Fastly Help Guides Archive

Generated: Fri, 31 Jan 2020 15:38:23 +0000

Support guides

Getting started

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

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

Basics

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

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

About the web interface controls

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

In addition to being accessible via Fastly's application programming interface (API), Fastly services can also be accessed via a web interface for users with the appropriate access permissions.

NOTE: 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.

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 the Login link at the top right of almost all pages at the Fastly website.

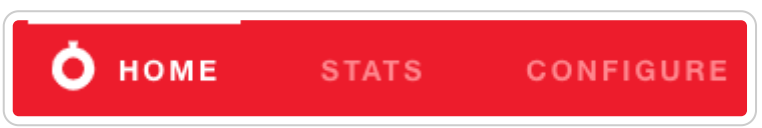
Once logged in to a Fastly account, the web interface controls appear as appropriate based on the roles and permissions assigned to you.

The default control groups appear as follows from left to right across the top of the interface:

- the All services (Home) page
- the Stats page
- the Configure controls
- the user menu

About the All services page

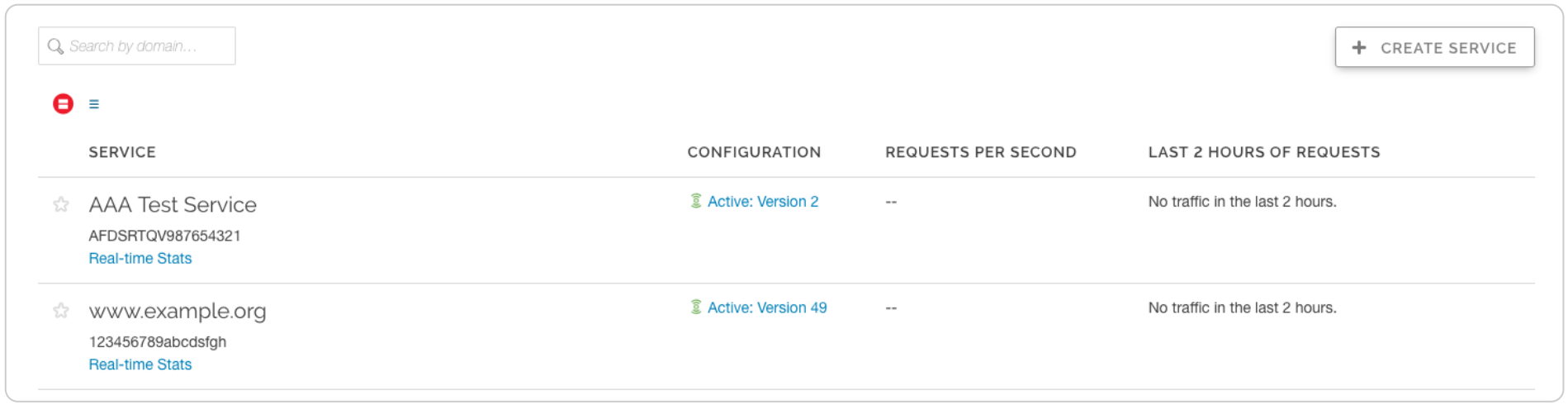
The All services page displays a summary of all your services, sorted by requests per second. It appears automatically when users with the appropriate access permissions log in to the Fastly web interface. You can access it by clicking **Home** next to the stopwatch icon.



The All services page allows you to:

- access the real-time stats information for a particular service by clicking the Real-time Stats link under the service name
- 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 type must be an exact match to find the desired service.

About the Stats page

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

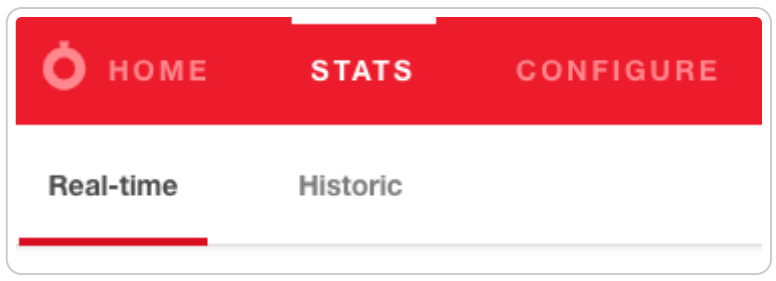
From here, you can access the:

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

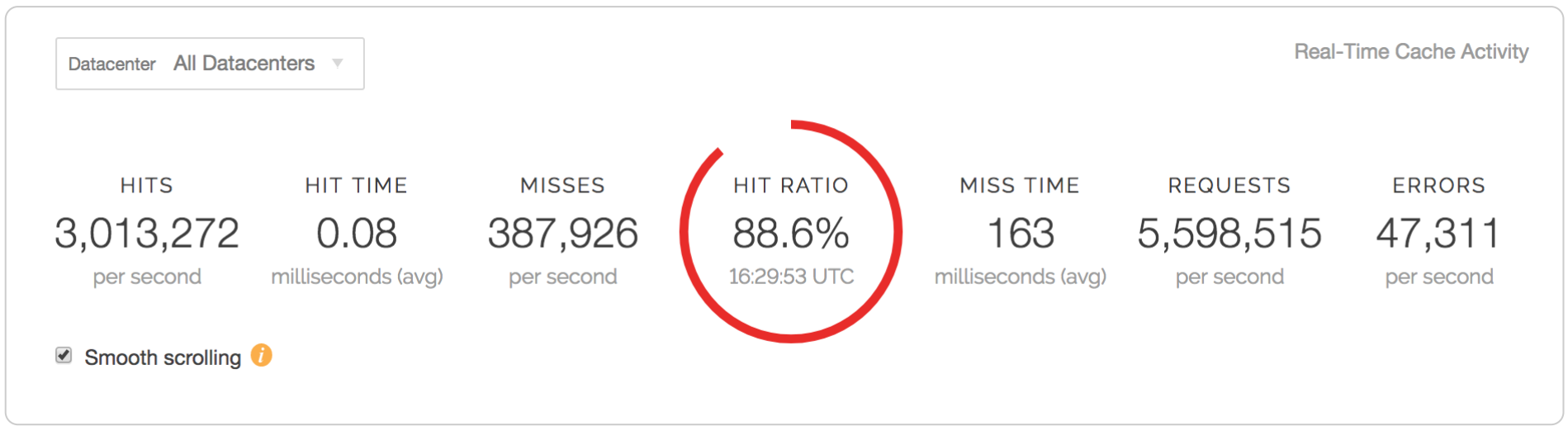
The Stats link appears automatically for logged in users with the appropriate access permissions.

Viewing the Real-time stats

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



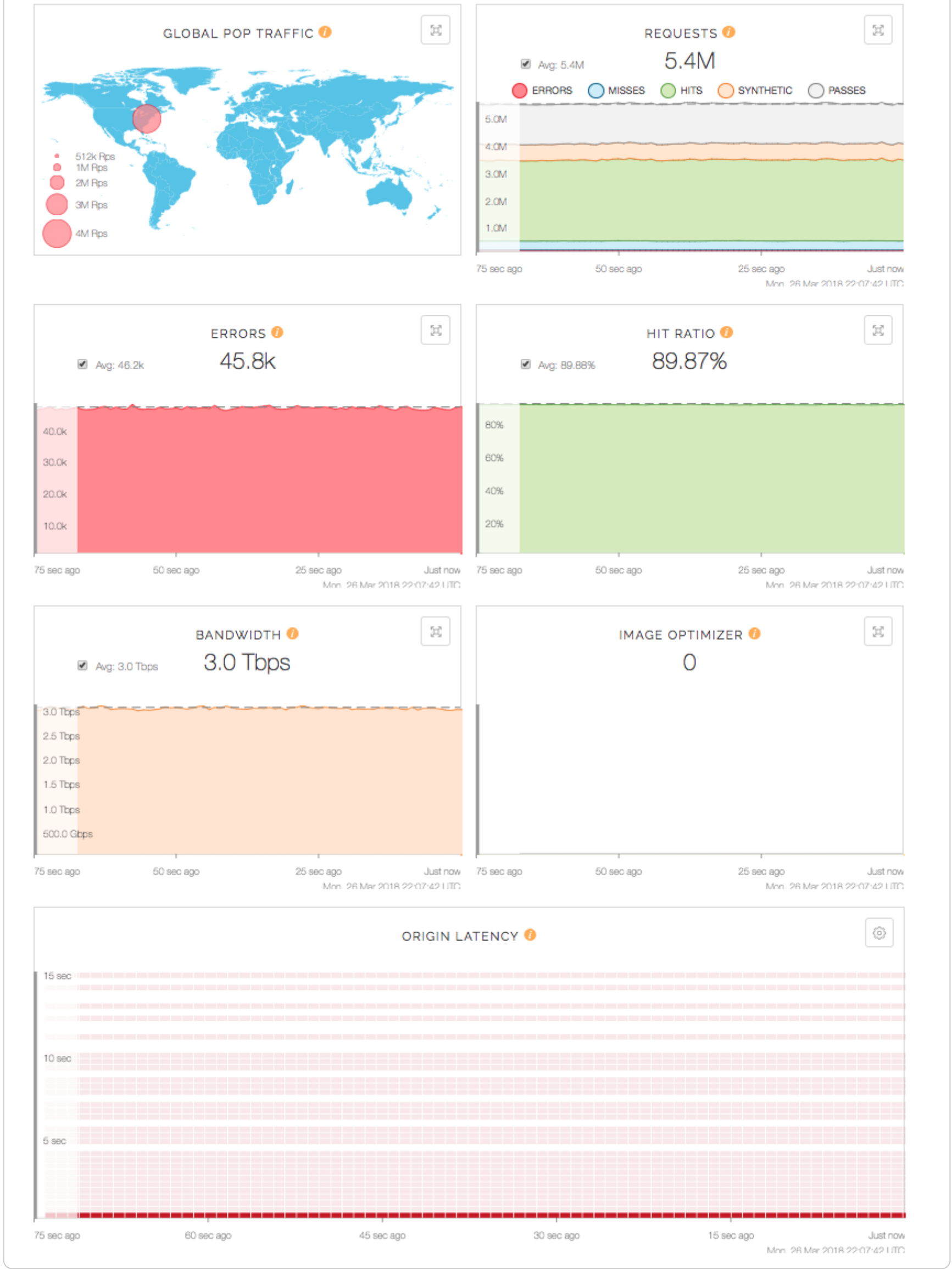
The data on this page may appear grayed out or blank to some users, with no information displayed in the controls, when a customer's service does not receive enough requests for Fastly to display meaningful information about it in real time. If you've just created your service, you might see a message saying there's nothing to see yet.



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

- **Hits:** the number of times requested data is found in cache
- **Hit Time:** the amount of time spent processing cache hits
- **Misses:** the number of times requested data is not found in cache
- **Hit Ratio:** the percentage of content being accessed that is currently cached by Fastly, defined as the proportion of cache hits to all cacheable content (hits + misses)
- **Miss Time:** the amount of time spent processing cache misses
- **Requests:** the total number of requests received for your site by Fastly
- **Errors:** the number of error requests that occurred

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



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 Optimization Requests:** when enabled, a graph displaying the number of responses that came from the Fastly Image Optimizer service over time.
- **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.

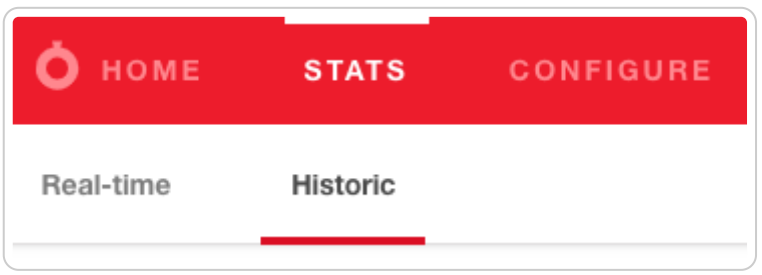
One minute after real-time measurement data in these graphs rolls off the screen, it becomes available for retrieval on the historic stats page. You may not see any traffic right away because of the following:

- **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 page provides a visual interface to our Stats API for a selected Fastly service. 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 versus 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 over time 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 shows the number of HTTP Info (1XX), Success (2XX), and Redirect (3XX) statuses served for your site using Fastly.
- **Status 3XX Details** metrics shows the breakdown between the number of HTTP Status 301s, 302s, 304s, and other 3XX requests.
- **HTTP Client and Server Errors** metrics shows the number of HTTP Client Errors (4XX), and Server Errors (5XX) served for your site by Fastly.
- When enabled, **Image Optimization Requests** metrics show you the number of responses that came from the Fastly Image Optimization service.

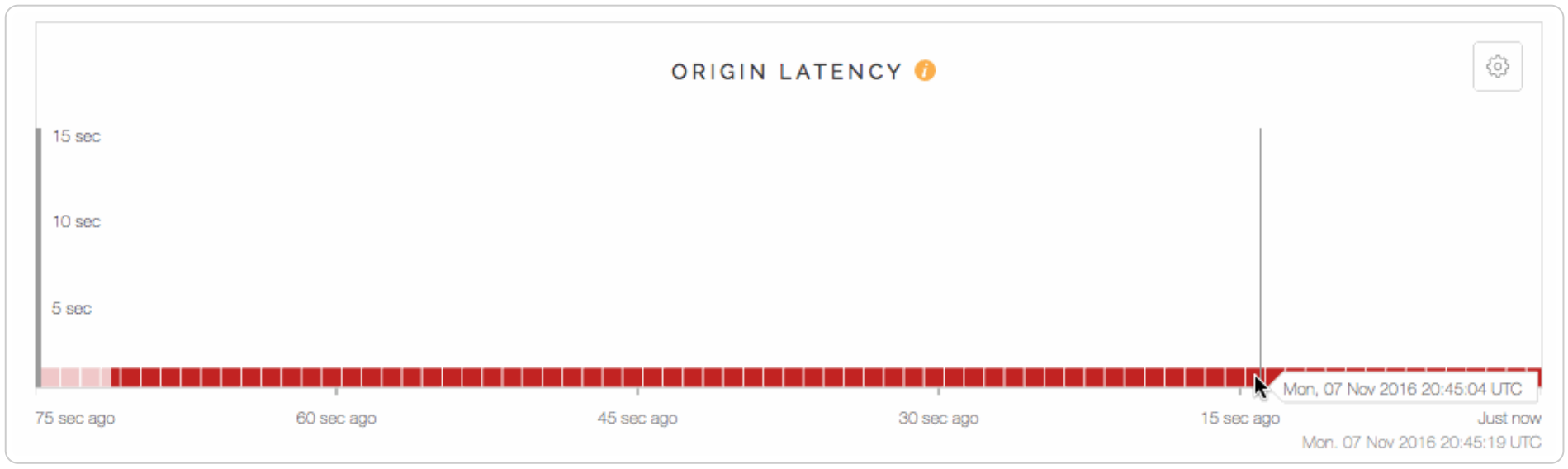
Once you start to see your caching and performance metrics, you also can interact with your stats graphs.

Working with stats graphs

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

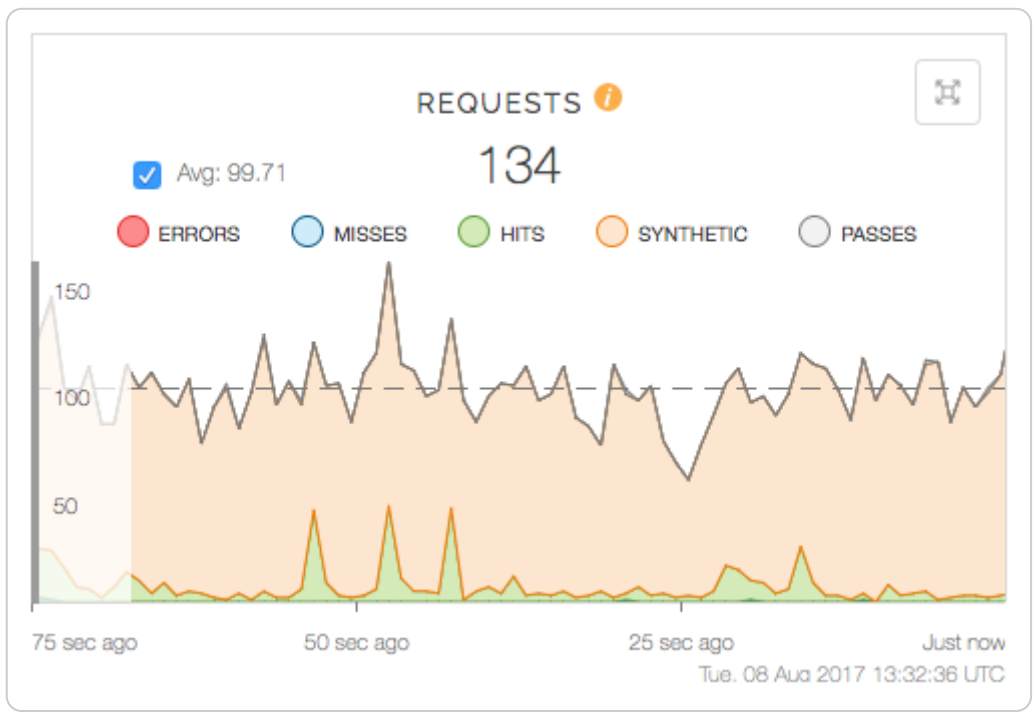
Viewing the real-time stats timestamp indicator

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 average link

The average line appears as a dashed line on some graphs. To hide the average line for a graph, deselect the **Avg** checkbox.



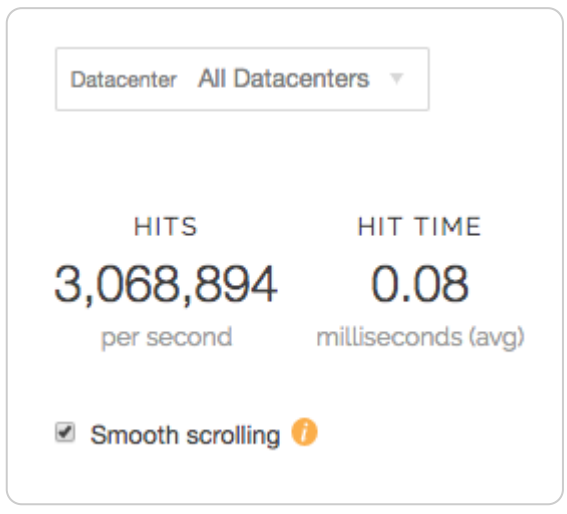
Expanding and minimizing graph views

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 shorter time frame.

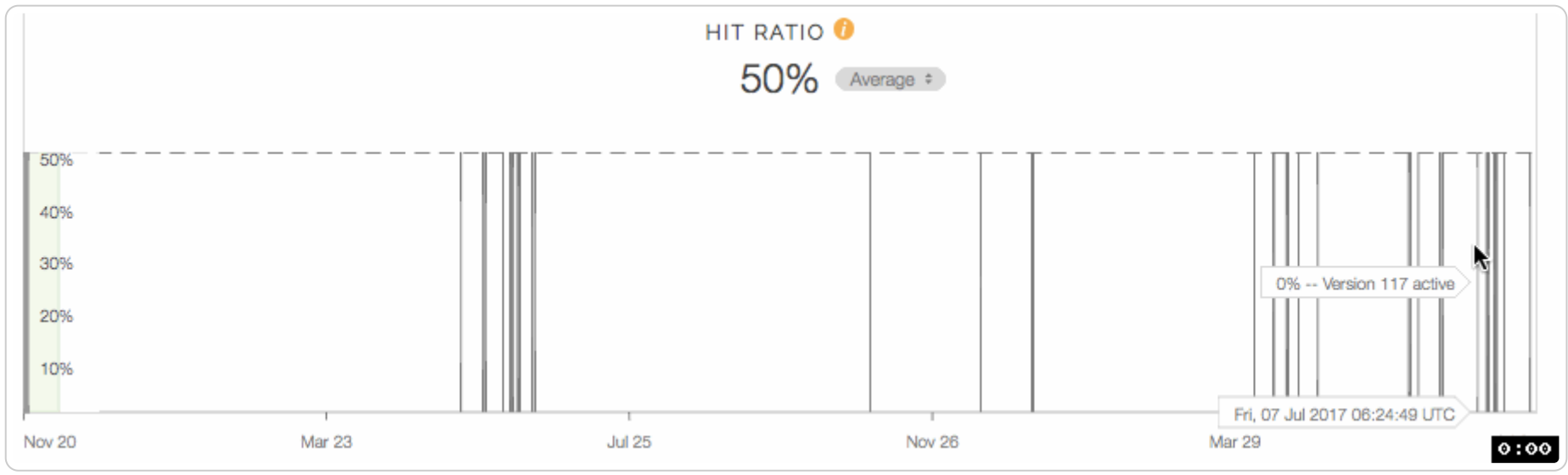
Disabling smooth scrolling

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 service version activation

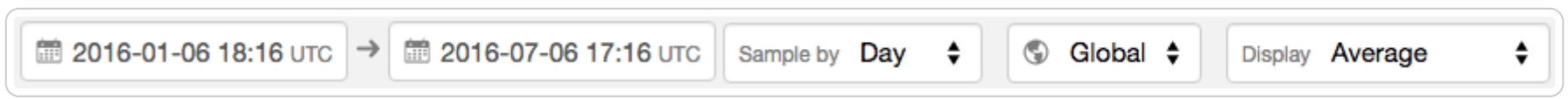
Service version activations appear as vertical lines on the Historic graphs. Hovering your cursor over any line displays the version's number and its activation timestamp.



IMPORTANT: You cannot retrieve minutely historical statistics data older than 90 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.

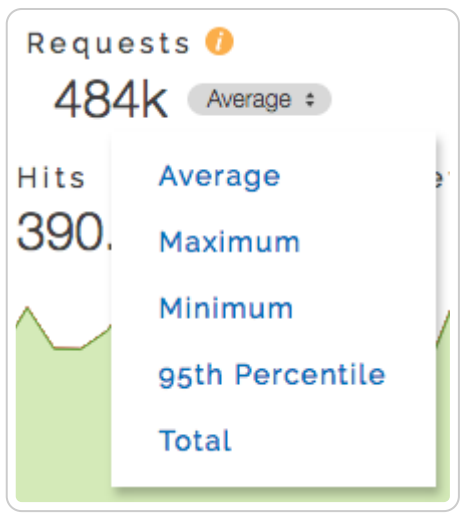


For all displayed graphs, you can choose:

- 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

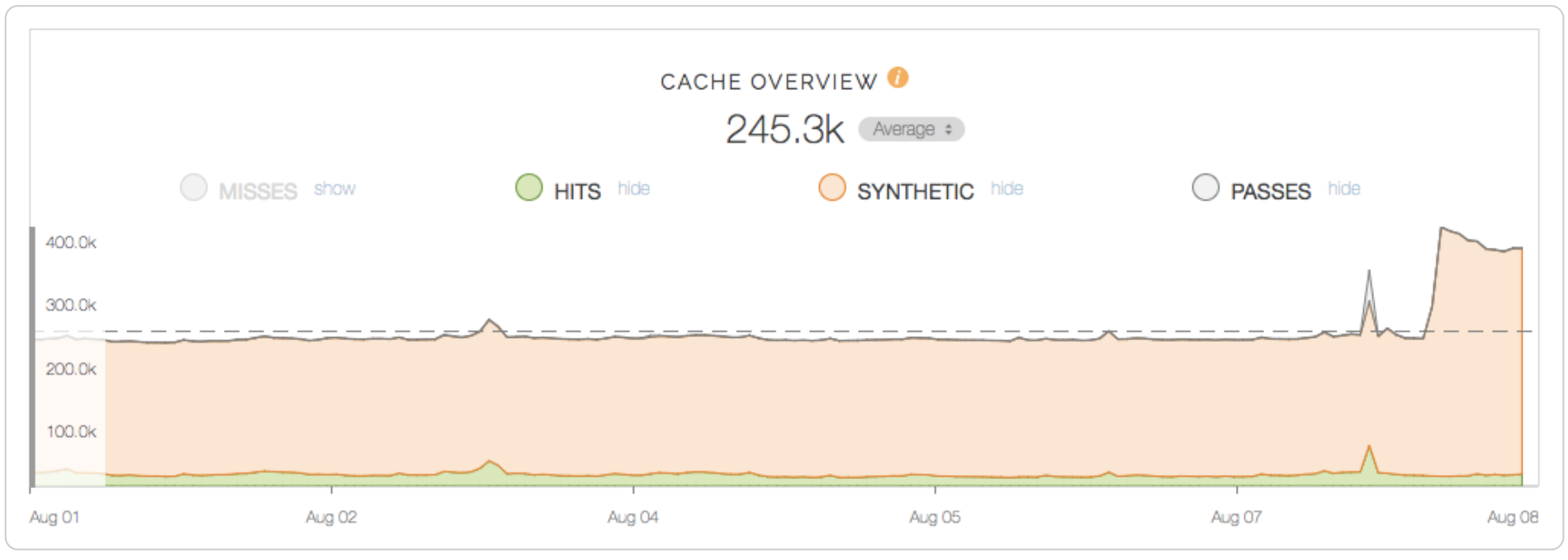
Changing the stats displayed in a graph

You can change the statistics displayed in each graph. For example, notice the Average button on this Requests graph:



Clicking this button on any graph (including those on the Real-time stats page) lets you change the display of the graph's data to an average, a 95th percentile, a minimum, a maximum, or a total. When set to average, the graph displays the average as a dashed line.

You can also exclude certain data entirely. For example, in this Caching Overview graph, hovering the cursor over the word "Hits" next to any of the data values displays a small, clickable **hide** link. Clicking this link will hide that value in the graph's display.



Notice that the actual numbers of the hidden data still appear grayed out in the controls, but the hide link changes to a show link and the graph itself doesn't display the hidden data at all.

About the Configure page

The Configure page allows you to define exactly how each instance of your cache should behave and deliver content from data sources. The Configure page appears automatically for logged in users with the appropriate access permissions.

You use the Configure page to create versions of each of your service's configuration settings and then use the controls to activate or deactivate them.

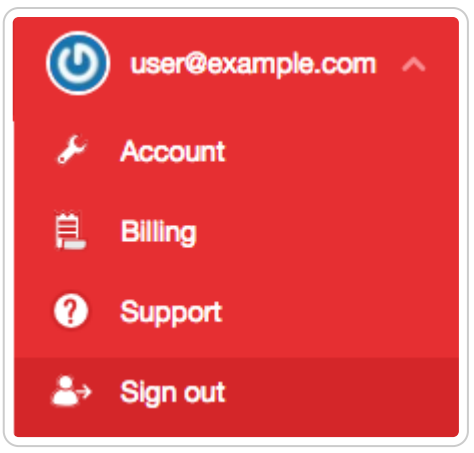
Specifically, you can configure and manage:

- the domains used to route requests to a 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 service
- how logging should be performed and where server logs should be sent (as specified by an rsyslog endpoint)
- 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 Configure page also allows you to compare differences between two configuration versions.

About the user menu

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



Depending on your access permissions, it provides access to a variety of account-specific and personal settings information. Specifically, it gives you access to Account details, Billing information and access to Support. It also provides a way for you log out of the web interface.

About the Account controls

Selecting Account from the user menu displays account-related details for your login with specifics about your Company Profile which include:

- **Company settings** where you'll find details about your company (e.g., its name and the phone number, the IP allowlist) as well as the location to 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
- **Transport Layer Security** (TLS) lets you add and manage your domains on one of Fastly's shared TLS certificates
- **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

You'll also find Personal Profile information here. Specifically:

- **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

About the Billing controls

Selecting Billing from the user 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

---

Always-on DDoS mitigation

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

Browser recommendations when using the Fastly web interface

https://docs.fastly.com/en/guides/browser-recommendations-when-using-the-fastly-web-interface

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.

Content and its delivery

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.

Fastly POP locations

https://docs.fastly.com/en/guides/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 the current and planned locations of all Fastly POPs. In addition, our datacenter API endpoint provides a list of all Fastly POPs, including their precise latitude and longitude locations.

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

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

Fastly continues to grow its network footprint, adding 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.

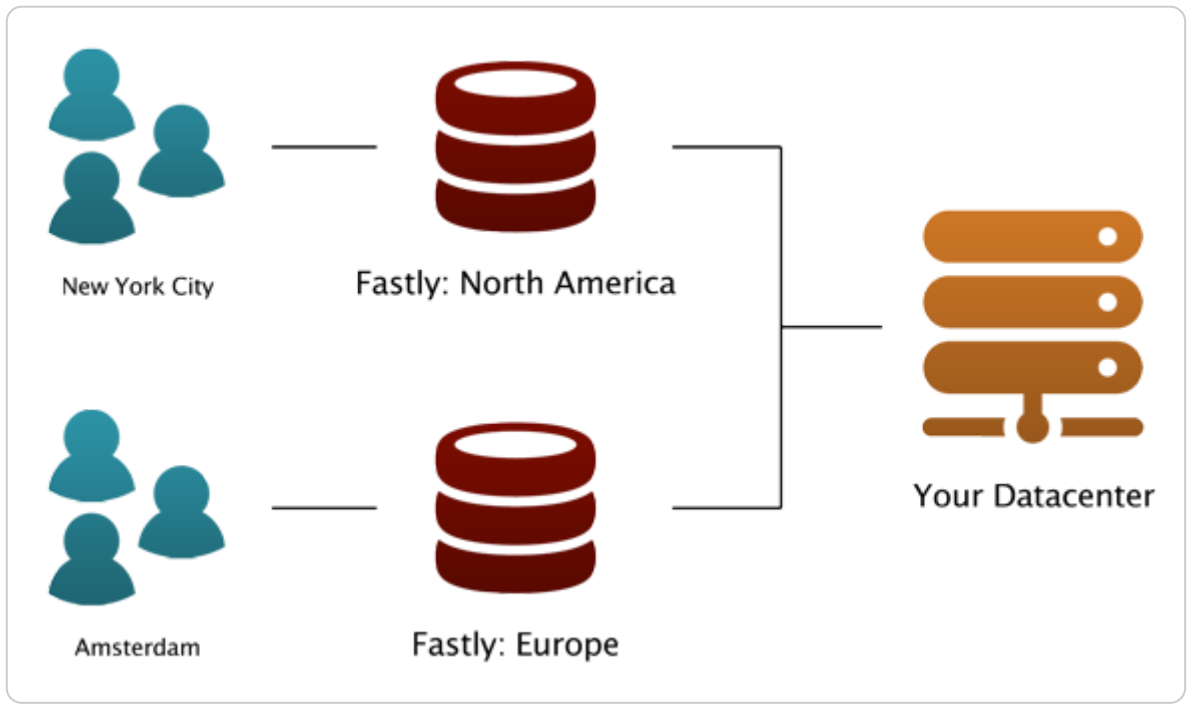
Getting started with Fastly

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

In this guide, we explain what Fastly does and how best to use it with your site.

How Fastly works

Fastly works by storing the content of your website on servers all over the world and quickly delivering that content to your users. We do this using Varnish, an open source web application accelerator.



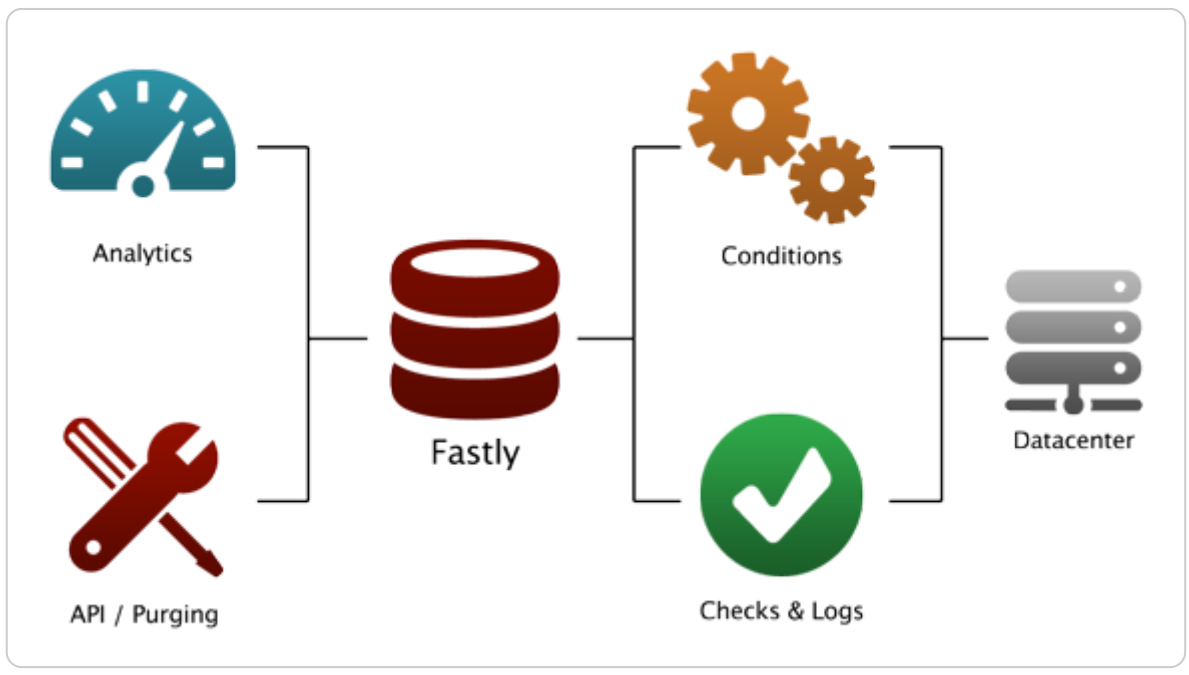
We track the geo-location of each user and make sure they are connecting to a server that is closest to them. This makes your site faster by reducing the time spent waiting for data to be sent from the server to the user.

We also give you full control over when and how we store content from your servers. You can set a Time To Live (TTL) for any path on your site and instantly invalidate or purge any path on your site using our Purge API.

By using these tools, you only have to generate pages one time for the site for many millions of page views. This saves time for your users and costs on your server bills.

Advanced features

Fastly also provides many advanced features that help you monitor how your data is accessed and customize your content delivery.



- **Instant Purging** allows you to have better control over when and how content is updated. You can update your data when you want and as often as you want, rather than waiting up to 24 hours to change data at the edge.
- **Real Time Analytics** provides a top level view of your network and how your site is performing. Every second, we compile the relevant data about all of your traffic into an easy-to-read report.
- **Conditions** change how requests are routed, what headers to send, and how content is cached.
- **Health Checks** monitors the status of one or many of your back end servers. This way if anything goes wrong with your servers, you immediately know about it.
- **Streaming Logs** are quickly and easily configured to send information from your servers anywhere and in the format you want.
- **Varnish Configuration Language (VCL)** allows you to modify nearly every aspect of an HTTP request and response. You can upload VCL files with specialized configurations to your account.
- The **Fastly API** can programmatically handle your configuration. This allows you to write scripts to handle basic configuration tasks and create your own administrative views (so they can be directly coupled with your existing admin software).

Getting started

If you don't have an account, sign up and create your first service. Feel free to choose the developer trial plan so you can test how Fastly works on your site. The interface will step you through everything you need to set up and configure your first service for your site.

To explore more about basics, check out our guide describing how caching and CDNs work. Then consider exploring our guides to services, purging, and shielding.

If you want information on advanced features, especially related to things like load balancing or the Varnish Configuration Language that we support, check the advanced configuration section of our help files or the API Reference, which includes a full reference to the Fastly API.

If you are having problems, send us a message at support@fastly.com.

How caching and CDNs work

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 files 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:

- `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 cache control tutorial .

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 cache control docs. In order of preference:

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

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 datacenter (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

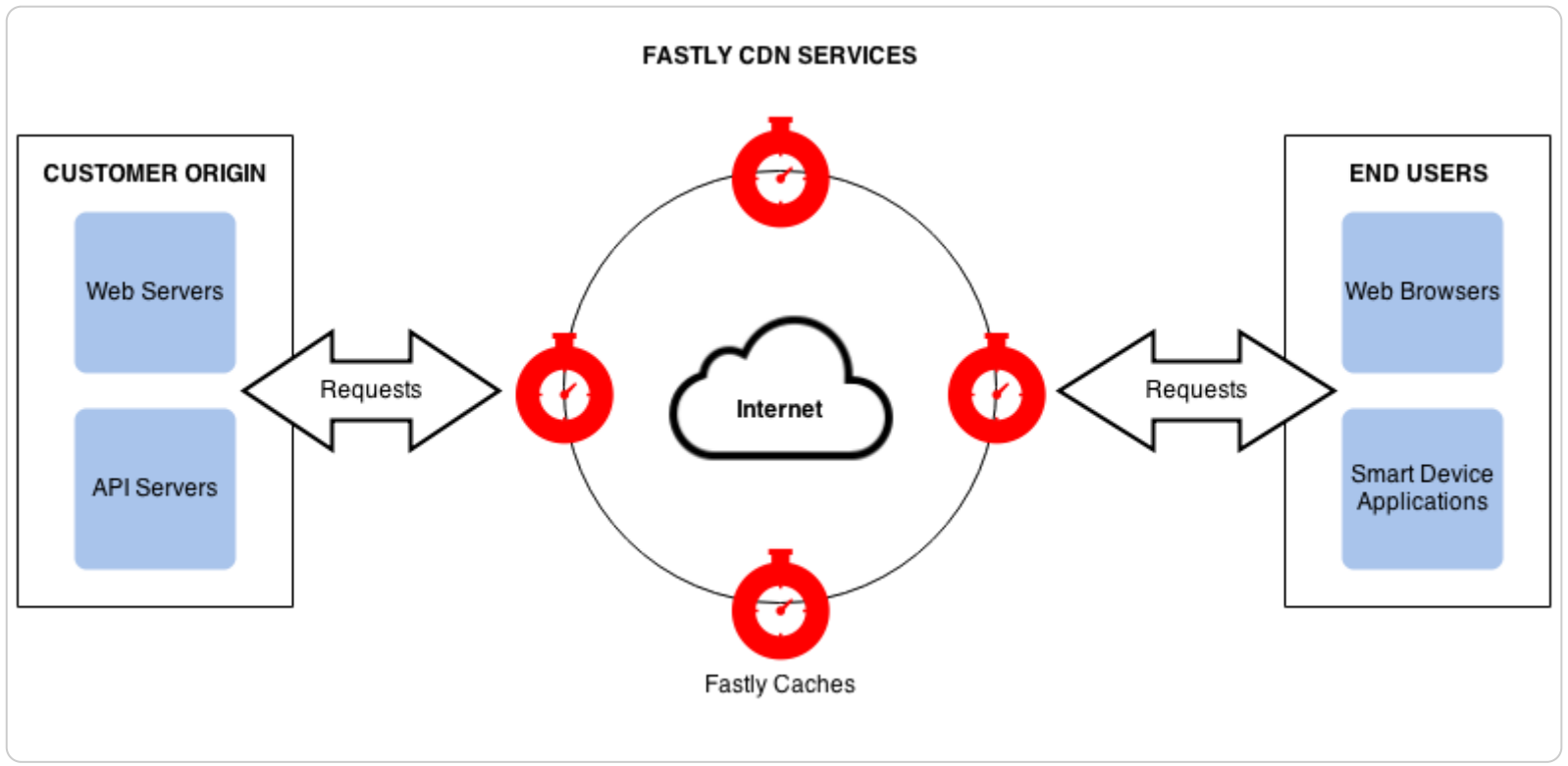
- Wikipedia: Reverse Proxy
- Fastly's cache control docs
- Mark Nottingham's caching tutorial
- Surrogate header W3C tech note

How Fastly's CDN Service works

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."

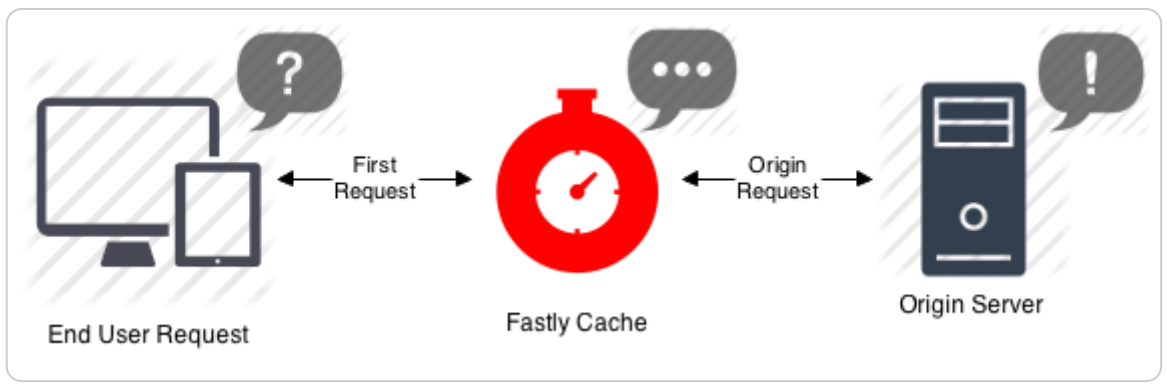


Fastly's delivers its CDN service from key access points to the internet called "points of presence" (POPs). Fastly places POPs where their connectivity to the internet reduces network transit time when delivering content to end-users. Each POP has a cluster of Fastly cache servers. When end users request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

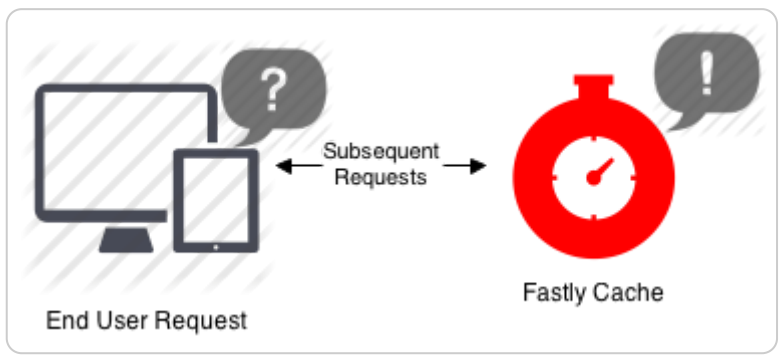
Fastly's caches only receive and process your end user requests for content objects. You decide which objects will be cached, for how long, who can access them, whether they are to be encrypted when transmitted over the internet, and when the objects will be deleted from the caching service. You make 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 datacenter 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.

HTTP status codes cached by default

https://docs.fastly.com/en/guides/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
`302`	Moved Temporarily
`404`	Not Found
`410`	Gone

