and why you'd want to use Fastly. You've created a Fastly service and added a domain and origin. You've configured a variety of caching settings, set up logging, and learned how to invalidate cache by purging. And last but not least, you've learned how to update DNS records and configure TLS through Fastly to secure your website.

You now know everything you need to know to start using Fastly with your own website or web application. With some minor modifications, these instructions could be adapted to work with virtually any website or application. For example, you could use a client library or a plugin to configure your application to interact with the Fastly API. Feel free to refer back to this tutorial as you set things up. We can't wait to see what you build with Fastly!

What's next

Learn more about Fastly's products and features by exploring our documentation on https://docs.fastly.com and https://developer.fastly.com. If you have questions, contact our support team at support@fastly.com.