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;
  }
  # ...
}
```

Self-provisioned Fastly services

https://docs.fastly.com/en/guides/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
- managing domains on TLS certificates

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.

Sign up and create your first service

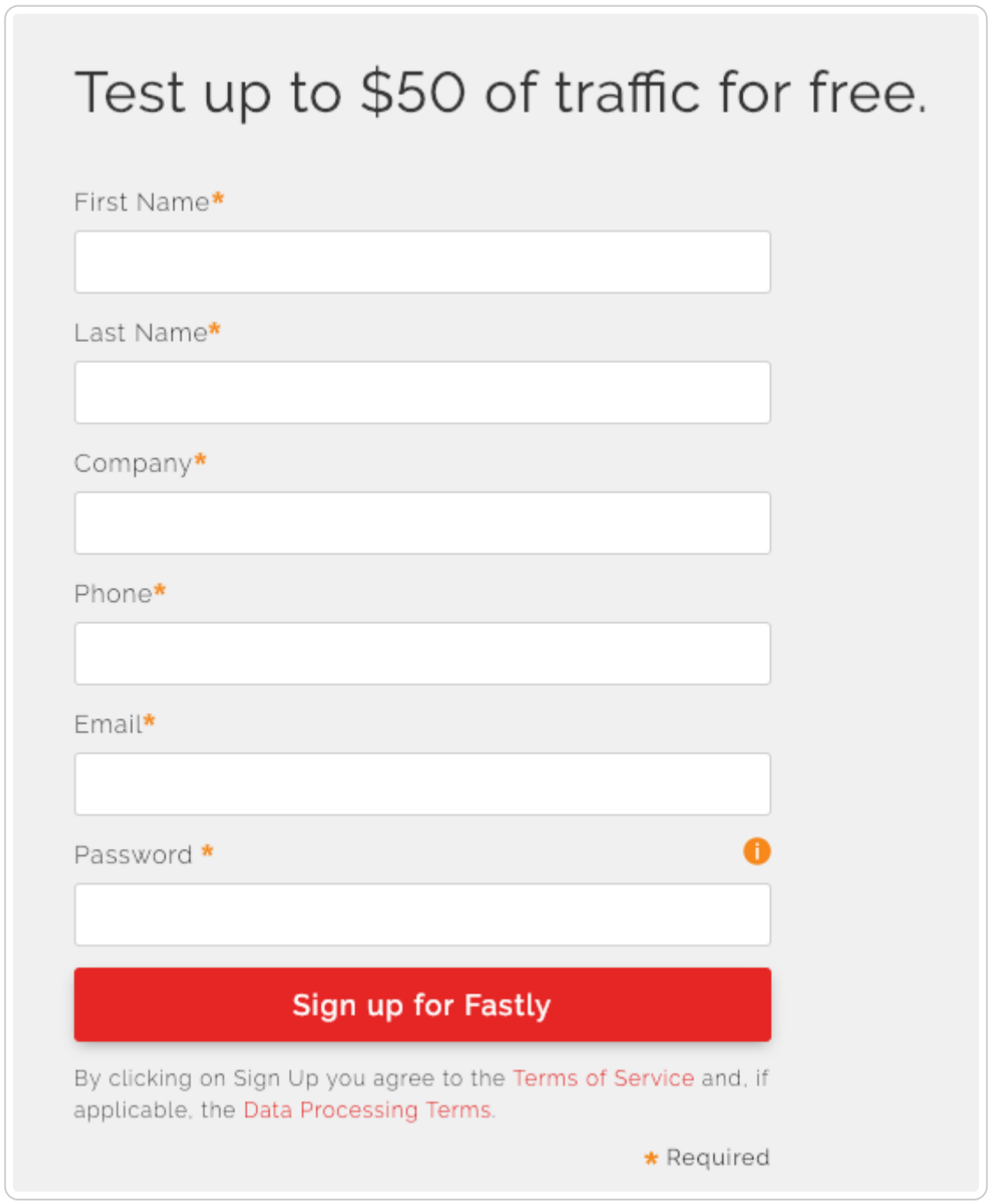
https://docs.fastly.com/en/guides/sign-up-and-create-your-first-service

To get started using Fastly, sign up for an account and then create your first service. Once you've created your service you can explore various configurations as much as you need before going live. Fastly won't start serving your traffic until you specifically set the CNAME DNS record for your domain.

Sign up at Fastly.com

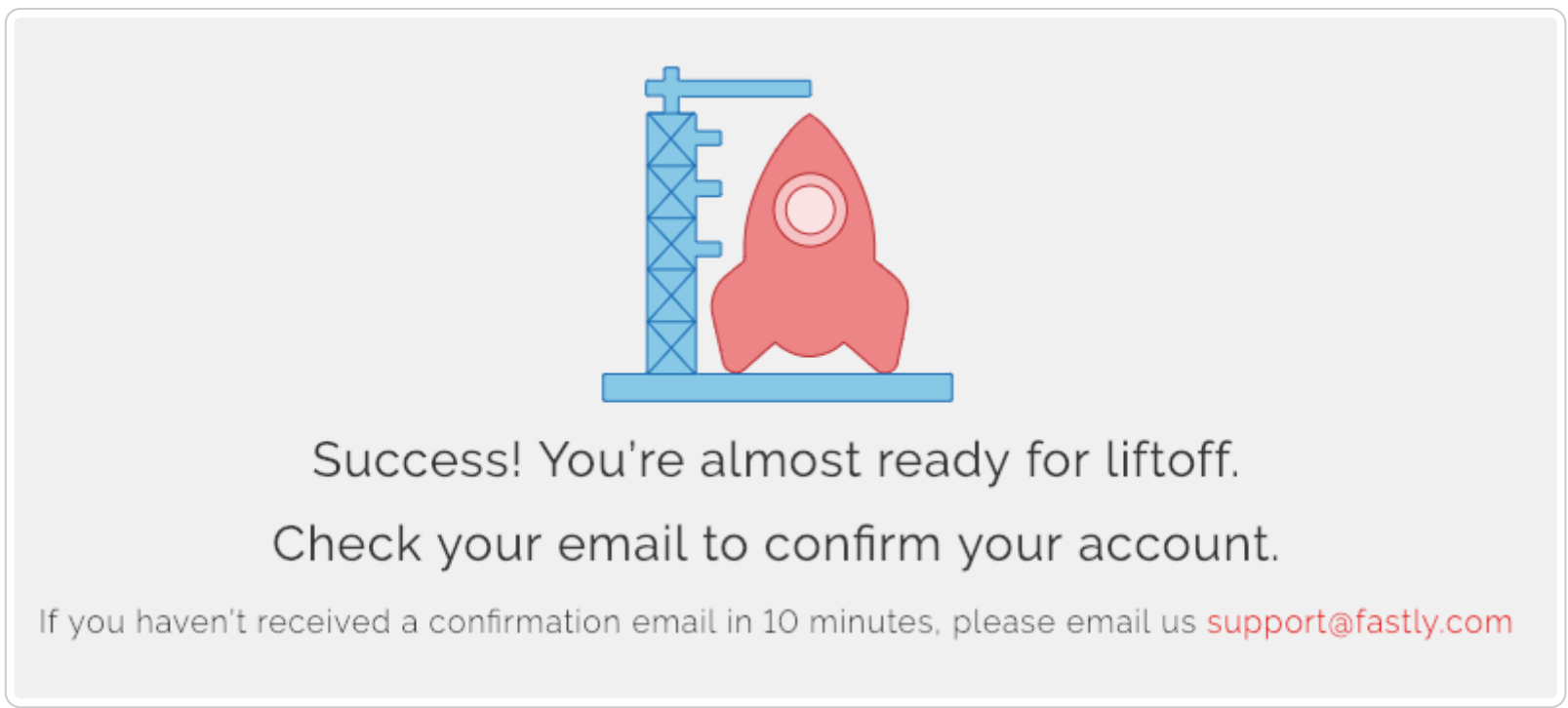
Before you do anything else, you must sign up for a Fastly account.

1. Click on any **Sign Up** button on the Fastly.com website or simply point a browser to the signup form.
2. When the signup form appears, fill in all the fields with your contact information. All the fields are required.



NOTE: You'll be able to change your password and email address any time after signup. Without a valid email we can't send you account verification details during account setup. Without a valid telephone number, we can't assist you with specific kinds of account lockout issues.

3. Click the **Sign up for Fastly** button. The confirmation screen will appear with instructions on what to do next and you'll be sent an e-mail that contains a verification link.



4. Check your inbox and find the confirmation email we sent you.
5. Click the verification link (we need to make sure you're not a spam robot and verify your email). The verification link will immediately take you to your empty home page.

Experiment with your first service

Once you've signed up at Fastly.com, we'll guide you through the process of creating your first service. When you receive your account verification email from us, we'll log you into the application automatically. All you'll need to do to start creating your first service is click the **Get Started** button.

Use our configuration page with your first service to experiment with and test initial configurations safely, without impacting any traffic to your origin (traffic won't flow there until you update your DNS records). When you're ready, we'll guide you through the process of creating a domain and a host, setting up your TLS, and finally activating your first service.

Test your configuration experiments

As you experiment with service configurations, you can test their success before you start serving traffic through Fastly. Simply open **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.

You can continue to explore various configuration settings for as long as you like, testing things repeatedly as you go.

Start serving traffic through Fastly

Once you're ready, all you need to do to complete your service setup and start serving traffic through Fastly is set your domain's CNAME DNS record to point to Fastly. For more information, see the instructions in our Adding CNAME records guide.

Next steps

Consider exploring these areas next:

- Working with services
- Log streaming
- Health Checks
- Diagnostics

Working with services

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. The Fastly web interface allows you to create new services or edit existing ones and then activate new versions of them that include your changes once you have things configured the way you want. The web interface also allows you to do other things with existing services, like rename them, compare them to each other, deactivate or reactivate them, and delete them.

Creating a new service

You might want to create a new service version to do things like:

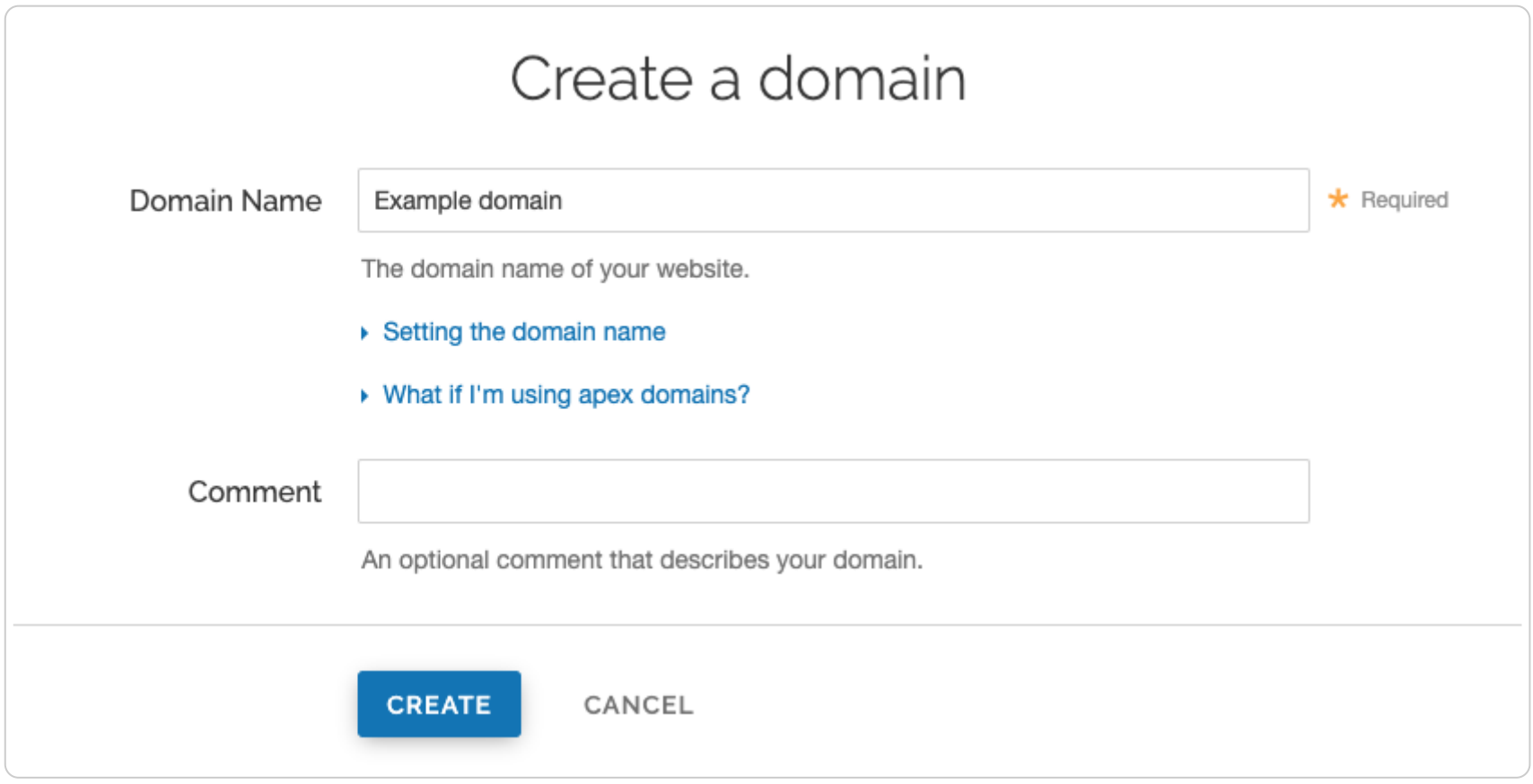
- add a new website you control to your list of web properties
- add a new domain to your growing list of existing domains already served by Fastly
- isolate traffic metrics for specific digital assets, like a site's images

Creating a service requires you to first create a domain and then a host.

Creating a new domain

To create a new domain, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. Click the **Create service** button. The empty state page appears.
3. Click the **Domains** link. The Domains page appears.
4. Click the **Create domain** button. The Create a domain page appears. Our guide to working with domains describes more about domains and what you can do with them.

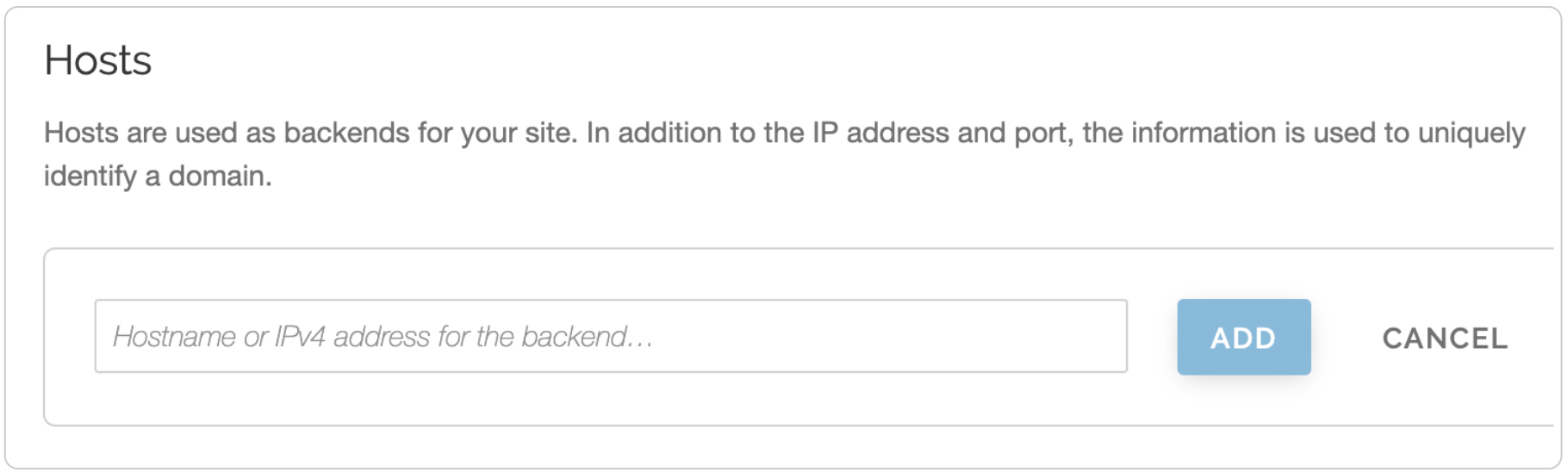


5. Fill out the **Create a domain** fields as follows:
   - In the **Domain Name** field, type the name users will type in their browsers to access your site.
   - In the **Comment** field, optionally type a comment that describes your domain.
6. Click the **Create** button. A new domain appears on the Domains page but the Activate button remains inactive until you add a host

Creating a new host

Once you create your domain, you can create a new host by following the steps below:

1. Click the **Origins** link. The Origins page appears.
2. Click the **Create a host** button. The Hosts field appears.

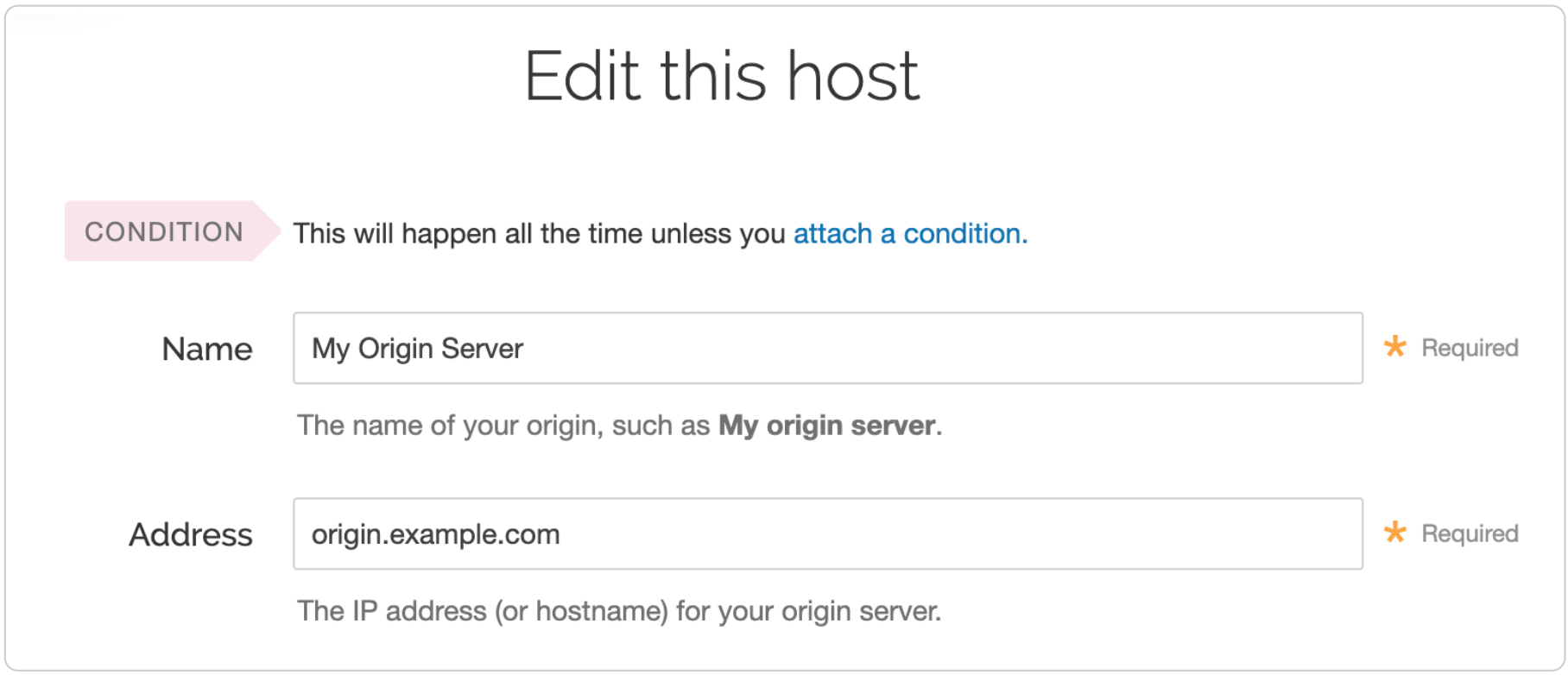


3. Fill out the **Hosts** field by typing the hostname or IP address of your origin server. Entering a hostname automatically enables Transport Layer Security (TLS) and assigns port 443. Entering an IP address disables TLS and assigns port 80.
4. Click **Add** to add your host.

Editing a host

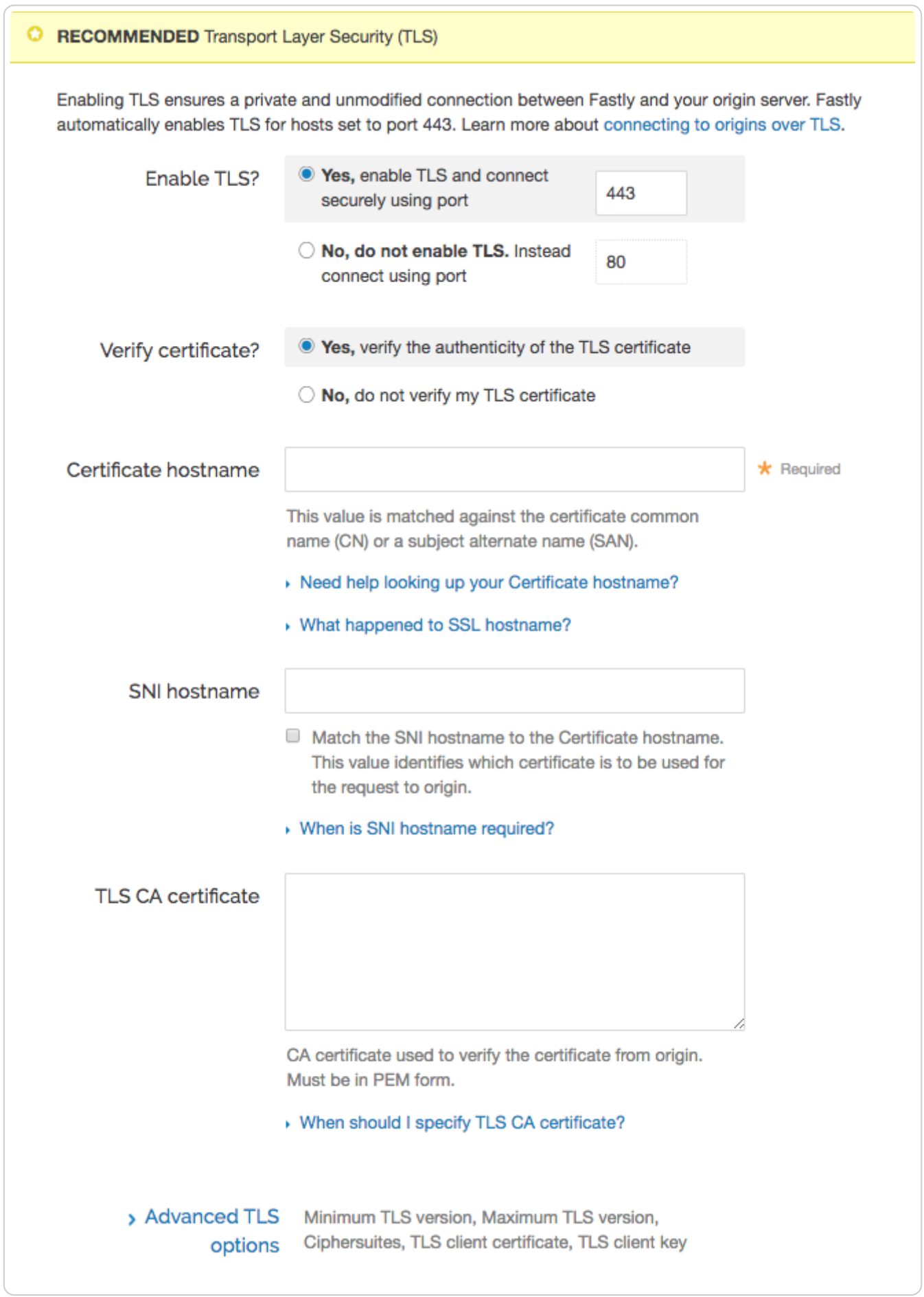
After you've created your host, you can edit the host's details by following the steps below:

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, type the name of your server (for example, `My Origin Server`). This name is displayed in the Fastly web interface.
   - In the **Address** field, optionally edit the IP address (or hostname) of your origin server.

Although we recommend setting up TLS when you set up a host, it's not required for your initial exploration and configuration testing.



3. Fill out the **Transport Layer Security (TLS)** area as follows:
   - Leave the **Enable TLS?** default set to **Yes** if you want to enable TLS and 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.
   - Leave the **Verify certificate?** 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 attack. Consider uploading a CA certificate instead of disabling certificate validation.

   - In the **Certificate hostname** field, type your certificate 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 specify your SNI hostname. This is generally only required when your origin is using shared hosting, such as Amazon S3, or when you use multiple certificates at your origin. See Setting the TLS hostname for more information. * In the **TLS CA certificate** field, optionally include your TLS CA certificate. You may want to provide the CA certificate if you're using a certificate that is either self-signed or signed by a certification authority (CA) not commonly recognized by major browsers. See Specifying a TLS CA certificate for more information.
4. Click the **Update** button. The new service appears in the list of services available.
5. Click the **Activate** button at the top right of the screen. A confirmation window appears.

6. Click the **Confirm and Activate** button to confirm you want to activate your new service. The **Configure** page appears displaying details about the configuration settings of the first version of your new service.

Testing your configuration experiments

As you experiment with service configurations, you can test their success before you start serving traffic through Fastly. Simply open **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.

You can continue to explore various configuration settings for as long as you like, testing things repeatedly as you go.

Serving traffic through Fastly

Once you're ready, all you need to do to complete your service setup and start serving traffic through Fastly is set your domain's CNAME DNS record to point to Fastly. For more information, see the instructions in our Adding CNAME records guide.

Editing your services

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

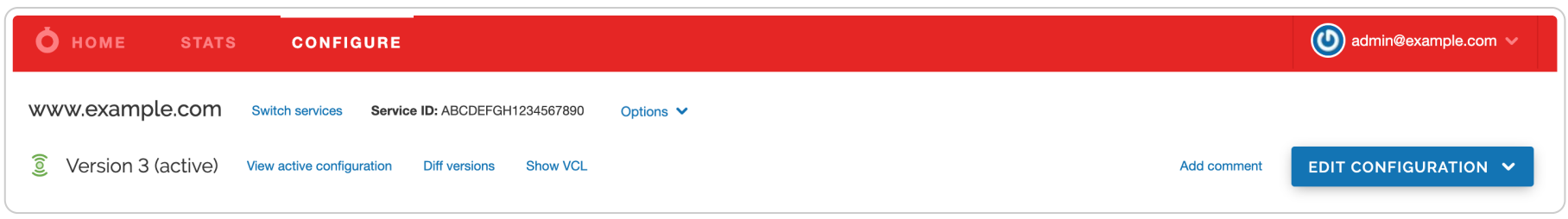
- change the amount of time information is retained in cache memory for a service
- configure a service to temporarily serve stale content should your origin server need to be unavailable for an extended period of time (for example, taken offline for maintenance)
- decrease the amount of time Fastly 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 in order 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 and click the **Configure** link. The Configure page appears.



2. Click the **Edit Configuration** button. The Edit Configuration menu appears.

3. 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.

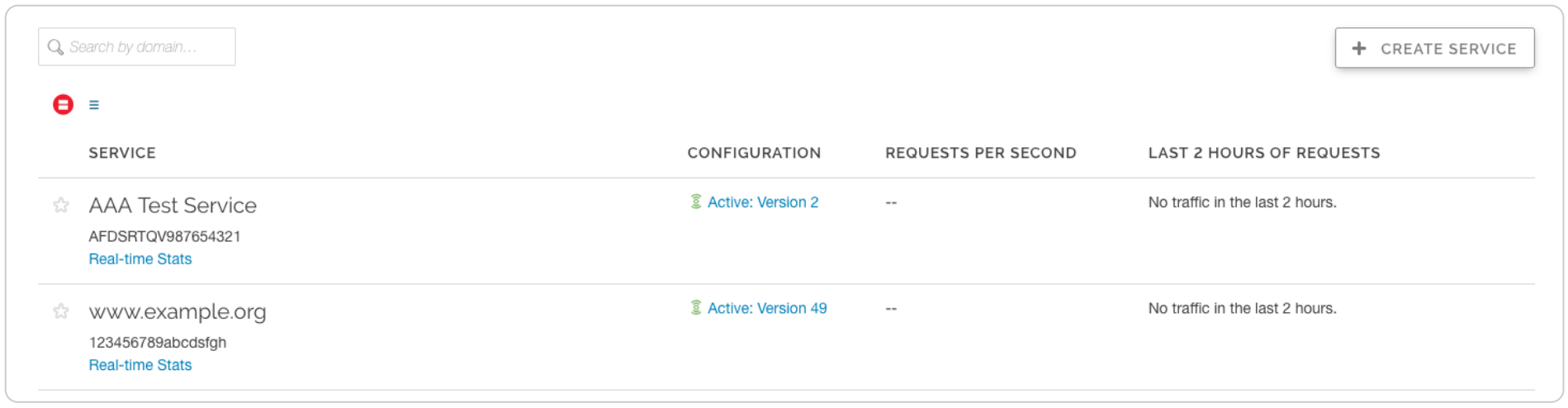
   The service version page appears, listing the version.
4. Click **Activate**. The new version of the service is activated and appears in the event log.

Other things you can do

In addition to creating or editing services, you can view all your services, view a condensed list of 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.

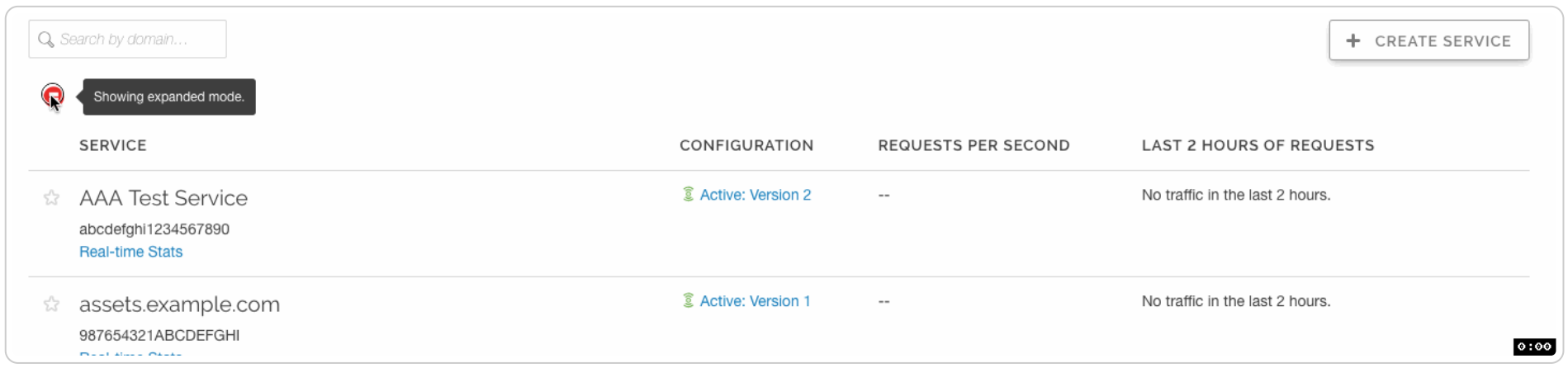
Viewing all services

To view all your services, log in to the Fastly web interface. The All services page appears displaying a summary of all your services, sorted by requests per second.



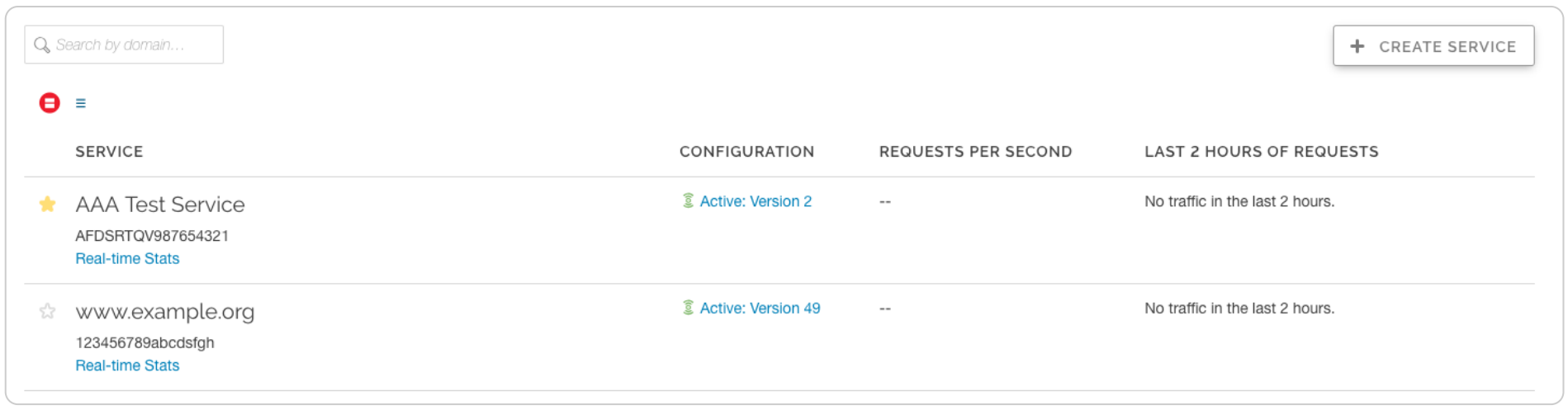
Viewing a condensed list of all services

If you have a lot of services, you can view a condensed list of all your services. On the All services page, click the icon with three lines above the list of services.



Star services

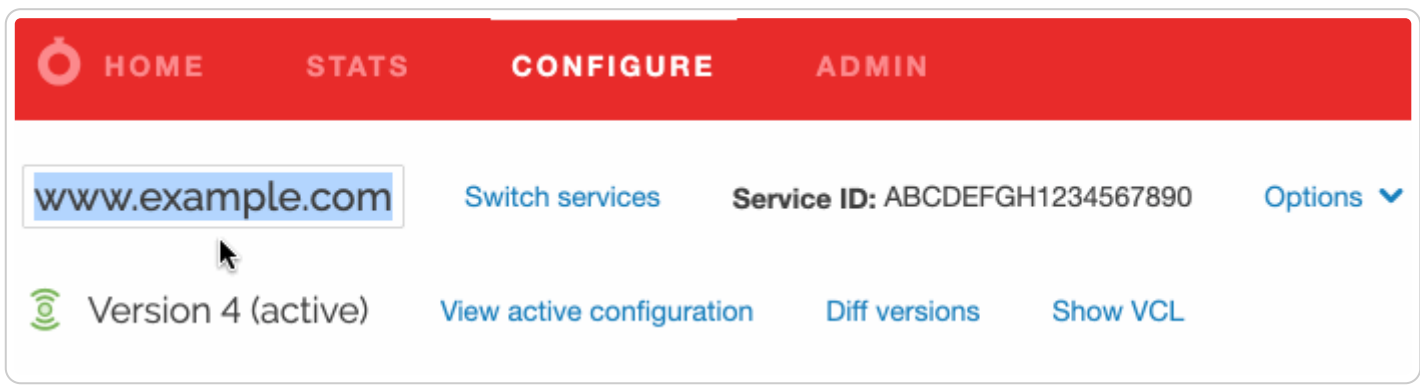
If you have a lot of services, you can star the services you use most often to pin them to the top of the All services page. Click the star next to a service to pin it to the top of the page.



Renaming services

To rename your service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Select the service name text box and type a new service name.



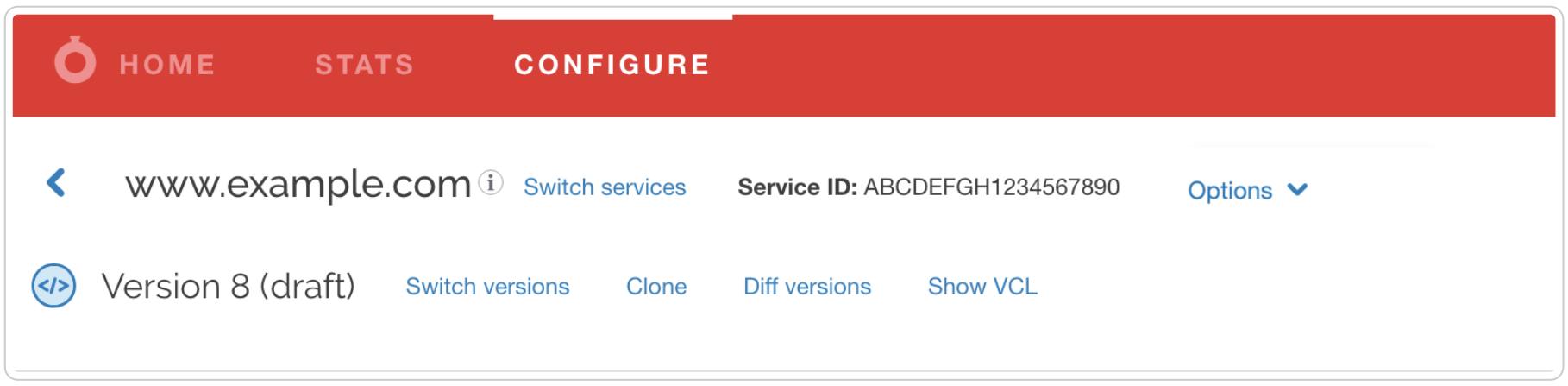
4. Press **enter**. The newly renamed service name appears.

Comparing different service versions

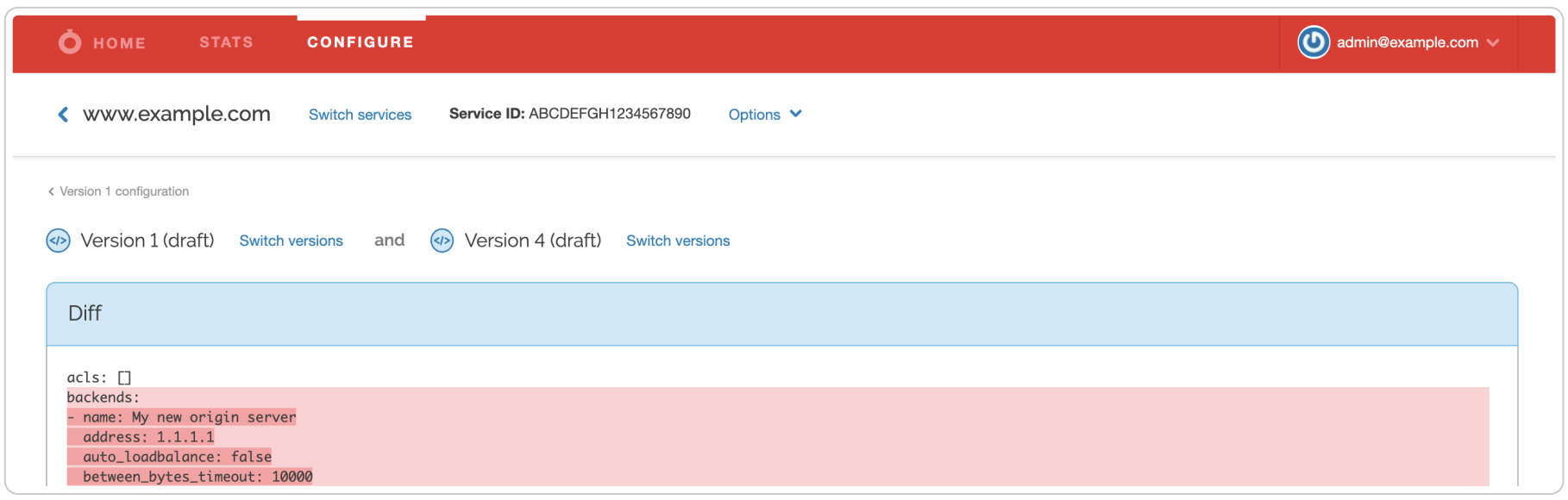
To compare two versions of a service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.

2. From the service menu, select the appropriate service.
3. Click the **Diff versions** link located under the Service ID.



The Diff versions page appears. Removals are highlighted in red with a minus sign at the beginning of the line. Additions are highlighted in green with a plus sign at the beginning of the line.



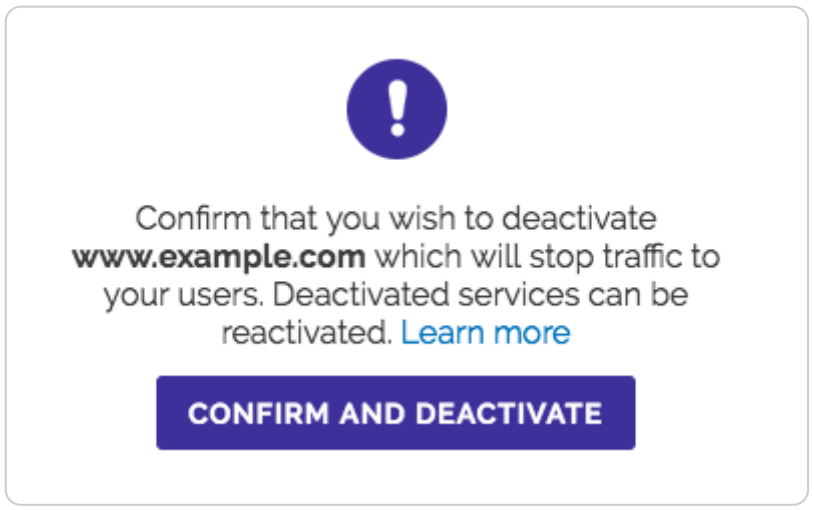
You can change the compared service versions by selecting a different version number in the selection menus.

Deactivating a service

To deactivate a service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
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.

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:

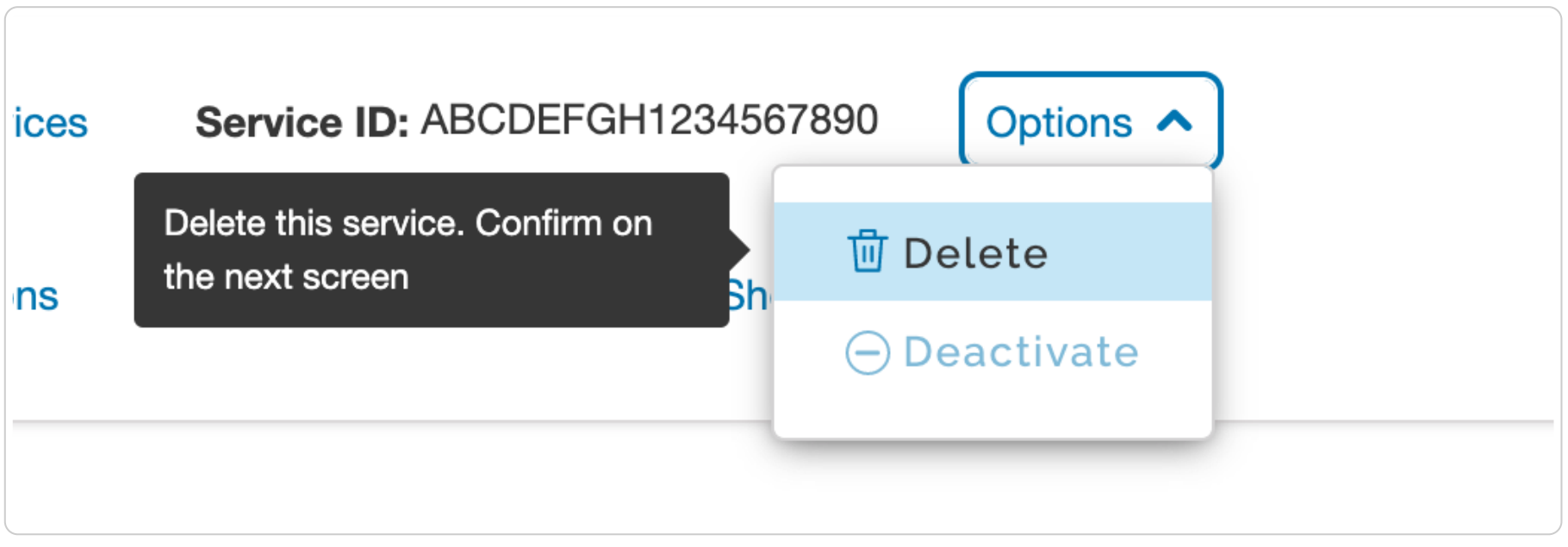
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click **Activate**. The service is reactivated.
5. If you removed the DNS CNAME records for the service's domains when you deactivated the service, you should add new DNS CNAME records now.

Deleting a service

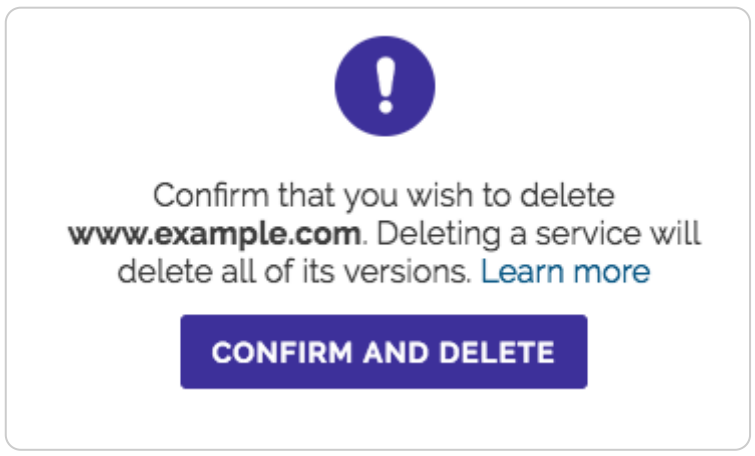
Fastly allows you to delete any service you create, along with all of its versions. Fastly does not offer a way to delete specific versions of a service, however. Service versions are meant to be an historic log of the changes that were made to a service. To undo changes introduced by a particular service 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
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.



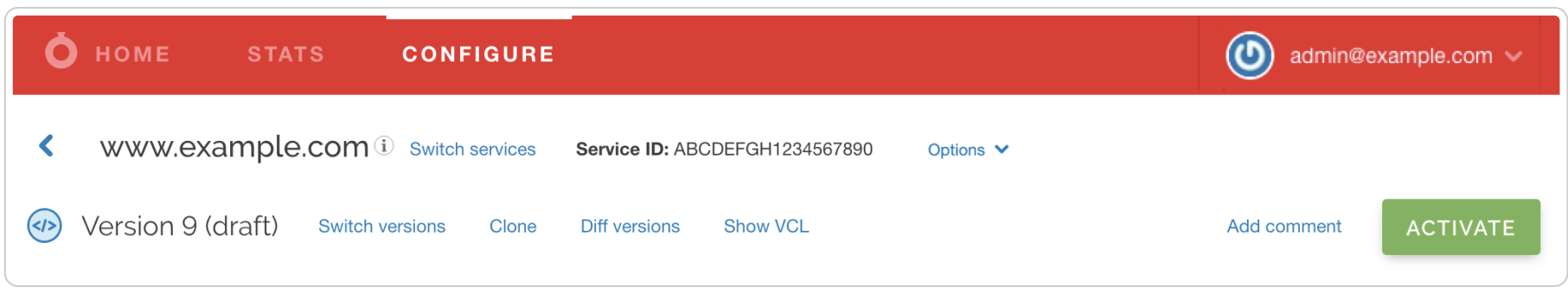
6. Click the **Confirm and delete** button to confirm that you want to delete the service.

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.

Adding comments to service versions

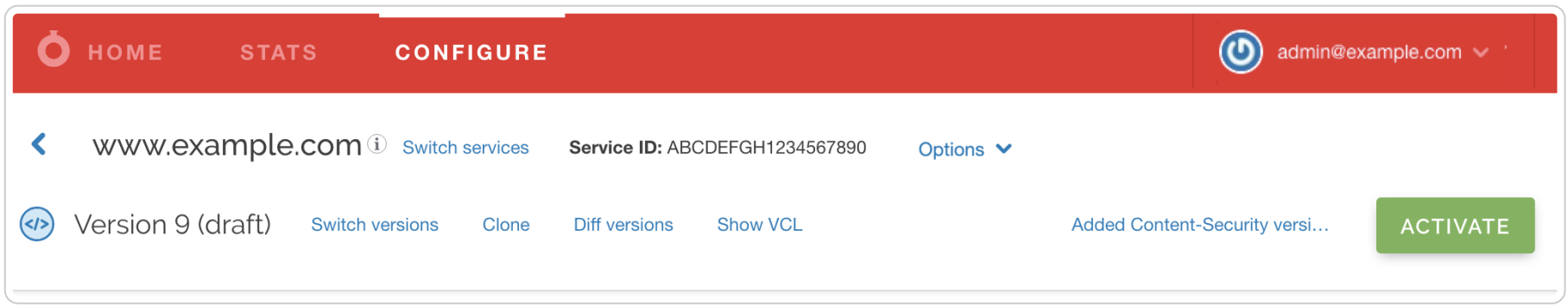
Service versions can include comments to label them or identify work in them. You can add and update version comments on both locked and activated service versions.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Add comment** link on the upper, right side of the web interface.



The comment window appears.

5. In the **Comment** field, type 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.

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.

Domains & Origins

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/getting-started#_domains-and-origins

Adding CNAME records

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.

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 either a Shared TLS Certificate or Shared TLS Wildcard Certificate service, use one of the following HTTP/1.x and HTTP/2 enabled hostnames:

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

When you purchase one of these certificate services, Fastly Support will add your domains to a specific TLS Certificate, usually differentiated by a certificate letter (e.g., `a`, `a2`, `b`, `c`). You'll need to add the appropriate certificate letter to the beginning of the Fastly hostname noted above for use in your CNAME record. For example, if your domain was added to our `a` certificate and was being routed through Fastly's entire global network, the above hostname would become:

`a.shared.global.fastly.net.`

IMPORTANT: You must use the assigned Fastly TLS hostname provided by Fastly Support. Using the incorrect Fastly hostname will cause a TLS Certificate mismatch error for HTTPS (Port 443) traffic.

Customer-specific hostnames

If you've purchased our Customer-Provided TLS Certificate Hosting Service option, we'll assign you to a specific domain map that uses the following format:

`[name].map.fastly.net.`

Free TLS wildcard Certificate

If you plan to accept both HTTP (port 80) and HTTPS (port 443) connections and you're using Fastly's free shared TLS wildcard certificate, use:

`[name].global.ssl.fastly.net.`

IMPORTANT: The free TLS hostname 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. If you want to use your own domain (`www.example.com`), see the TLS-enabled hostname section above.

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 our Testing setup before changing domains guides.
- 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.

Testing setup before changing domains

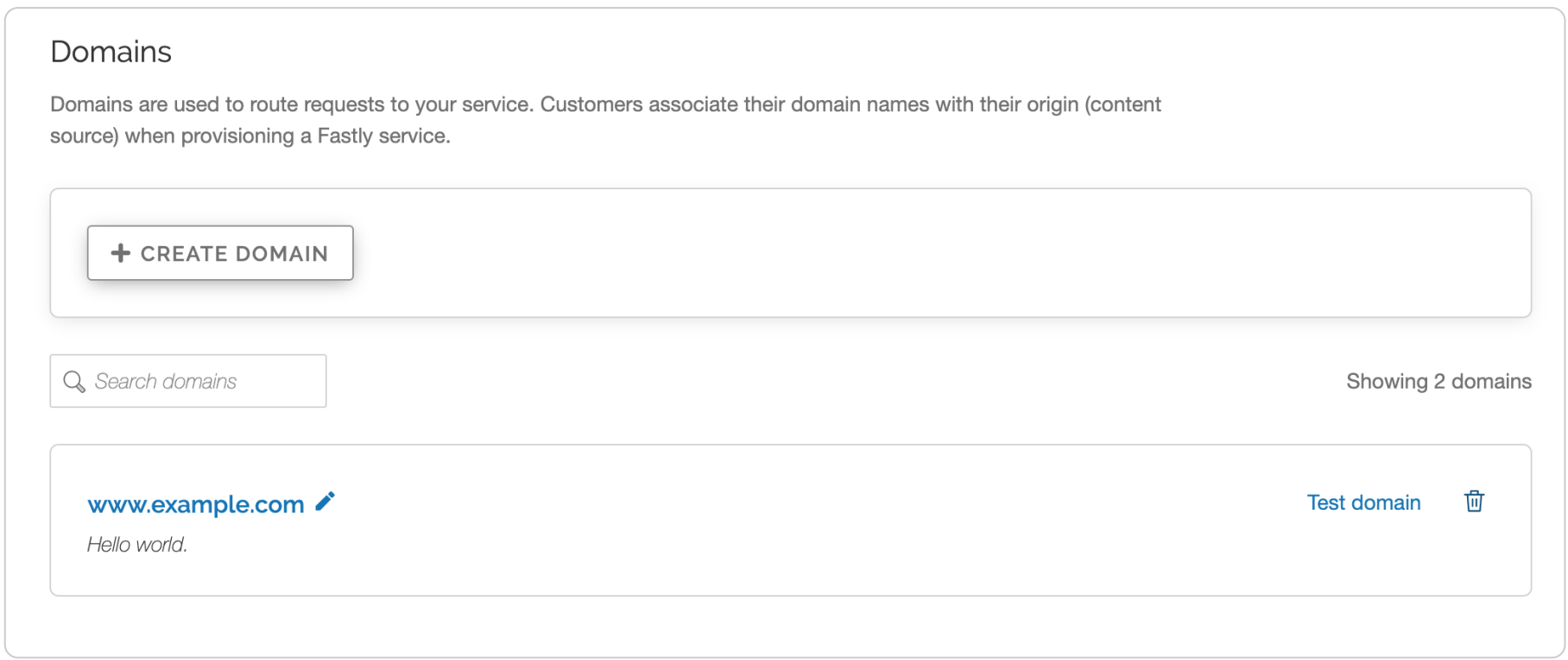
https://docs.fastly.com/en/guides/testing-setup-before-changing-domains

After you deploy your service, 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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 Mac OSX 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.

Working with domains

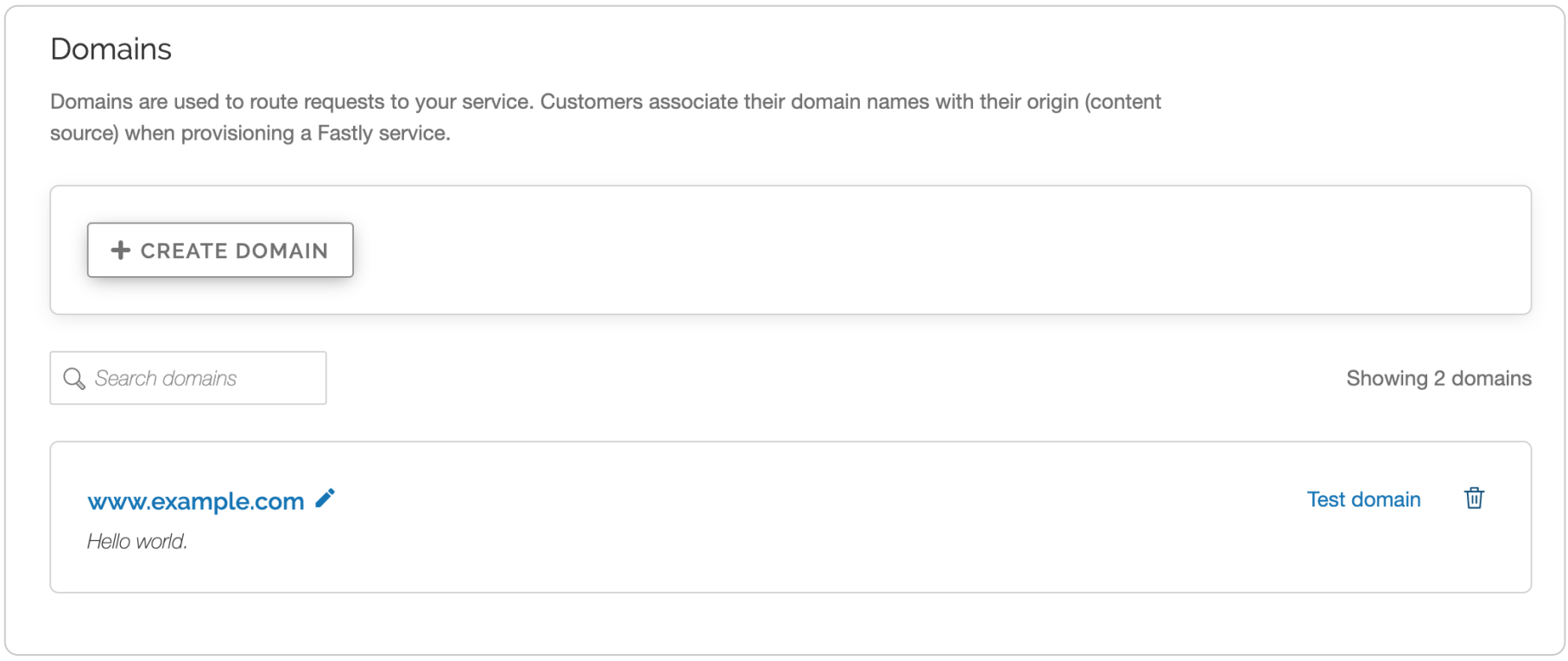
https://docs.fastly.com/en/guides/working-with-domains

Domains are used to route requests to your service. You associate your domain names with your origin when provisioning a Fastly service, and you can add, edit, or remove domains from your service at any time.

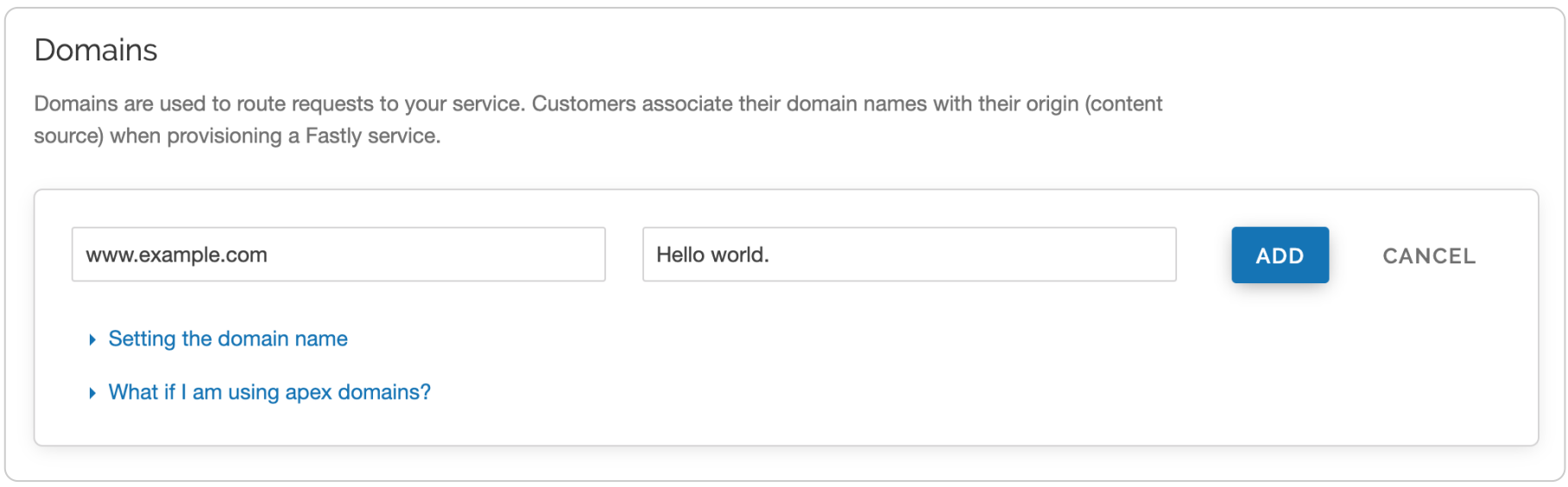
Creating a domain

Follow the steps below to add a domain to your service:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type your domain name. The domain name is used to properly route requests to your website, and ensures 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`.

     ★ **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`.

   - In the **Comment** field, optionally type a comment that describes the domain.
6. Click the **Add** button. Your new domain appears in the list of domains.
7. 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.
8. Click the **Activate** button to deploy your configuration changes.

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>/domai
n -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

You can test a domain to see how your service is being pulled through our network. This can help you identify DNS issues and problems with your Fastly configuration. See our testing guide for instructions.

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.

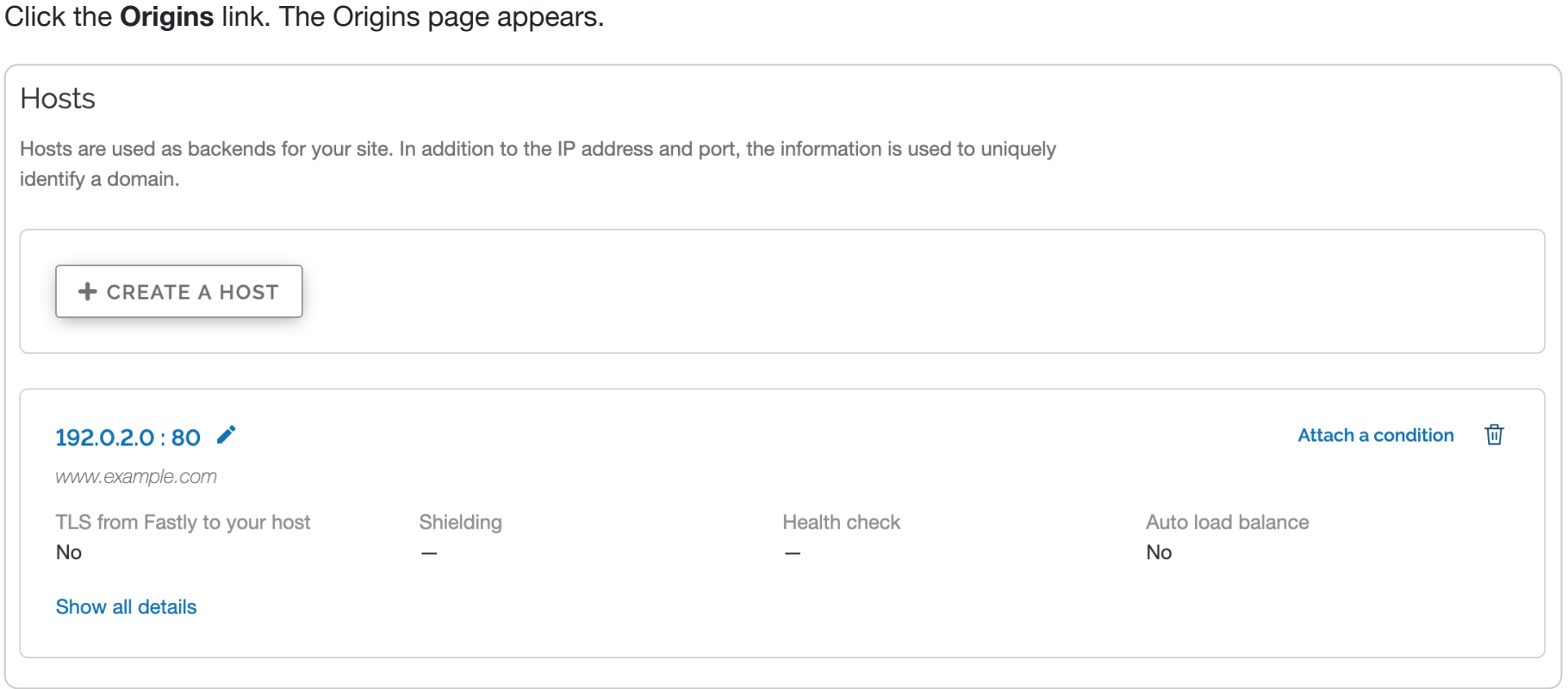
Working with health checks

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

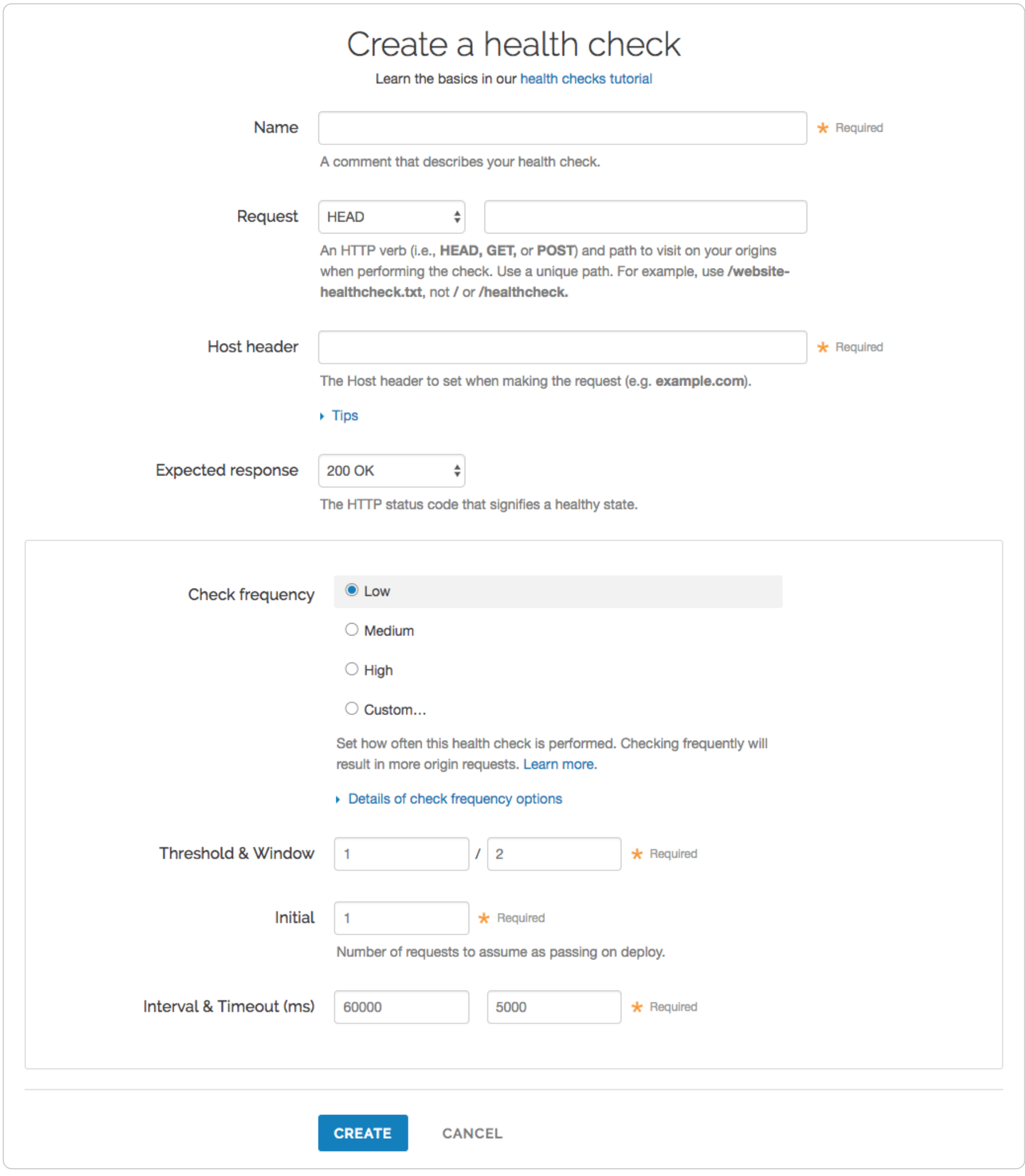
Health checks monitor the status of your hosts. Fastly performs health checks on your origin server based on the Check frequency setting you select in the Create a new health check page. The Check frequency setting you select specifies approximately how many requests per minute Fastly POPs are checked 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."

Creating a health check

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type 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, type 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 datacenter, where "healthy" means 1 out of 2 must pass.
     - **Medium:** One request every 15 seconds from each datacenter, where "healthy" means 3 out of 5 must pass.
     - **High:** One request every 2 seconds from each datacenter, where "healthy" means 7 out of 10 must pass.
     - **Custom:** A custom frequency you specify.
   - In the **Threshold & Window** fields, type 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, type 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, type 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.

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.

Performance

These articles describe how to adjust the performance of Fastly's services beyond standard configuration methods.

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

Shielding

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

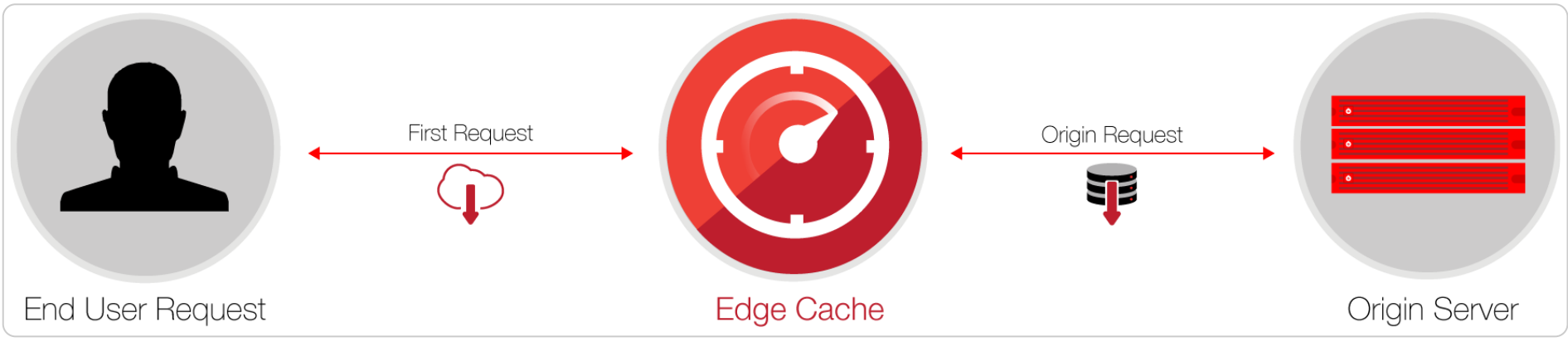
Fastly's shielding service feature allows you to designate a specific Point of Presence (POP) as a shield node to your origins. Once enabled, all requests to your origin will go through the datacenter you designate, increasing the chances of getting a HIT for a given resource. If a different POP doesn't have a specific object, it will query the shield (provided it's not down for maintenance) instead of your origins.

How shielding works

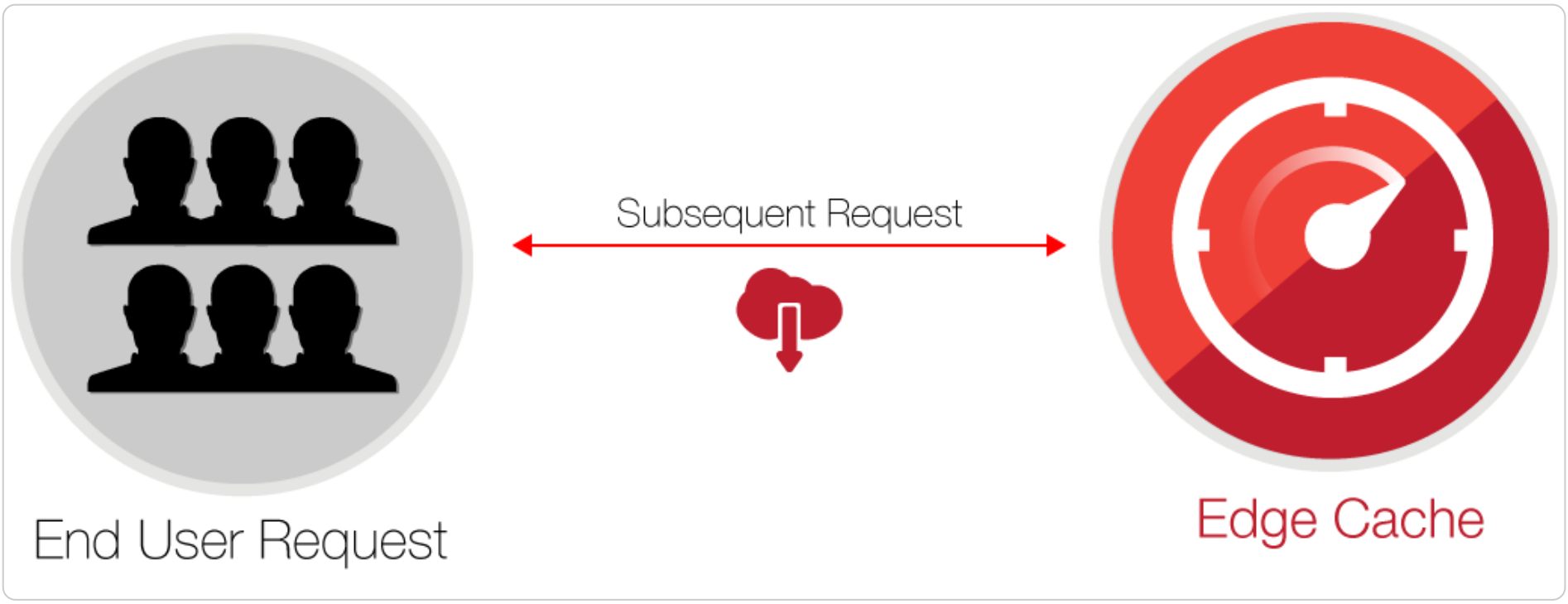
When a user requests brand new content from a customer's server and that content has never been cached by any Fastly POP, this is what happens to their request when shielding is and is not enabled.

Without shielding enabled

Without shielding enabled, when the first request for content arrives at POP A, the POP does not have the content cached. It passes the request along to a customer's origin server to get the content. Once the content is retrieved, POP A caches it and sends it on to the user.



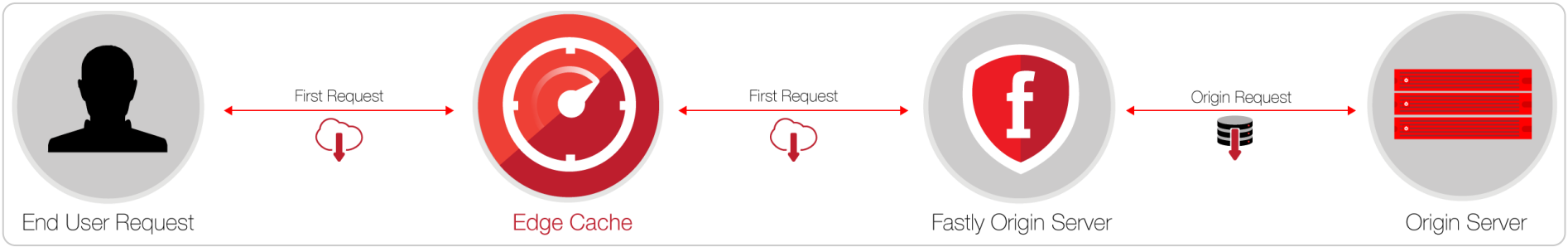
When a second request for that same content arrives at POP A, the content is already cached. No request goes to the customer's origin server. It's merely sent from the cached copy.



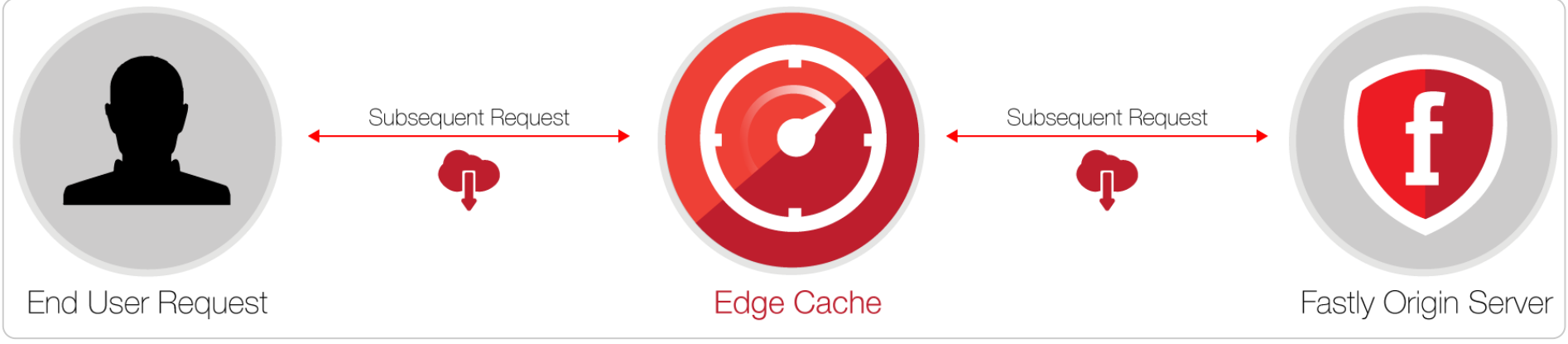
If the second request were to arrive at POP B instead of POP A, however, the request would once again be passed along to the customer's origin server. It would then be cached and passed back to the end user, just like POP A did when that info was first requested.

With shielding enabled

With shielding enabled, when the first request for content arrives at the POP A, that POP does not have the content cached. It passes the request along to the shield POP, which also doesn't have the content cached. The shield POP passes the request along to the customer's origin server. It then caches the content that's retrieved and passes it along to POP A. POP A then passes the content along to the user.



When a second request for that same content arrives at POP A, the content is already cached, so no request goes to the shield POP or the customer's origin server.



If the second request were to arrive at POP B instead of POP A, however, the request would be passed along to the shield POP. That shield POP already has a cached copy from the first request to POP A. No future requests for the content would be passed along to the customer's origin server until the shield POP's cached copy of it expires.

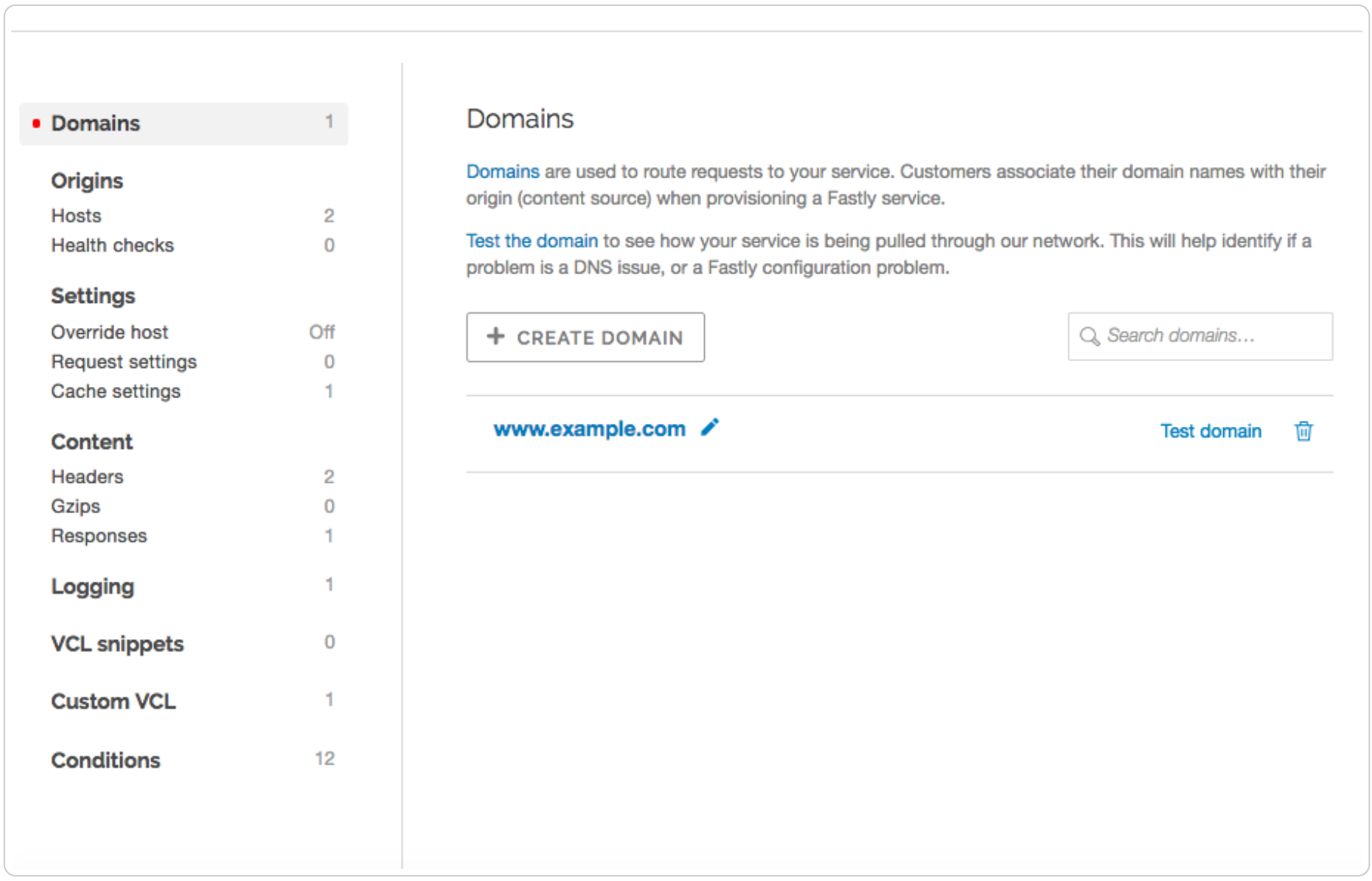
Enabling shielding

> **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 and click the **Configure** link.
3. From the service menu, select the appropriate service.
4. Click the **Edit configuration** button and then select **Clone active**. 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 datacenter to use as your shield keeping the following in mind:
   - Generally, we recommend selecting a datacenter 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).
   - 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 article Understanding Cache HIT and MISS with Shielding Services.

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:

```
1  if (req.http.fastly-ff) {
2    set client.identity = req.http.Fastly-Client-IP;
3  }
```

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. Also, 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.

When you purge an object from Service B and not from Service A, the shield will serve the old object that you wanted to purge to the edge, since the purge went out to Service B and not Service A. You will want to purge the object from both Service A and Service B. However, this opens the door for confusion and error.

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:

```
1  Surrogate-Control: max-age=3600
2  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");
  }
}
```

Configuration

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

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

Basics

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

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

Conditionally changing a URL

https://docs.fastly.com/en/guides/conditionally-changing-a-url

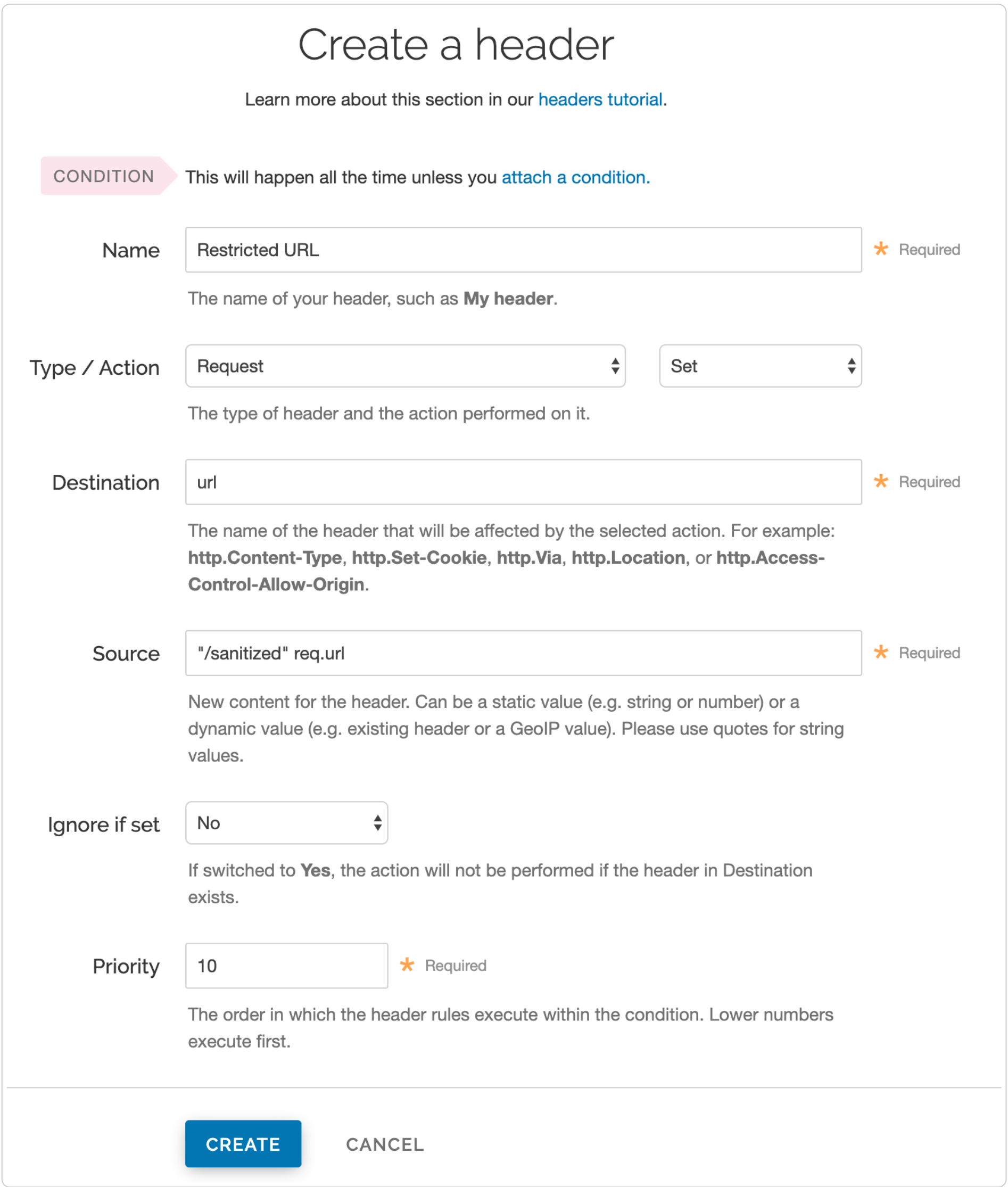
To conditionally change a URL based on the domain, include VCL that looks something like this:

```
1  if (req.http.host ~ "^restricted") {
2    set req.url = "/sanitized" req.url;
3  }
```

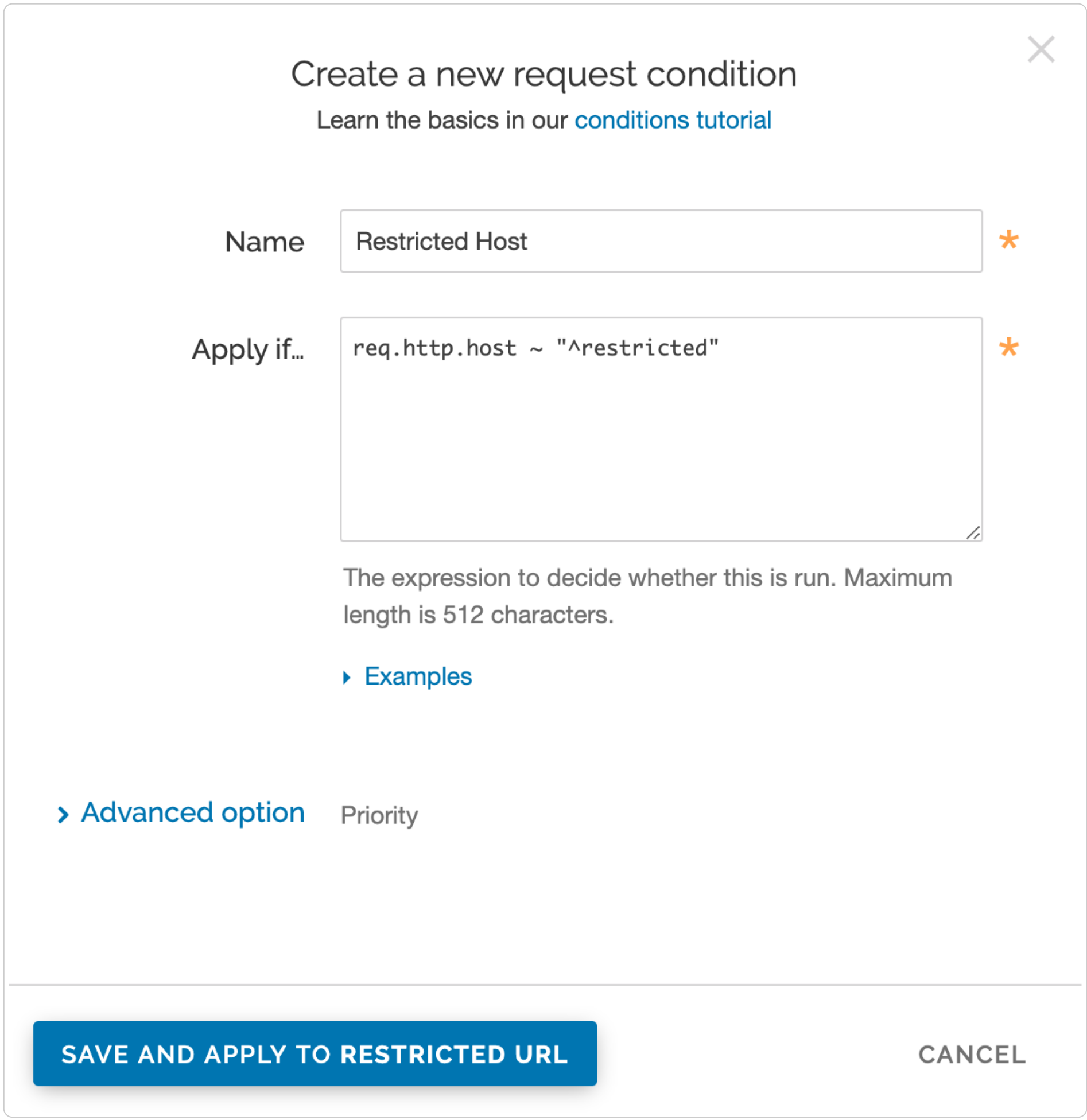
If you have shielding enabled, however, add the following code instead to avoid rewriting the URL twice:

```
1  if (req.http.host ~ "^restricted" && req.url !~ "^/sanitized") {
2    set req.url = "/sanitized" req.url;
3  }
```

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:



Conditions

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

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

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.*`
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

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 type 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

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.

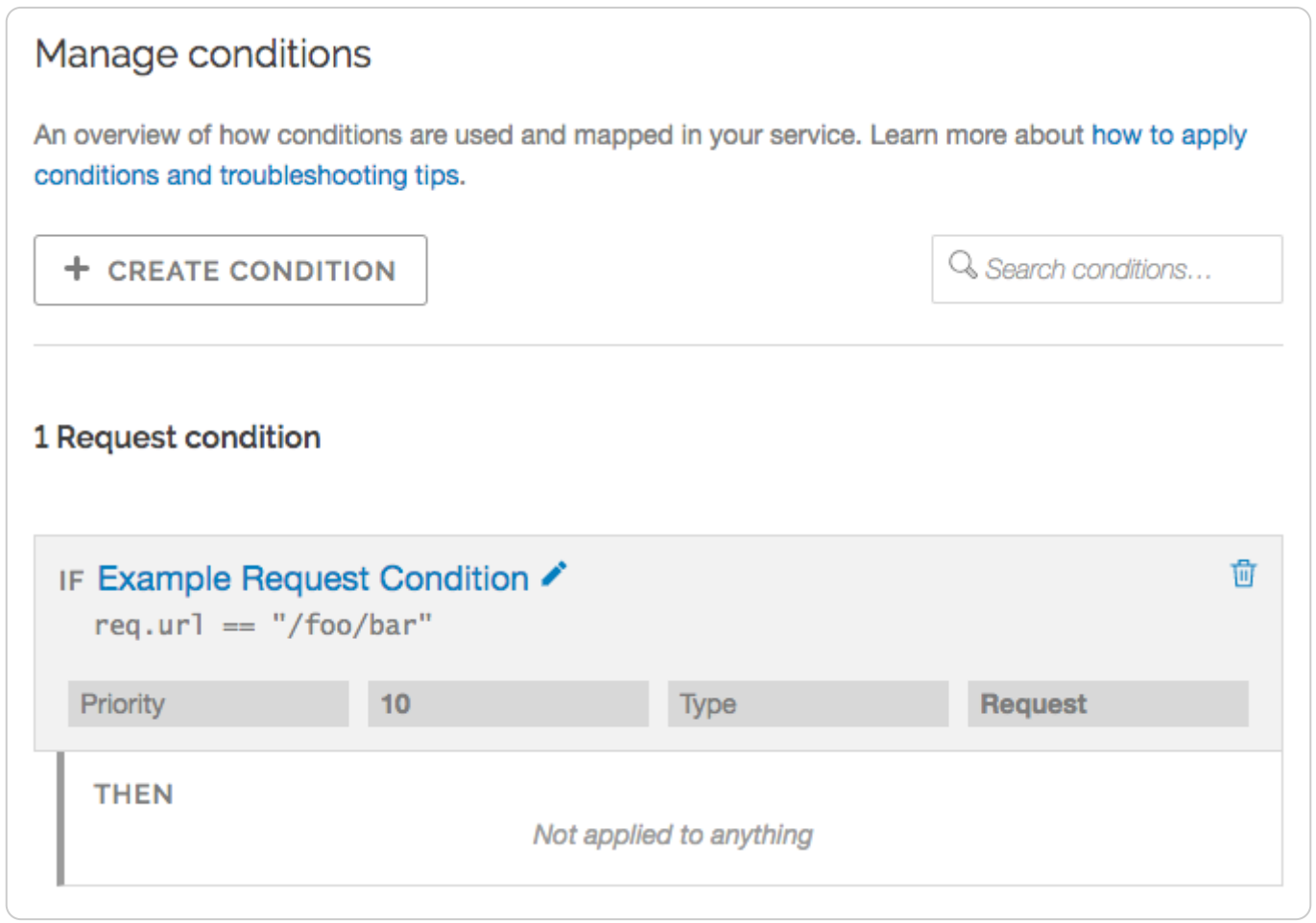
Conditions on the Manage conditions page

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.

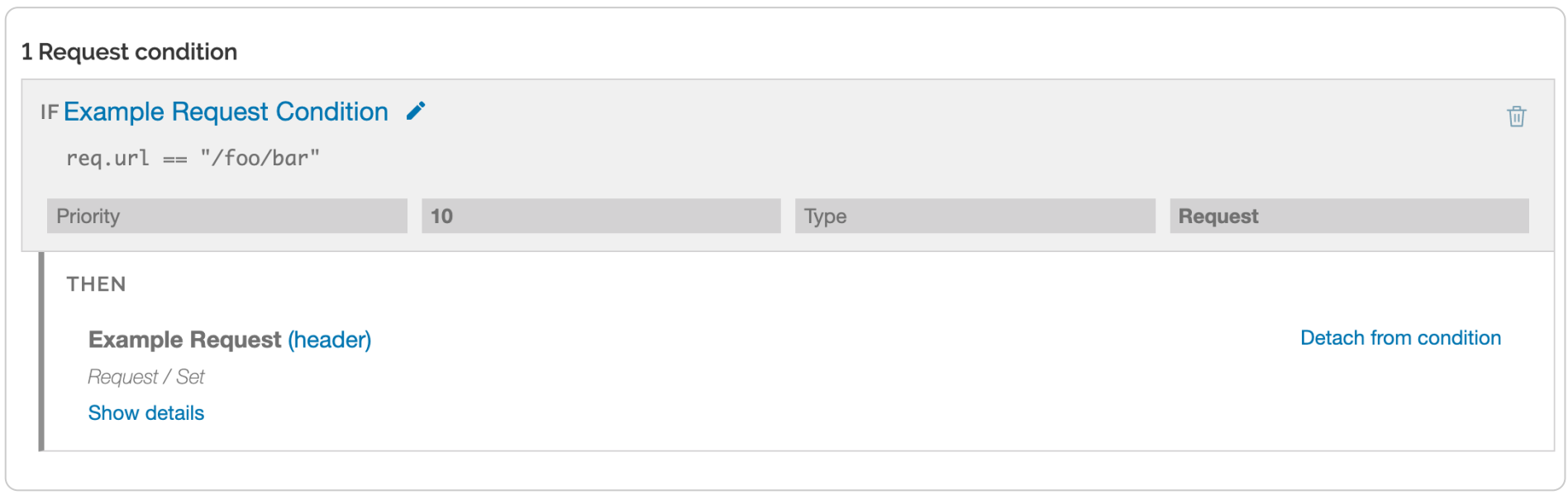
To view conditions on the Manage conditions page:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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:

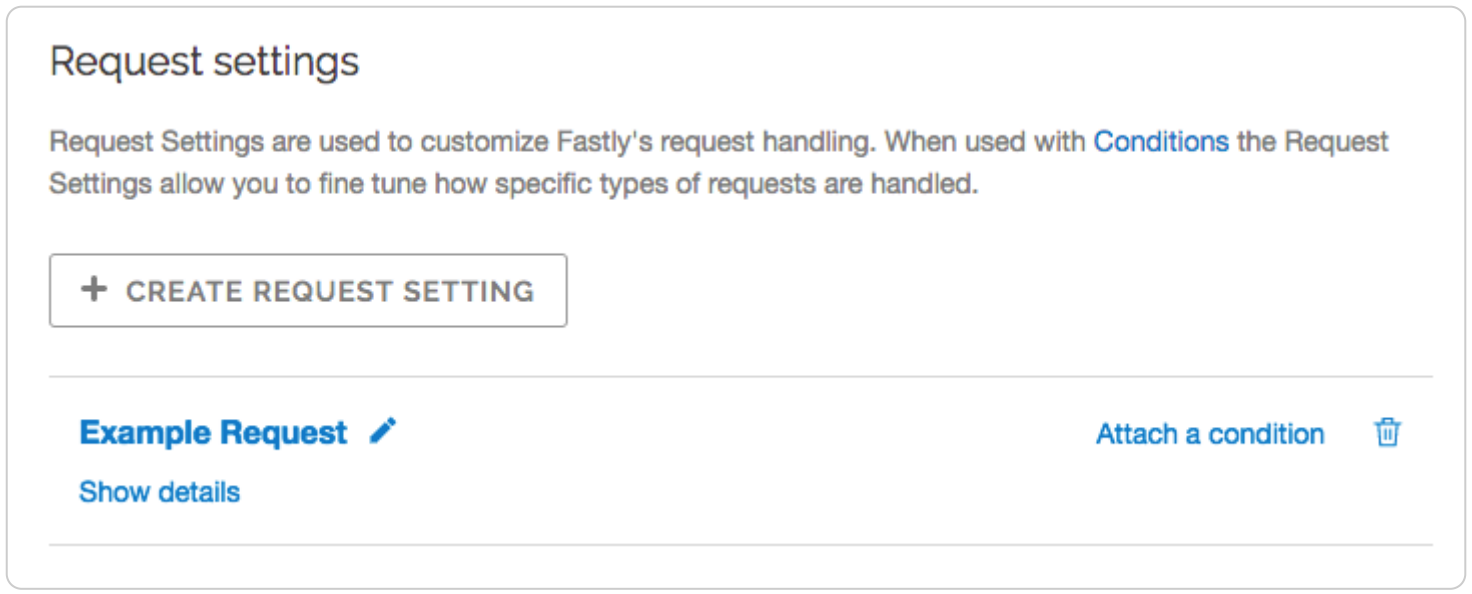


The Example Request Condition shown above currently isn't applied to a configuration object (as indicated by "Not applied to anything"). If it was, it would instead appear similar to this:

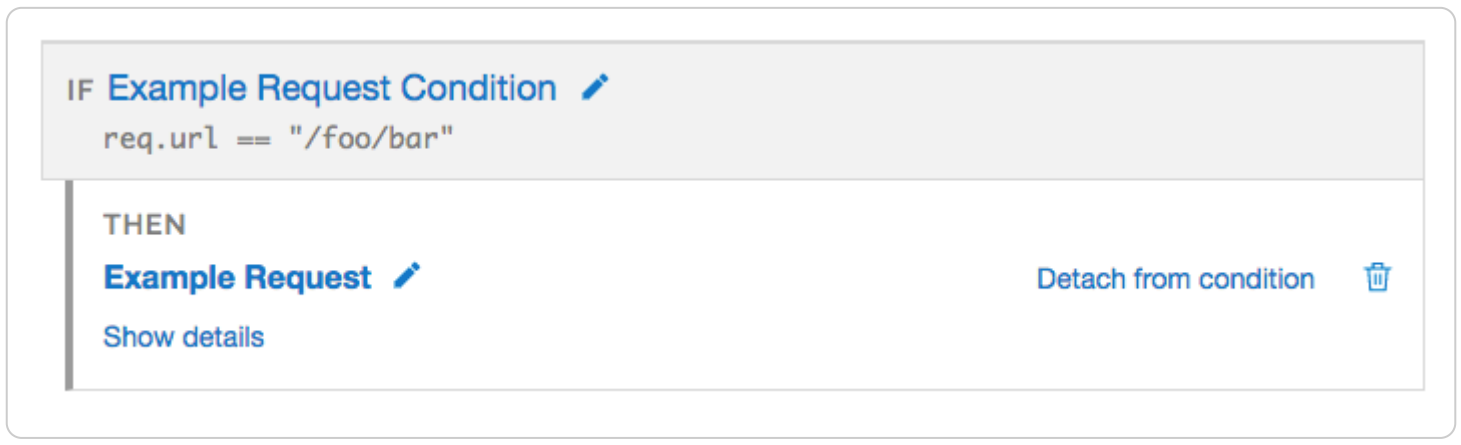


Conditions attached directly to a configuration object

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

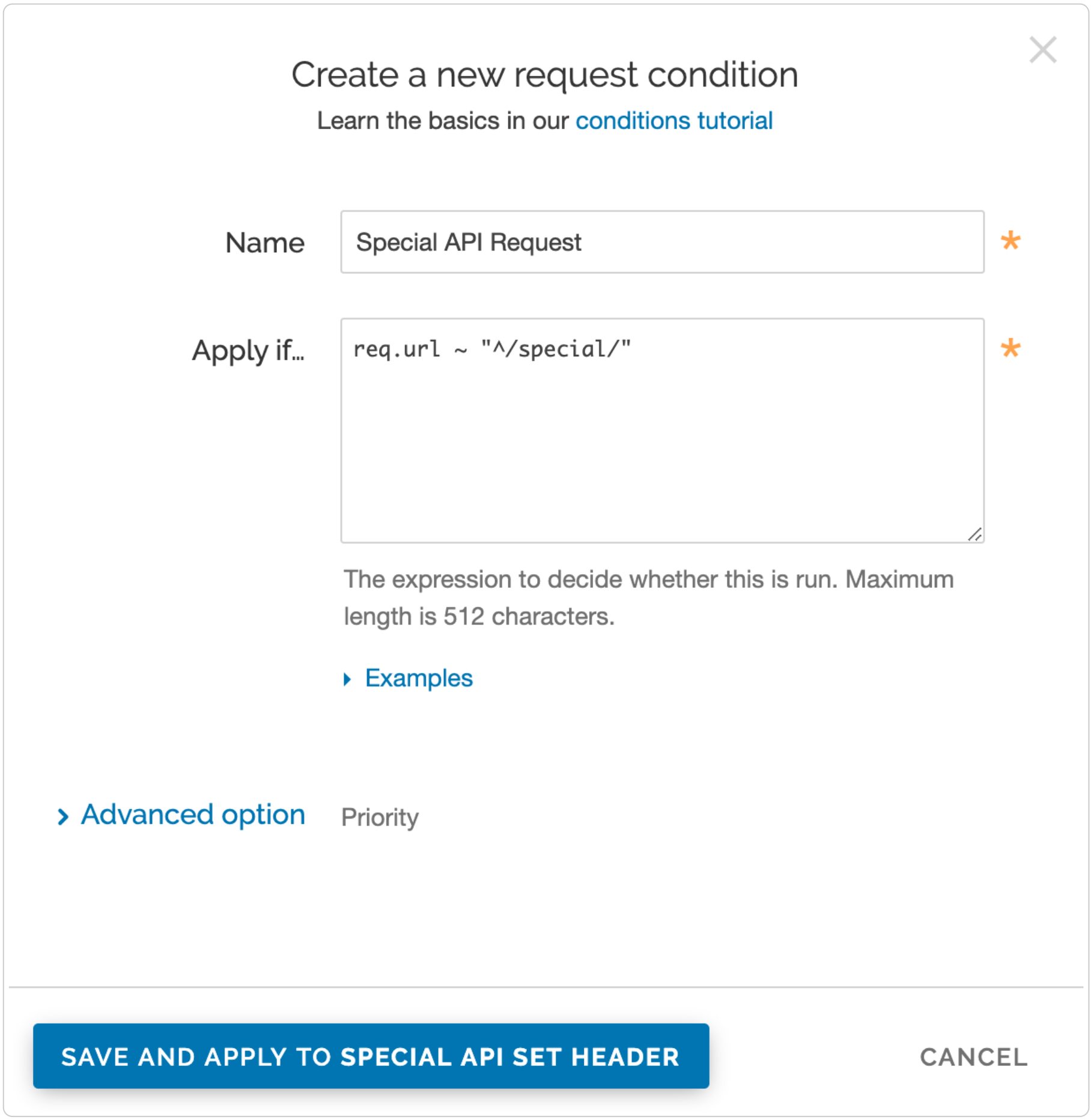
To create a new origin server that will handle the special API traffic, follow the instructions for connecting to origins. You'll add specific details about your API server when you fill out the **Create a host** fields:

- In the **Name** field, type a name for your API server (for example, `Special API Server`).
- In the **Address** field, type 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.

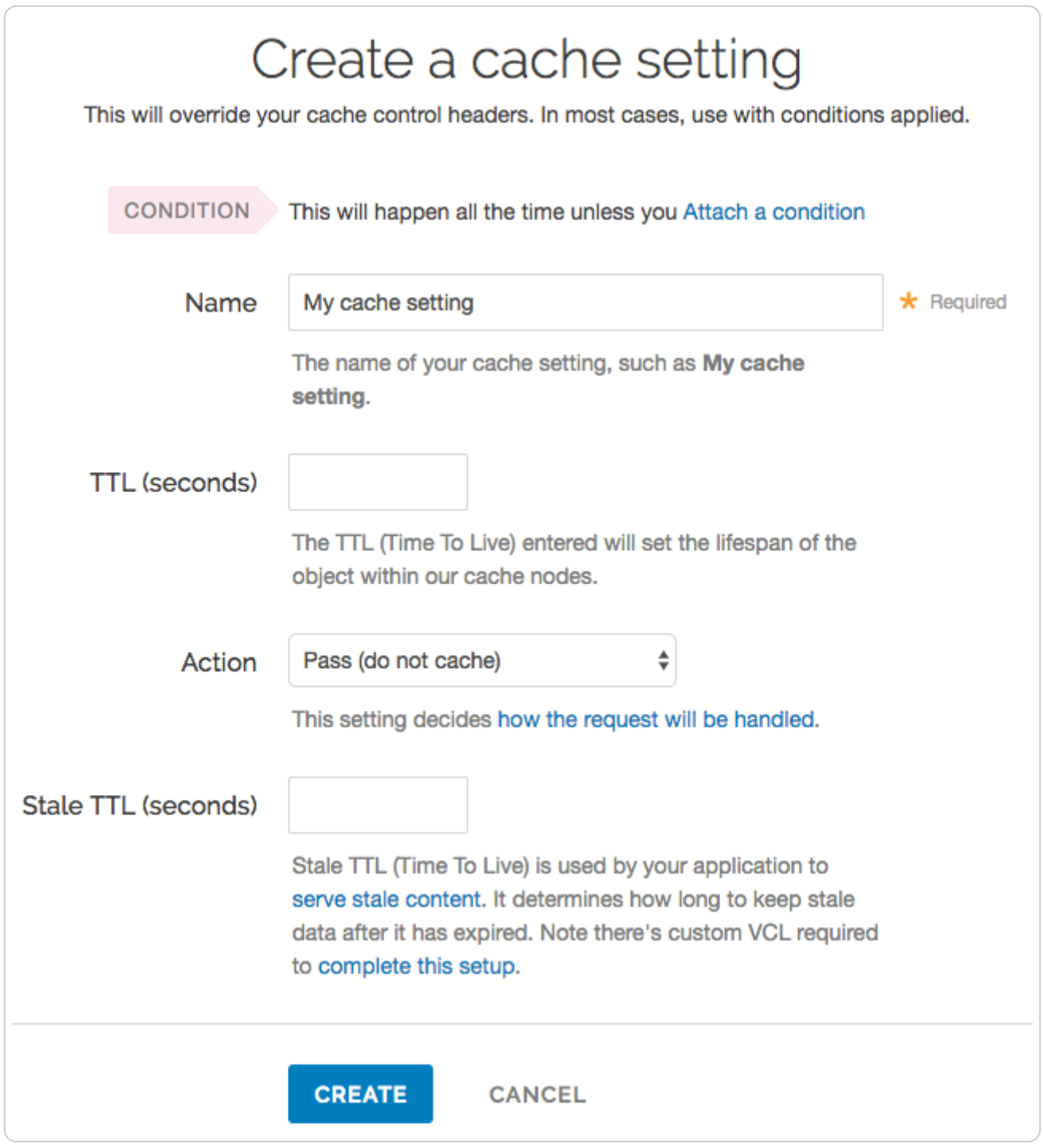


3. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example `Special API Request`).
   - In the **Apply if** field, type the appropriate request condition that will be applied (for example, `req.url ~ "^/special/"` 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.

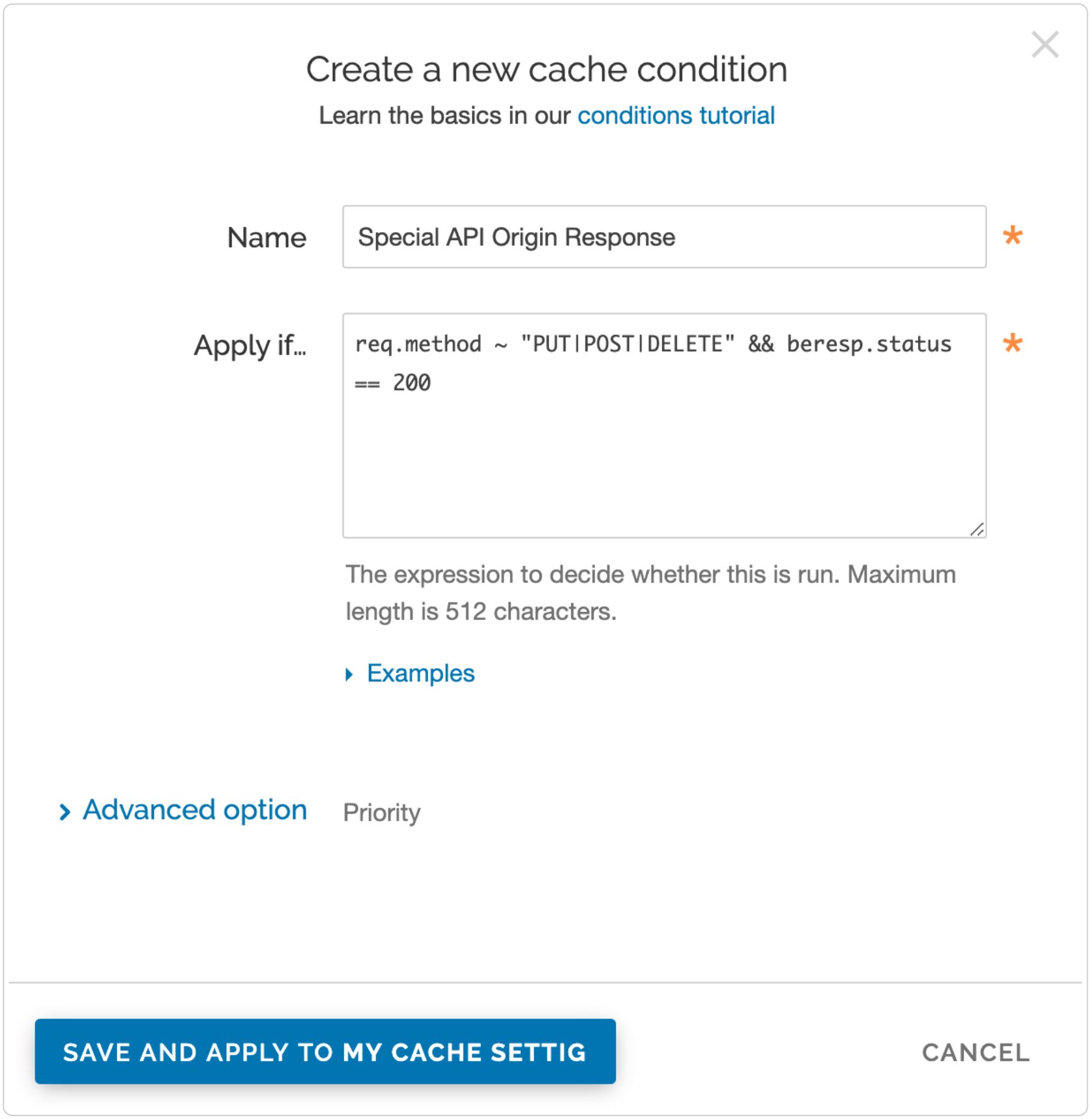


3. Fill out the **Create a cache setting** fields as follow:
    - In the **Name** field, type a descriptive name for the new cache settings.
    - Leave the **TTL (seconds)** field set to its default value.
    - From the **Action** menu, select **Pass (do not cache)**.
    - Leave the **Stale TTL (seconds)**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

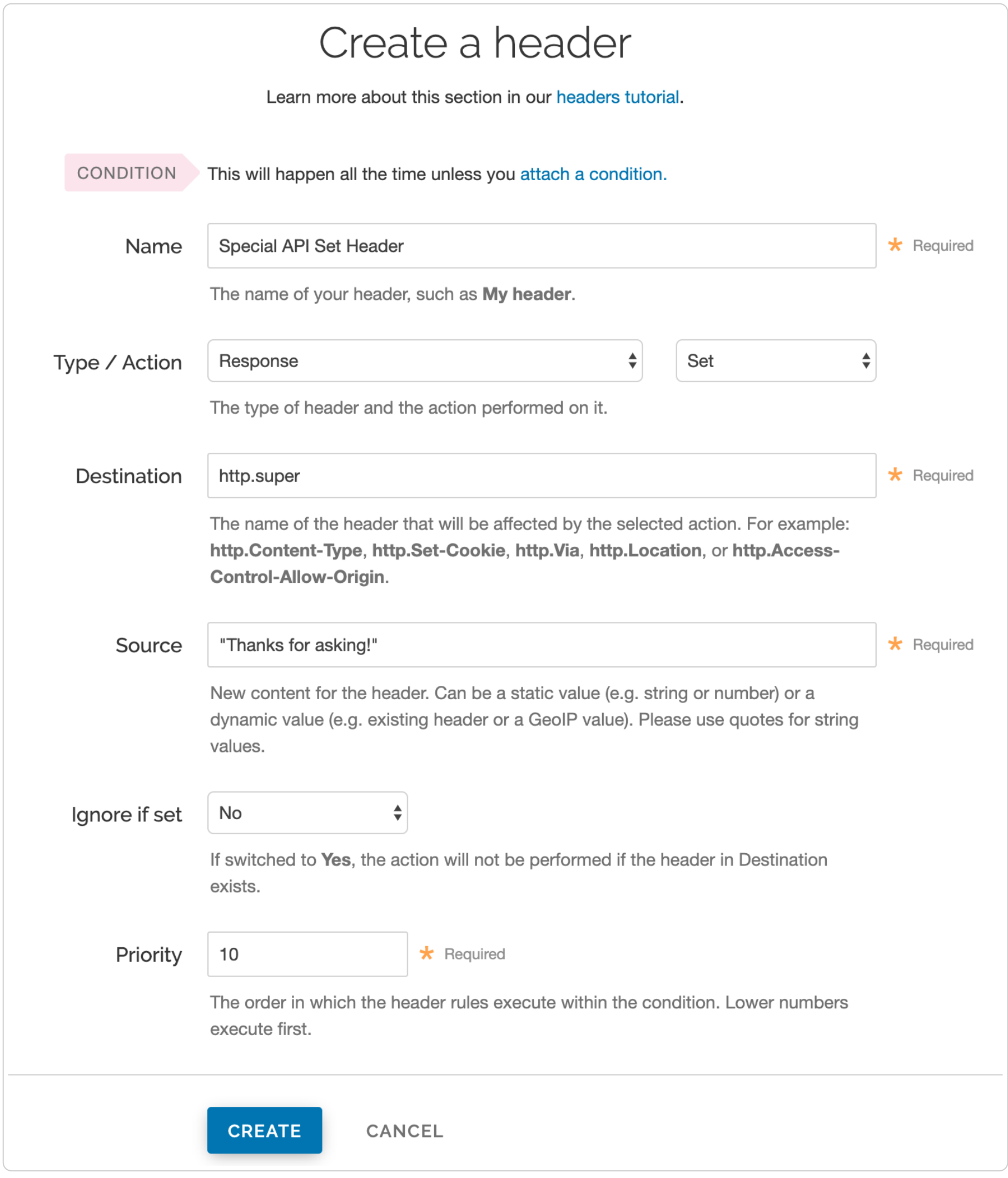


3. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `Special API Origin Response`).
   - In the **Apply if** field, type the appropriate request condition that will be applied (for example, `req.method ~ "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.

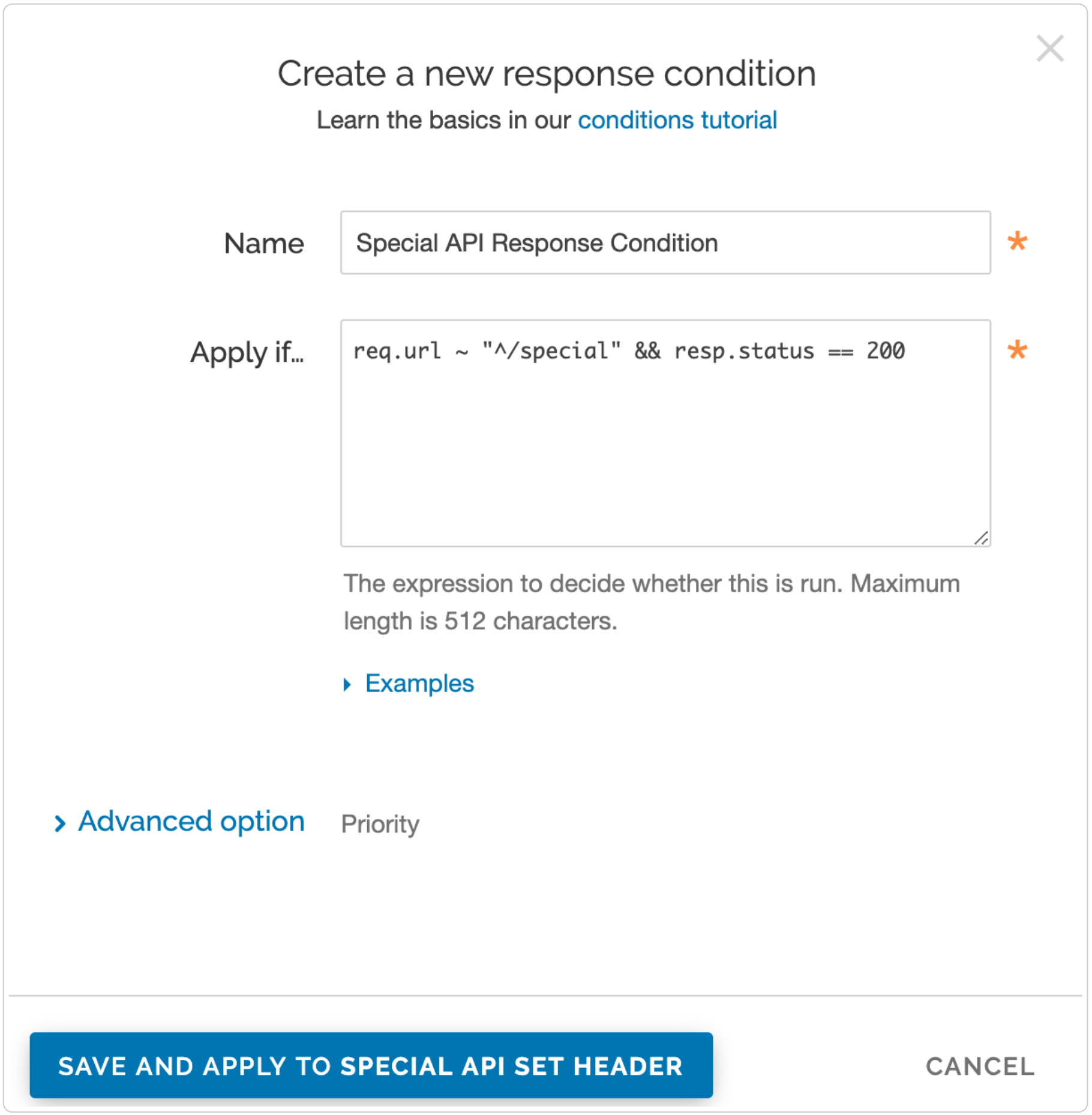


3. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a descriptive name for the new header (for example, `Special API Set Header`).
   - From the **Type** menu, select **Response** and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header that will be affected by the action (for example, `http.super`).
   - In the **Source** field, type 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, type a descriptive name for the new condition (for example, `Special API Response Condition`).
   - In the **Apply if** field, type 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).

Dictionaries

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

https://docs.fastly.com/en/guides/about-edge-dictionaries

Fastly offers updatable, global, Edge Dictionaries that allow you to store data as key-value pairs and 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",
  ...
}
```

for use with a service.

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 our guide on 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 user 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 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.

Private Edge Dictionaries

https://docs.fastly.com/en/guides/private-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
 }
```

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 API

https://docs.fastly.com/en/guides/working-with-dictionaries-using-the-api

Edge Dictionaries allow you to create logic that doesn't need to be attached to a configuration service version. Edge Dictionaries are made up of dictionary containers and dictionary items. Attaching dictionary containers to a service version allows you to turn frequently repeated statements into single function statements that act as a constant.

Create an empty dictionary container within a service

To use a dictionary container, start by creating an empty one within an unlocked version of a service.

Before an Edge Dictionary can be manipulated, its 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 dictionary's existence.

For example, if you were creating a referer blocklist via the API, you would make an API call by running this command:

```
curl -X POST -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'name=referer_blocklist' 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": "referer_blocklist",
  "service_id": "<service_id>",
  "updated_at": "2017-05-03T16:11:41+00:00",
  "version": <version_number>,
  "write_only": false
}
```

Activate the service associated with the dictionary container

For an Edge Dictionary to appear in generated VCL so it can be referred to later, the version associated with the dictionary container must be activated.

In our referer blocklist example, you would make this API call to activate service version associated with the empty dictionary container you created:

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/service/<service_id>/version/<version_number>/activate
```

The response would be this:

```
{
  "number": <version_number>,
  "active": true,
  "service_id": "<service_id>"
}
```

Add dictionary items

Once the dictionary container becomes associated with the configuration of a service, you can begin populating it with dictionary items. Our guide to working with dictionary items provides more detail on manipulating dictionary items. In the example shown here, you would use the following API call for each URL you want to add to your referer blocklist:

```
curl -X POST -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'item_key=example-referer.org&item_value=true' "https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/item"
```

The response for each URL added would look similar to this:

```
{
  "dictionary_id": "<dictionary_id>",
  "service_id": "<service_id>",
  "item_key": "example-referer.org",
  "item_value": "true"
}
```

Once the blocklisted URLs are added as items in your dictionary container, you can find them in your generated VCL by looking for a table similar to this:

```
table referer_blocklist {
  "example-referer.org": "true",
  "another-referer.net": "true",
  "sample-referer.com": "true",
}
```

Using a service to call dictionaries

When you create Edge Dictionaries via API calls, the dictionary contents aren't tied to any single version of your service.

The logic needed to *interact* with the table of information the Edge Dictionary creates, however, is always tied to a service version.

For example, adding a new referer to your blocklist requires that you specifically interact with the Edge Dictionary at some point after you create it. You could do this via API calls because its data would not require a service version activation. The dictionary was created via API calls not via custom VCL.

Specifically, you would set the host of the referer to a header by including custom VCL like this:

```
// versioned vcl
sub vcl_recv {

  # capture host of referer into a header
  set req.http.Referer-Host = regsub(req.http.Referer, "^https?://?([^:/\s]+).*$", "\1");

  # check if referrer host is in blocklisted table
  if (table.lookup(referer_blocklist, req.http.Referer-Host)) {
    # ResponseObject: forbidden-referrer
    error 900 "Fastly Internal";
  }
  #end condition
}

sub vcl_error {

  if (obj.status == 900) {
    set obj.http.Content-Type = "";
    synthetic {""};
    return(deliver);
  }
}
```

Custom VCL examples

These examples illustrate how to use Edge Dictionaries in custom VCL. The dictionaries are created via API calls and are displayed in the tables.

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

Example: Referer blocklist

This example returns a 403 error message if the referer is in the dictionary.

```
// dictionary items can be added, updated, removed via the API
// does not require cloning and activating versions
table bad_actors {
  "example.com" : "nope",
  "fastly.com" : "nope",
}
// versioned vcl
sub vcl_recv {
  set req.http.Referer-Host = regsub(req.http.Referer, "https?://([^/]+)/.*", "\1");
  set req.http.Referer-Check = table.lookup(bad_actors, req.http.Referer-Host, "yes");
  if (req.http.Referer-Check == "nope") {
    error 403;
  }
}
```

Example: CORS origin database

This example adds the origins in the dictionary to the `Access-Control-Allow-Origin` header.

```
// dictionary items can be added, updated, removed via the API
// does not require cloning and activating versions
table acceptable_origins {
  "http://example.com" : "yes",
  "http://fastly.com" : "yes",
}
// versioned vcl
sub vcl_deliver {
  set req.http.CORS = table.lookup(acceptable_origins, req.http.Origin, "nope");
  if (req.http.CORS == "yes") {
    set resp.http.Access-Control-Allow-Origin = req.http.Origin;
  }
}
```

Example: TTL database

This example sets the TTLs for the URLs in the dictionary.

```
// dictionary items can be added, updated, removed via the API
// does not require cloning and activating versions
table ttls {
  "/" : "60",
  "/public" : "86400",
  "/api" : "3600",
  "/foo" : "7200",
  "/user" : "5"
}
// versioned vcl
sub vcl_fetch {
  /* cut URL down to first directory, or just / */
  if (req.url.path ~ "^(/[^/\?]*)") {
    /* should always be true */
    set beresp.ttl = std.atoi(table.lookup(ttls, re.group.1, "30"));
  }
```

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

https://docs.fastly.com/en/guides/working-with-dictionaries-using-the-web-interface

Edge Dictionaries allow you to create logic that doesn't need to be attached to a configuration service version. Edge Dictionaries are made up of dictionary containers and dictionary items. You can use dictionary items to create and store key-value pairs. Attaching dictionary containers to a service version allows you to turn frequently repeated statements into single function statements that act as a constant.

Viewing dictionaries

Depending on how you created your dictionaries, you can view them either under VCL Snippets or the Data link.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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

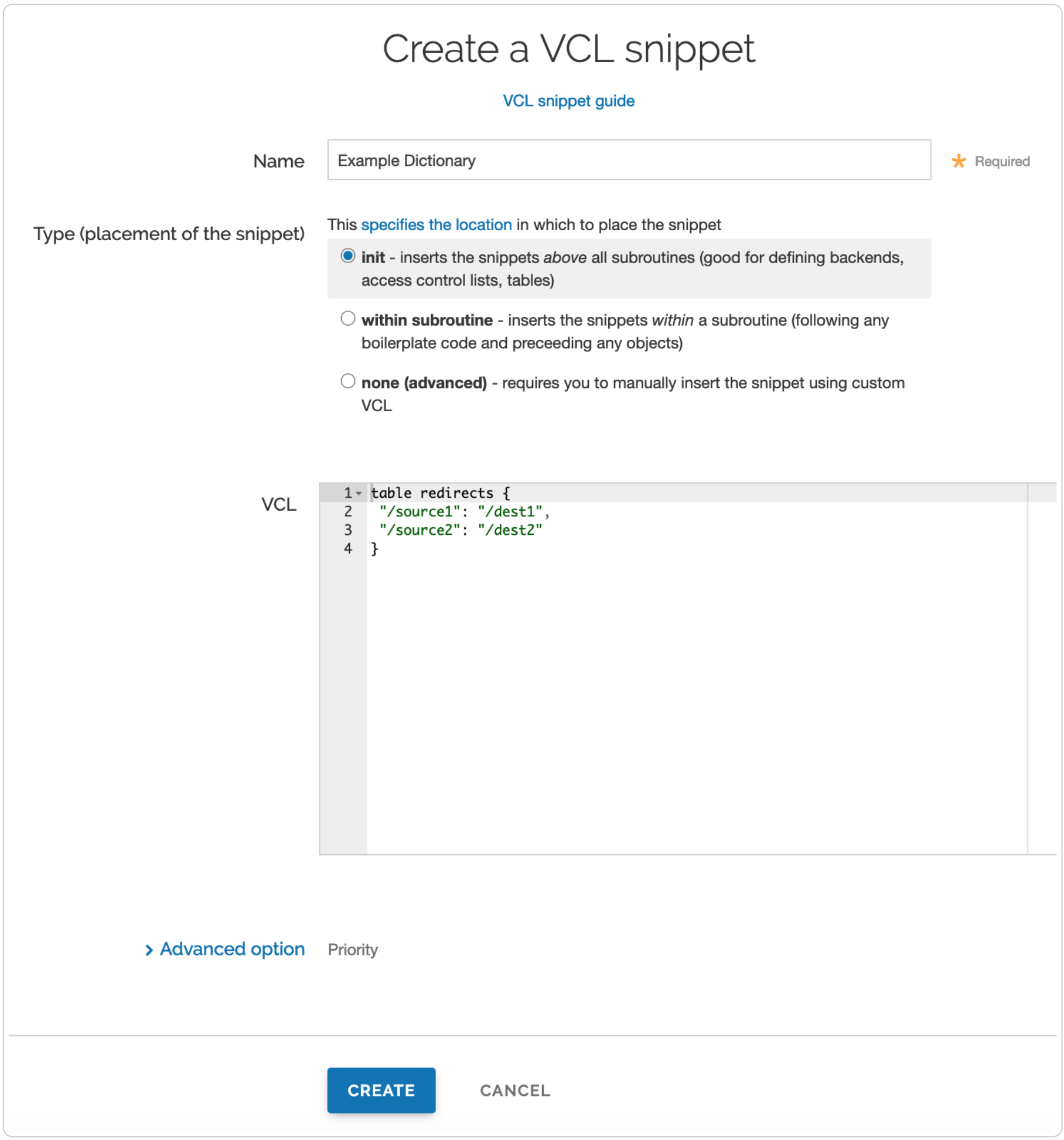
You can create a dictionary by either using VCL Snippets or via the web interface.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.

3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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.

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

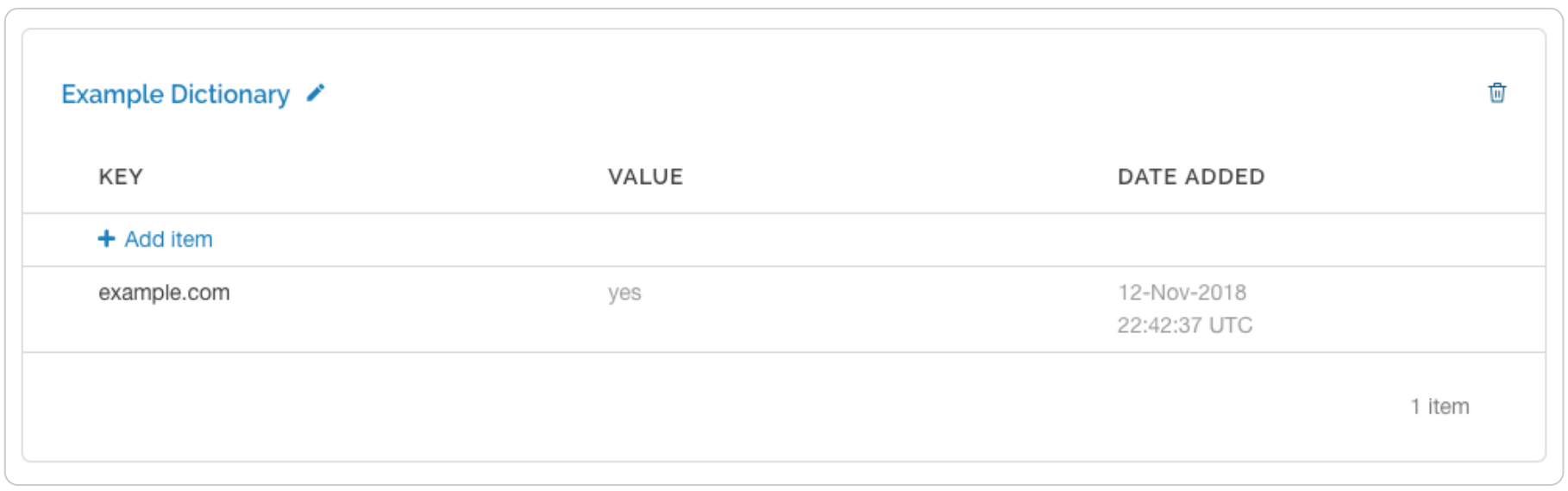
Start by creating a dictionary container using the following steps:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type the unique identifier for some item of data (e.g., `example.com`).
3. In the **Value** field, type 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.



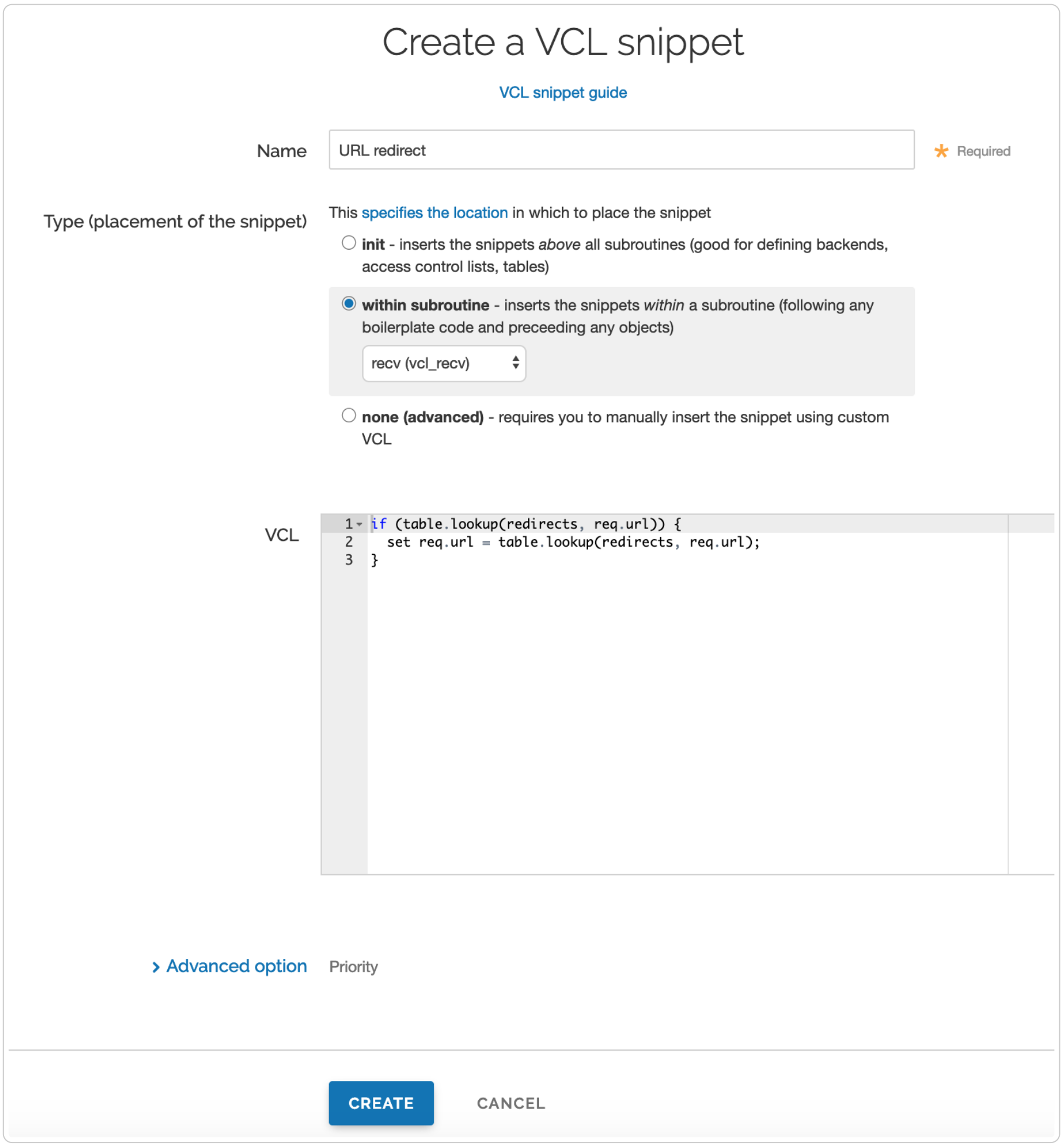
Using a dictionary

Once you've created a dictionary, you can start using it.

Using a dictionary with VCL Snippets

To start using your dictionary with VCL Snippets, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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:

   ```
   1   if (table.lookup(redirects, req.url)) {
   2     set req.url = table.lookup(redirects, req.url);
   3   }
   ```

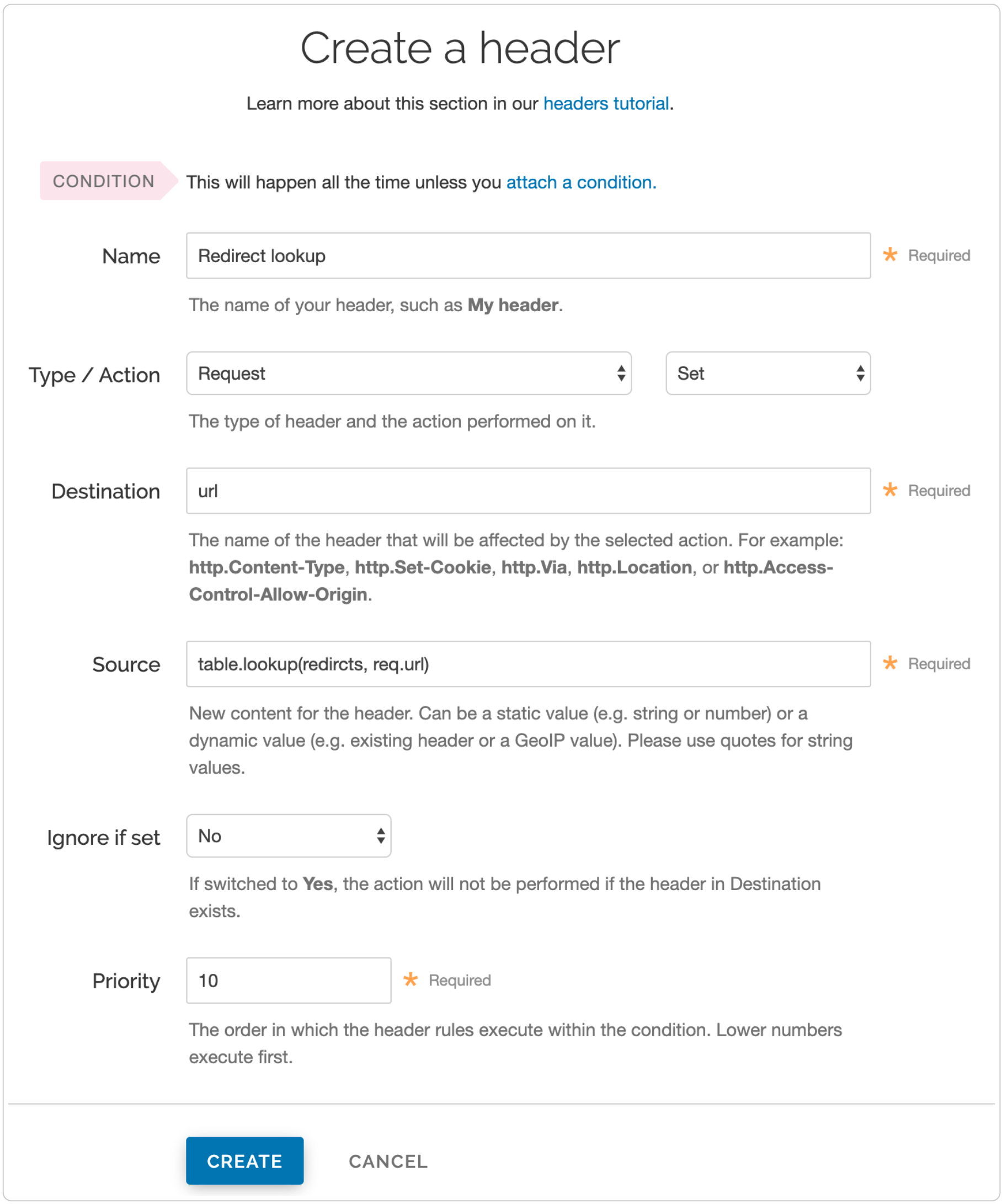
   where `table.lookup` checks the dictionary for the desired contents you want. The first parameter is the table being looked in and the second parameter is the key you're looking for. If the key exactly matches the second parameter, that value is returned. Be aware that regex doesn't work with a dictionary lookup.
10. Click **Create** to create the snippet.
11. Click the **Activate** button to deploy your configuration changes.

Using a dictionary via the web interface

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 and click the **Configure** link.

2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

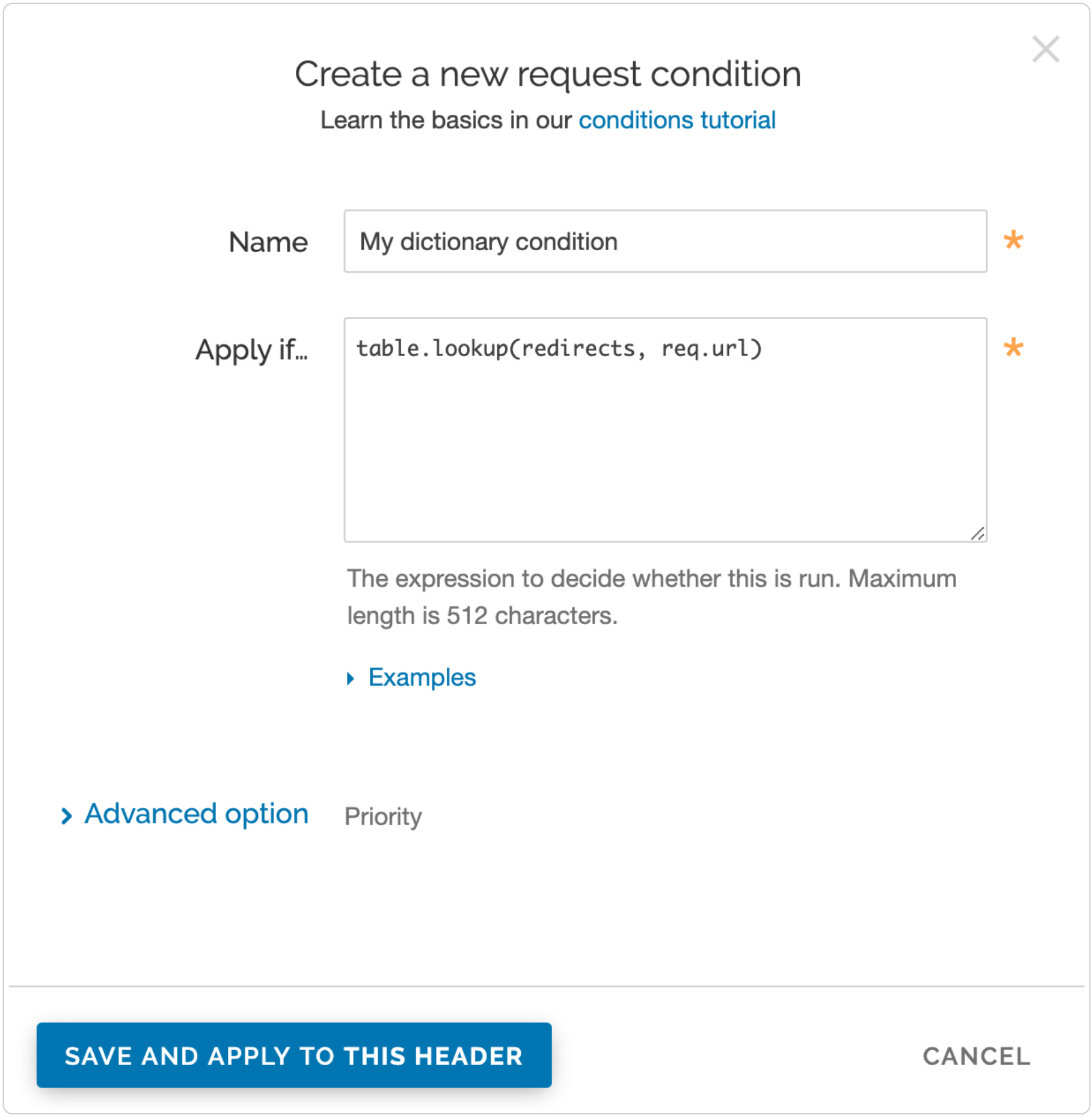


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type the name of your header rule (for example, `Redirect lookup`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header affected by the selected action (e.g., `url`).
   - In the **Source** field, type 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.

Specifying when to use the dictionary

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, type a name for your condition (e.g., `My dictionary condition`)
4. In the **Apply if** field, type 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

Keeping in mind their limitations, dictionaries can be edited using VCL Snippets or via dictionary containers and the items within them can be edited via the web interface.

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.

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

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

Keeping in mind their limitations, dictionaries can be deleted using VCL Snippets or dictionary containers and the items within them can be deleted via the web interface.

Deleting a dictionary using VCL Snippets

You can delete a dictionary using VCL Snippets by following the steps below:

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.

Deleting a dictionary container via the web interface

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.

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 Dictionary items using the API

https://docs.fastly.com/en/guides/working-with-dictionary-items-using-the-api

A dictionary item is a key-value pair that makes up an entry in a dictionary container in an Edge Dictionary. Once you create an Edge Dictionary and associate the dictionary container with a service, any dictionary items created will appear in your generated VCL.

For example, if you were using Edge Dictionaries to control geolocation redirects, the table would appear similar to this:

```
table geoip_redirect {
  "GB"  :  "www.example.co.uk",
  "IE"  :  "www.example.co.uk",
  "IT"  :  "www.example.com.it",
  "AU"  :  "www.example.com.au",
}
```

Finding a dictionary container's ID

If you already have a dictionary container associated with an active version of your service, you can add, update, or delete the items in it as long as you know the `dictionary_id`.

In our geolocation example, you would find your `dictionary_id` using the following API call:

```
curl -H 'Fastly-Key: FASTLY_API_TOKEN'
https://api.fastly.com/service/<service_id>/version/<version_number>/dictionary/geoip_redirect
```

which would return this response:

```
{
  "version": <version_number>,
  "name": "geoip_redirect",
  "id": "<dictionary_id>",
  "service_id": "<service_id>"
}
```

Adding new items to a dictionary

You can add new dictionary items without having to increment your service version number. For example, this API call to a geolocation table to add a new dictionary item:

```
curl -X POST -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'item_key=NZ&item_value=www.example.com.au' "https://api.fastly.co
m/service/<service_id>/dictionary/<dictionary_id>/item"
```

returns this response:

```
{
  "dictionary_id": "<dictionary_id>",
  "service_id": "<service_id>",
  "item_key": "NZ",
  "item_value": "www.example.com.au"
}
```

The table in the generated VCL would then be updated with the new dictionary item and look like this:

```
table geoip_redirect {
  "GB"  :  "www.example.co.uk",
  "IE"  :  "www.example.co.uk",
  "IT"  :  "www.example.com.it",
  "AU"  :  "www.example.com.au",
  "NZ"  :  "www.example.com.au",
}
```

Listing dictionary items

You can view all of the dictionary items in an Edge Dictionary. For example, this API call:

```
curl -H "Fastly-Key: FASTLY_API_TOKEN" https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/items
```

returns this response:

```
{
  "dictionary_id": "<dictionary_id>",
  "service_id": "<service_id>",
  "item_key": "some_key",
  "item_value": "some_value",
  "created_at": "2016-04-21T18:14:32+00:00",
  "deleted_at": null,
  "updated_at": "2016-04-21T18:14:32+00:00"
}
```

Upserting dictionary items

You can create and update dictionary items regardless of whether or not they exist. For example, the following API call to the geolocation table to update an existing dictionary item or create it if it doesn't exist:

```
curl -X PUT -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'item_value=www.example.co.aq' "https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/item/AQ"
```

returns this response:

```
{
  "dictionary_id": "<dictionary_id>",
  "item_key": "AQ",
  "item_value": "www.example.co.aq",
  "service_id": "<service_id>"
}
```

The table in the generated VCL would then be updated with the new dictionary item and look like this:

```
table geoip_redirect {
  "GB"  :  "www.example.co.uk",
  "IE"  :  "www.example.co.uk",
  "IT"  :  "www.example.com.it",
  "AU"  :  "www.example.com.au",
  "NZ"  :  "www.example.com.au",
  "AQ"  :  "www.example.co.aq",
}
```

Updating dictionary items one at a time

You can also update any dictionary item without having to increment your service version number. For example, the following API call to the geolocation table to update an existing dictionary item:

```
curl -X PATCH -H 'Fastly-Key: FASTLY_API_TOKEN' -d 'item_value=www.example.co.uk' "https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/item/NZ"
```

returns this response:

```
{
  "dictionary_id": "<dictionary_id>",
  "item_key": "NZ",
  "item_value": "www.example.co.uk",
  "service_id": "<service_id>"
}
```

The table in the generated VCL would then be updated with the new dictionary item and look like this:

```
table geoip_redirect {
  "GB"  :  "www.example.co.uk",
  "IE"  :  "www.example.co.uk",
  "IT"  :  "www.example.com.it",
  "AU"  :  "www.example.com.au",
  "NZ"  :  "www.example.co.uk",
  "AQ"  :  "www.example.co.aq",
}
```

Batch updating dictionary items

You can update up to 1,000 dictionary items with a single API call. The following actions are available within a batch update:

- **Upsert** - Creates an item if it doesn't exist, otherwise modifies the existing one.
- **Create** - Creates a new item, but will not update an existing one.
- **Update** - Updates an existing item, but will not create a new one if it doesn't exist.
- **Delete** - Permanently deletes the item from the dictionary.

For example, to batch update existing dictionary items in the geolocation table, create a new file called `batch.json` that contains the following JSON-encoded data:

```
{
  "items": [
    {
      "op": "create",
      "item_key": "JP",
      "item_value": "www.example.co.jp"
    },
    {
      "op": "update",
      "item_key": "GB",
      "item_value": "www.example.co.uk"
    },
    {
      "op": "delete",
      "item_key": "IT"
    }
  ]
}
```

Then you can make the following API call:

```
curl -X PATCH -H 'Content-Type: application/json' -H 'Fastly-Key: FASTLY_API_TOKEN' -d @batch.json "https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/items"
```

See the API documentation for more information.

Deleting a dictionary item

WARNING: Dictionary item deletions are permanent. Fastly does not store data. If you delete a dictionary item, the entry is gone forever from all versions of your service.

To remove an item from your table, use this API call:

```
curl -X DELETE -H 'Fastly-Key: FASTLY_API_TOKEN' https://api.fastly.com/service/<service_id>/dictionary/<dictionary_id>/item/NZ
```

Unlike creation and update of dictionary items, the API call returns no response.

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.

Domains & Origins

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

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.

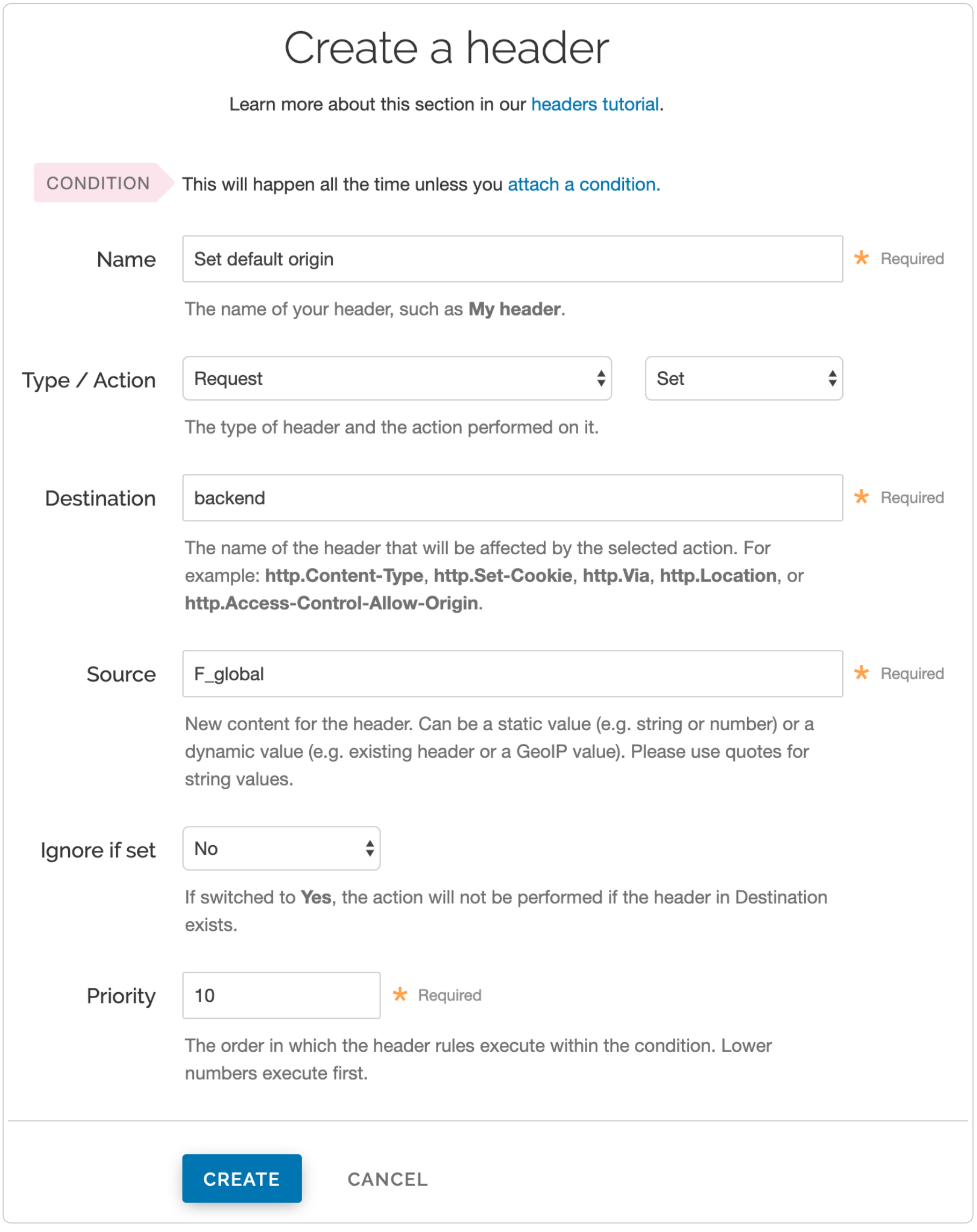
Using the web interface

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type `backend`.
   - In the **Source** field, type the name of 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.

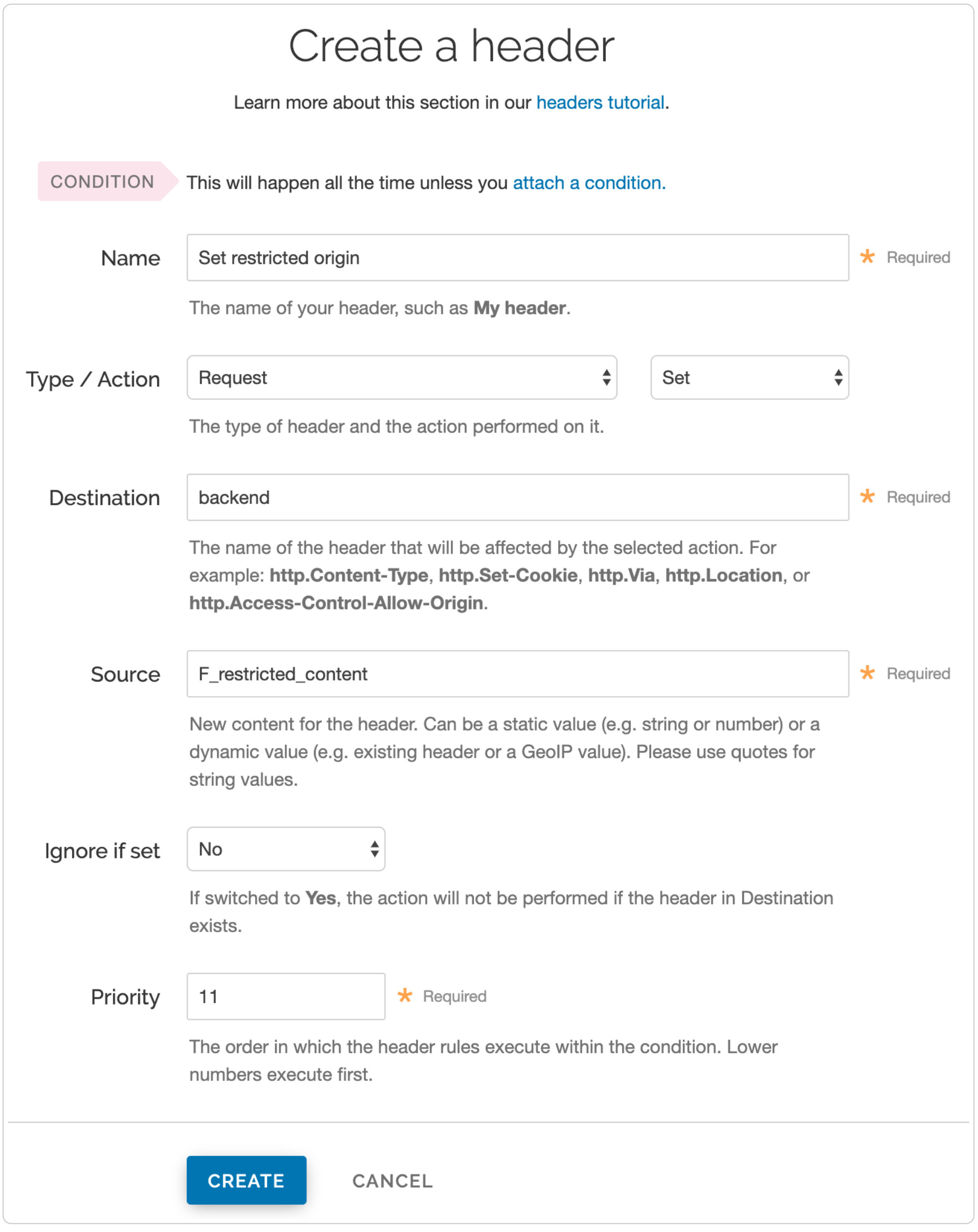
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

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, type the name of your header rule (for example, `Set restricted origin`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.

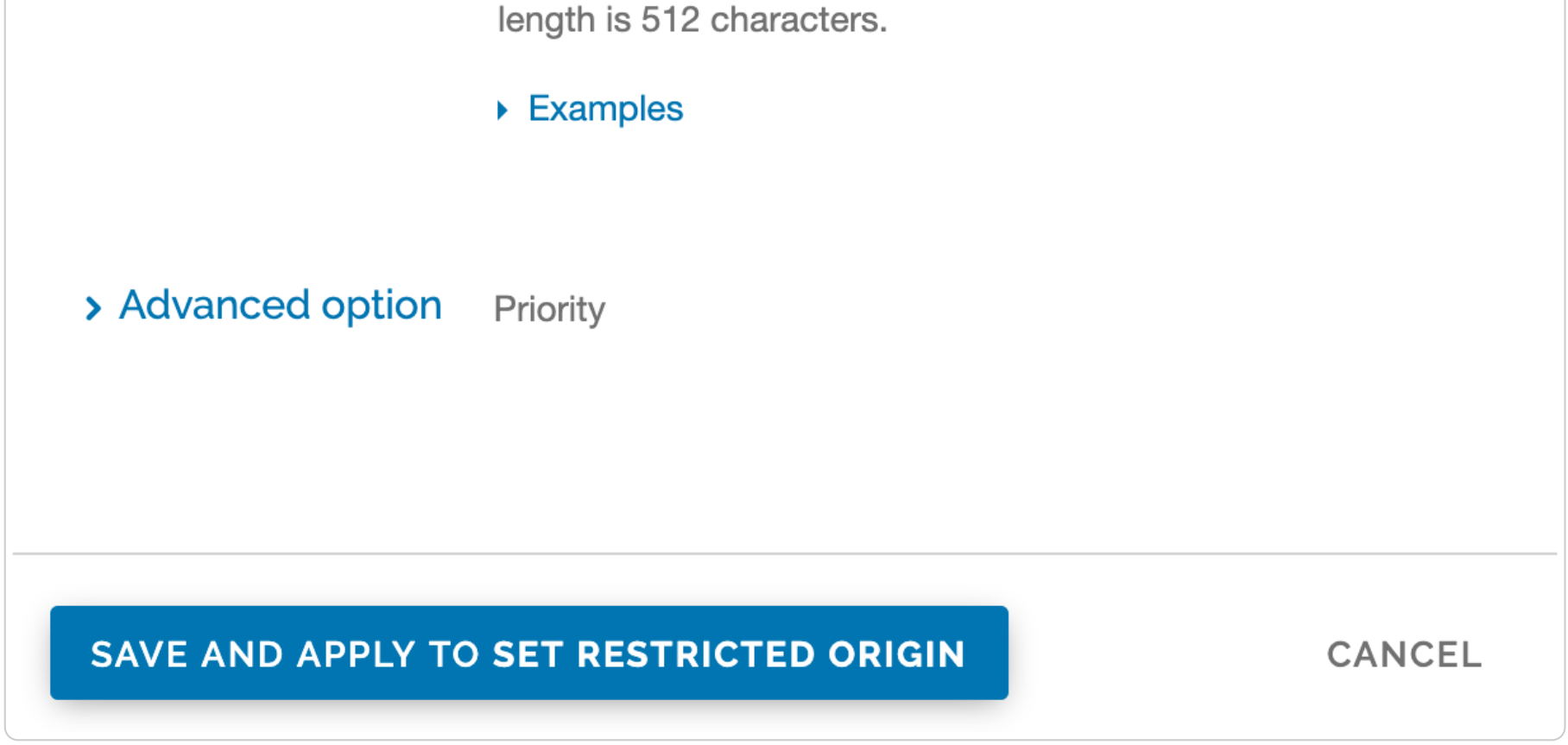
- In the **Destination** field, type `backend`.
- In the **Source** field, type the name of restricted origin server you want to serve content to the users residing in the countries specified in the condition (here it's `F_restricted_content`). Preview the VCL to find the name of the origin server.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `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, type a descriptive name for the new condition (for example, `From Restricted Location`).
   - In the **Apply if** field, type a request condition. For example, to send all users in Asia and Europe to the restricted origin server, type `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;
}
```

Connecting to origins

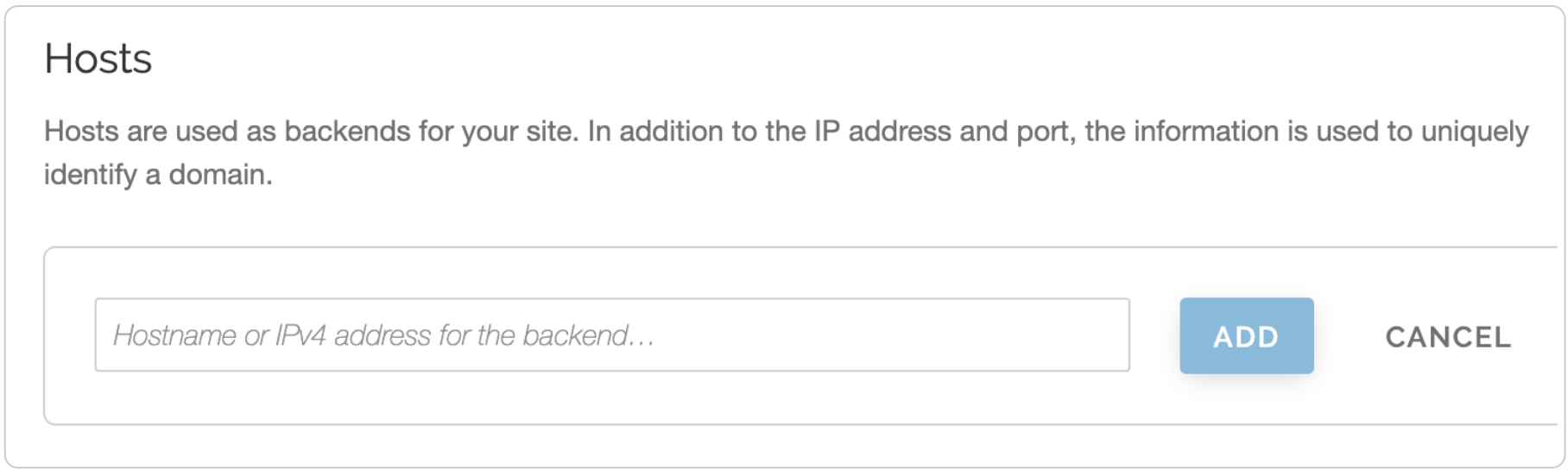
https://docs.fastly.com/en/guides/connecting-to-origins

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

Adding a host

To add a host, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. Click the **Create a host** button. The Hosts field appears.

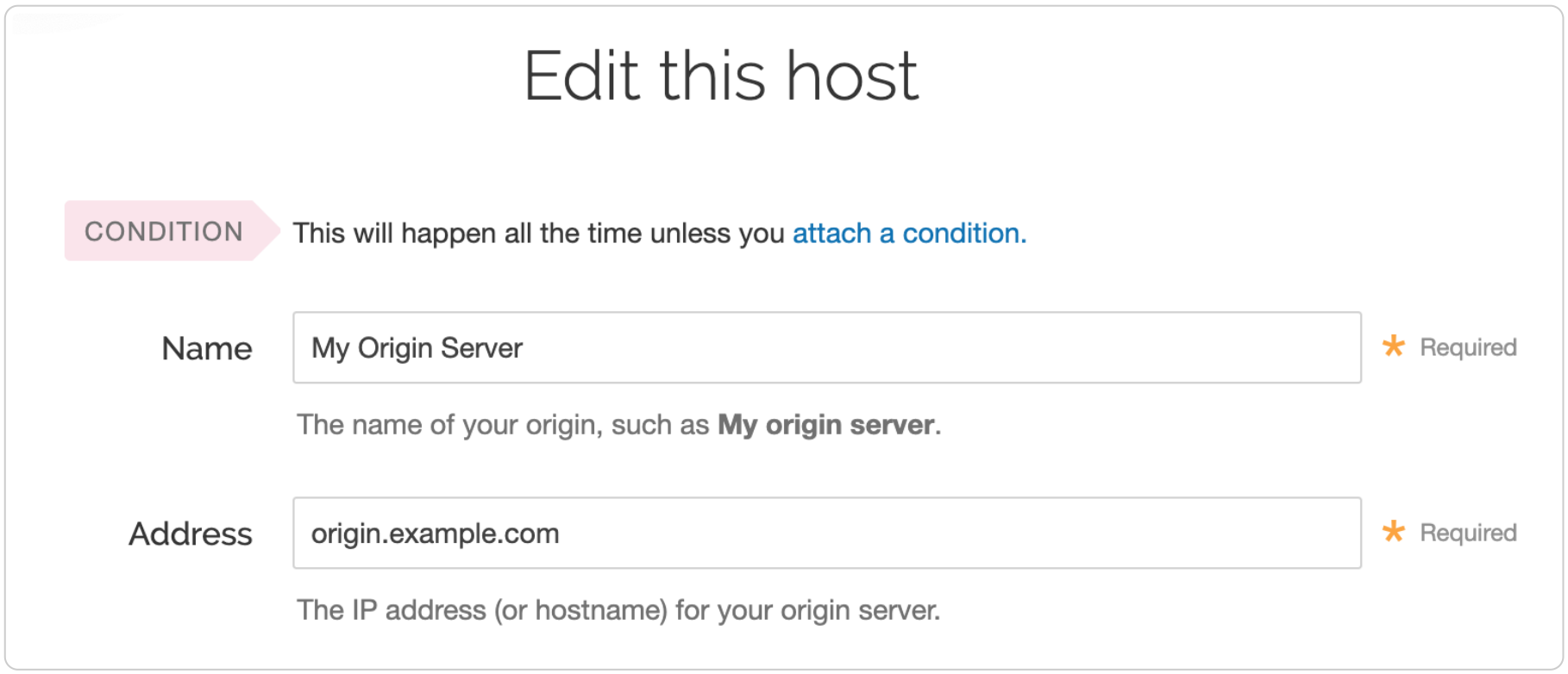


6. Fill out the **Hosts** field by typing the hostname or IP address of your origin server. Entering a hostname automatically enables Transport Layer Security (TLS) and assigns port 443. Entering an IP address disables TLS and assigns port 80.
7. Click **Add** to add your host.

Editing a host

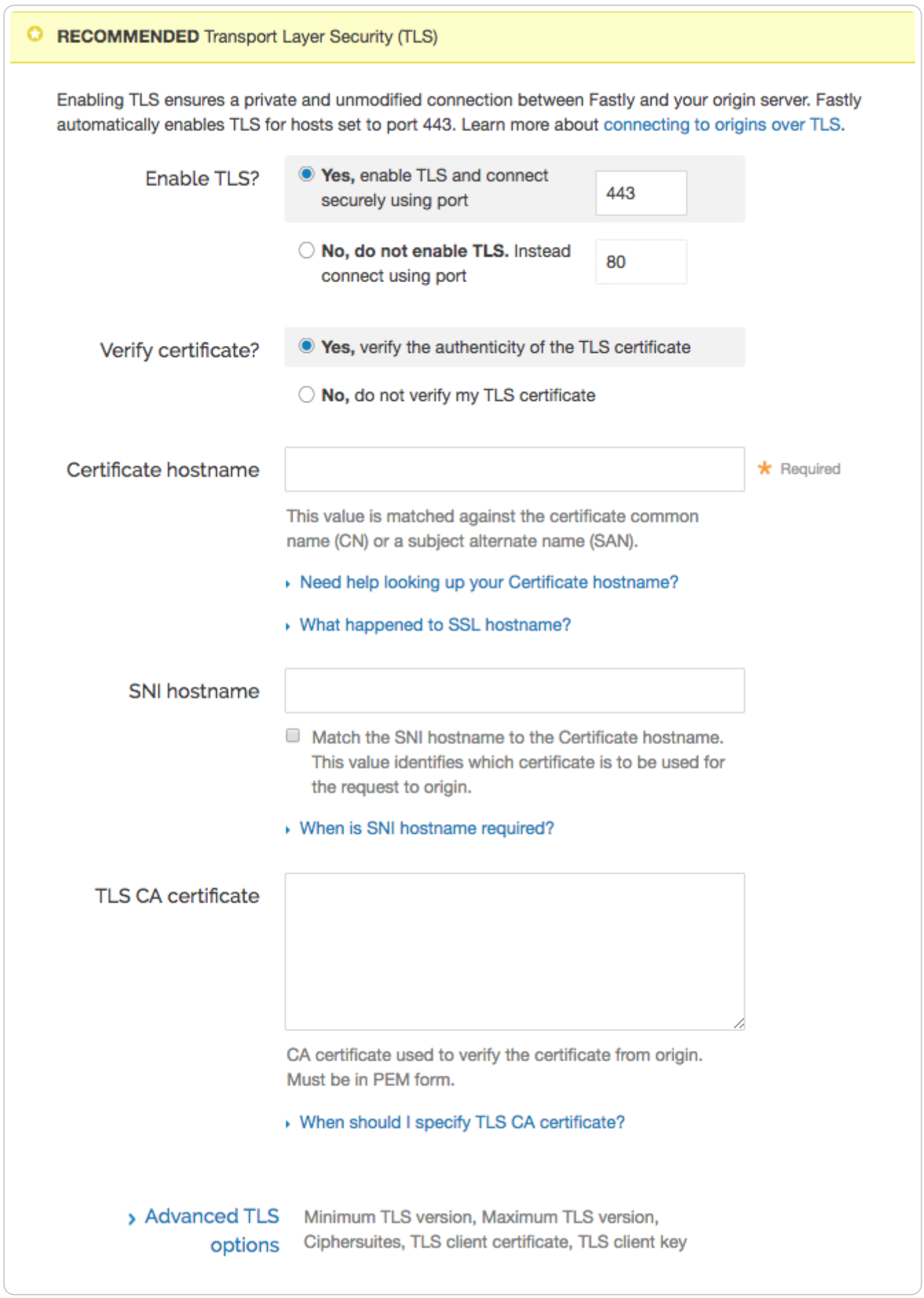
After you've created your host, you can edit the host's details by following the steps below:

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, type the name of your server (for example, `My Origin Server`). This name is displayed in the Fastly web interface.
   - In the **Address** field, type 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.



3. Fill out the **Transport Layer Security (TLS)** area as follows:
    - Leave the **Enable TLS?** 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.
    - Leave the **Verify certificate?** 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 attack. Consider uploading a CA certificate instead of disabling certificate validation.

    - In the **Certificate hostname** field, type 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.
    - If you are specifying an SNI hostname, see the section below.
    - If you are specifying a TLS CA certificate, see the section below.
4. Fill out the remaining **Create a host** fields 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, type a weight in the **Weight** field.

5. Click the **Advanced options** link and decide which of the optional fields to change, if any:
   - In the **Maximum connections** field, optionally type 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 type the number of errors allowed before an origin is considered down.
   - In the **Connection timeout** field, optionally type how long, in milliseconds, to wait for a connection timeout. The default is 1000 milliseconds.
   - In the **First byte timeout** field, optionally type how long, in milliseconds, to wait for a first byte timeout. The default is 15000 milliseconds.
   - In the **Between bytes timeout** field, optionally type how long, in milliseconds, to wait between bytes. The default is 10000 milliseconds.
6. In the **Override host** field, optionally type 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 override host quick configuration. Keep in mind the following:
   - If you're using Amazon S3 as your origin, type `<yourbucket>.s3.amazonaws.com`.
   - If you're using Google Cloud Storage as your origin, type `<your bucket name>.storage.googleapis.com`.

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

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

And that's all you need to do. Everything else is optional, but just in case you'd like to set them, we've included the information below.

Setting the TLS hostname

Normally we check the server certificate against the hostname portion of the address for your origin entered in the **Create a host** window. Checking the certificate is done by using the value of the Certificate Hostname field in your origin TLS settings. To have Fastly verify the certificate using a different hostname, specify it via the **SNI Hostname** field under **Advanced options**.

This information also gets sent to the server in the TLS handshake. If you are using Server Name Indication (SNI) to put multiple certificates on your origin, specifying it in the **SNI Hostname** field will select which one is used.

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. 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 of the **TLS options** page.

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.

If you don't enter an actual value in your certificate hostname, the .host value is used by default to verify the certificate. The .host value is the actual IP address or virtual hostname you enter in the **Address** field on the **Host** area of the **Origins** page. This value is matched against the certificate common name (CN) or a subject alternate name (SAN).

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
nothing	www.example.org	the .host value from the **Address** field	www.example.org
www.example.com	www.example.org	www.example.com	www.example.org

If Certificate hostname contains…	and SNI hostname contains…	then the Certificate Validation value will be…	and the SNI value will be…
nothing	nothing	the .host value from the **Address** field	nothing

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 name that matches the wildcard certificate. 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	*.example.com
live.example.com	live.example.com
*.example.com	*.example.com

If you set the certificate hostname to `*.example.com`, Fastly will treat it as a literal. When using that as the certificate hostname, `*.example.com` is the only option for the SNI hostname.

Specifying a TLS CA certificate

If you're using a certificate that is either self-signed or signed by a certification authority (CA) not commonly recognized by major browsers (and unlikely to be in the Ubuntu bundle that we use), you can provide the certificate in PEM format via the **TLS CA certificate** field. 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-----
```

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 appropriate text boxes on the **TLS options** page.

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

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** under **Advanced options**. We recommend setting both to the most up-to-date TLS protocol, currently 1.2, 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_2
```

Replace `-tls1_2` with `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 1.2 of the TLS protocol by default. For backend connections from our edge nodes to customer origins, Fastly supports TLS 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 under **Advanced options** 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.

Enabling global POPs

https://docs.fastly.com/en/guides/enabling-global-pops

The sun never sets on the Fastly empire, but how can you take full advantage? Simply set your CNAME record to `nonssl.global.fastly.net` for non TLS traffic. You'll now have access to all of our worldwide POPs as they come online. We don't restrict POP access. Instead, you control it.

How to check if your CNAME is set to nonssl.global.fastly.net

Run the following command in your terminal:

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

Your output should appear similar to the following:

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

If you don't see `nonssl.global.fastly.net.` in your output, then your CNAME isn't properly set. We link to instructions for setting your CNAME for a number of popular providers.

Instead of using the above command in your terminal, you can also use various online DNS checking tools, such as the OpenDNS Cache Check.

Limiting POP use to North America and the European Union

You can route your traffic through Fastly's North American and European Union POPs only. If you're not using TLS, simply set your CNAME record to `nossl.us-eu.fastly.net.` instead of `nonssl.global.fastly.net.`. If you're using TLS, see our guide on CNAME records to find the appropriate entry.

Failover configuration

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

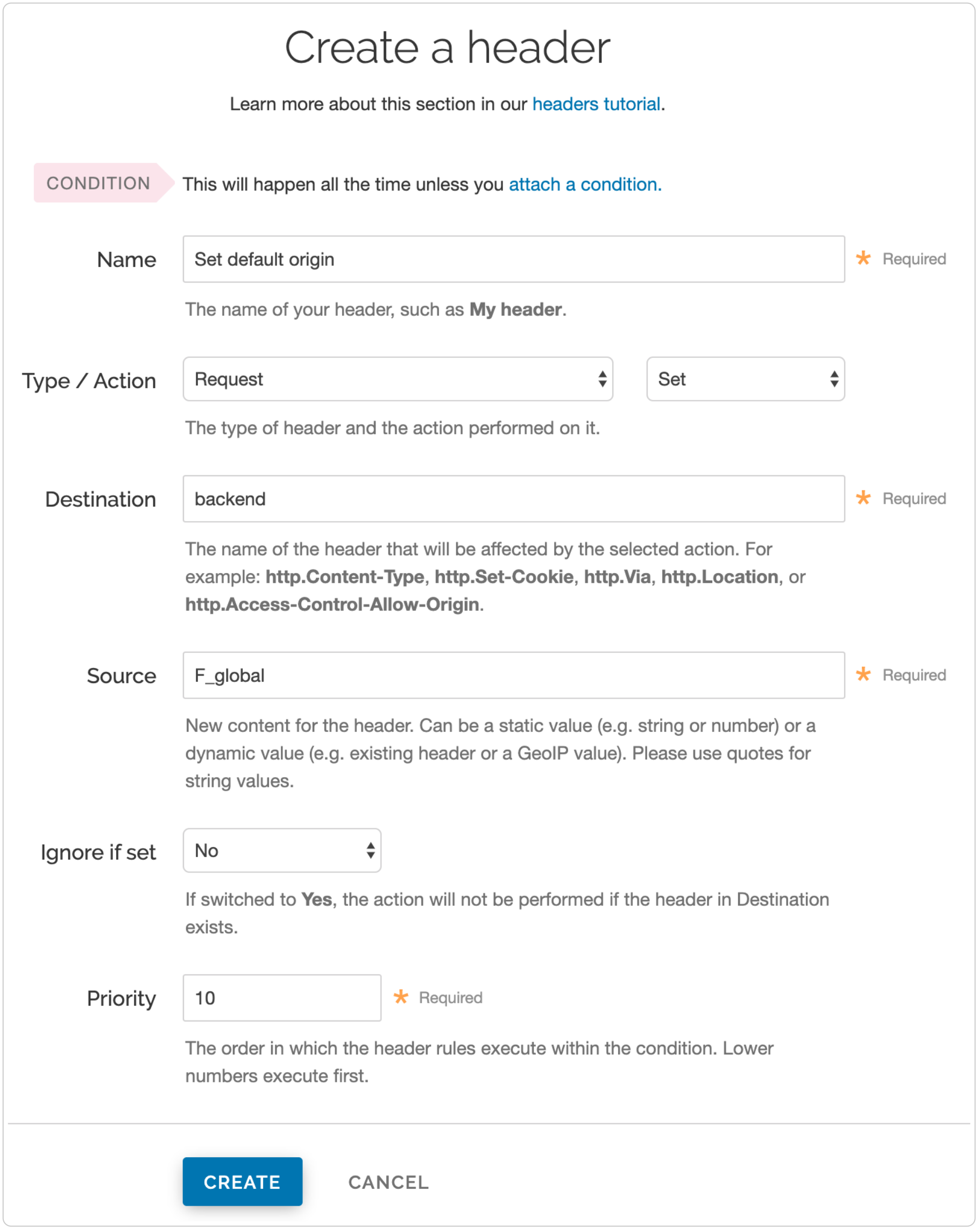
To configure a failover origin server you must turn off automatic load balancing for both the server that will act as your primary origin server and the server that will become your failover origin server.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

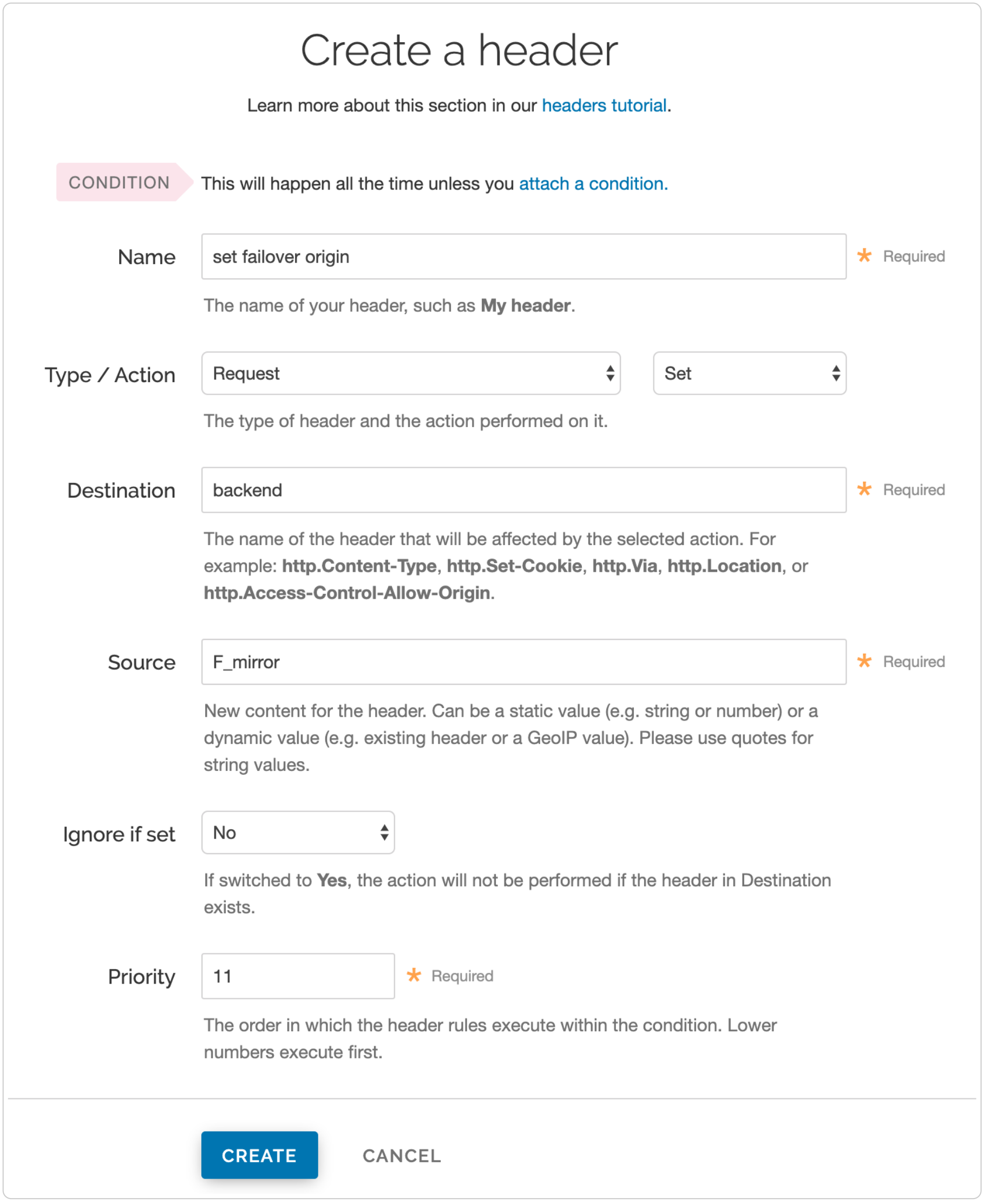
Configure the primary and failover origin servers

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. The Create a header window appears.



3. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type 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, type the name of the header that will be affected by the selected action.
   - In the **Source** field, type where the new content for the header comes from.
   - 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. The Create a header window appears.

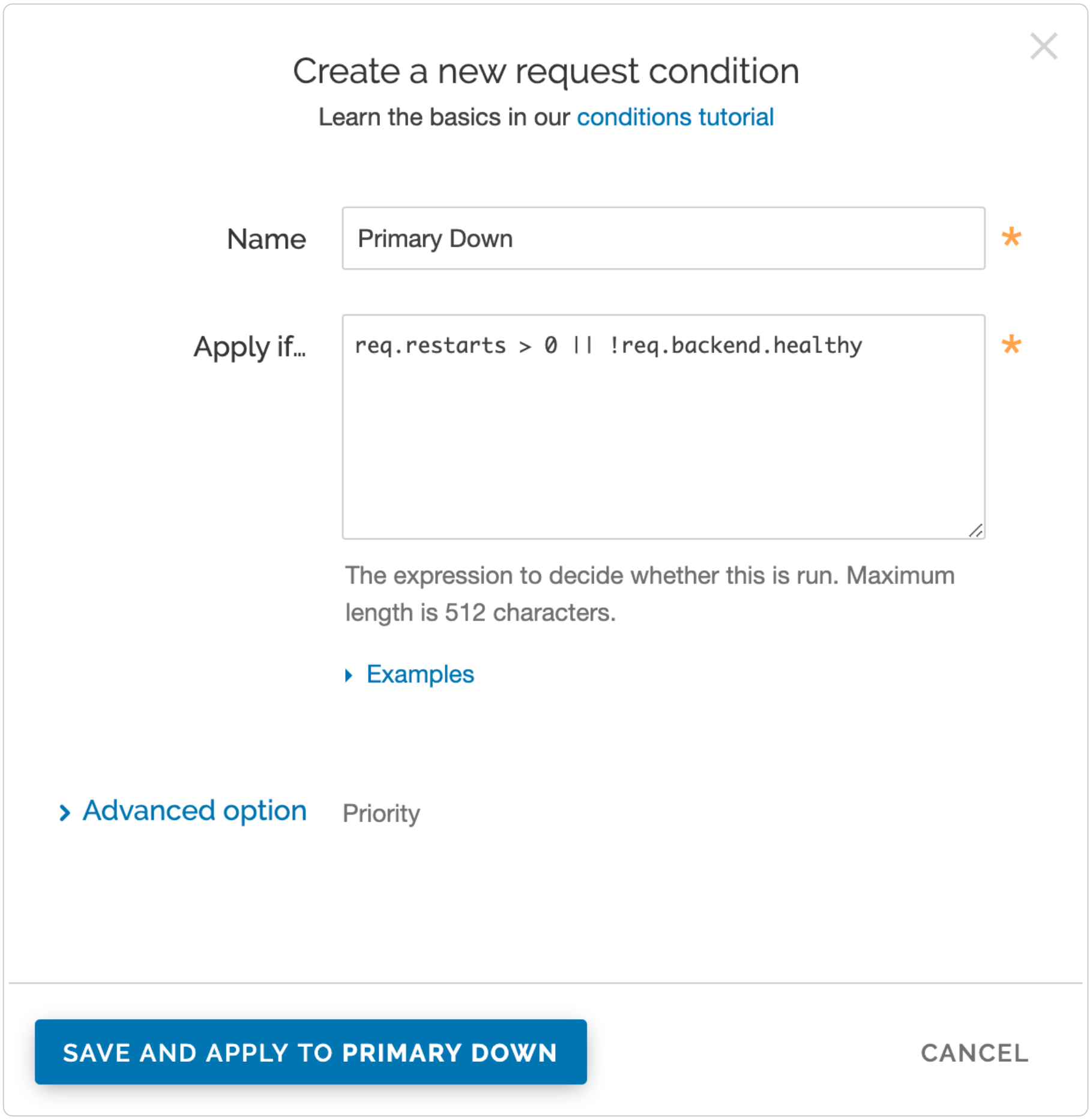


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type a descriptive name for the header. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header that will be affected by the selected action.
   - In the **Source** field, type where the new content for the header comes from.
   - Leave the **Ignore if set** control at the default setting.
   - In the **Priority** field, type 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, type a descriptive name for the new condition (for example, `Primary Down`).
   - In the **Apply if** field, type 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, type `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

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 IPv6, follow the instructions below as appropriate for your CNAME record.

NOTE: Fastly doesn't support IPv6 connections to origin servers.

Enabling IPv6 on Non-TLS- and TLS-enabled hostnames

You can enable IPv6 dualstack (IPv4 and IPv6) functionality for your hostname by prefixing your CNAME record with `dualstack`. For example, if you're on our "g" shared SAN certificate, you have the following dualstack options:

- `dualstack.g.shared.global.fastly.net` (dualstack global map for HTTP/2 support)
- `dualstack.g.shared.us-eu.fastly.net` (dualstack US-EU map for HTTP/2 support)
- `dualstack.g.ssl.global.fastly.net` (dualstack global map for HTTP/1 only)
- `dualstack.g.ssl.us-eu.fastly.net` (dualstack US-EU map for HTTP/1 only)

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 use a customer-specific hostname, contact Fastly Support and we'll provide you with an IPv6 map or enable 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

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 connecting to origins. 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, type a name for the non-secure server (for example, `Server Name (plain)`).
- In the **Address** field, type 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 connecting to origins 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:

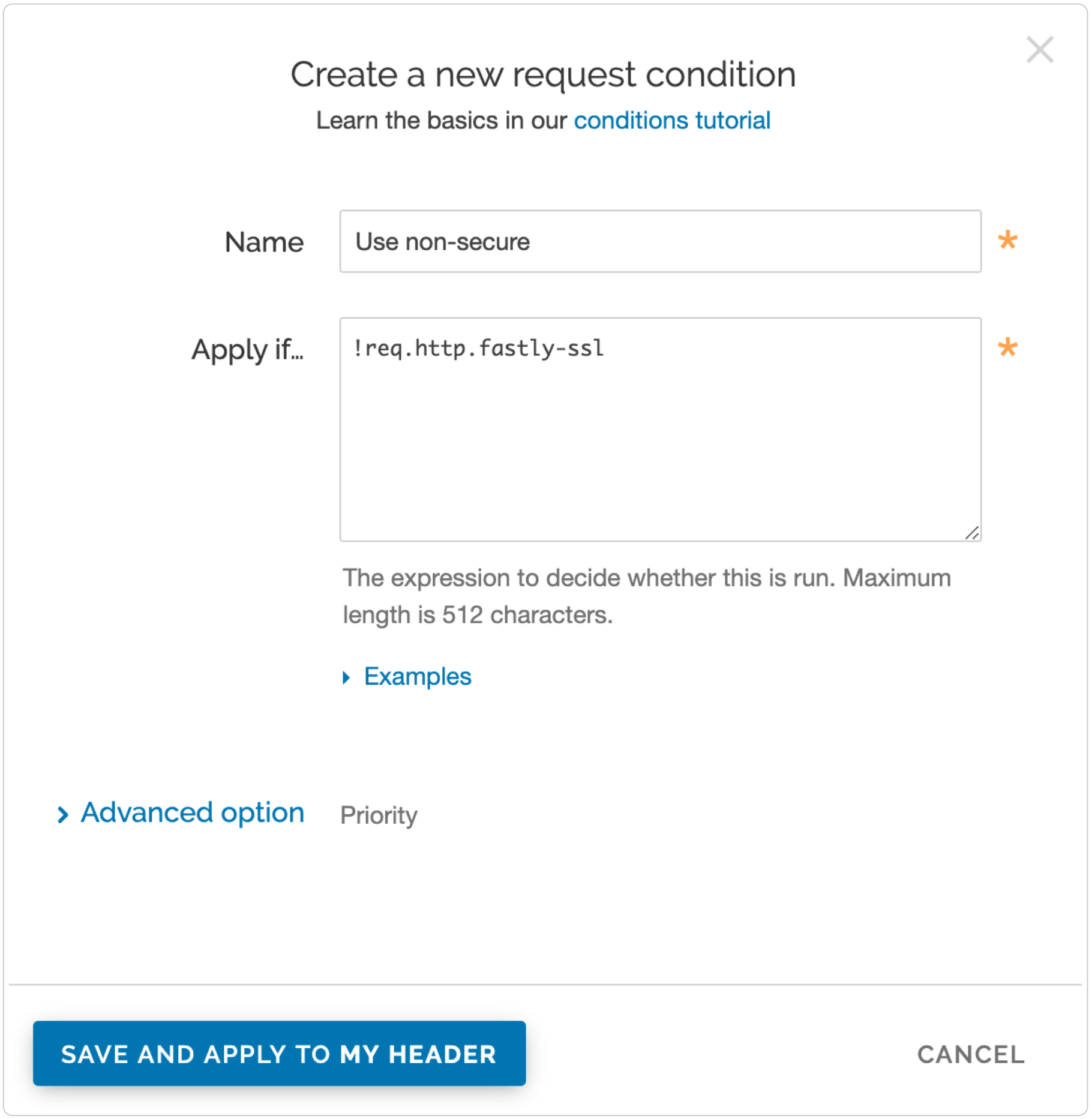
- In the **Name** field, type a name for the non-secure server (for example, `Server Name (secure)`).
- In the **Address** field, type the address of the non-secure server (for example, `server.example.com`).
- In the **Transport Layer Security (TLS)** section, leave the **Enable TLS?** default set to **Yes**.

Conditionally send traffic to origins

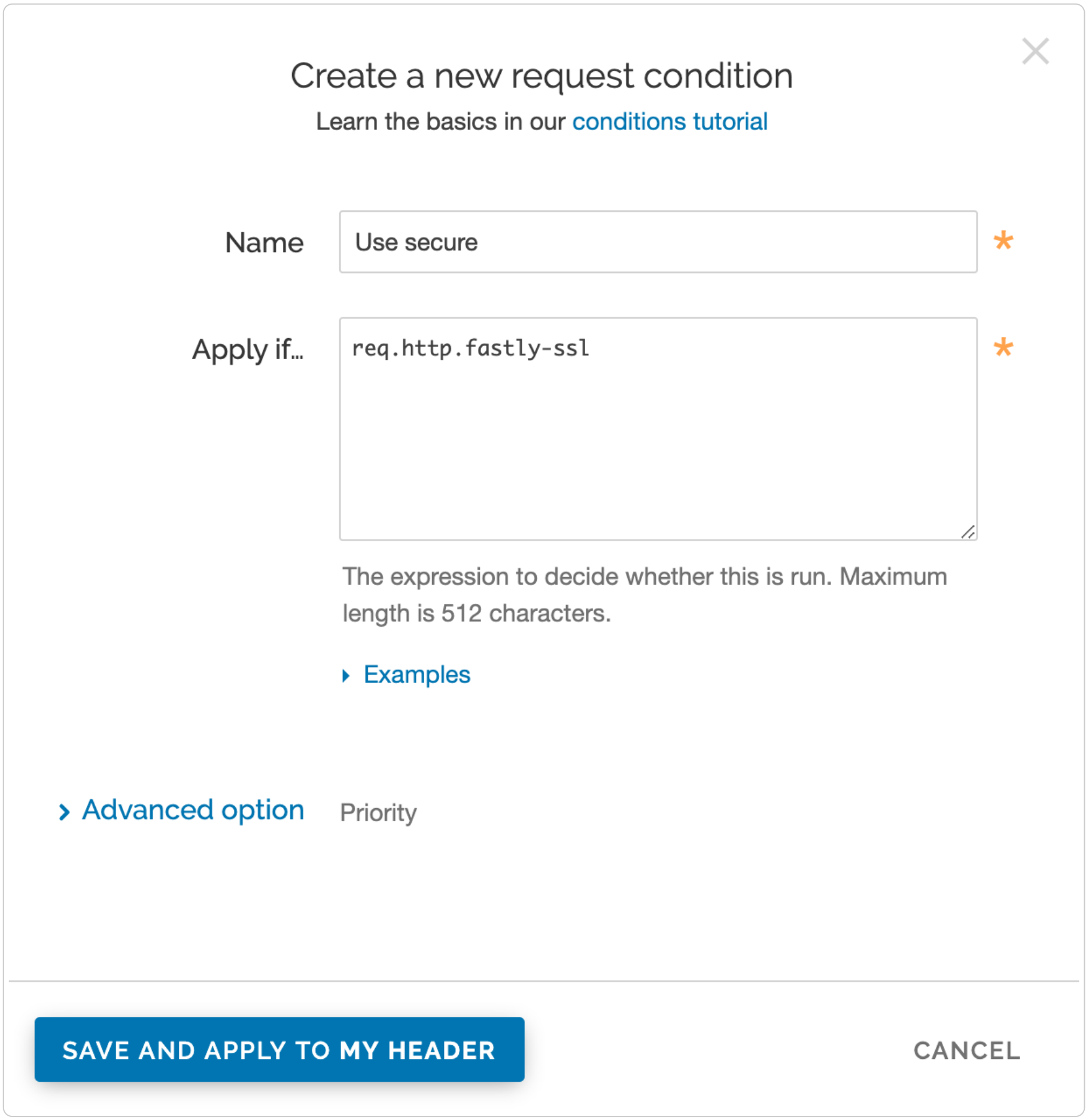
To conditionally determine which server receives secure and non-secure requests, Fastly relies on the presence or absence of a specific header when the backend is selected. When an incoming connection is received over TLS, Fastly sets the `req.http.fastly-ssl` header to determine which server to use.

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, type the name of the condition specifying use of the non-secure server (for example, `Use non-secure`).
   - In the **Apply if** field, type `!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, type the name of the condition specifying use of the secure server (for example, `Use secure`).
   - In the **Apply if** field, type `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.

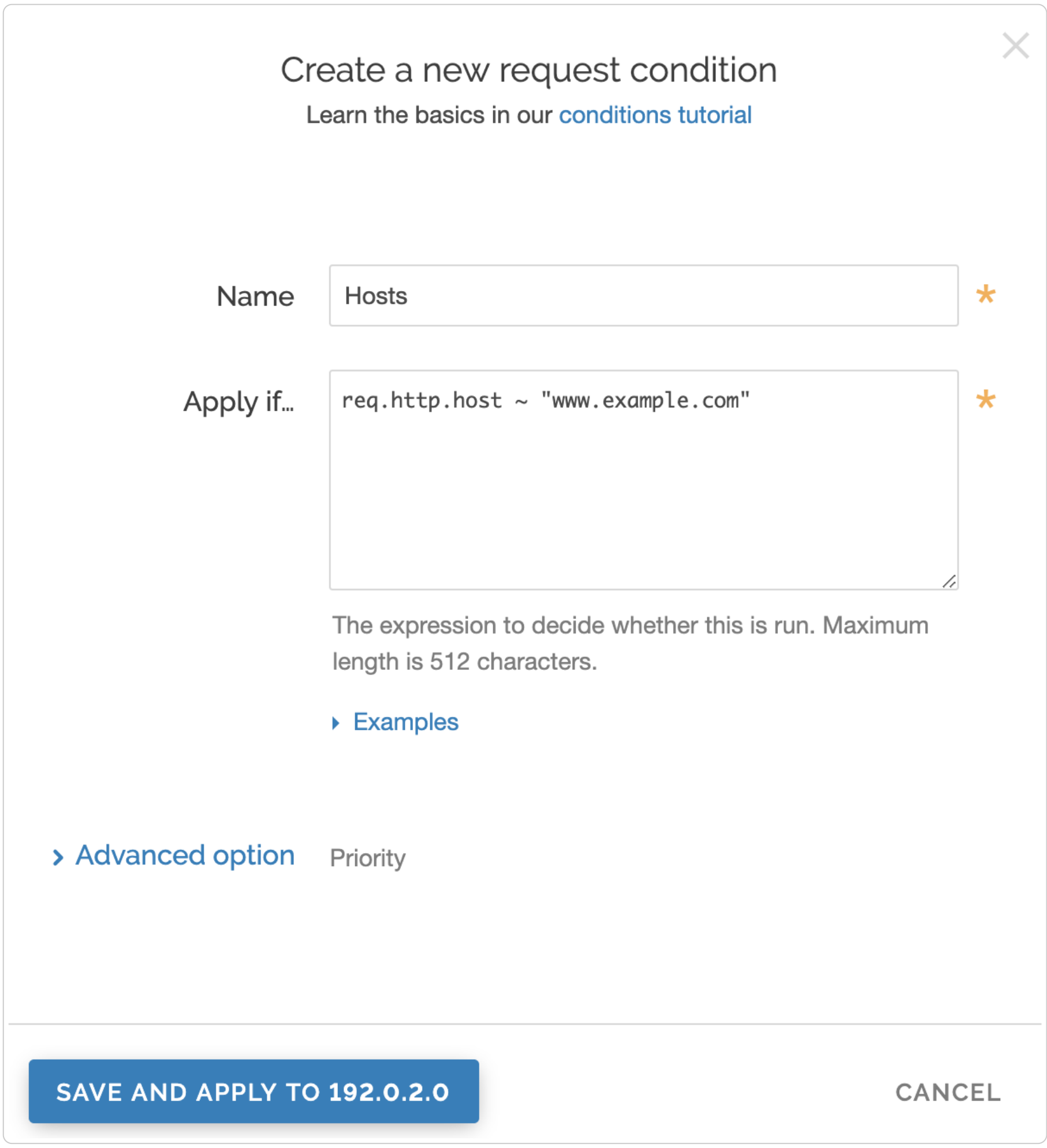
> **Routing assets to different origins**
>
> https://docs.fastly.com/en/guides/routing-assets-to-different-origins

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.



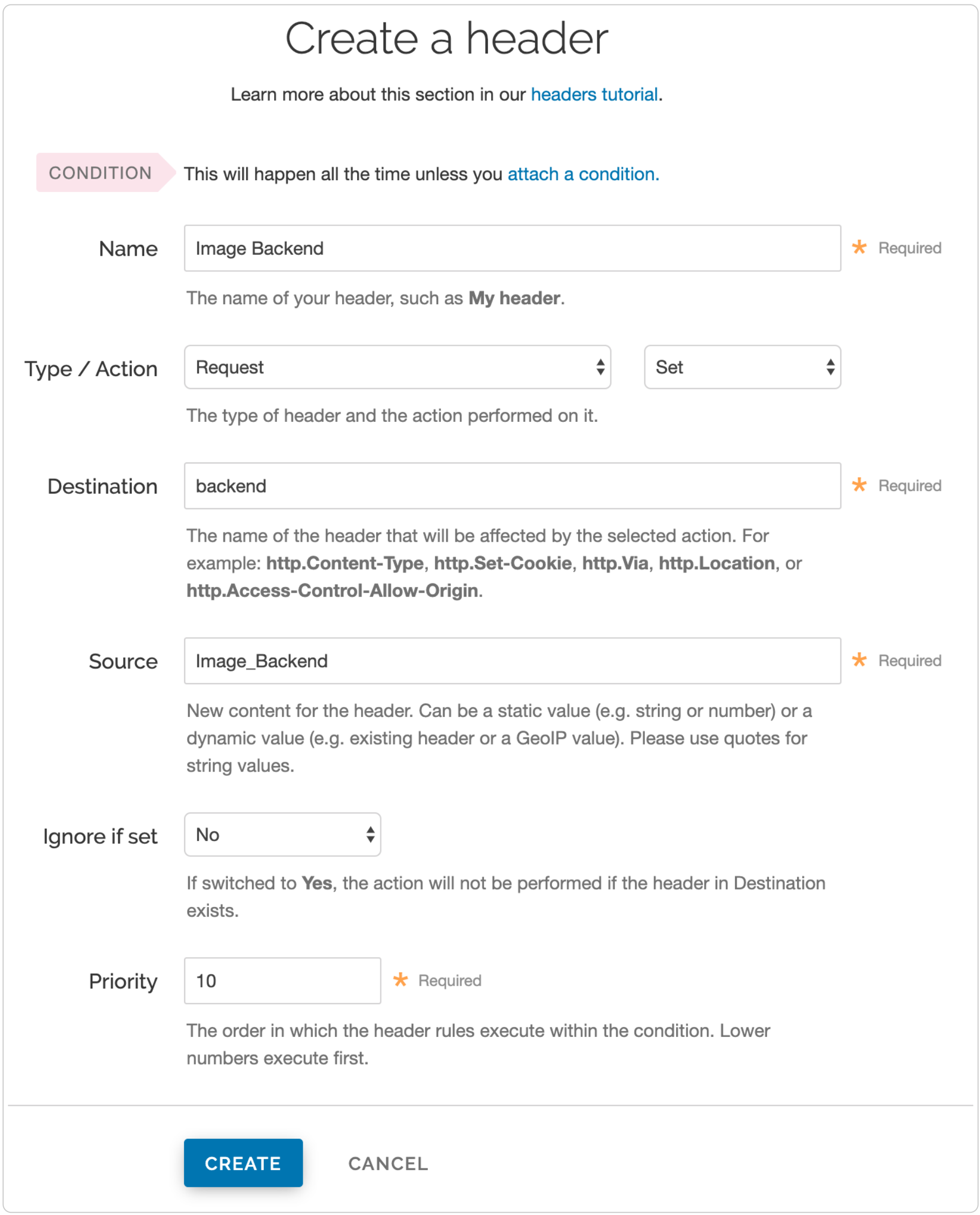
6. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a human-readable name for the condition.
   - In the **Apply if** field, type the conditions that you want to apply to your origin server. For example, for hosts, you could type `req.http.host ~ "www.example.com"`. Or, for content-type / URL, you could type `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.

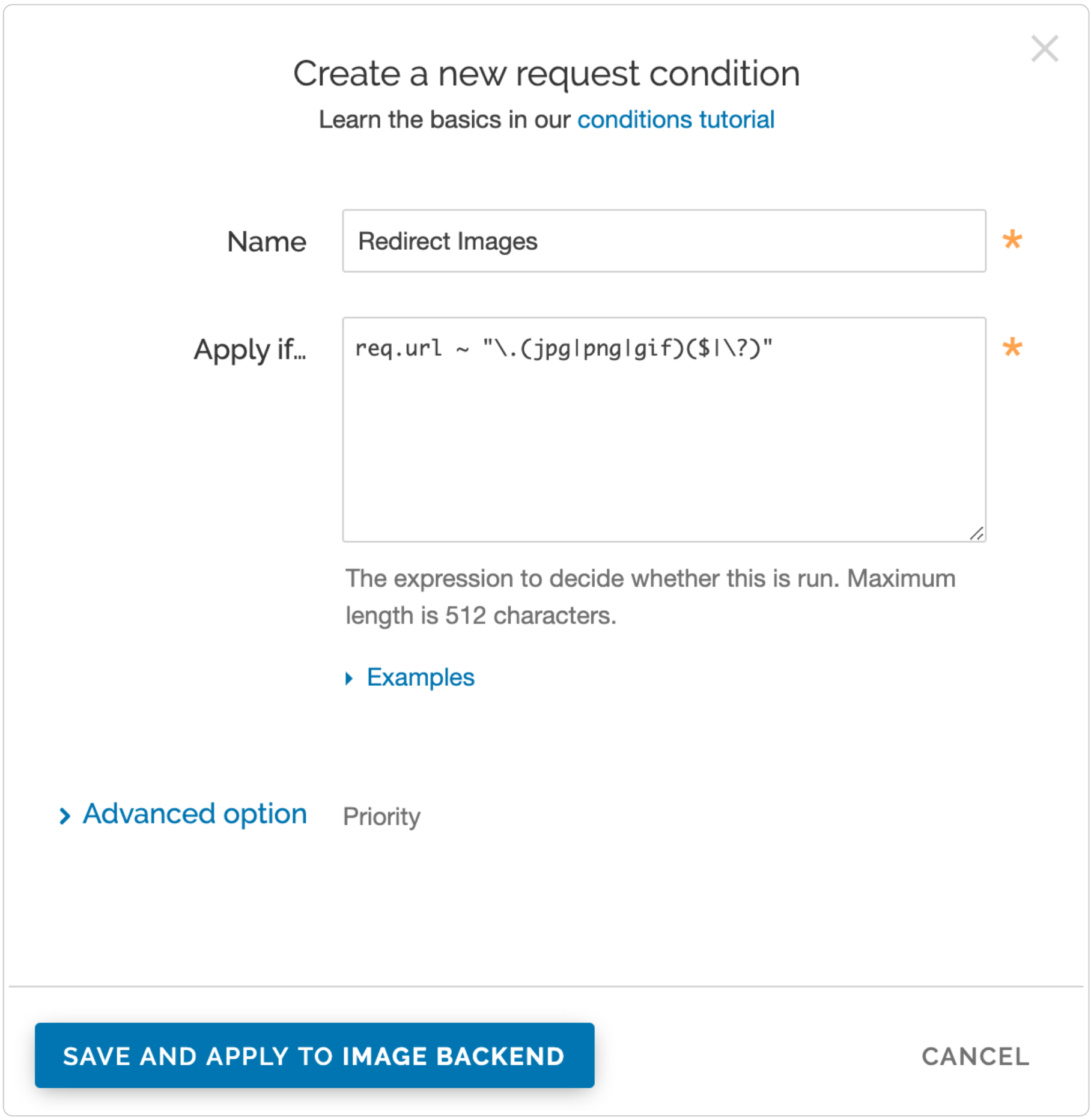
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.

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, type `Image Backend` (or any meaningful, preferred name).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `backend`.
   - In the **Source** field, type `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, type `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.



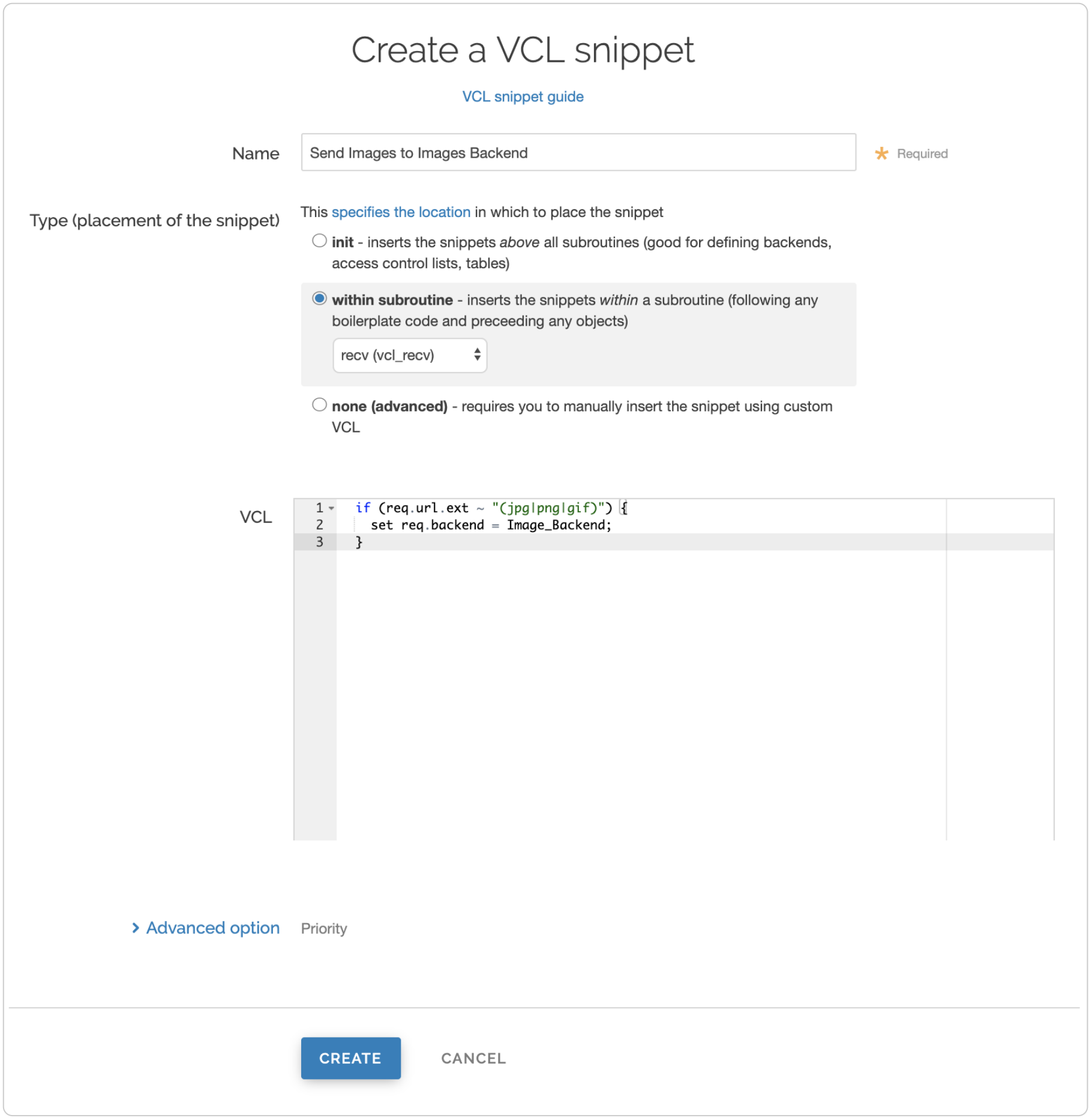
9. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type `Redirect Images` (or any meaningful, preferred name).
   - In the **Apply if** field, type `req.url ~ "\.(jpg|png|gif)($|\?)"`.
10. Click the **Save and apply to** button. The condition appears on 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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:

```
1  if (req.url.ext ~ "(jpg|png|gif)") {
2    set req.backend = Image_Backend;
3  }
```

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

Setting up redundant origin servers

https://docs.fastly.com/en/guides/setting-up-redundant-origin-servers

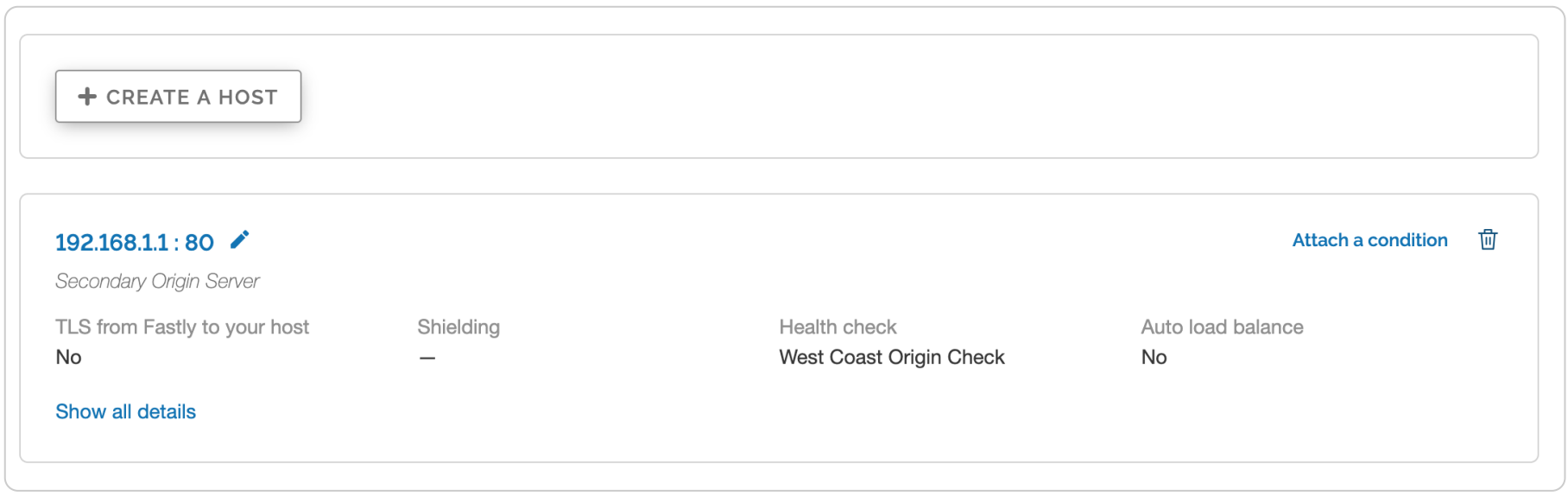
Sometimes you want to set up two different origin servers, one as a primary and one as a backup in case the primary becomes unavailable. You can do this via the web interface or using custom VCL.

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.

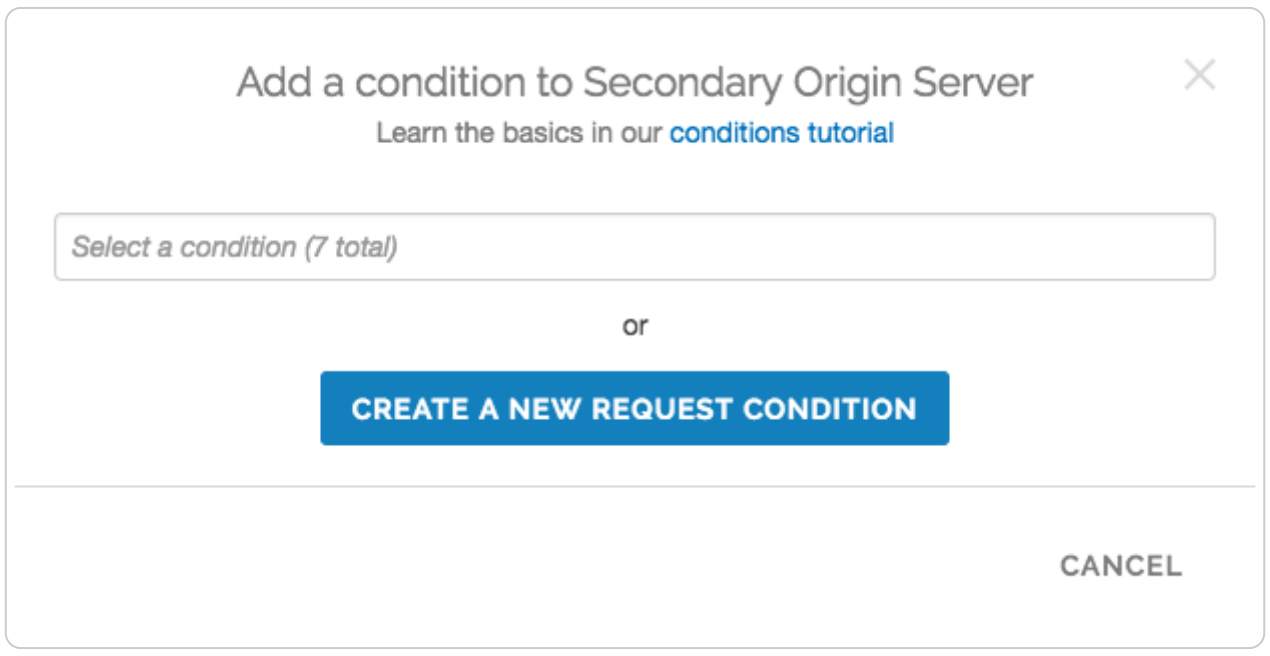
Using the user interface

Set up redundant origins via the user interface using these steps.

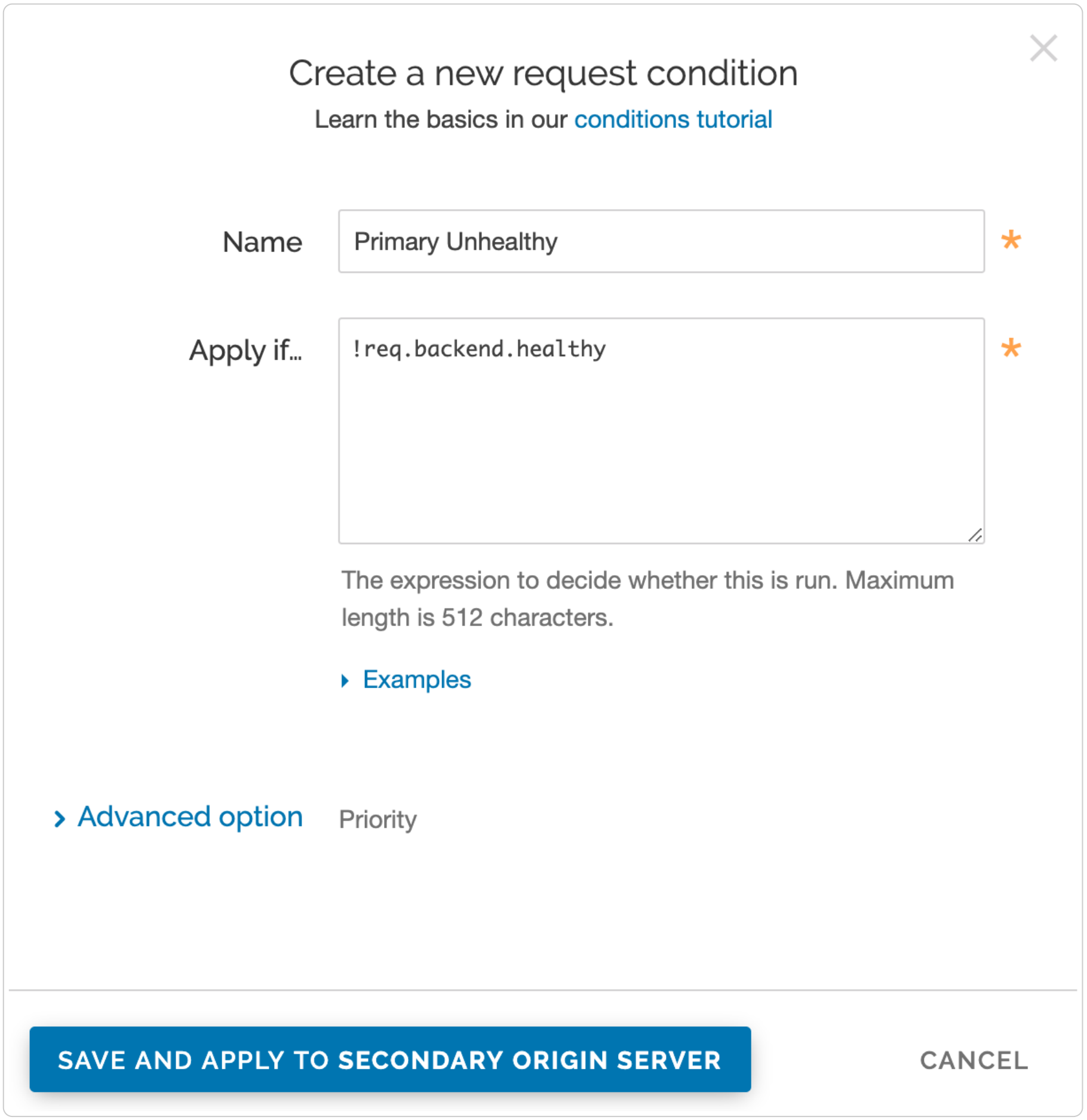
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.
5. In the **Health 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.



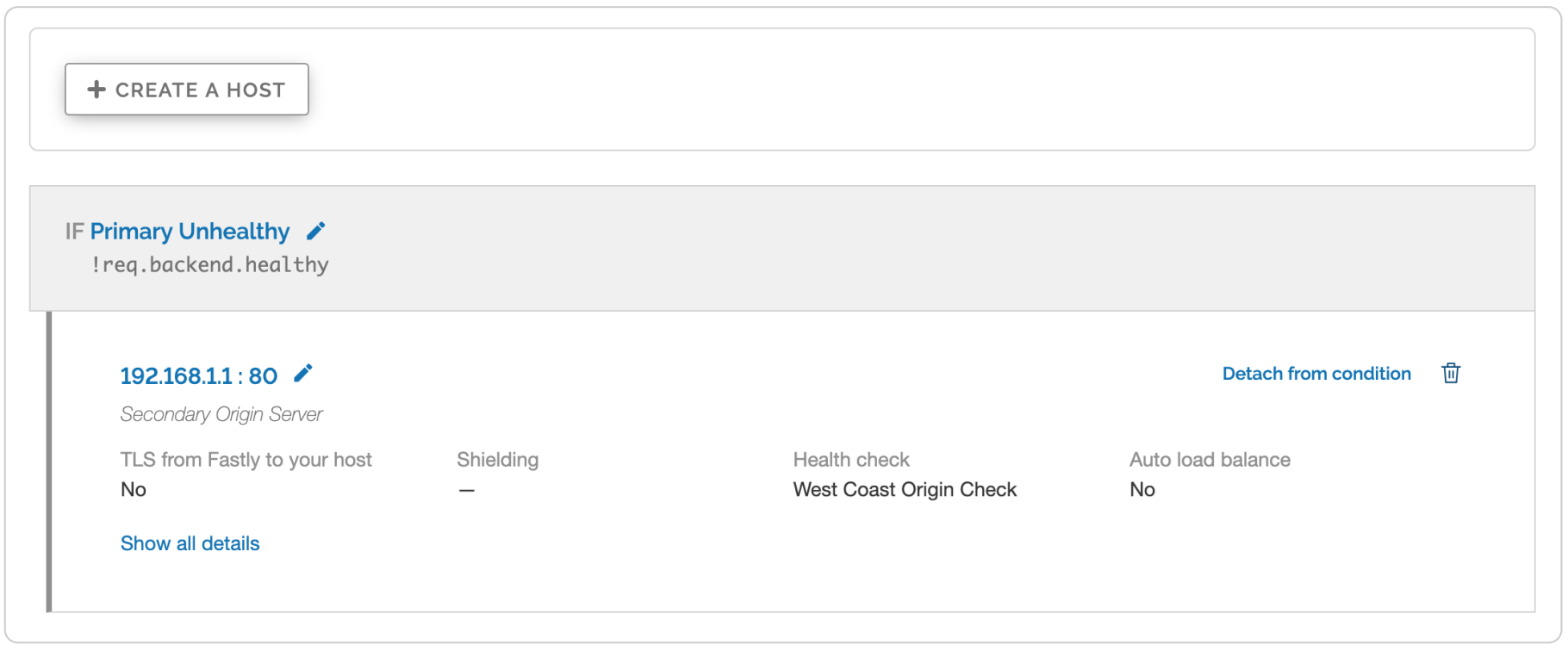
The Add a condition window appears.



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, type the name of your request condition (for example, `Primary Unhealthy`).
   - In the **Apply if** field, type `!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

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 `<your-bucket>.s3.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`.

Overriding a host at the host level

You can add an override a host at the host level. Once you've added a host, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Origins** link. The Origins page appears.

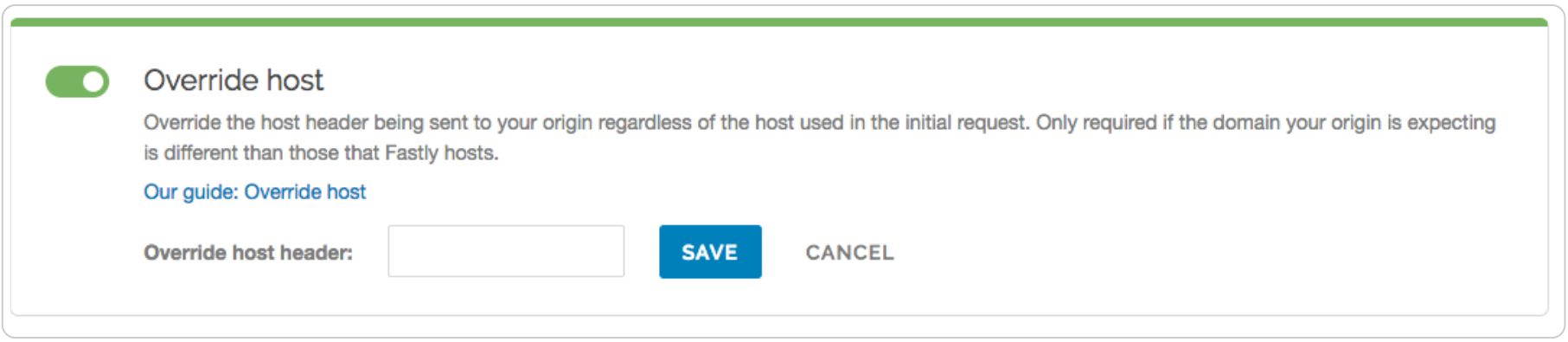
5. In the Hosts area, click the pencil icon next to the host you want to edit. The Edit this host page appears.
6. Towards the bottom of the **Edit this host** page, click the **Advanced options** link.
7. In the **Override host** field, type the hostname of your override host header based on the origin you're using. The value in this field will take precedence over anything you've set using the override host quick configuration. Keep in mind the following:
   - If you're using Amazon S3 as your origin, type `<yourbucket>.s3.amazonaws.com`.
   - If you're using Google Cloud Storage as your origin, type `<your bucket name>.storage.googleapis.com`.
8. Click the **Update** button.
9. 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type the hostname of your override host based on the origin you are using:
   - If you are using Amazon S3 as your origin, type `<yourbucket>.s3.amazonaws.com`.
   - If you are using Google Cloud Storage as your origin, type `<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.

Using Fastly with apex domains

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 files or streaming video.

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 Brazil, India, and South 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.

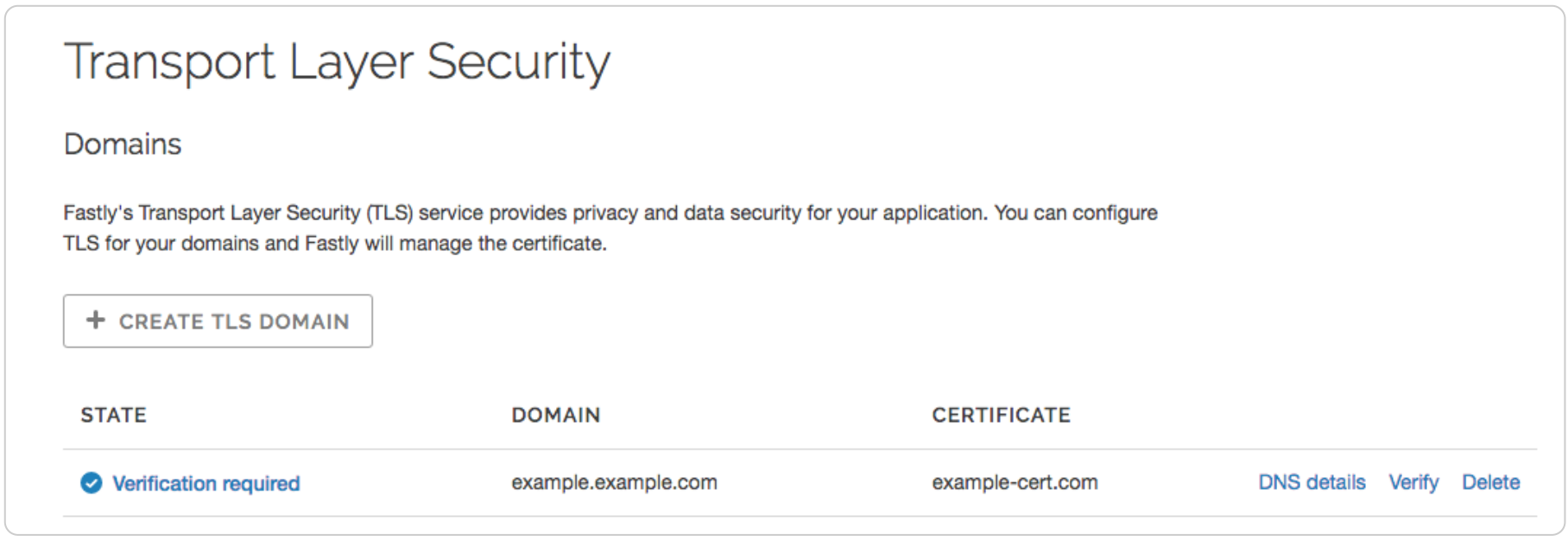
Finding anycast IP addresses

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.

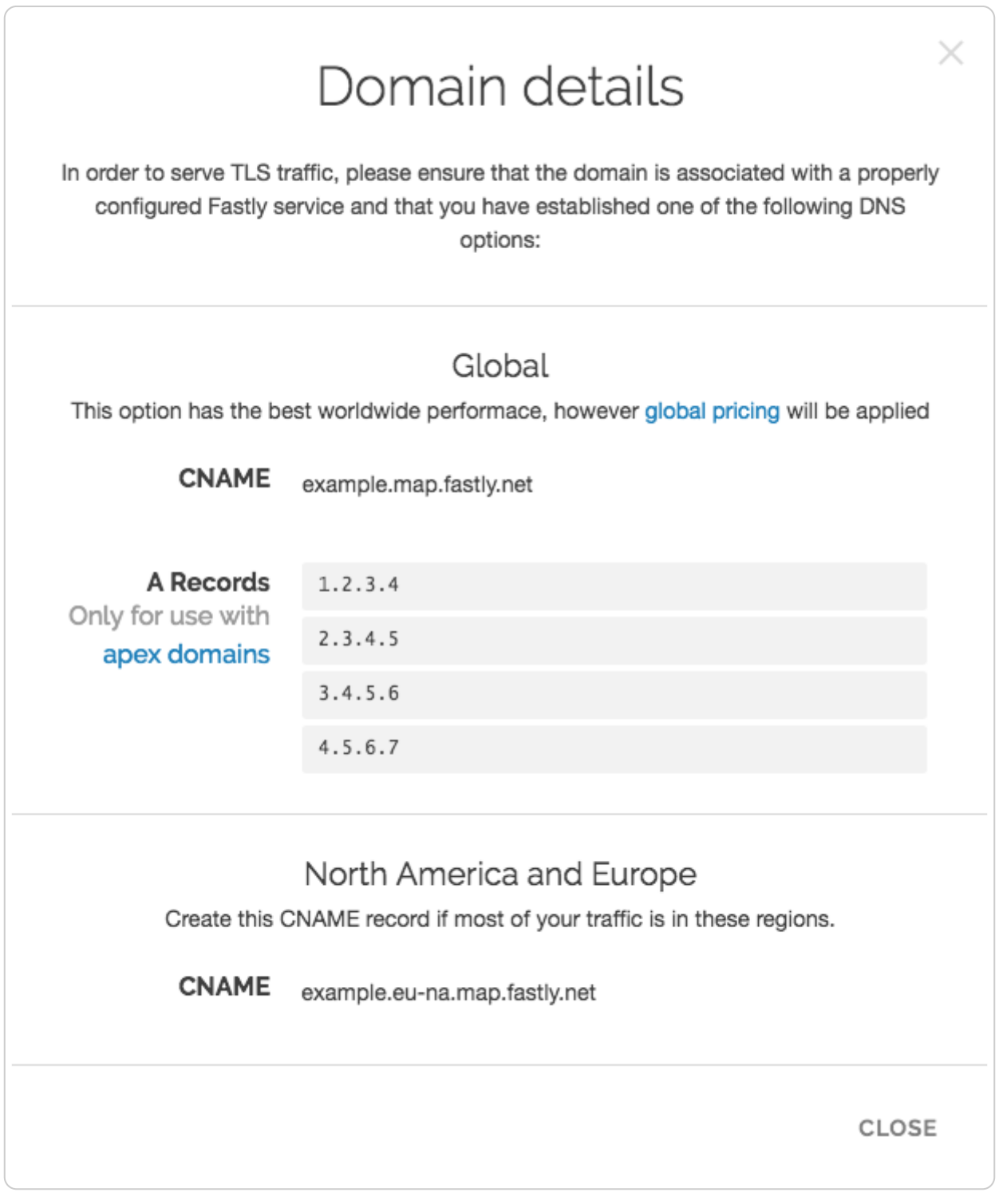
When you have TLS configured

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 **Account** link from the user menu. Your account information appears.
2. Click the **Transport Layer Security** link. The Transport Layer Security page appears.



3. Click the **DNS 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.

Request settings

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

https://docs.fastly.com/en/guides/configuration#_request-settings

Checking multiple backends for a single request

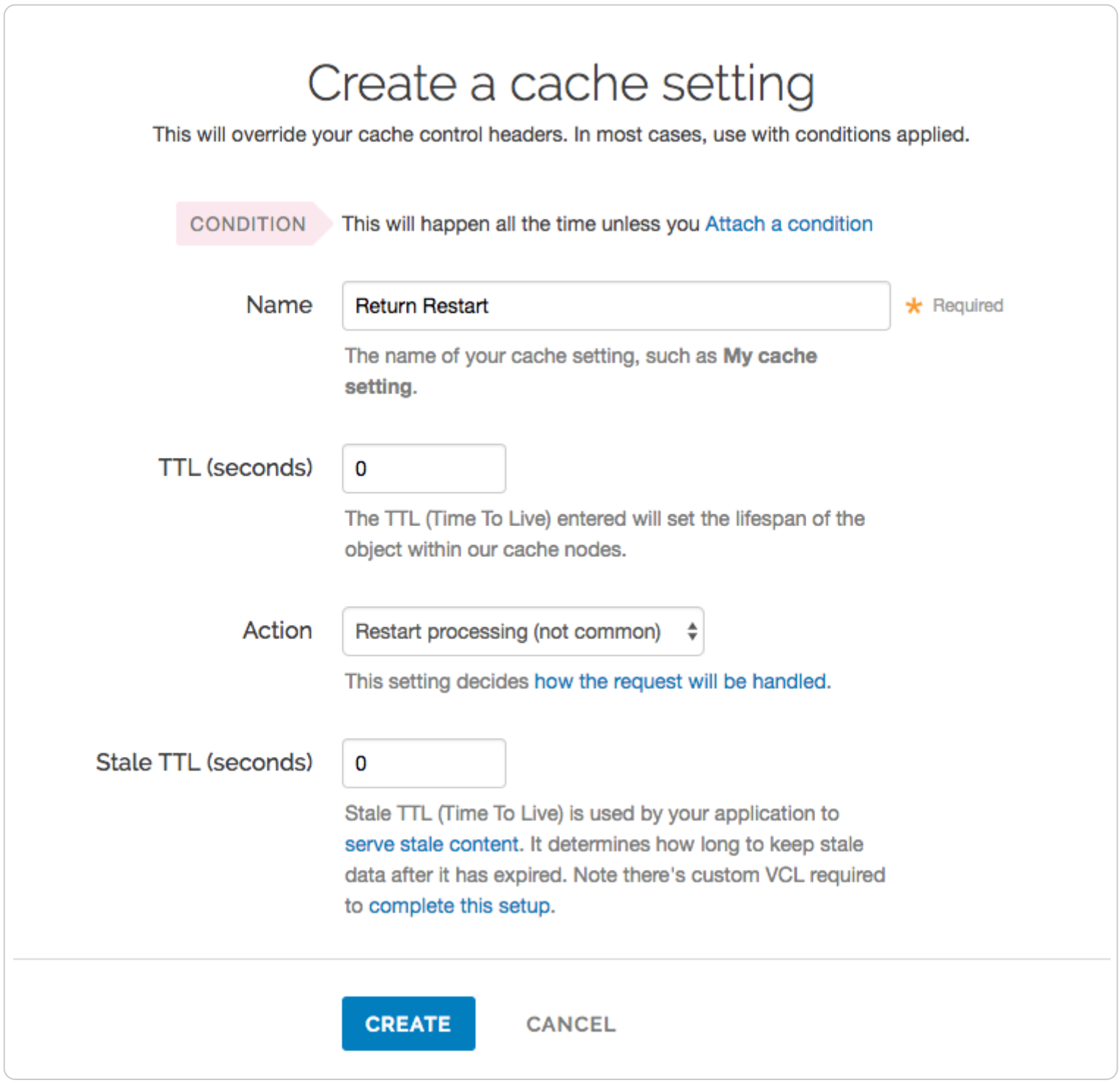
https://docs.fastly.com/en/guides/checking-multiple-backends-for-a-single-request

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.

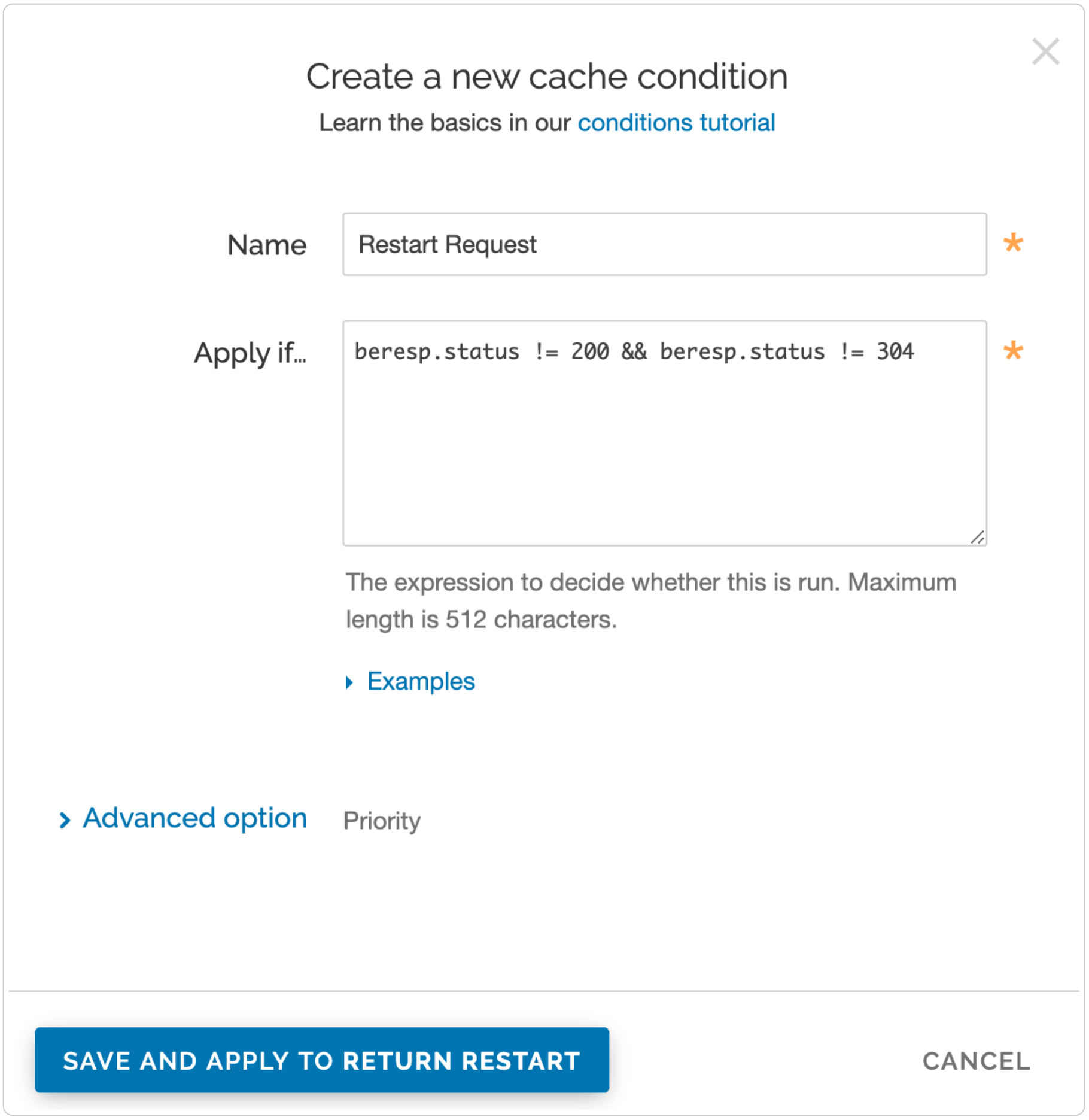
Create a new cache setting rule

Follow these steps to create a cache restart within vcl_fetch.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type `Return Restart` (or any meaningful, preferred name).
   - In the **TTL (seconds)** field, type `0`.
   - From the **Action** menu, select **Restart processing**.
   - In the **Stale TTL (seconds)** field, type `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.

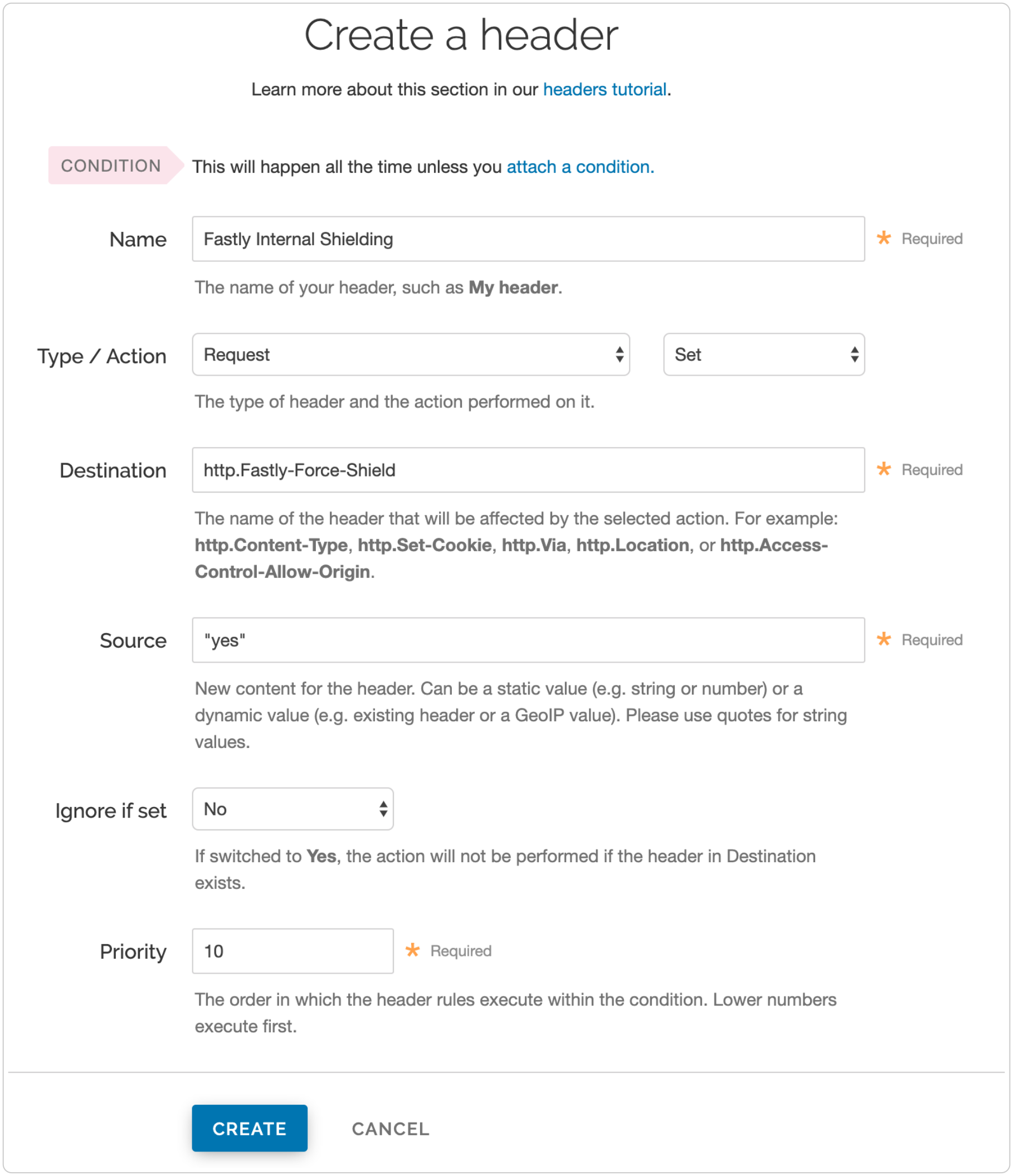


9. Fill out the **Create a new cache condition** fields as follows:
   - In the **Name** field, type `Restart Request` (or any meaningful, preferred name).
   - In the **Apply if** field, type `beresp.status != 200 && beresp.status != 304`.
10. Click the **Save and apply to** button to create the condition.

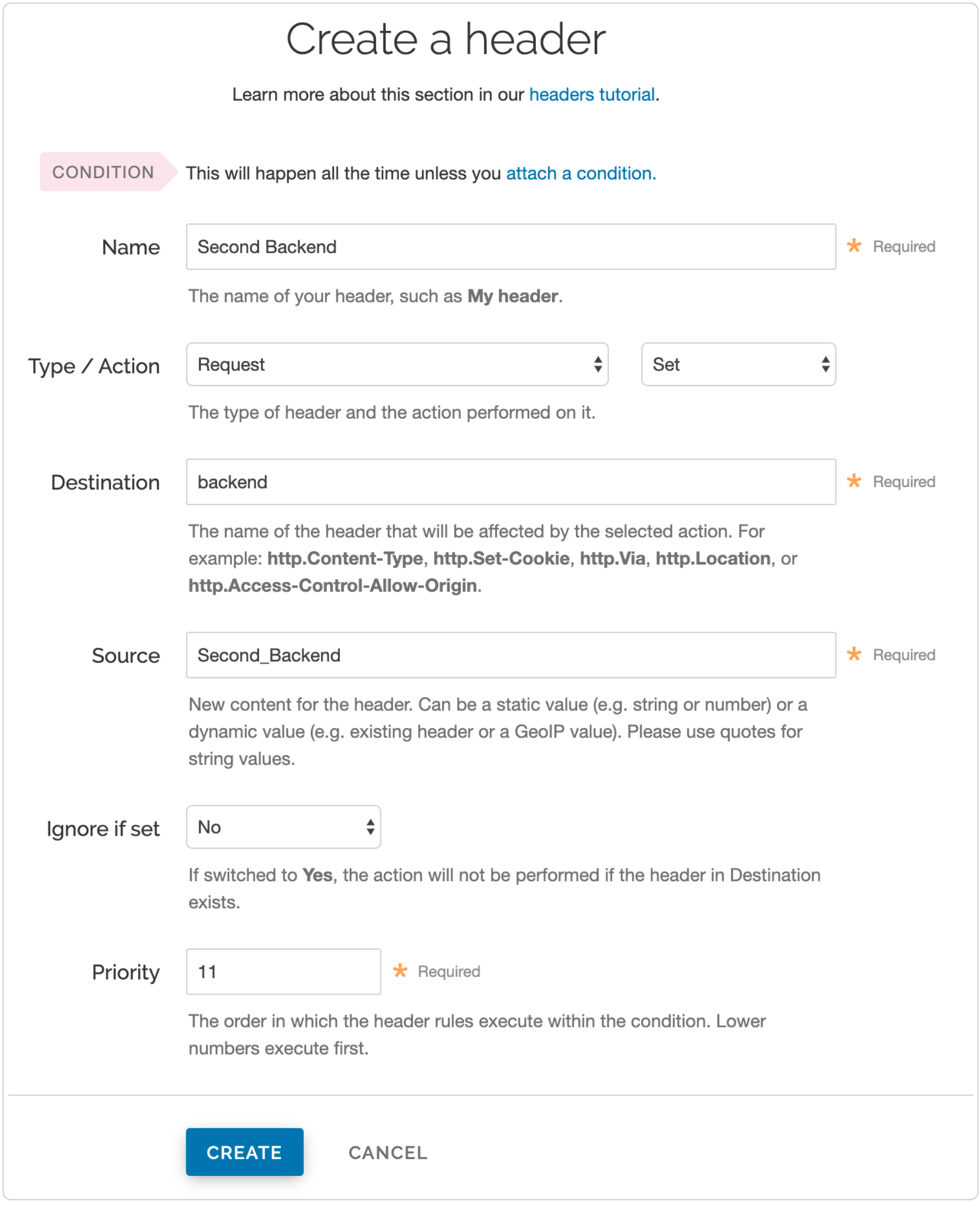
Create new request headers

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, type `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, type `http.Fastly-Force-Shield`.
   - In the **Source** field, type `"yes"`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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.

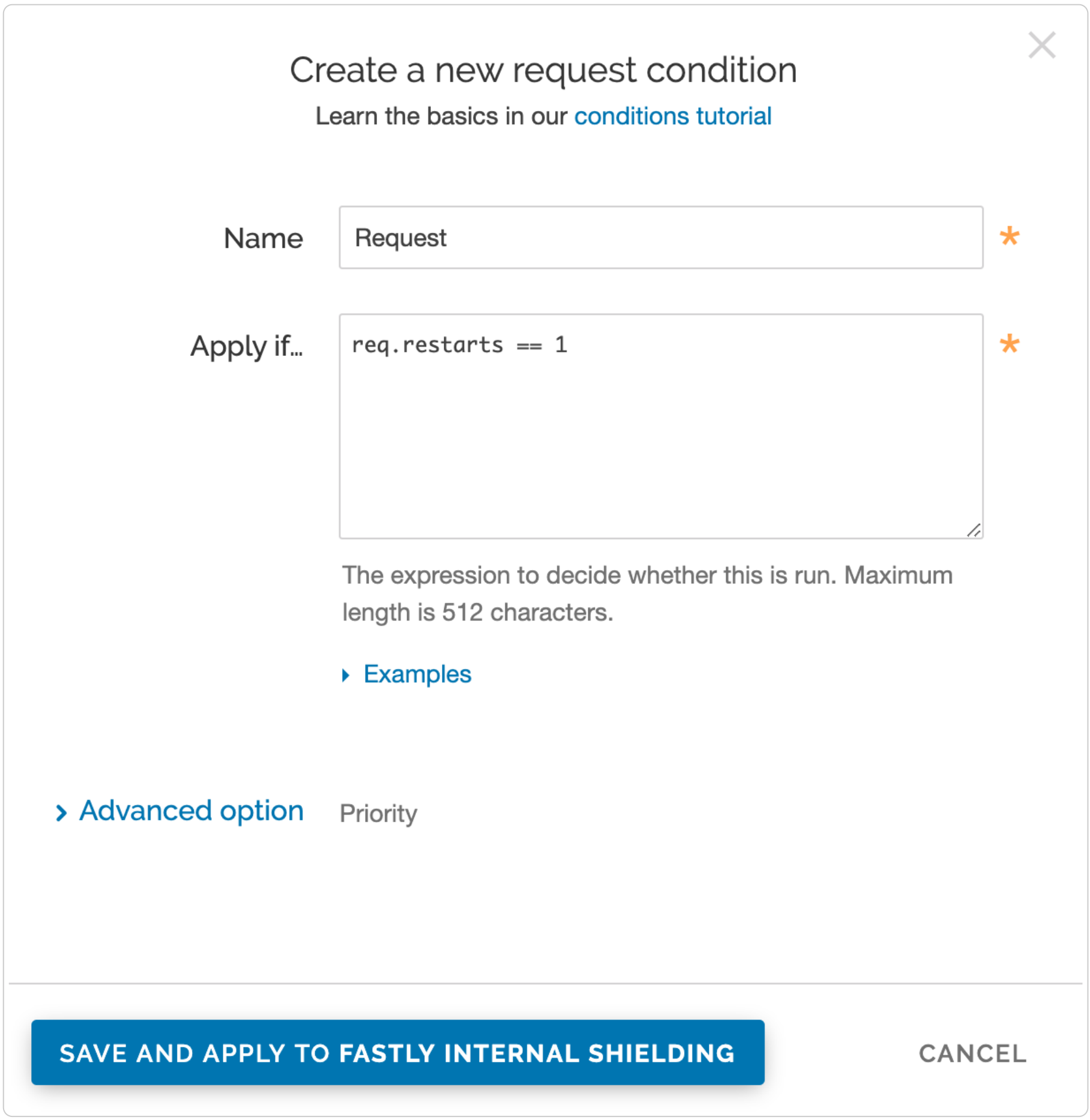


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Second Backend` (or any meaningful, preferred name).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `backend`.
   - In the **Source** field, type `Second_Backend` (this should match the name of your other backend).
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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, type `Req.request` (or any meaningful, preferred name).
   - In the **Apply if** field, type `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.

How request settings are applied

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

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.

Caching

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

Cache control tutorial

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

You are in full control of how Fastly caches your resources. The most preferred way of instructing Fastly is to use backend HTTP headers. The other way is to use the Varnish Configuration Language (VCL).

Backend HTTP headers

You can set four different types of HTTP headers which will have different effects on our caches and on web browsers. If you use more than one type, they are prioritized in the order listed below:

Surrogate-Control

Format:

```
Surrogate-Control: max-age=(time in seconds)
```

Example:

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

will cache something on our caches for one hour. This header gets stripped and is only visible to Fastly caches.

Cache-Control: s-maxage

Format:

```
Cache-Control: s-maxage=(time in seconds)
```

This is the same as `Surrogate-Control`, except the header is not stripped and will be respected by Fastly caches and any caches between Fastly and the browser, but not the browser itself.

Cache-Control: max-age

Format:

```
Cache-Control: max-age=(time in seconds)
```

This header will be respected by Fastly caches, any caches between Fastly and the browser, and the browser itself.

Expires

This header caches content until it expires as specified. It will be respected by Fastly caches, any caches between Fastly and the browser itself. See section 5.3 of RFC7234 for an explanation of the format.

Do not cache

If you want to ensure that a resource is not cached by Fastly, send the following HTTP header with the origin response:

```
Cache-Control: private
```

If you just set `max-age=0` or an `Expires` in the past, Fastly may still use a single response to satisfy multiple outstanding requests that arrive while waiting on the origin (see Request collapsing), or may cache the object in a stale form so that it can be used in case of errors or asynchronous revalidation (see Serving stale content).

The `private` directive does not prevent content from being cached in the browser. If you need to prevent caching by both Fastly *and* web browsers, we recommend combining the `private` directive with `max-age=0` or `no-store`. For example:

```
Cache-Control: private, no-store
```

Fastly does not currently respect `no-store` or `no-cache` directives. Including either or both of these in a `Cache-Control` header has no effect on Fastly's caching decision, unless you alter this behavior using custom VCL.

Applying different cache rules for Fastly and browsers

Say you want Fastly to cache your content but you don't want it cached by browsers. The best way to do this would be to send Fastly both the `Cache-Control` header as you want it to go to the browsers, and use `Surrogate-Control` to tell us how long to cache for. For example:

```
1  Cache-Control: no-store, must-revalidate
2  Surrogate-Control: max-age=3600
```

Except for when the `Cache-Control` header is set to `private`, the `Surrogate-Control` header takes priority over `Cache-Control`, but unlike `Cache-Control` it is stripped so the browsers don't see it.

Configuring popular web servers

Apache Config

If you are using Apache, the easiest way to add headers is to use the mod_expires module. For example, to cache GIF images for 75 minutes after the image was last accessed by the cache server, you would add a directive like this under the VirtualHost (or globally). For example:

```
1  ExpiresActive On
2  ExpiresByType image/gif "access plus 1 hours 15 minutes"
```

You can also cache whole URL subtrees. For example:

```
1  <Location "/css">
2    ExpiresActive On
3    ExpiresDefault "access plus 1 year"
4  </Location>
5
6  <Location "/static/">
7    ExpiresActive On
8    ExpiresDefault "access plus 1 day"
9  </Location>
```

NGINX Configuration

To configure NGINX, add the expires directive. For example:

```
1  location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
2     expires 1h;
3  }
```

Alternatively, if you need more flexibility in modifying headers you can try the HttpHeadersMore Module.

Amazon S3 configuration

By default, S3 doesn't have a facility for setting Cache-Control headers across multiple objects, so you will have to do this file-by-file using the S3Explorer, or in an automated fashion by using a cloud library like boto. Remember that you can combine long cache time with instant purges to enhance your performance.

> ★ **TIP:** While it's difficult to get S3 to set `Surrogate-Control`, you can set `x-amz-meta-surrogate-control` instead on origin and Fastly will honor that.

```
from boto.s3.connection import S3Connection

connection = S3Connection('aws access key', 'aws secret key')

buckets = connection.get_all_buckets()

for bucket in buckets:
    for key in bucket.list():
        print('%s' % key)

        if key.name.endswith('.jpg'):
            contentType = 'image/jpeg'
        elif key.name.endswith('.png'):
            contentType = 'image/png'
        else:
            continue

        key.metadata.update({
            'Content-Type': contentType,
            'Cache-Control': 'max-age=864000'
        })
        key.copy(
            key.bucket.name,
            key.name,
            key.metadata,
            preserve_acl=True
        )
```

> ❗ **IMPORTANT:** The above example provides an S3 configuration option for customers with small- to medium-sized buckets. However, it iterates over every object in those buckets. If you have millions of objects this may not be the right approach. For millions of objects, we recommend using VCL. Be sure to contact us for assistance.

Custom Headers in Programming Languages and Frameworks

PHP

More information: https://php.net/manual/en/function.header.php

Example: add this to your PHP code before you send any output to cache certain HTML for an hour

```
header('Cache-Control: max-age=3600');
```

Django

More information: https://docs.djangoproject.com/en/dev/ref/request-response/#setting-headers

Example:

```
response = HttpResponse()
response['Cache-Control'] = 'max-age=3600'
```

Sinatra

More information: http://sinatrarb.com/documentation.html

Example:

```
get '/' do
    headers['Cache-Control'] = 'max-age=3600'
end
```

> ★ **TIP:** Expiration times in these examples are provided for guidance only. You can use longer expirations coupled with our purging API to make your site faster and your backend less loaded.

Caching configuration best practices

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.

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. When no cache related header exists for an object, you can specifically set a fallback TTL (sometimes called a "default TTL").

TIP: 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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 guide to changing caching times on backend headers describes this in more detail.

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

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 Cache Control Tutorial. 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 Configuration 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:

```
1  Surrogate-Control: max-age=31557600
2  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).

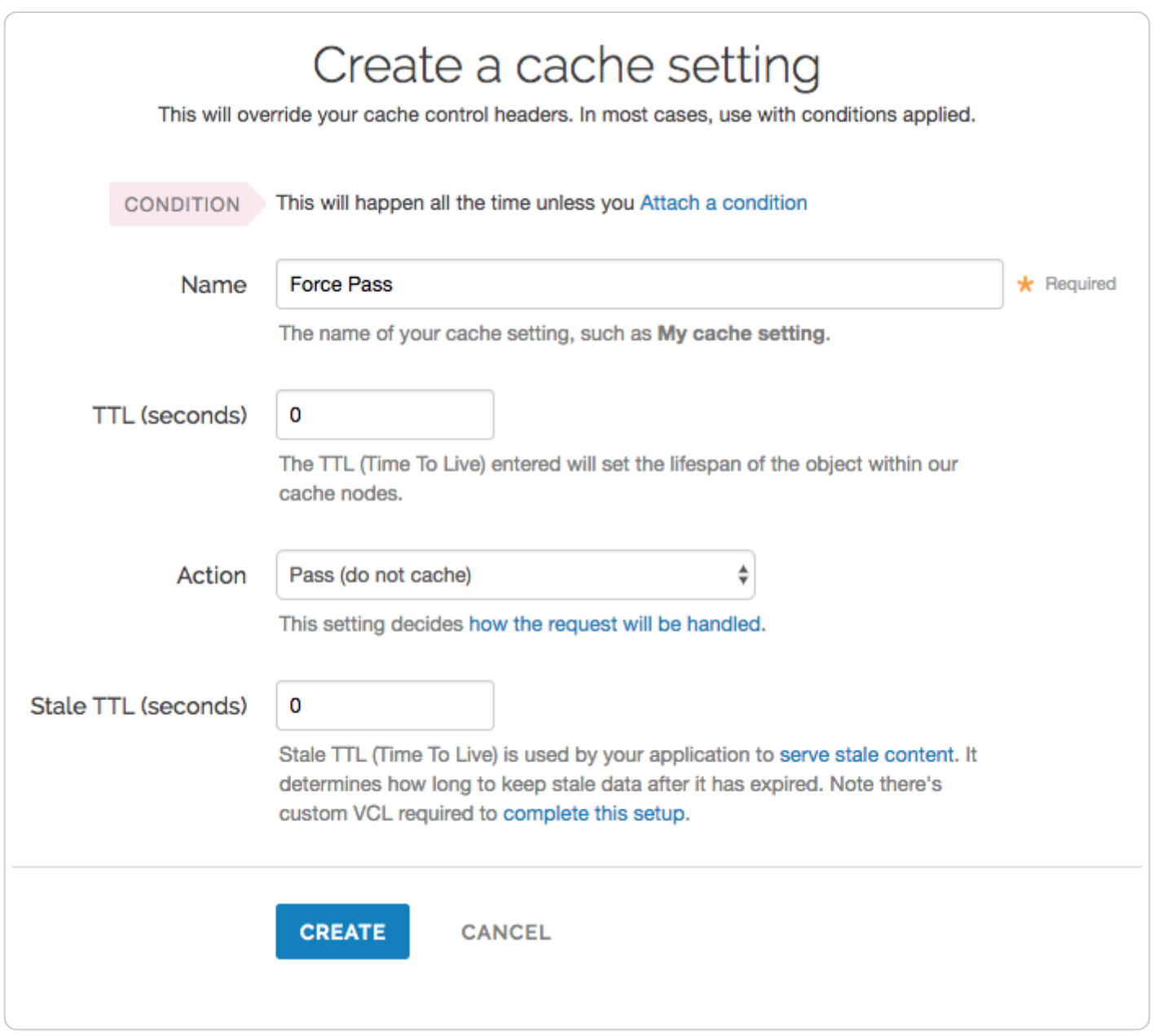
For Surrogate-Control, Fastly supports the `max-age`, `stale-if-error`, and `stale-while-revalidate` parameters.

For more information about controlling caching, see our Cache Control Tutorial.

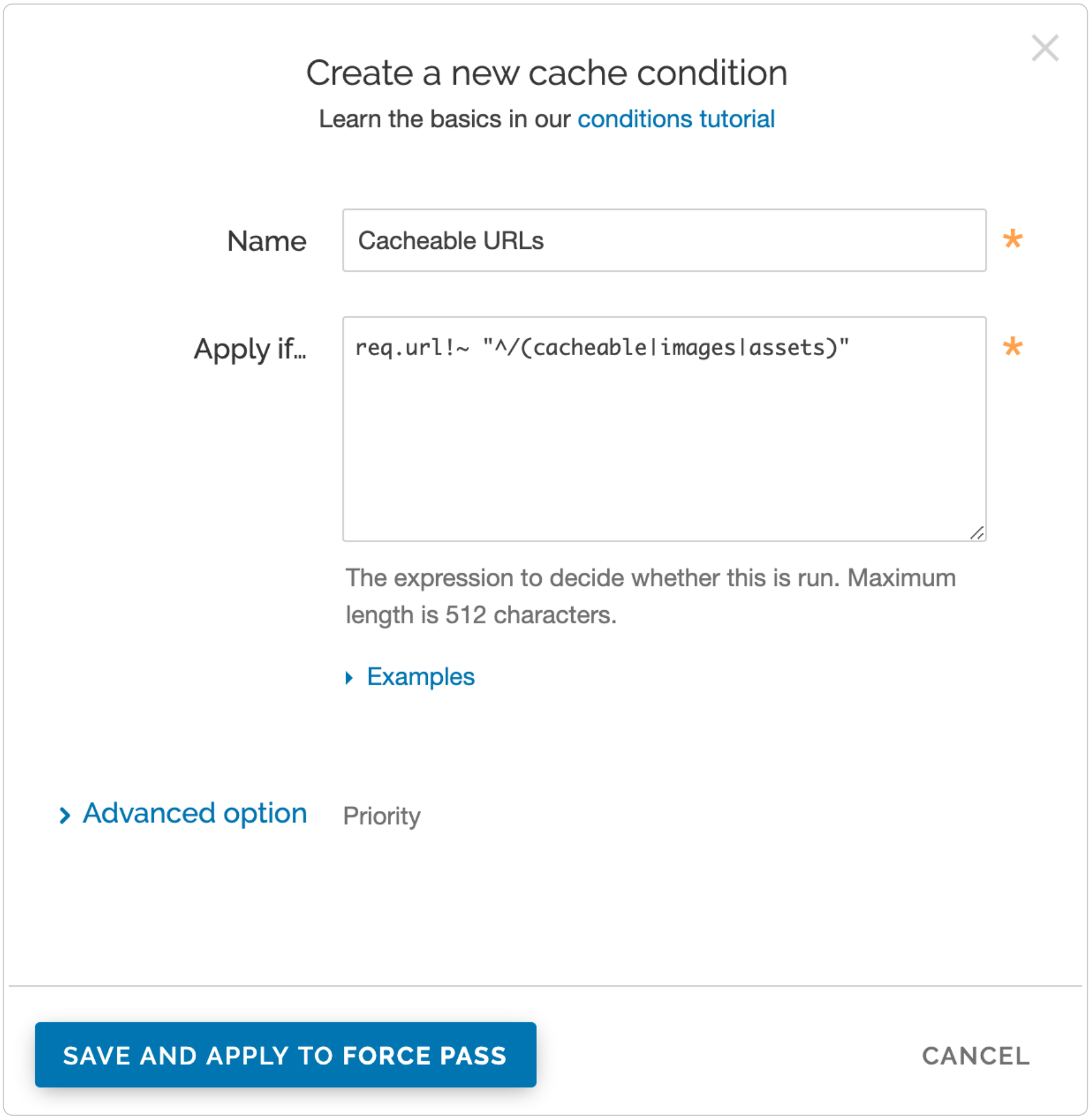
Conditionally preventing pages from caching

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

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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. Create a new cache setting and then click the **Create** button. The new cache setting you created appears on the Settings page.
7. Click the **Attach a condition** link to the right of the newly created cache setting. The Create a new cache condition window appears.



8. In the **Apply if** field, create a condition that matches the URLs you want.

   In this example, we created `req.url!~ "^/(cacheable|images|assets)"` to set the condition to look for URLs containing `/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.
9. Click the **Save and apply to** button.
10. Click the **Activate** button to deploy your configuration changes.

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

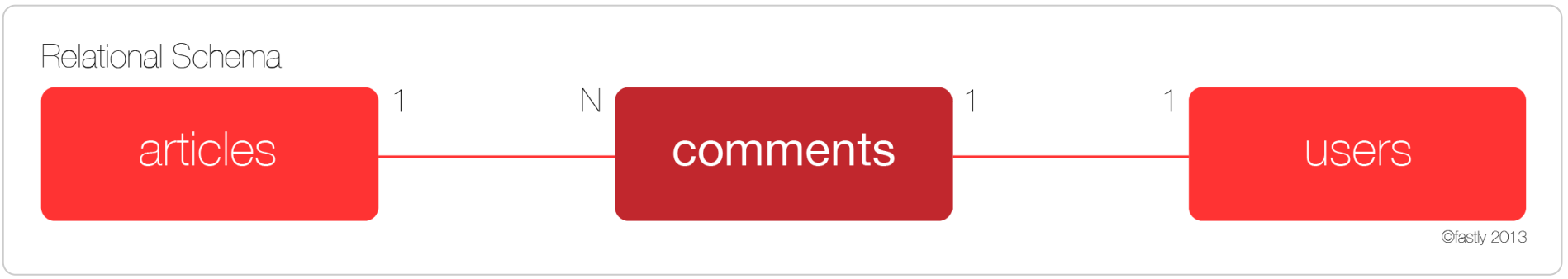
Enabling API caching

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 authored 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

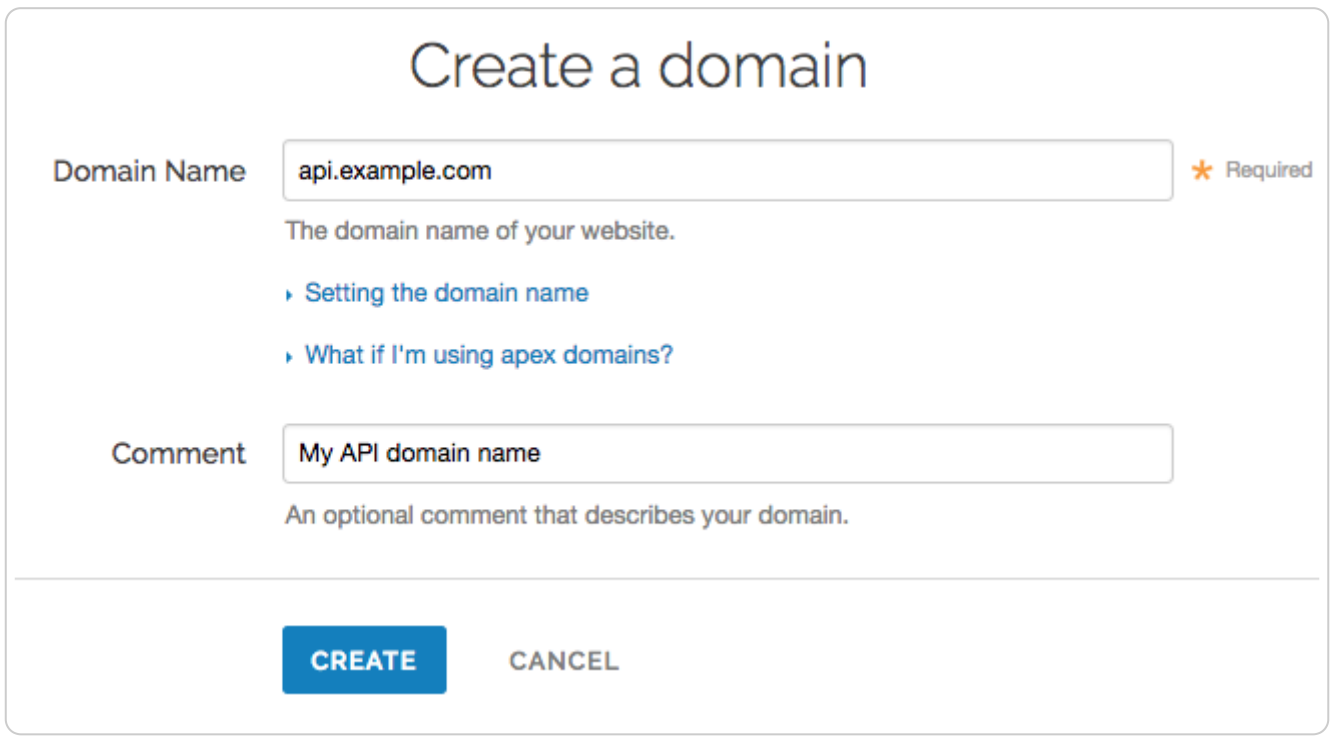
Follow the instructions for creating a new service. You'll add specific details about your API server when you fill out the **Create a new service** fields:

- In the **Name** field, type 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, type the domain name associated with your API (e.g., `api.example.com`).
- In the **Address** field, type 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 **Configure** page, click the **Configuration** button and then select **Clone active**. 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, type the domain name for the API.
   - In the **Comment** field, type 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 connecting to origins. You'll add specific details about your API server when you fill out the **Create a host** fields:

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

> **Implementing API cache control**
>
> 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 cache control tutorial.

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:

```
1  Surrogate-Control: max-age=86400
2  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.

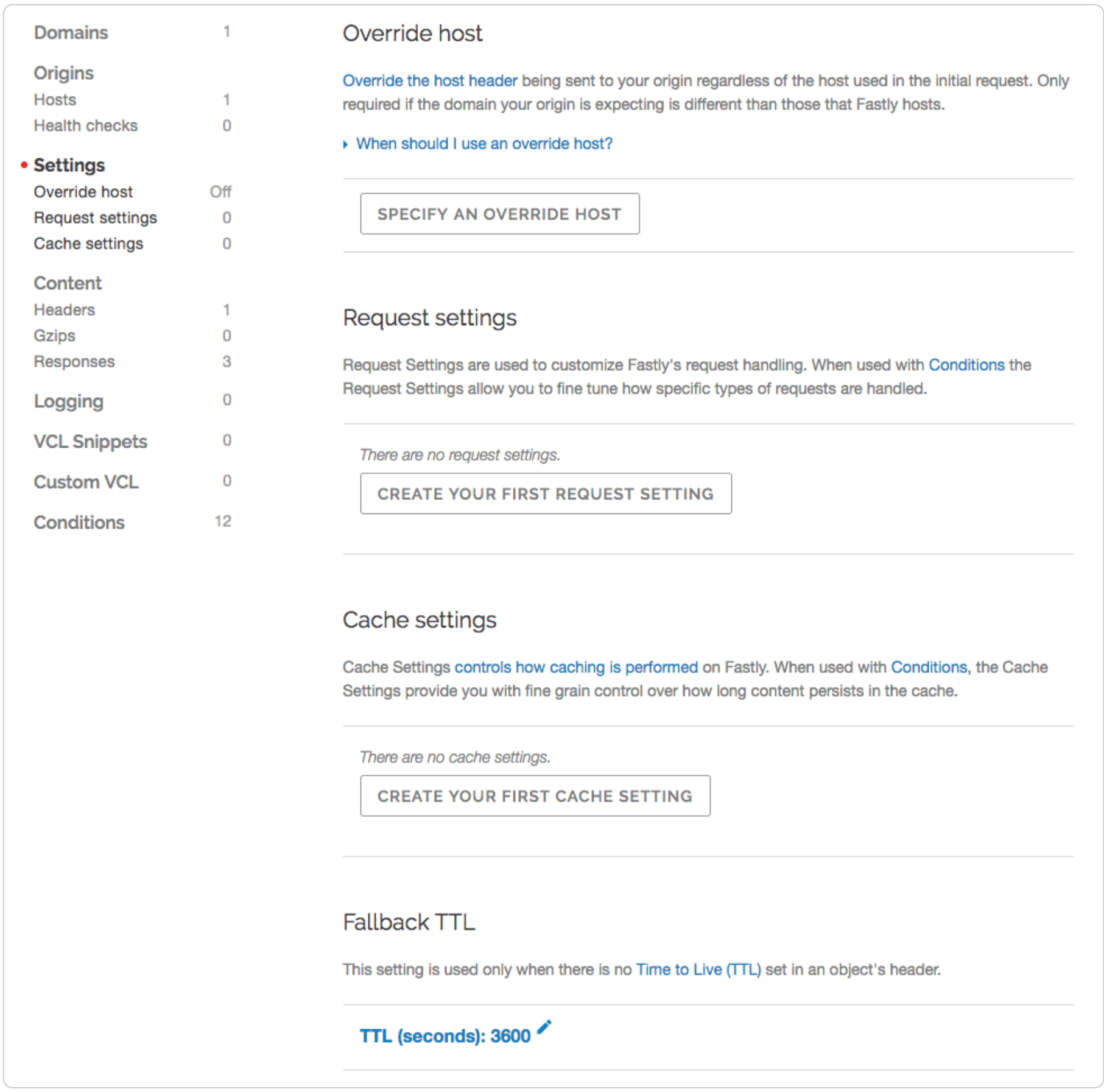
Overriding caching defaults based on a backend response

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

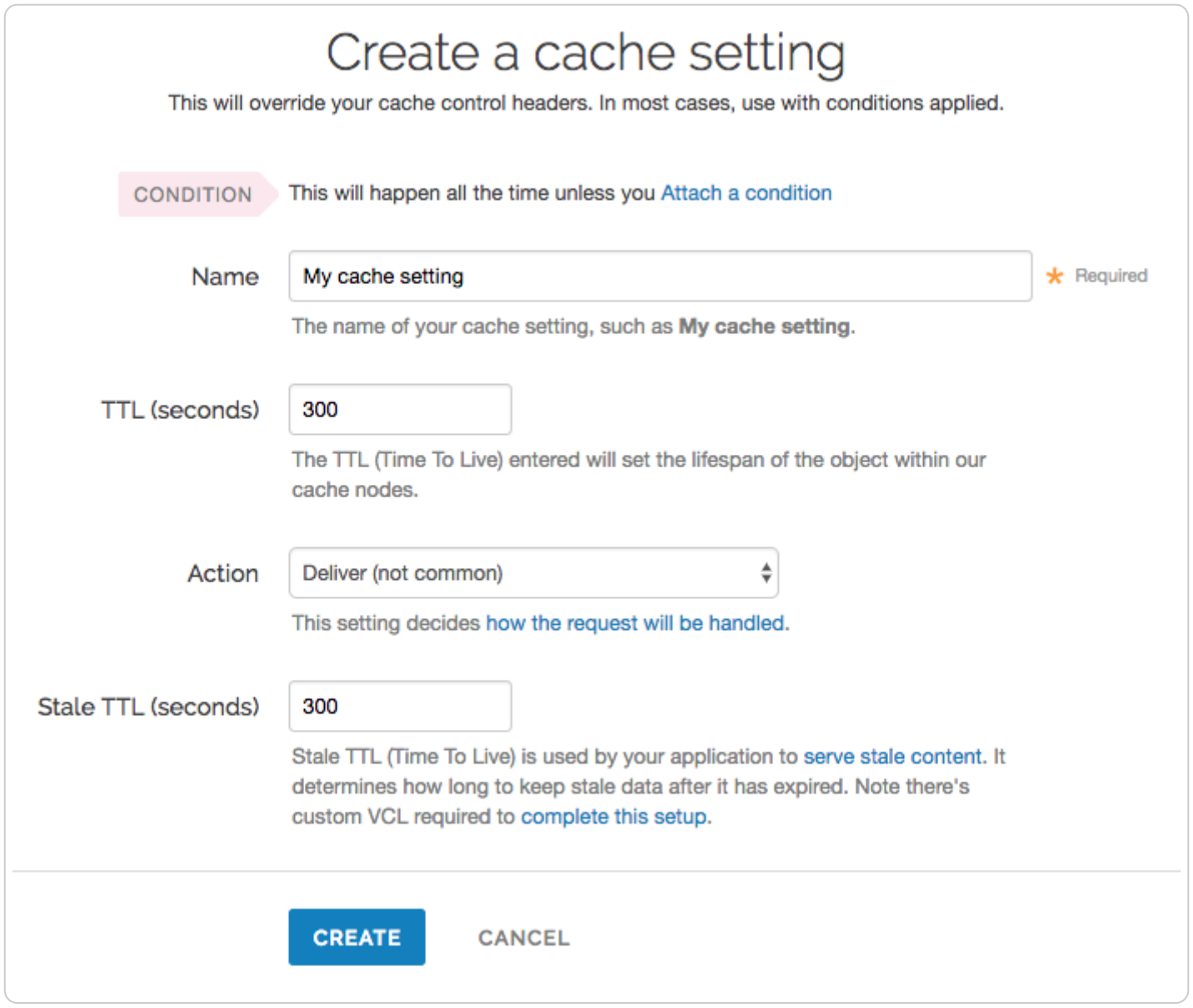
In certain situations you may want to conditionally apply a different caching policy based on a backend response. In this particular case we have backend that on occasion returns 404 errors (e.g., document not found). We don't want those responses to be cached for full caching period of a day but only 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

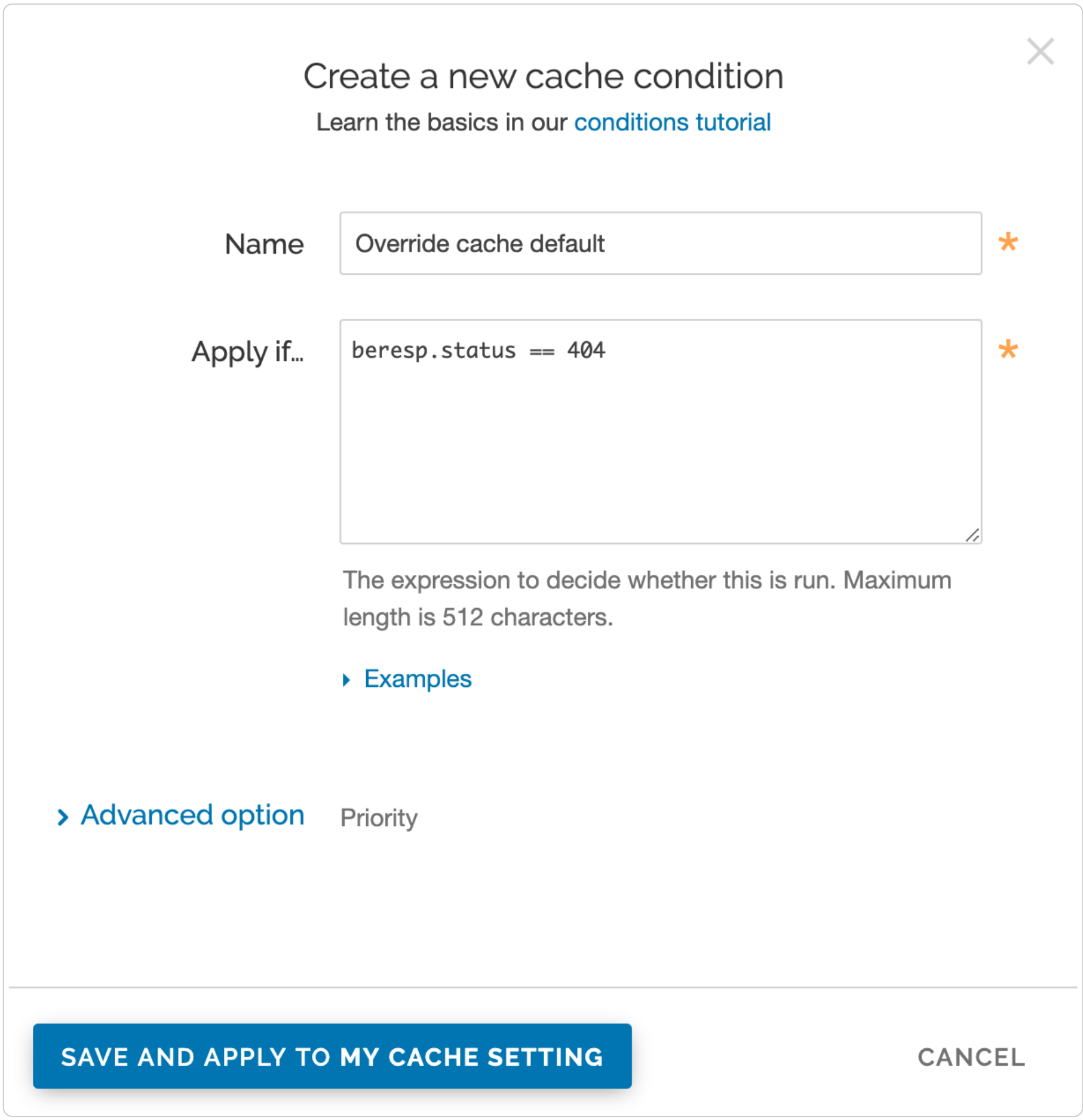


6. Fill out the **Create a cache setting** fields as follows:
   - In the **Name** field, type a descriptive name for the new cache settings.
   - In the **TTL (seconds)** field, type the amount of time, in seconds, to cache the objects (e.g., `300`).
   - From the **Action** menu, select **Deliver**.
   - In the **Stale TTL (seconds)** field, type 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, type a descriptive name for the new condition. For example, `Override cache default`.
   - In the **Apply if** field, type 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.

Segmented Caching

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

Fastly's Segmented Caching feature allows you to cache files of any size. Segmented Caching works by breaking files into smaller 1 MB segments in Fastly's cache then recombining or splitting these objects 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 files of any size. If Segmented Caching is not enabled, requests for files over 2 GB without Streaming Miss or 5 GB with Streaming Miss will result in errors.

How Segmented Caching works

When an end user makes a range request for a file 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 file that have been requested by the end user (along with rounding based on object size) will be cached rather than the entire file. Partial cache hits will result in having the cached portion served from cache and the missing pieces fetched from origin. (Requests for an entire file would be treated as a byte range request from 0 to end of file.)

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 `fastly.segmented_caching.is_inner_req` and requests from end users, also called "outer requests," will have a `true` value for `fastly.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 file 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:

- **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.
- **HTTP headers `If-Modified-Since` and `If-None-Match` are not supported.** Fastly only returns HTTP 200 OK and HTTP 206 Partial Content success status response codes.
- **URL purges must be authenticated.** Segmented caching allows you to purge all range objects for the file 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 file size.** You must explicitly enable Segmented Caching using a conditional statement in your VCL (e.g., based on file extension or path). Our instructions below contain an example of how to enable the Segmented Caching feature based on extension.
- **Files cached prior to enabling Segmented Caching are not used.** If you are enabling Segmented Caching on an existing service, whole files 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 file 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 files should use Segmented Caching.

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

2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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 files 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.

Headers

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

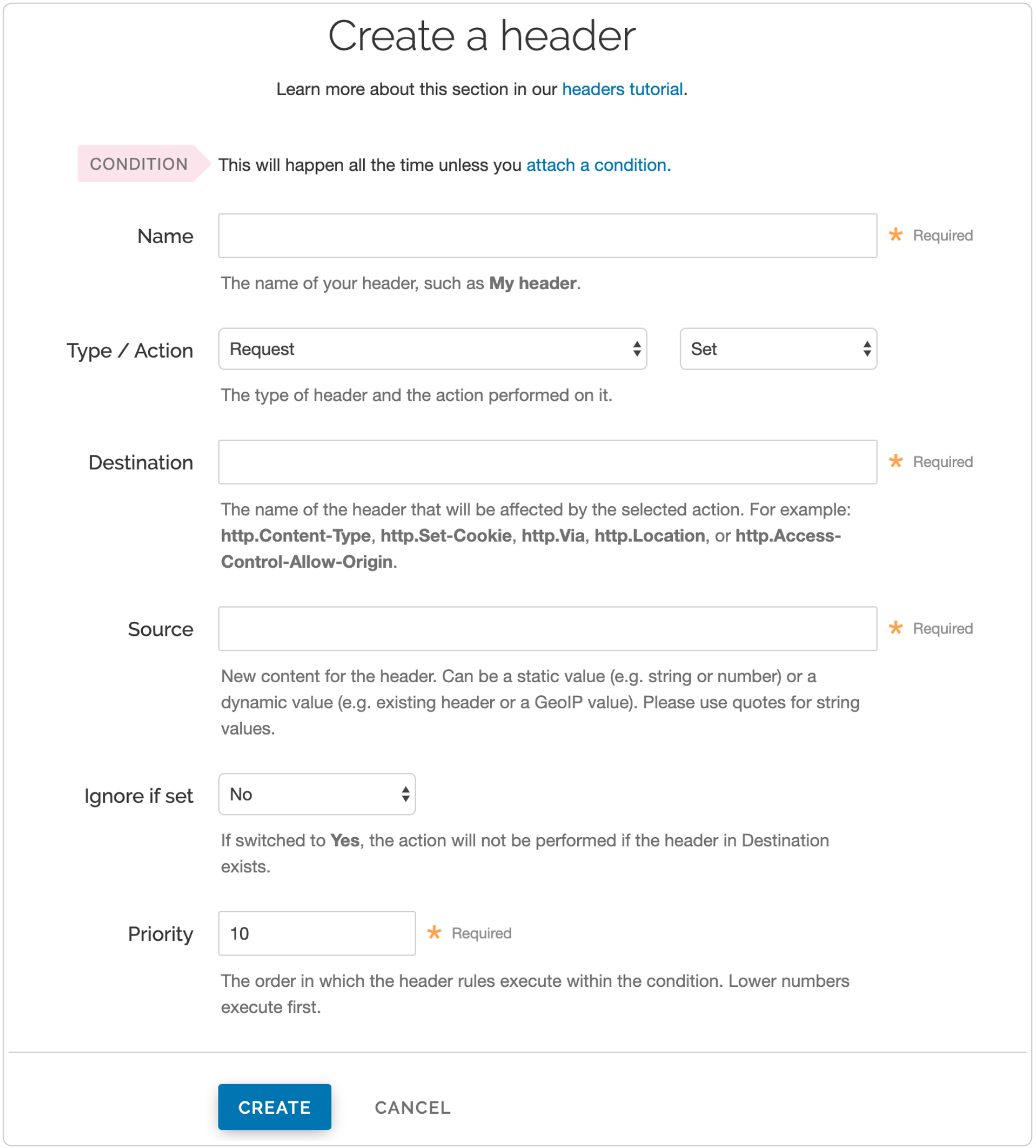
Adding or modifying headers on HTTP requests and responses

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.



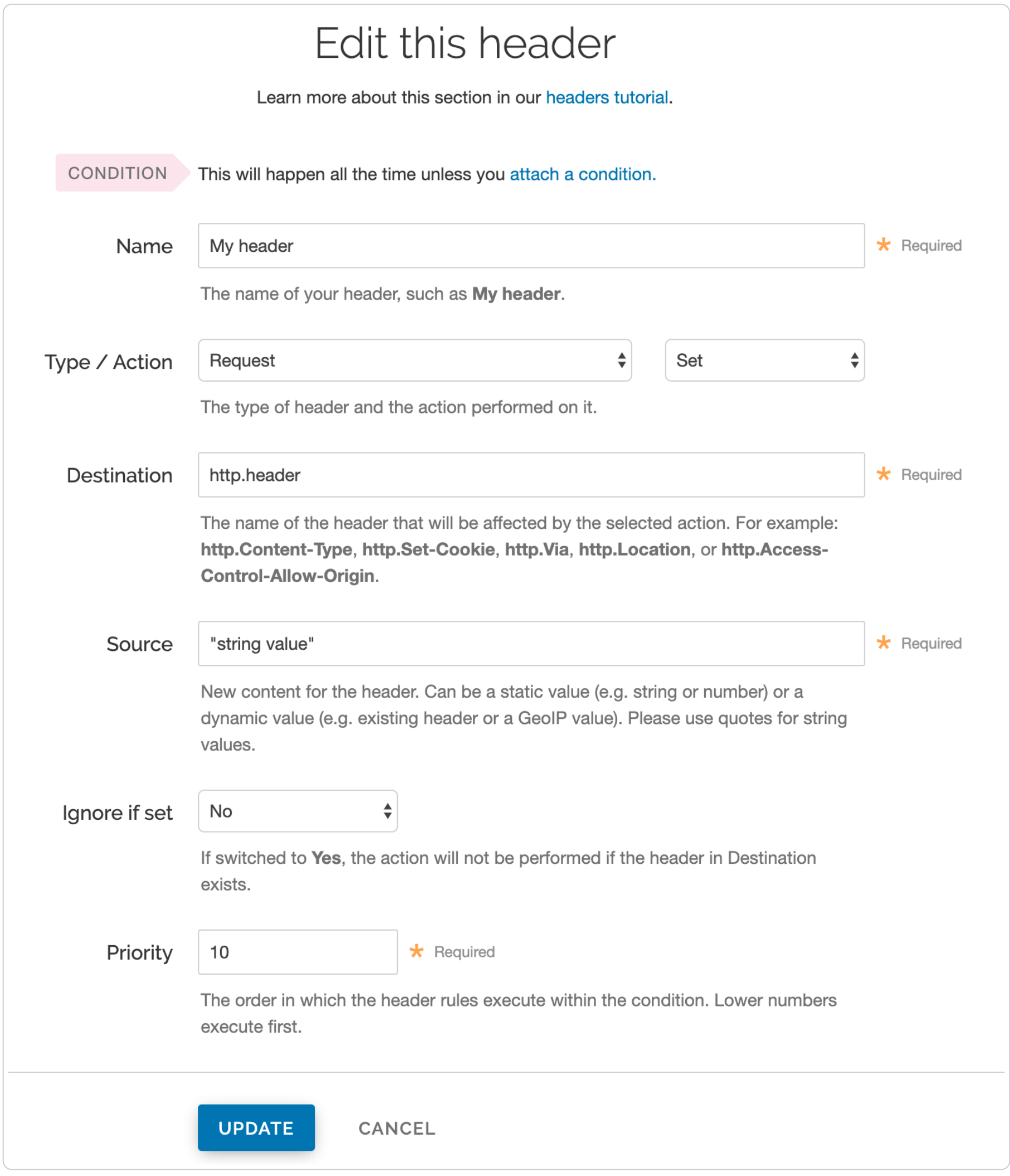
6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type the name of your header rule (for example, `My header`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type the name of the header affected by the selected action.
   - In the **Source** field, type 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, type 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type the name of the header affected by the selected action.
   - In the **Source** field, type 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, type 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.

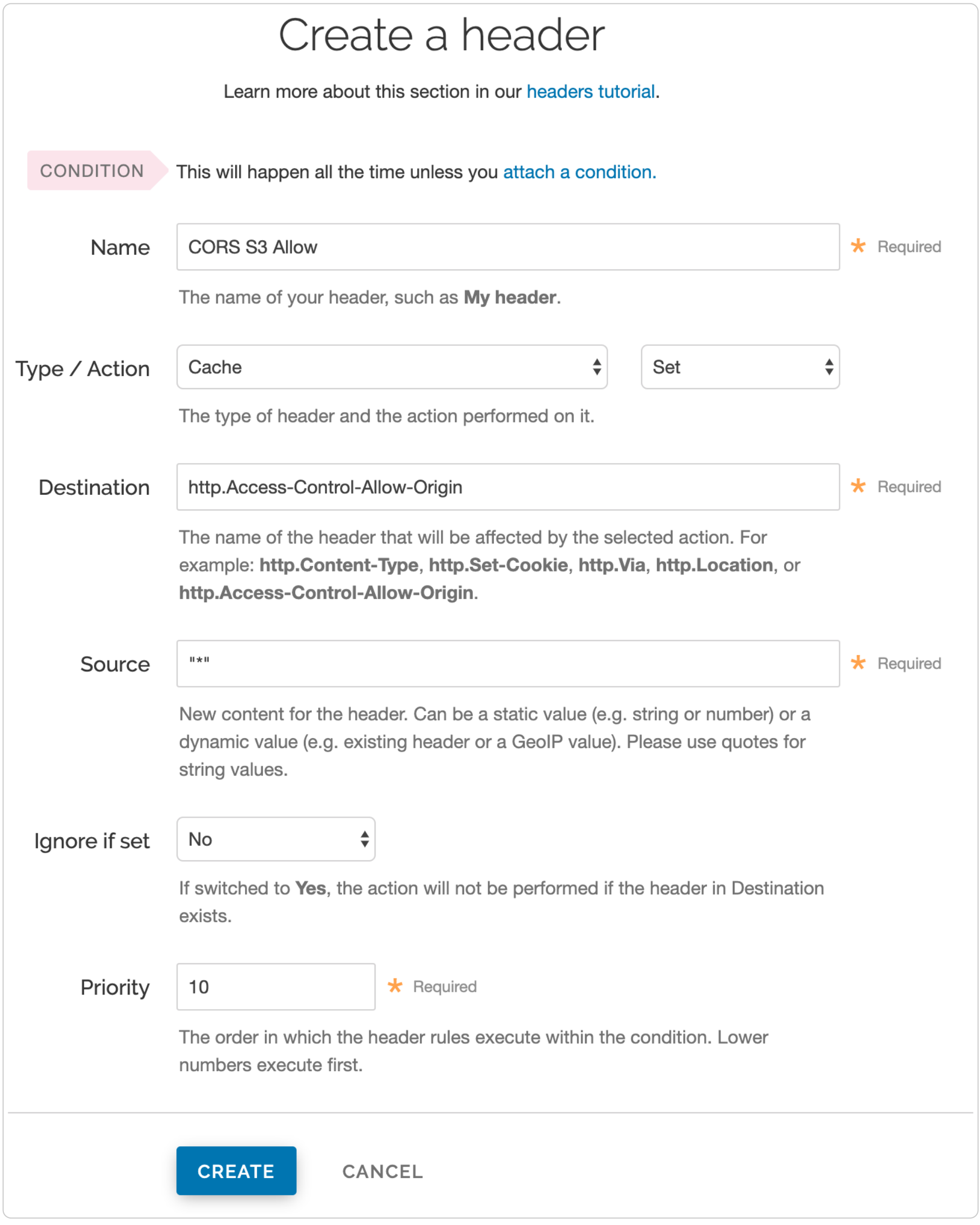
Name	Valid Types	Description
`server.identity`	Request, Cache, Response	A unique identifier for the Fastly server processing the request.
`server.region`	Request, Cache, Response	The region in which the Fastly server resides.
`server.datacenter`	Request, Cache, Response	The datacenter in which the Fastly 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 (see our geolocation article for more information).

Enabling cross-origin resource sharing (CORS)

https://docs.fastly.com/en/guides/enabling-cross-origin-resource-sharing

We recommend enabling CORS (Cross-Origin Resource Sharing) when using Amazon S3 as your backend server. 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type `http.Access-Control-Allow-Origin`.
   - In the **Source** field, type `"*"`.
   - 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:

```
Access-Control-Allow-Origin: http://example.tld
Access-Control-Allow-Methods: GET
Access-Control-Expose-Headers: Content-Length, Connection, Date...
```

Removing headers from backend response

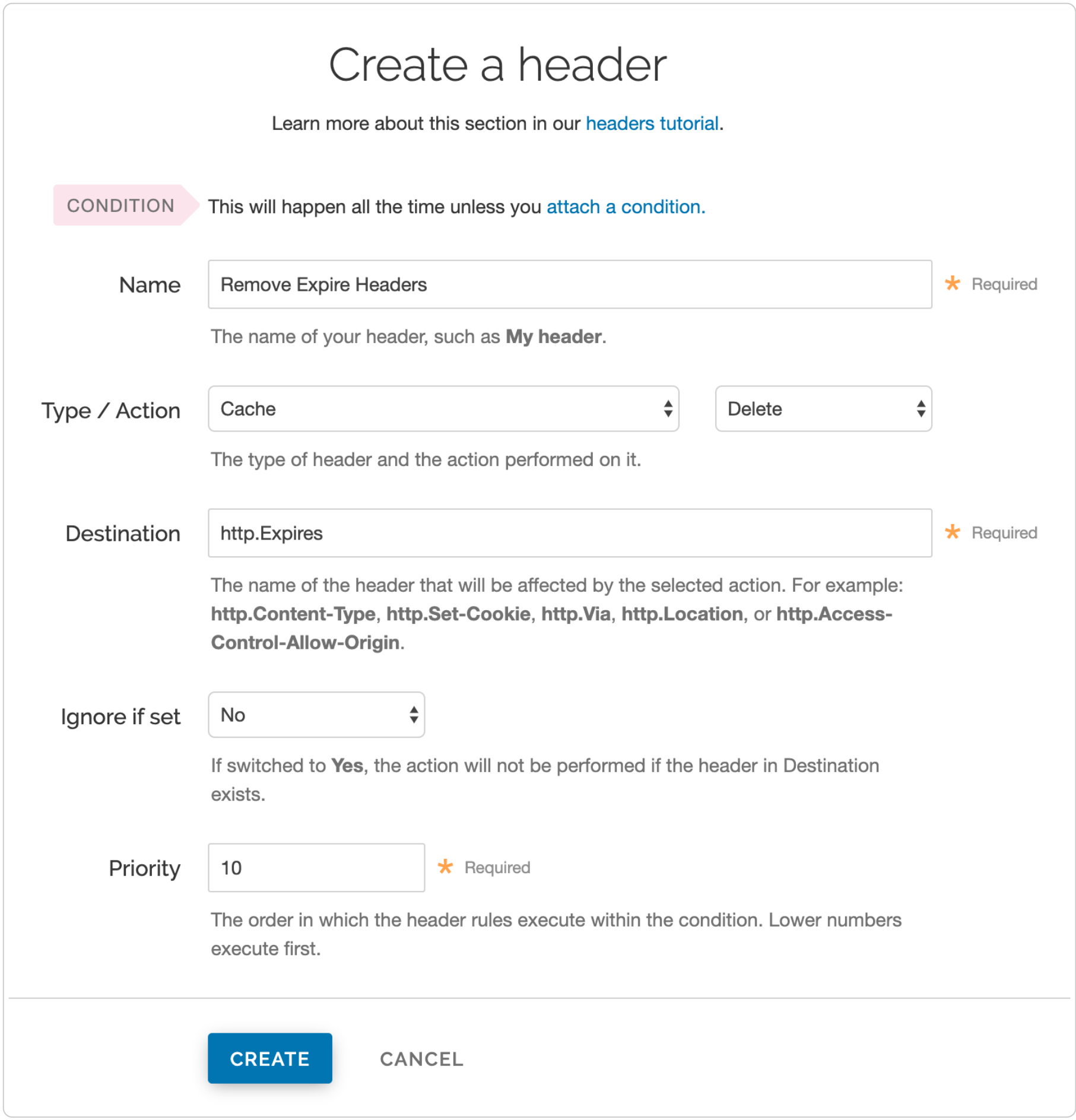
https://docs.fastly.com/en/guides/removing-headers-from-backend-response

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:

```
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Last-Modified: Wed, 18 Jul 2012 18:52:16 +0000
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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type the name of the header (e.g., `http.Expires`).
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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.

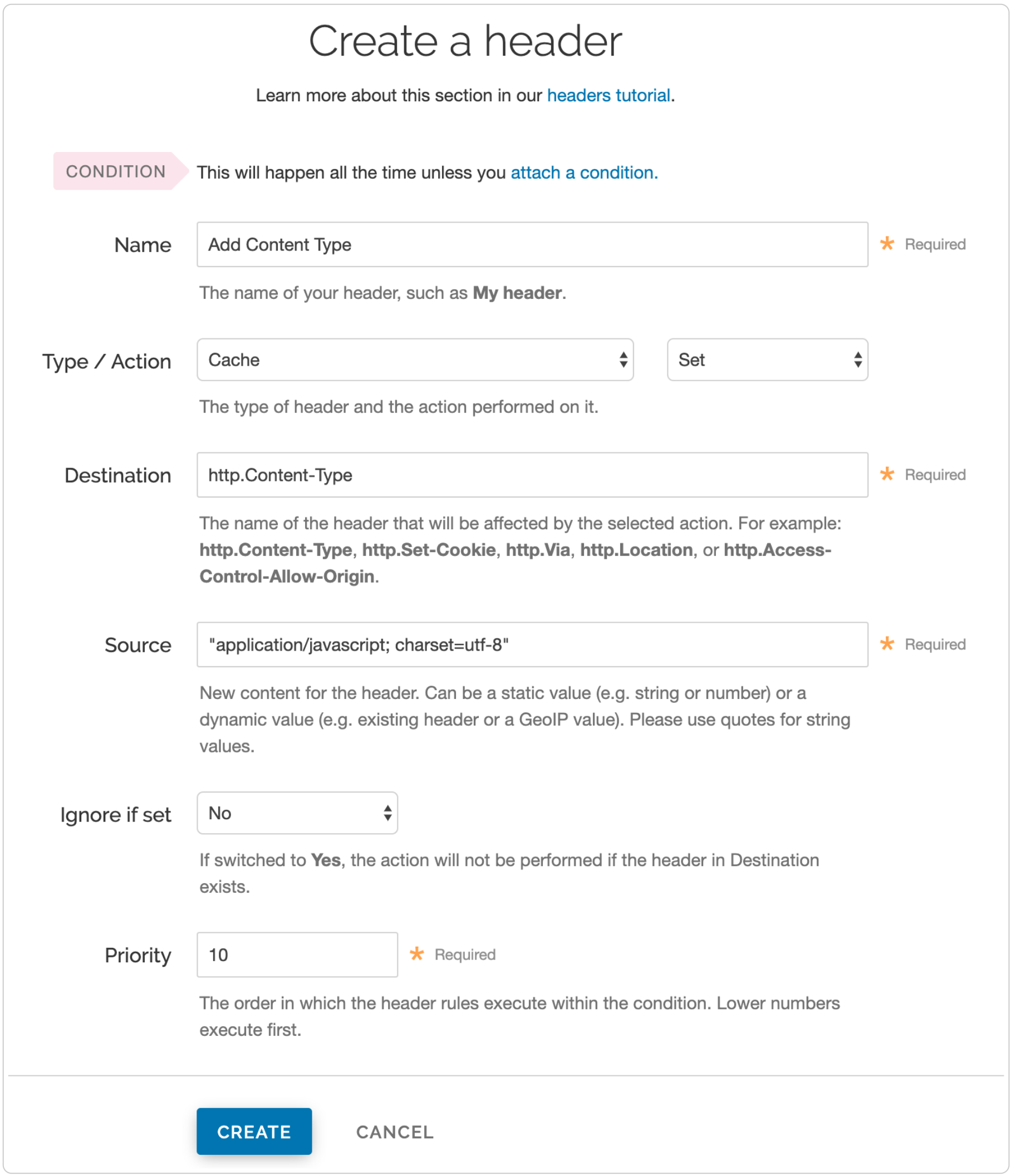
Setting Content Type based on file extension

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.

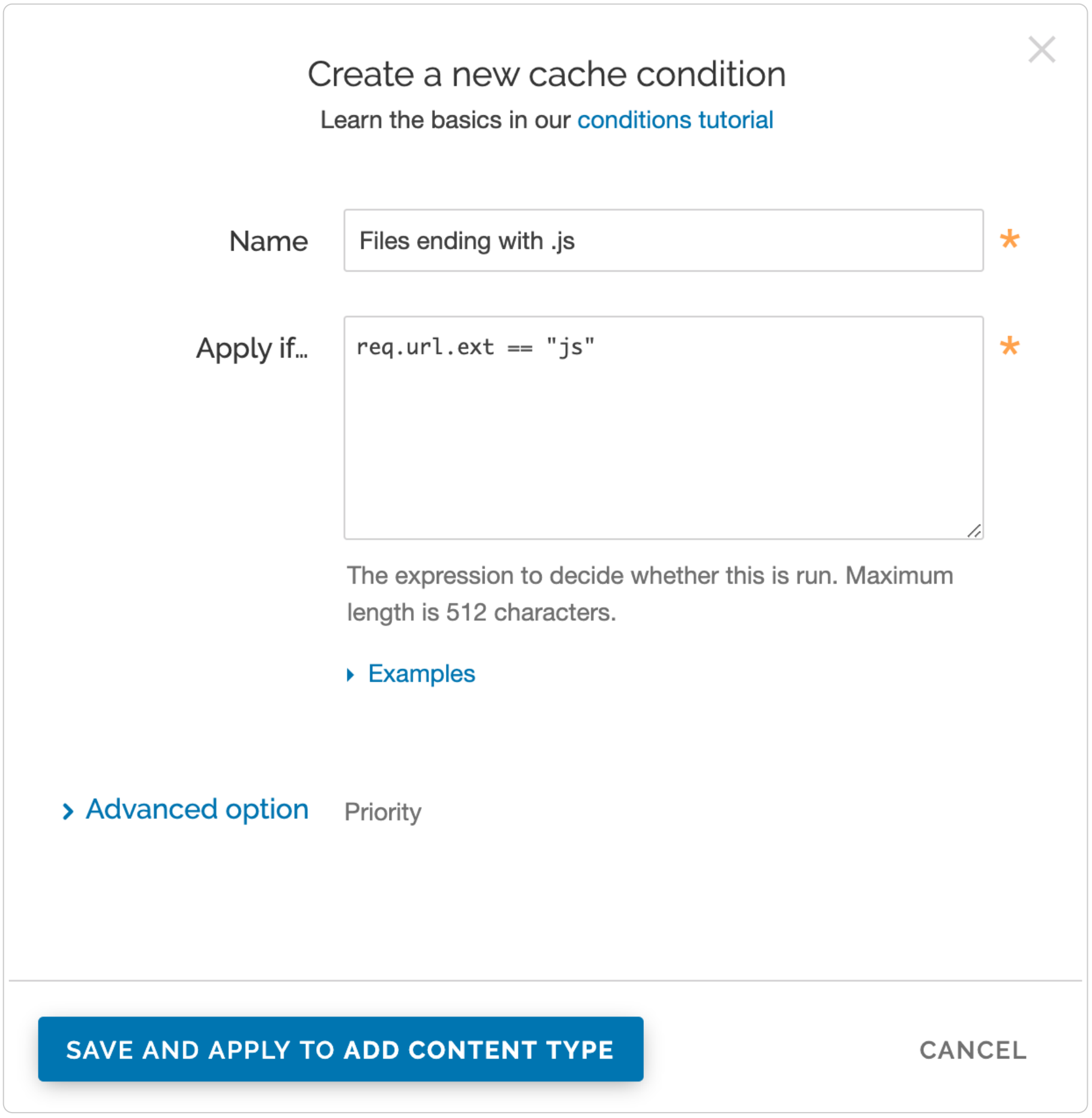
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type 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, type `http.Content-Type`.
   - In the **Source** field, type 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, type `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, type a descriptive name, such as `Files ending with .js`.
   - In the **Apply if** field, type 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 cache HIT and MISS headers with shielded services

https://docs.fastly.com/en/guides/understanding-cache-hit-and-miss-headers-with-shielded-services

Here's some help deciphering cache hit and miss headers when you have shielding enabled.

The cache headers

Let's look at the following requests for the same object if you had run a cURL command in your terminal (for example, `curl -svo /dev/null www.example.com`) to return the Fastly headers.

The first request for an object using the above cURL command might produce output something like this:

```
X-Served-By: cache-iad2120-IAD, cache-sjc3120-SJC
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
```

For this first request, the two cache-nodes in X-Served-By show that shielding is turned on, with `cache-iad2120-IAD` serving as the delivering cache node at the shield datacenter and `cache-sjc3120-SJC` serving as the delivering cache node at the "local" datacenter. The `X-Cache: MISS, MISS` indicates that the requested object was neither in the shield cache (a MISS) nor the local delivering node (also a MISS). The `X-Cache-Hits` reflects that same MISS information because it displays `0, 0`.

The second request for an object using the above cURL command might produce output something like this:

```
X-Served-By: cache-iad2120-IAD, cache-sjc3120-SJC
X-Cache: MISS, HIT
X-Cache-Hits: 0, 1
```

This second time, we hit the same local cache-node (`cache-sjc3120-SJC`) and got a HIT. The MISS listed for `cache-iad2120-IAD` reflects the state of that node the last time it was queried for that object and not its current state, which at the time of the first request, was a MISS. The object is now cached in both datacenters.

Waiting a minute or two and requesting the same object a third time using the above cURL command might produce output something like this:

```
X-Served-By: cache-iad2120-IAD, cache-sjc3122-SJC
X-Cache: MISS, HIT
X-Cache-Hits: 0, 1
```

This third request shows a new cache (`cache-sjc3122-SJC`) being selected from the local datacenter. It registers as a HIT as the object is cached in the local datacenter, with the MISS still reflecting the state of the shield datacenter when it was originally requested. The `X-Cache-Hits` shows `0, 1` reflecting the `0` from the shield datacenter and the `1` for the first hit on cache-sjc-3122-SJC.

Keep in mind that if the closest delivering cache node exists in the shield datacenter, you will only see a single server and HIT data such as:

```
X-Served-By: cache-iad2120-IAD
X-Cache: HIT
X-Cache-Hits: 1
```

After a purge of the object, requesting the object again via the above cURL command will produce results similar to the first request scenario. For example:

```
X-Served-By: cache-iad2120-IAD, cache-sjc3120-SJC
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
```

The x-cache header

Let's look at all the possible combinations of MISS and HIT we can see in the X-Cache header when a request is run through both an edge and a shield datacenter.

```
X-Cache: MISS, MISS
```

The header returned a MISS on both the shield and the edge, indicating the requested object was neither in the shield cache nor the edge cache.

```
X-Cache: MISS, HIT
```

The header was a HIT on the edge. When the object was previously cached on this datacenter, it had been a MISS on the shield. The object cached the MISS status on the shield on the edge.

```
X-Cache: HIT, MISS
```

The header returned a MISS on the edge and a HIT on the shield. This shows that the edge datacenter did not have the object, but when the request was forwarded to the shield datacenter, the object was found.

```
X-Cache: HIT, HIT
```

The header returned a HIT on the edge. On a previous request, the request was forwarded from the edge datacenter to the shield datacenter to retrieve the object. Because the shield had the object, the response back to the edge showed that the shield was an `X-Cache: HIT`. When the object was cached on the edge, the HIT status for the shield was cached along with it. On this subsequent request, the header shows the HIT on the shield as well as the HIT on the edge.

Understanding the X-Timer header

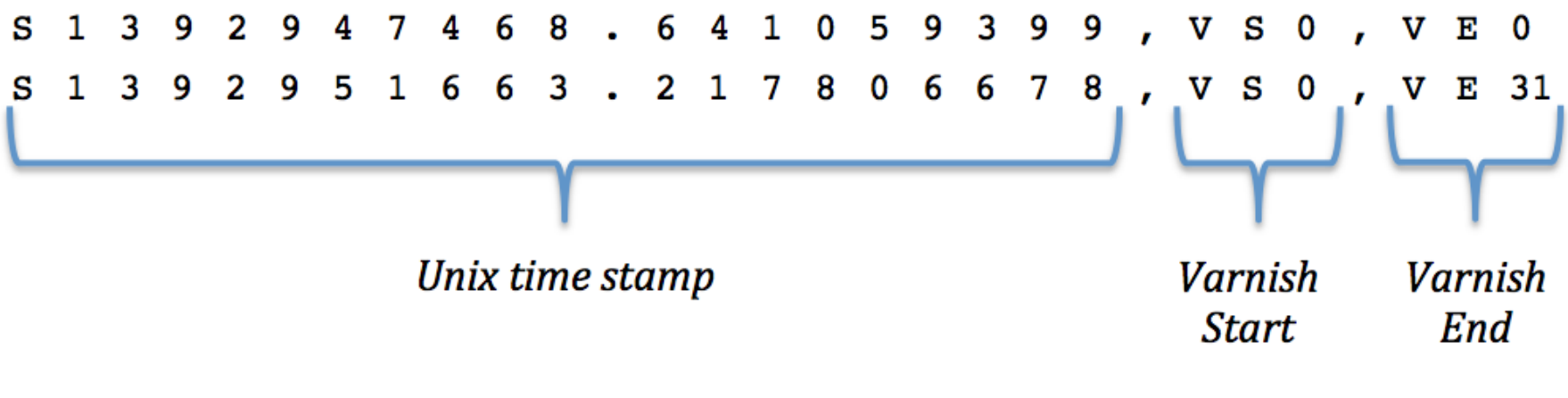
https://docs.fastly.com/en/guides/understanding-the-xtimer-header

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: Interested in functions and variables that allow you to control dates and times using custom VCL? We have a guide describing which ones we support.

Responses

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

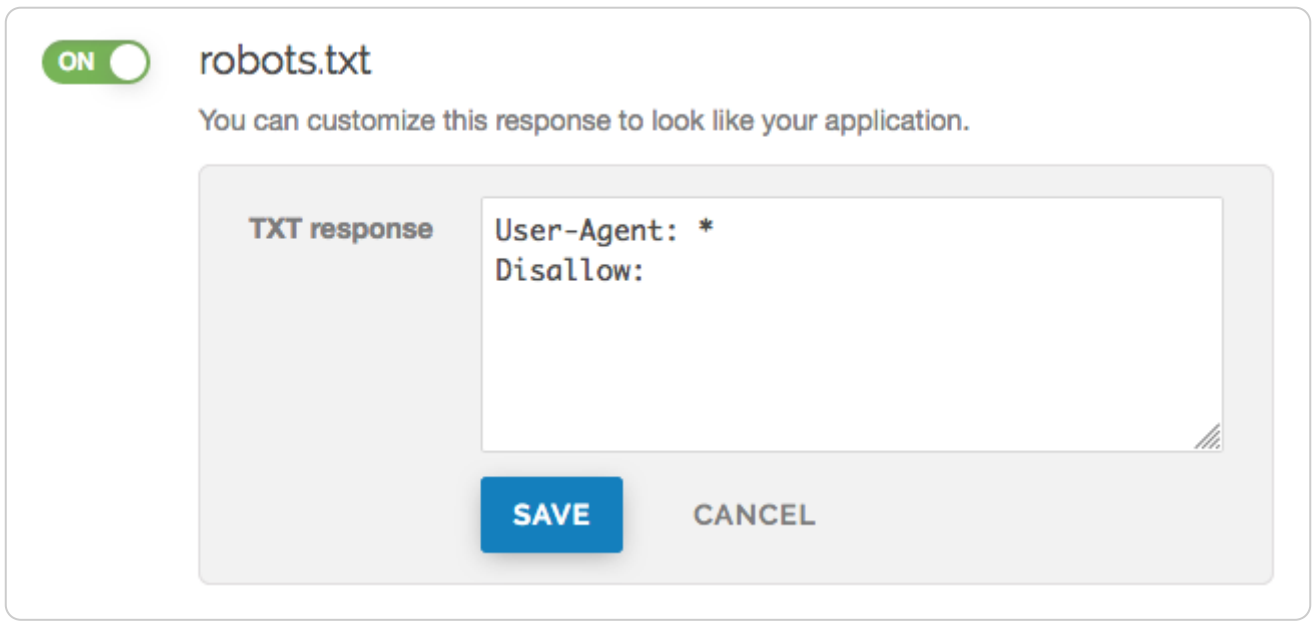
https://docs.fastly.com/en/guides/creating-and-customizing-a-robots-file

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

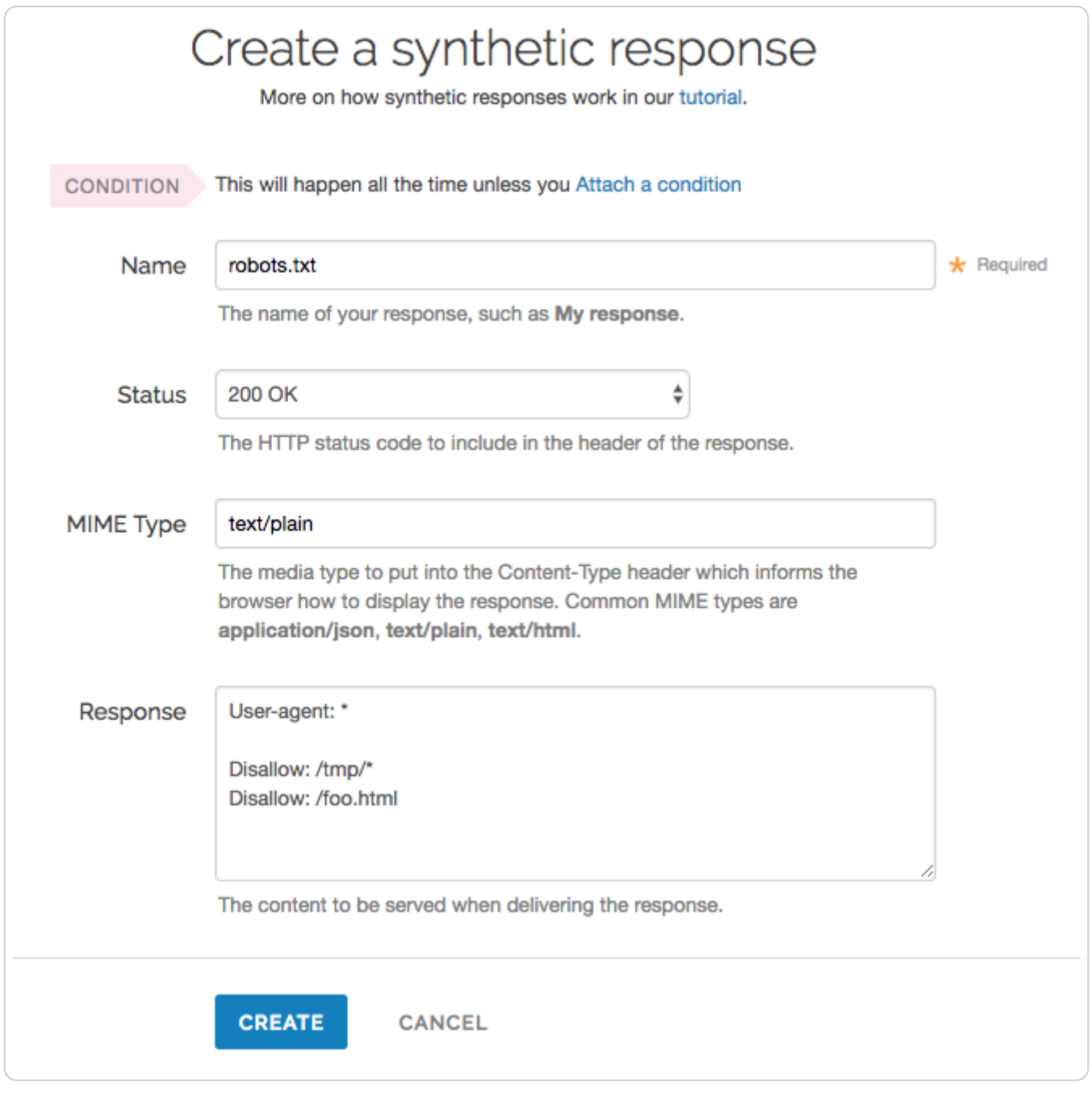


6. In the **TXT Response** field, customize the response for the robots.txt file.
7. Click the **Save** button to save the response.
8. Click the **Activate** button to deploy your configuration changes.

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:

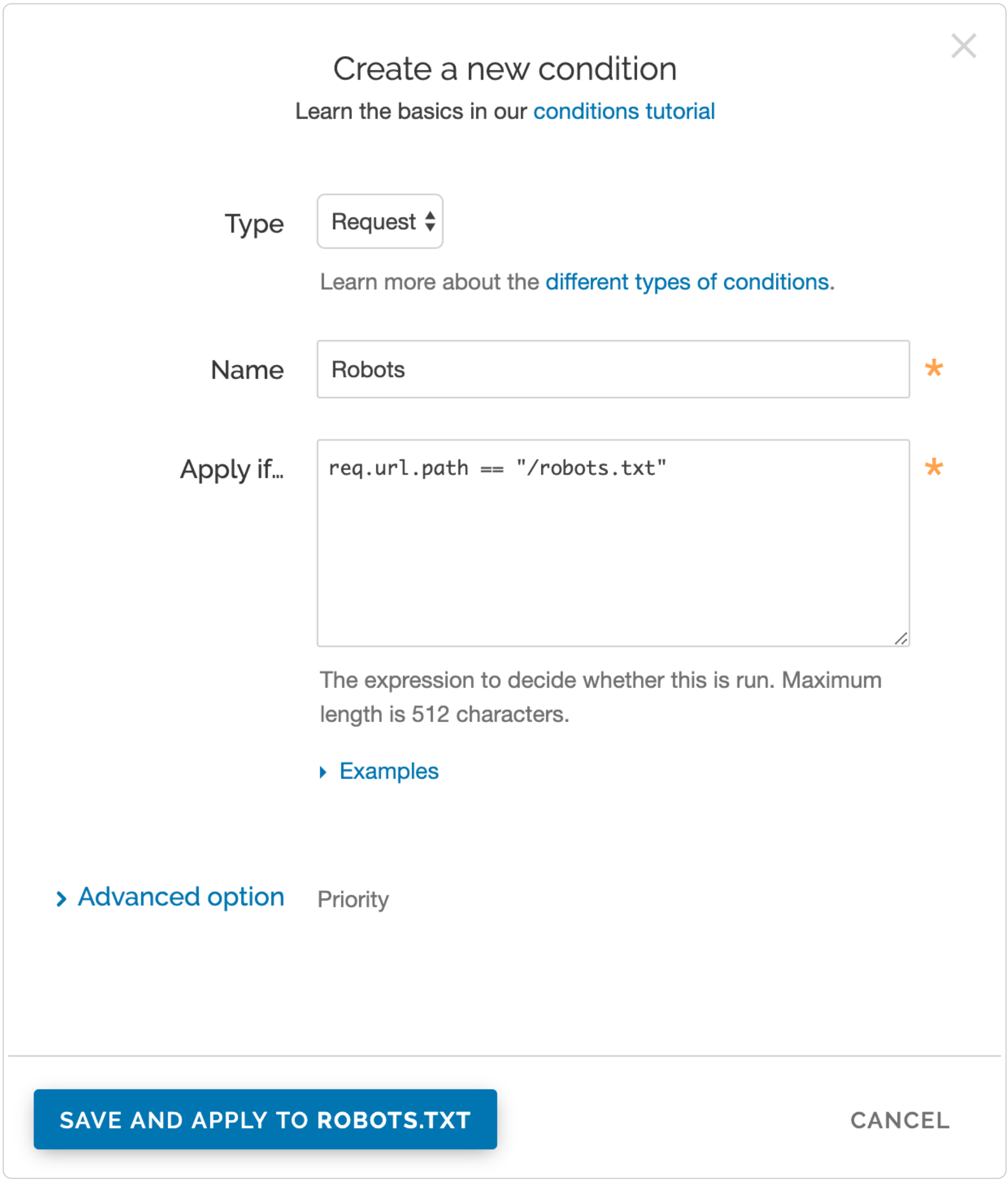
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **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, type an appropriate name. For example `robots.txt`.
   - Leave the **Status** menu set at its default `200 OK`.
   - In the **MIME Type** field, type `text/plain`.

- In the **Response** field, type 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.



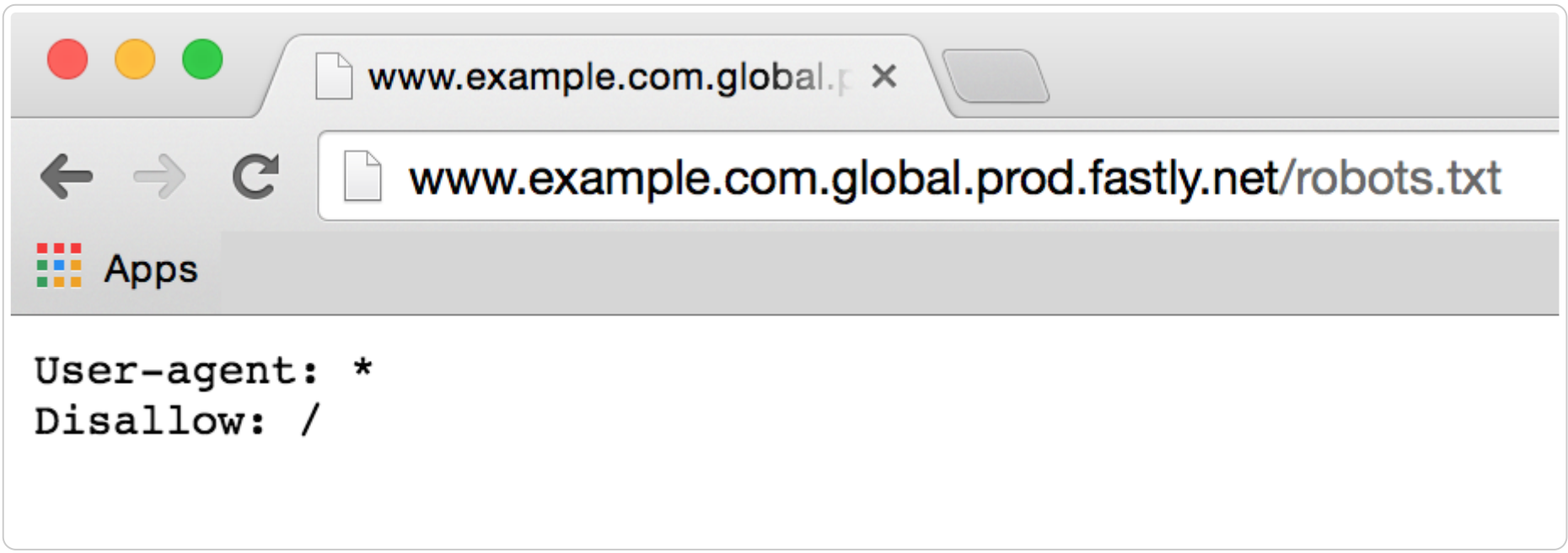
9. Fill out the **Create a condition** fields as follows:
   - From the **Type** menu, select the desired condition (for example, `Request`).
   - In the **Name** field, type a meaningful name for your condition (e.g., `Robots`).
   - In the **Apply if** field, type the logical expression to execute 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

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.

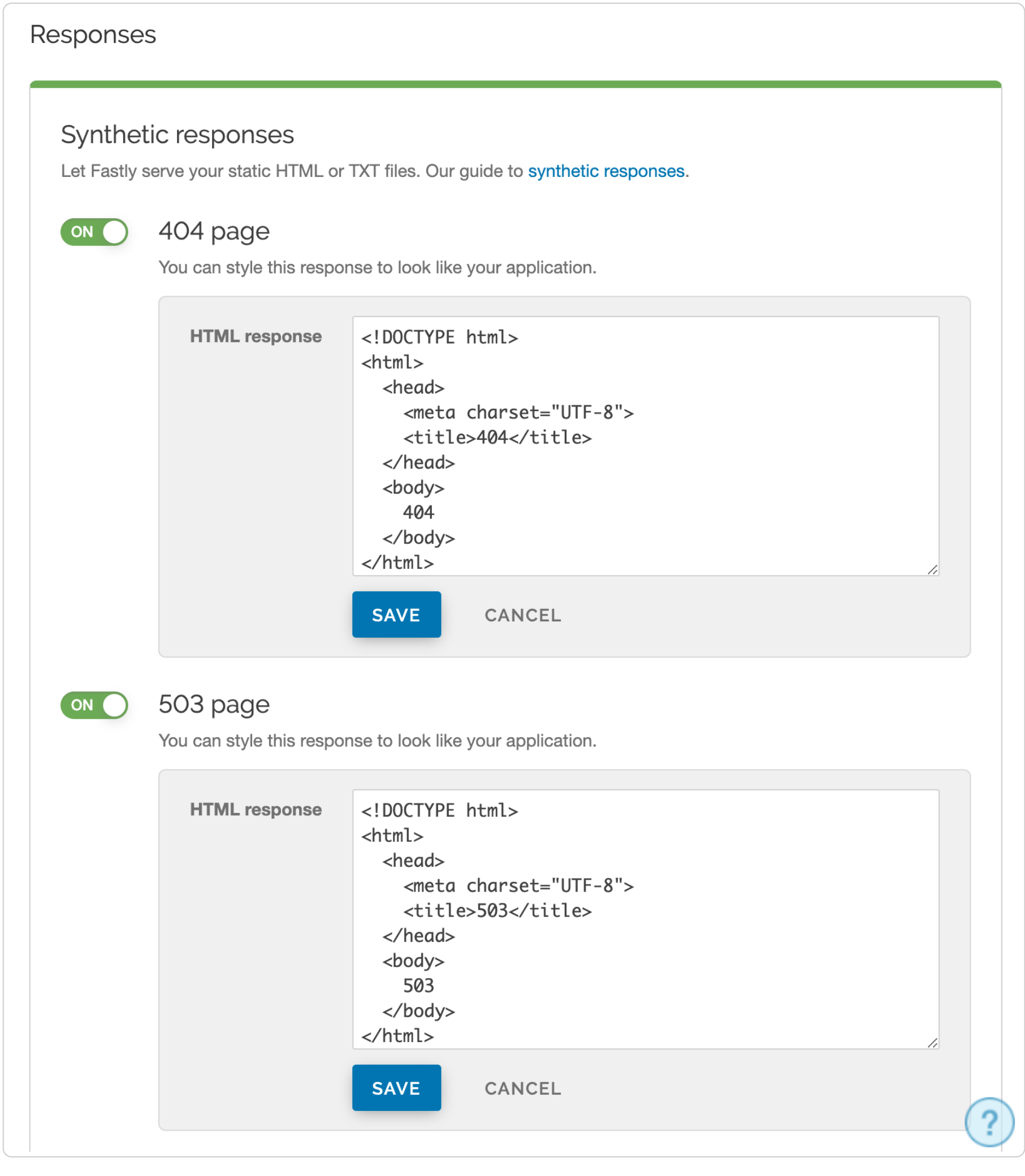
Fastly offers two quick configuration options for creating 404 and 503 error pages directly in the web interface, but you can also use the interface to create error pages for other status codes. If you're working with large blocks of content when styling your error pages, consider creating custom responses 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.



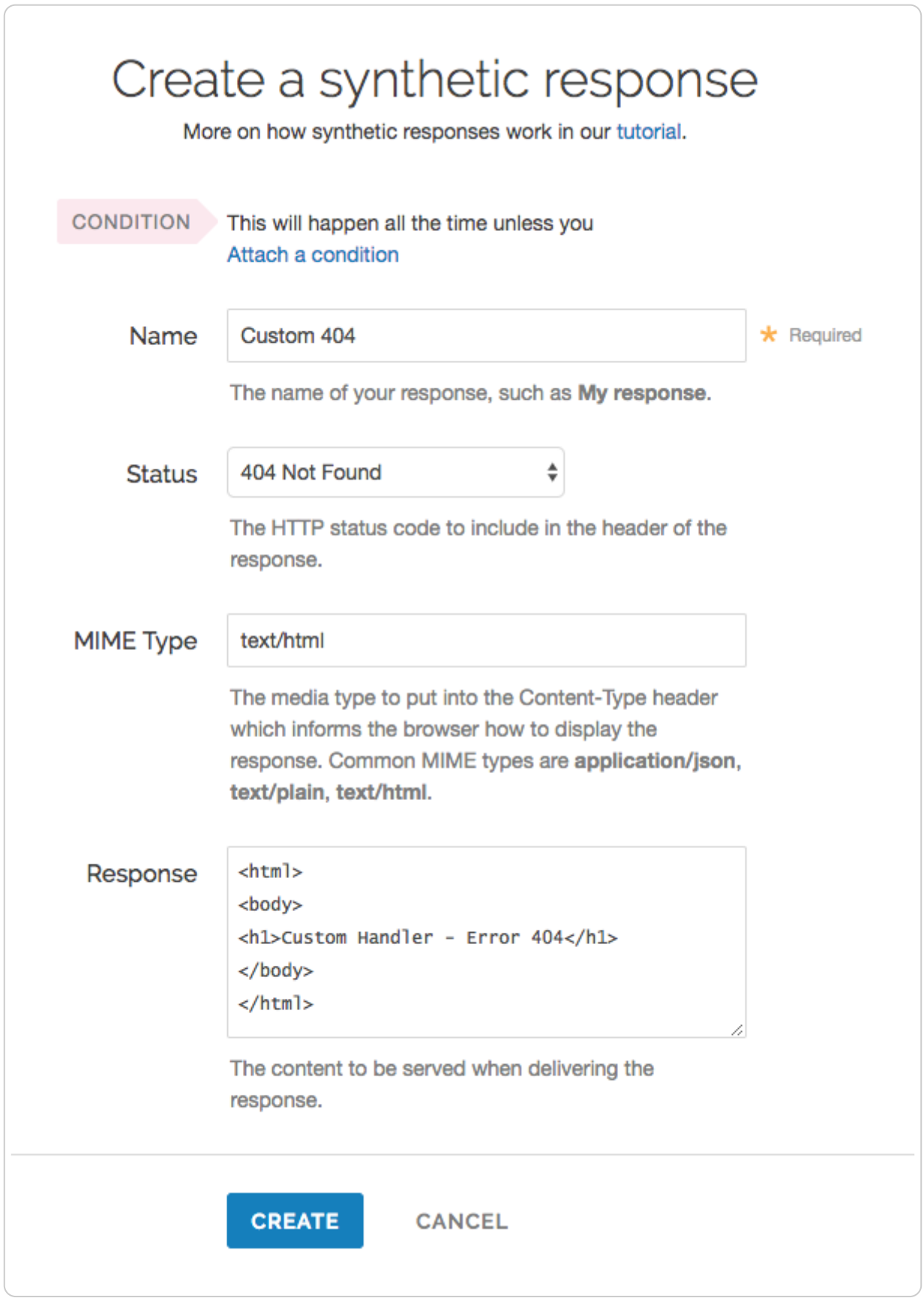
6. In the **HTML response** fields, customize the response for the 404 and 503 error pages.
7. Click the **Save** buttons to save the responses.
8. Click the **Activate** button to deploy your configuration changes.

Creating error pages for other status codes

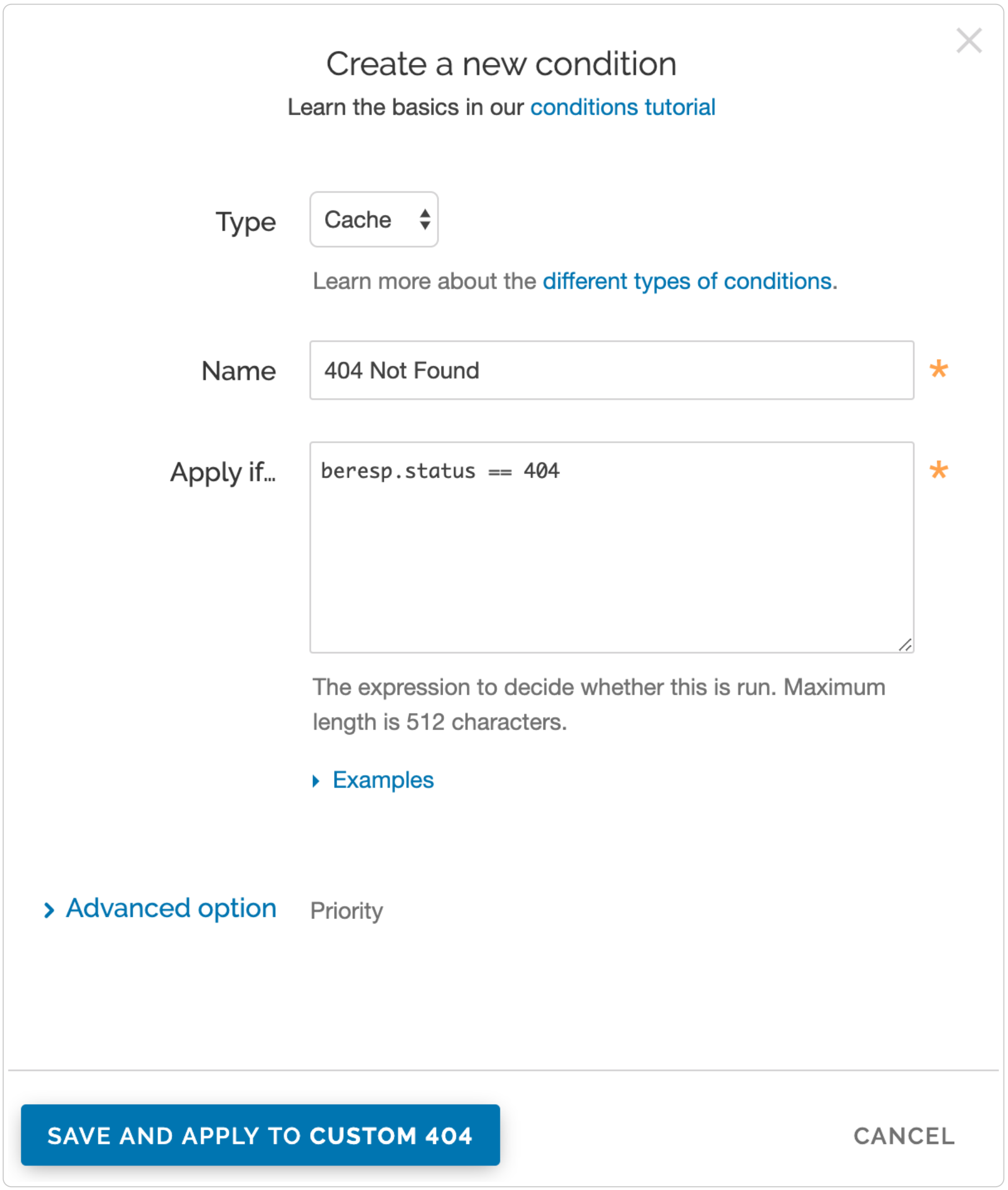
You can also create error pages for other HTTP status codes. We provide example HTML, but you can use any HTML you see fit. The response object will require that you use a condition in order for it to be served.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type 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, type a name for the condition you're creating (e.g., `404 Not Found`).
   - In the **Apply if** field, type the condition under which the new response occurs in the following format:

     `beresp.status == ###`

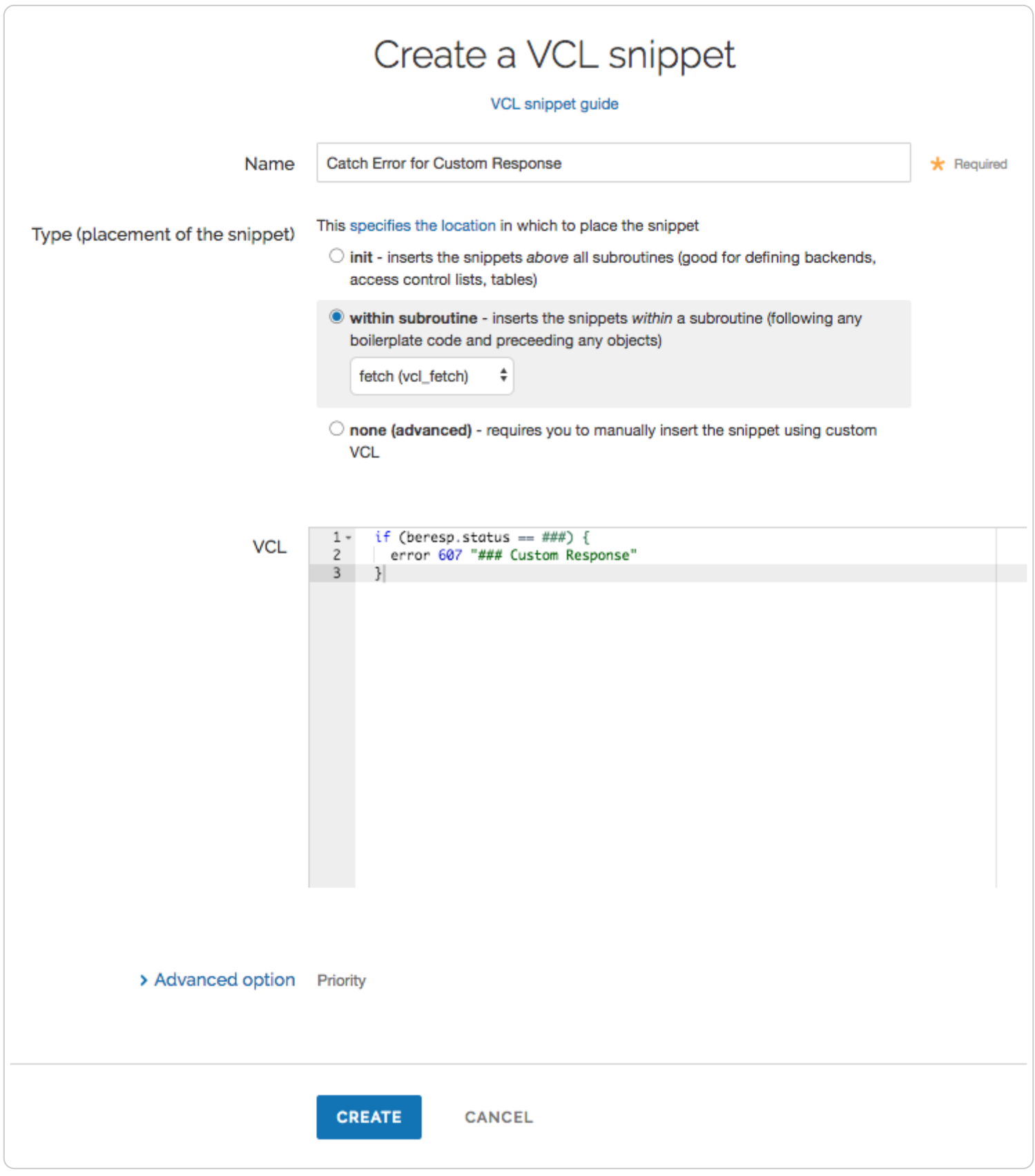
   where ### equals the status condition you're creating the response for. For example, using the value of `beresp.status == 404` in the **Apply if** field here tells Fastly to use this response object whenever origin servers return a 404 status. (See the Conditions guides for more detailed information 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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:

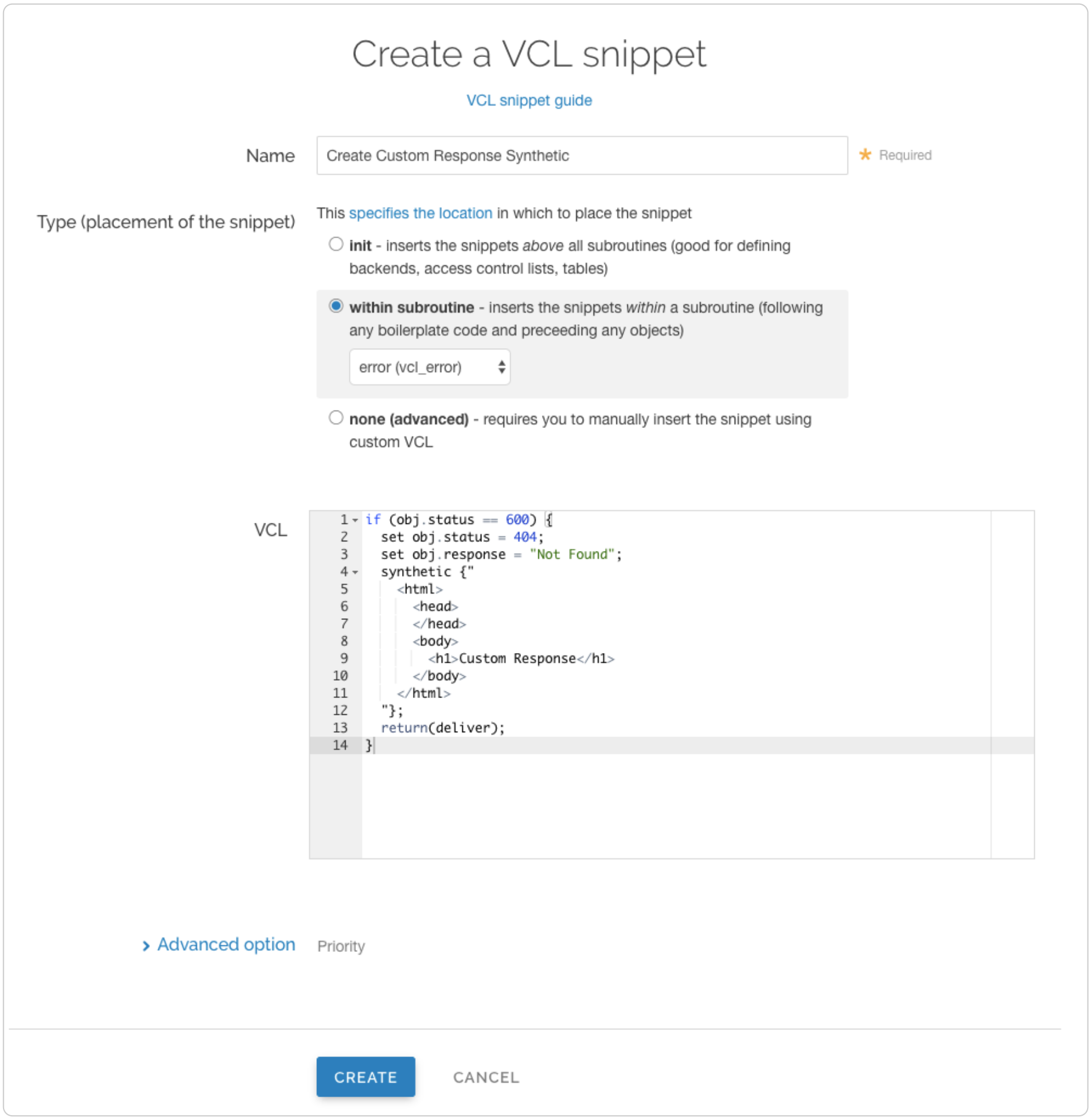
```
1  if (beresp.status == ###) {
2    error 600 "### Custom Response"
3  }
```

   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, type 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

https://docs.fastly.com/en/guides/generating-http-redirects-at-the-edge

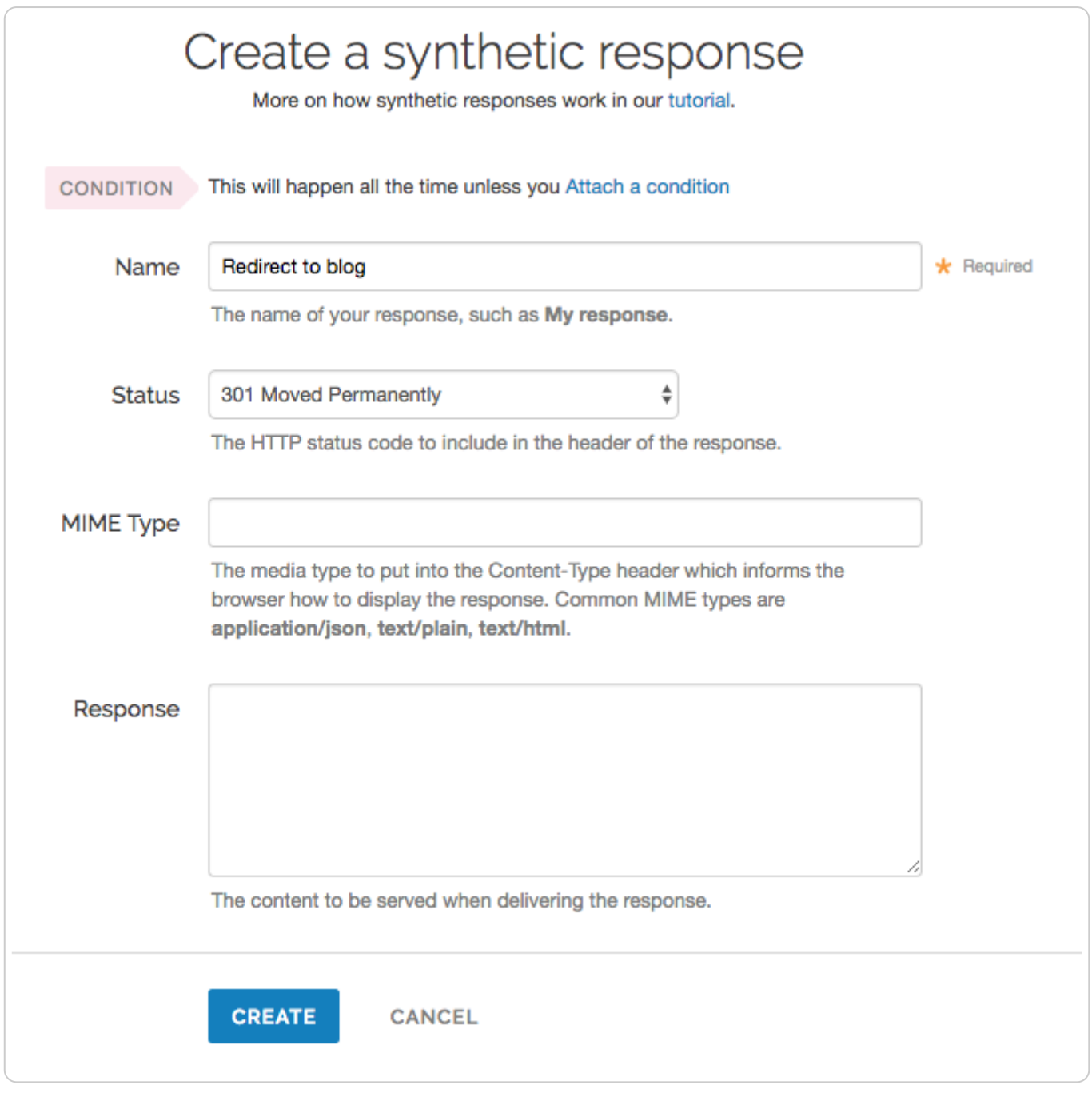
When users request information from your origin servers, you may want to redirect them for various reasons. For example, you may want to redirect them to pages that have been moved or updated since the last time they were requested. You can send these redirects from the edge rather than having to go to origin by creating a synthetic response with the appropriate redirect status code and then creating a content rule with the proper Location header.

> **TIP:** This guide describes how to create normal 301 (and 302) redirects from one URL to another. If you are interested in automatically redirecting all HTTP requests to HTTPS, our guide to forcing a TLS redirect describes an easier way to do this.

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.

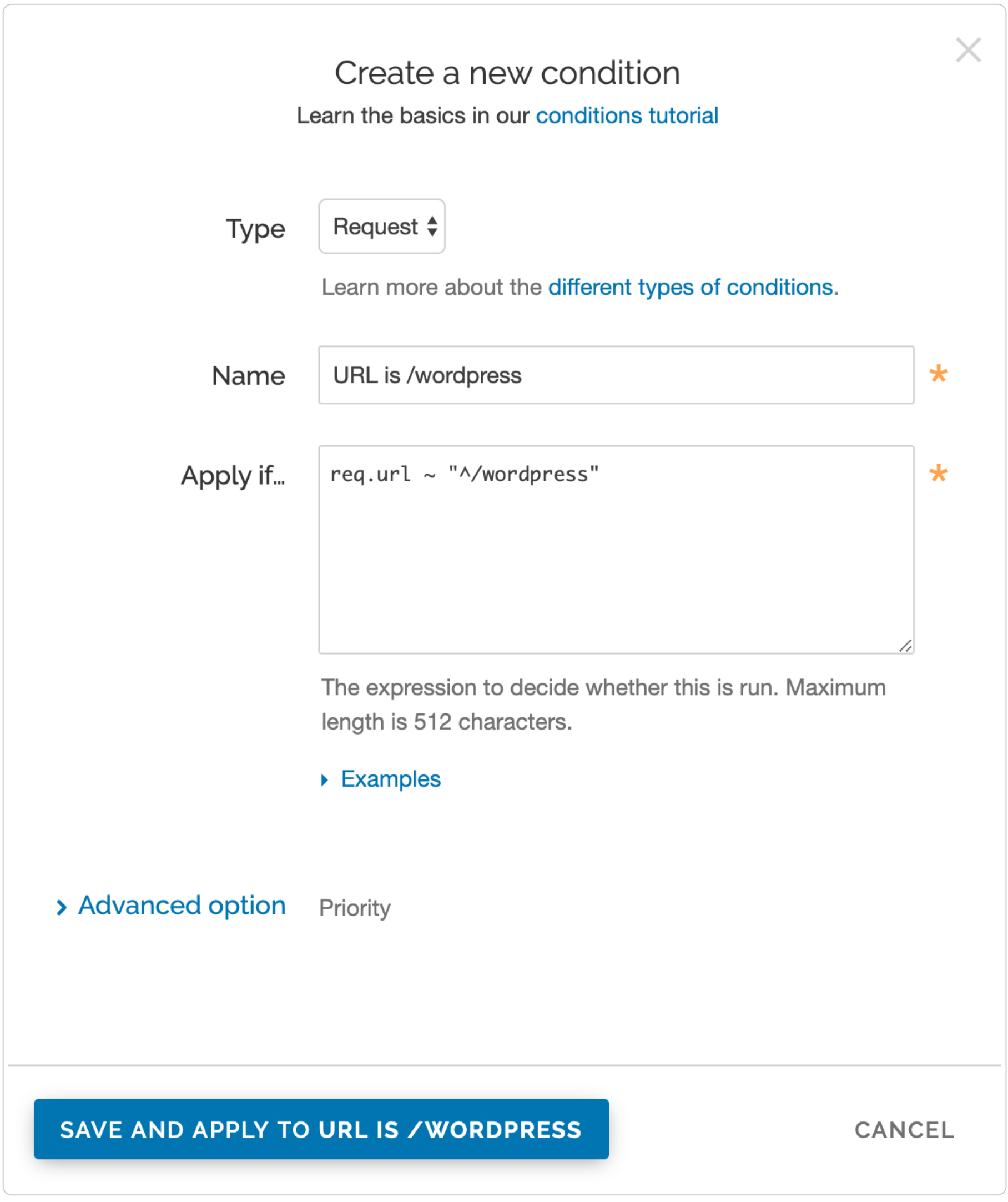
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **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, type 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.

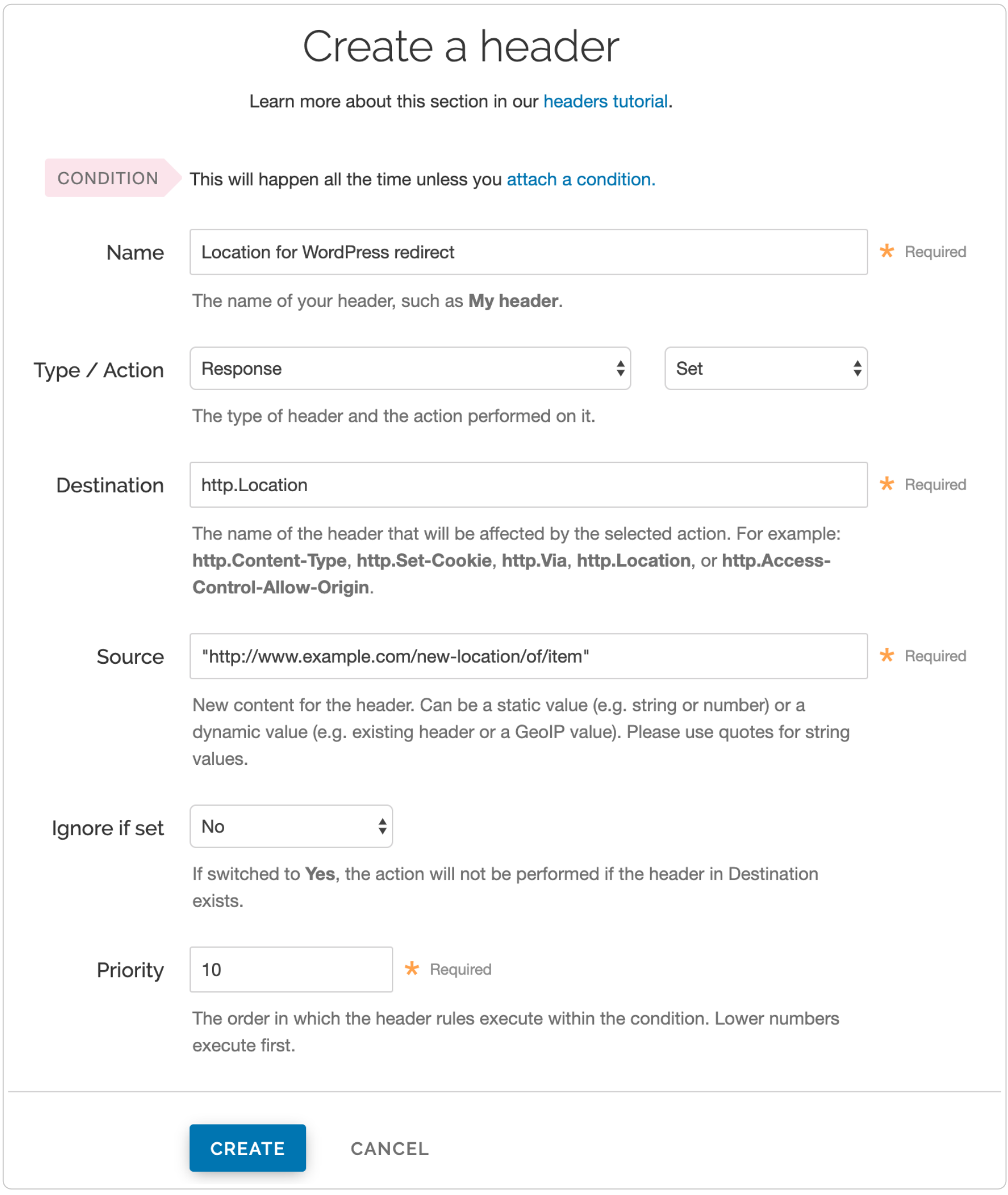


9. Fill out the **Create a new request condition** fields as follows:
   - From the **Type** menu, select the type of condition you want to create (e.g., `Request`).
   - In the **Name** field, type a meaningful name for your condition (e.g., `URL is /wordpress`). This name is displayed in the Fastly web interface.
   - In the **Apply if** field, type 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.

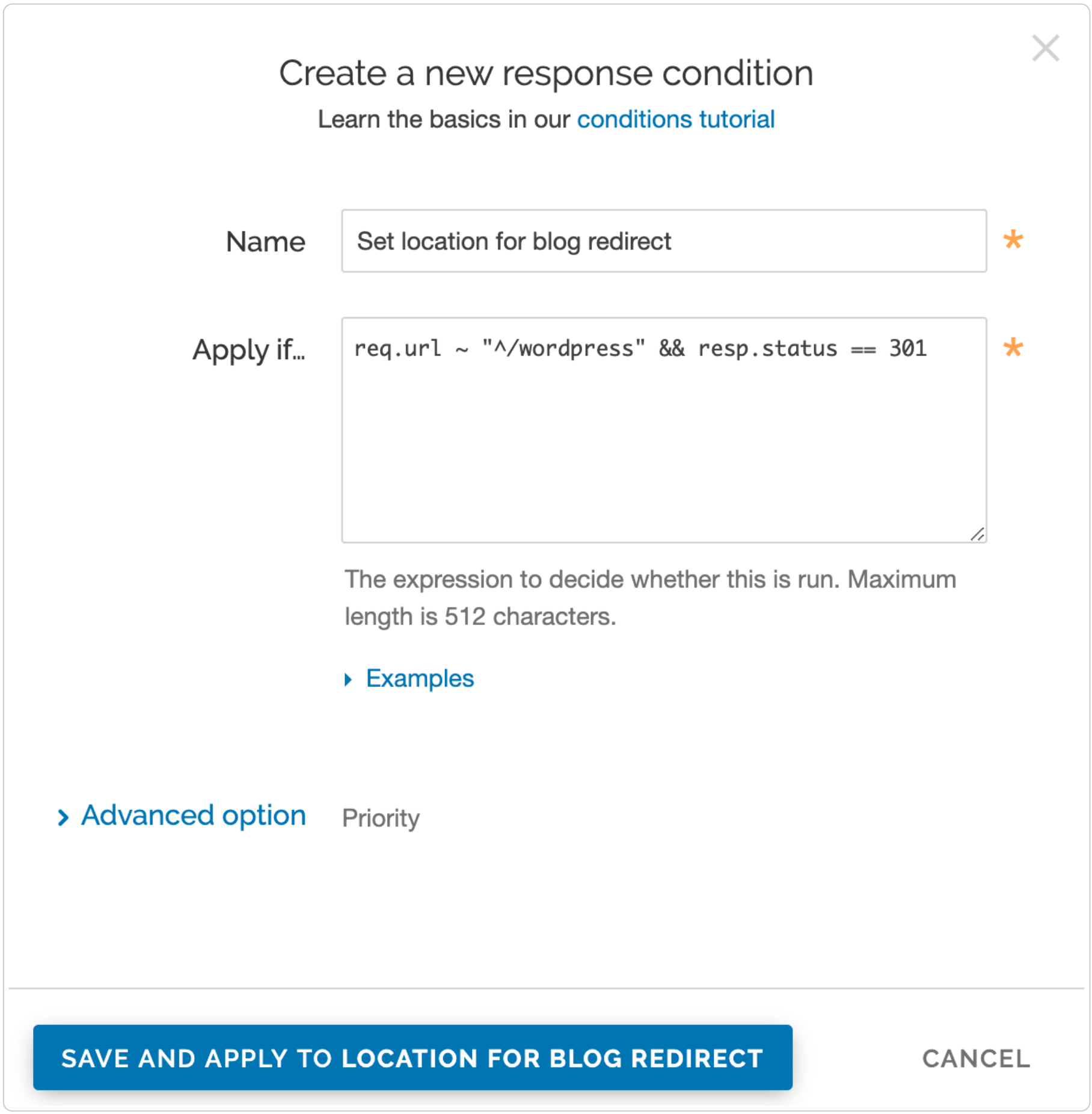
Create a new header and condition

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, type 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, type `http.Location`.
   - In the **Source** field, type 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. The Create a new response condition window appears.



5. Fill out the **Create a new response condition** fields as follows:
   - In the **Name** field, type a meaningful name for your condition (e.g., `Set location for blog redirect`).
   - In the **Apply if** field, type 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.
6. Click the **Save and apply to** button to create the new condition.
7. 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.

Responses tutorial

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.

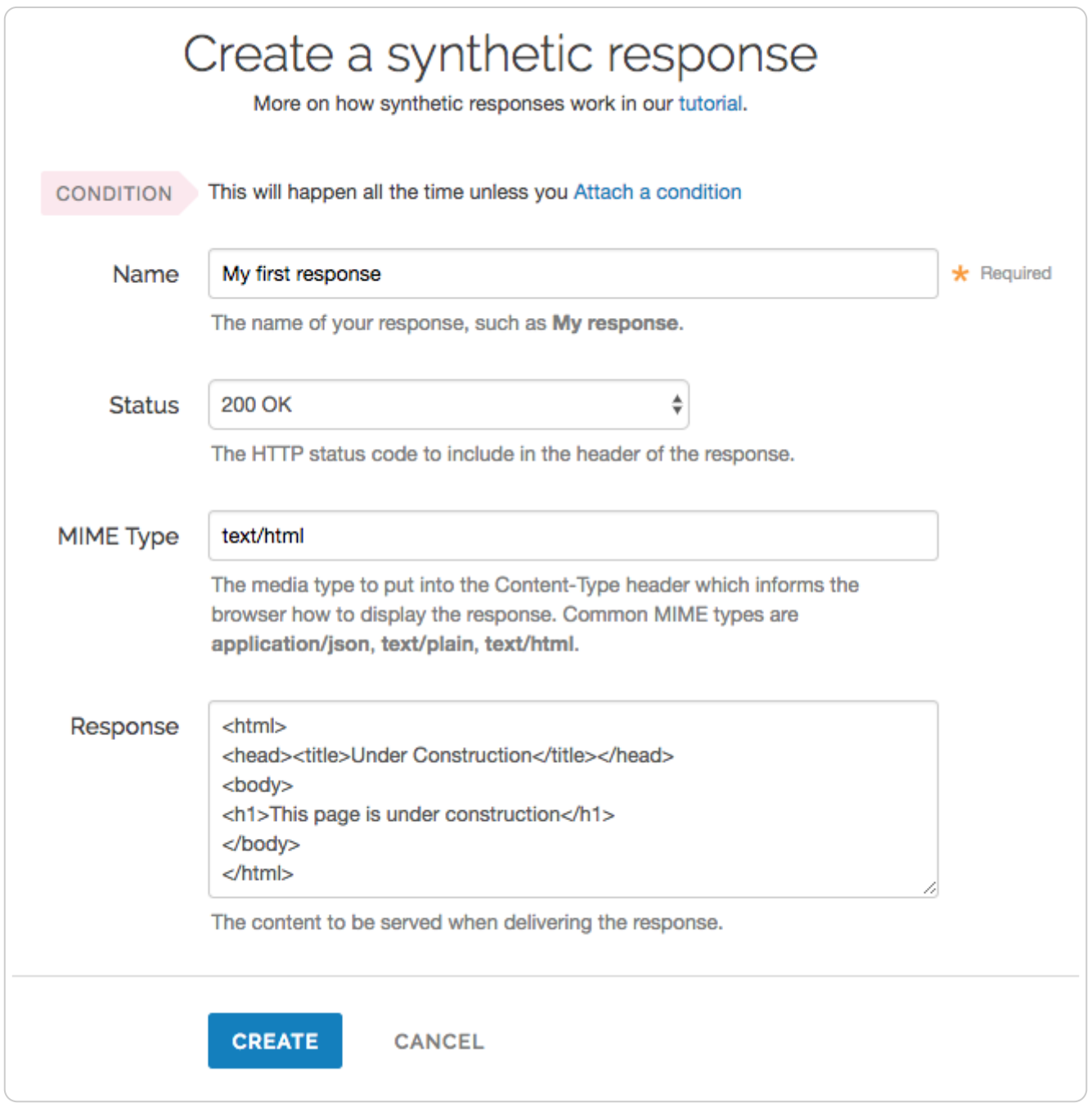
Creating an advanced response

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.



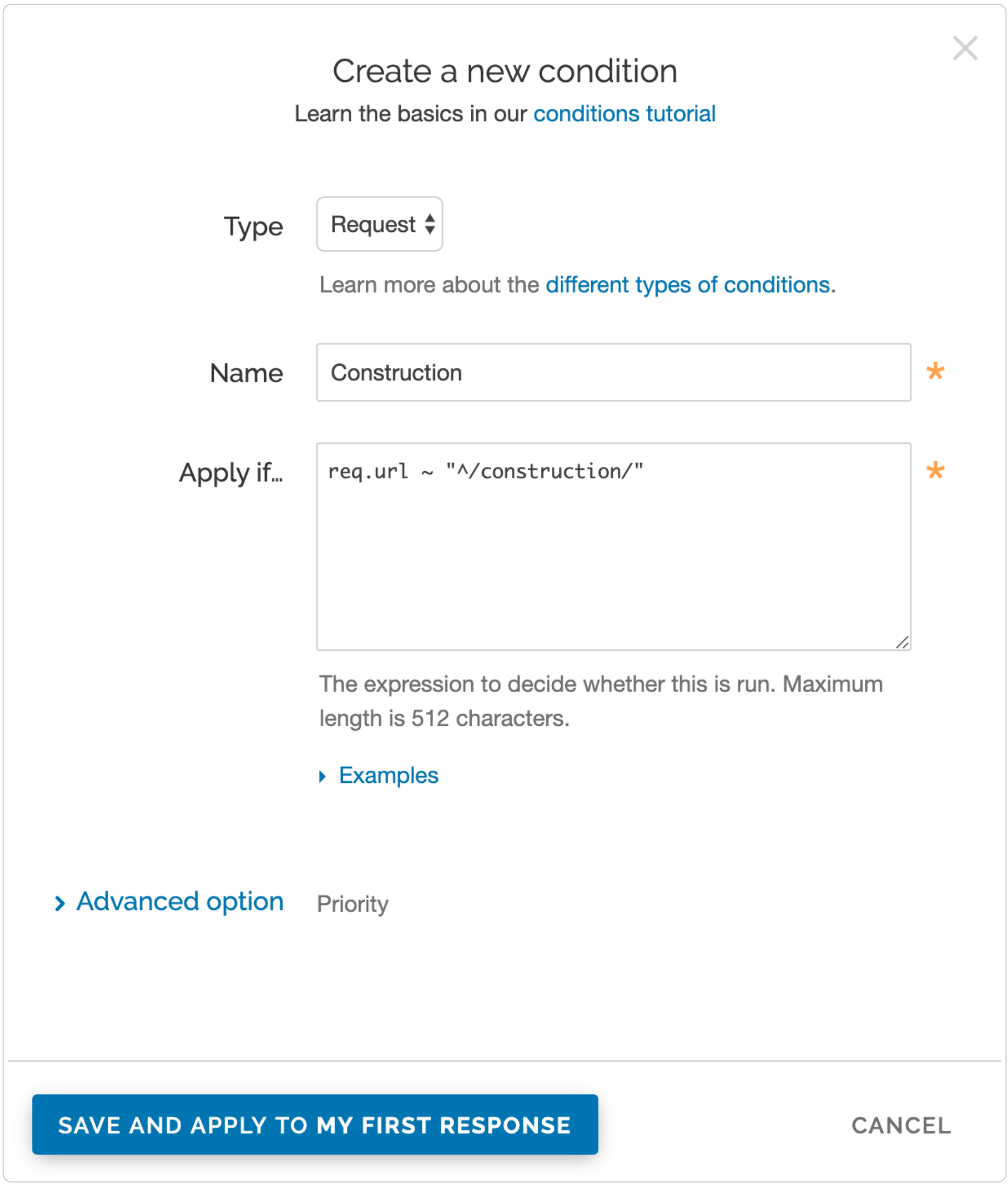
6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, type a human-readable name for the response (e.g., `My first response`).
   - From the **Status** menu, select the appropriate status (e.g., `200 OK`).
   - In the **MIME Type** field, type the content type of the response (e.g., `text/html`).
   - In the **Response** field, type 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, type a human-readable name for the condition so that it can be easily identified in the future.
   - In the **Apply if** field, type 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, type 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.

Performance

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

https://docs.fastly.com/en/guides/enabling-automatic-gzipping

Fastly's gzip feature dynamically fetches content from origin, compresses it, and then caches it. There are two ways to enable gzip:

- Enable the default gzip policy to compress content in files with the following extensions: `css js html eot ico otf ttf json svg`.
- Set up an advanced gzip policy to customize the content and conditions for compression.

WARNING: This feature doesn't work with our ESI feature. If you enable gzipping, Fastly will stop processing ESI language elements.

Enabling gzip

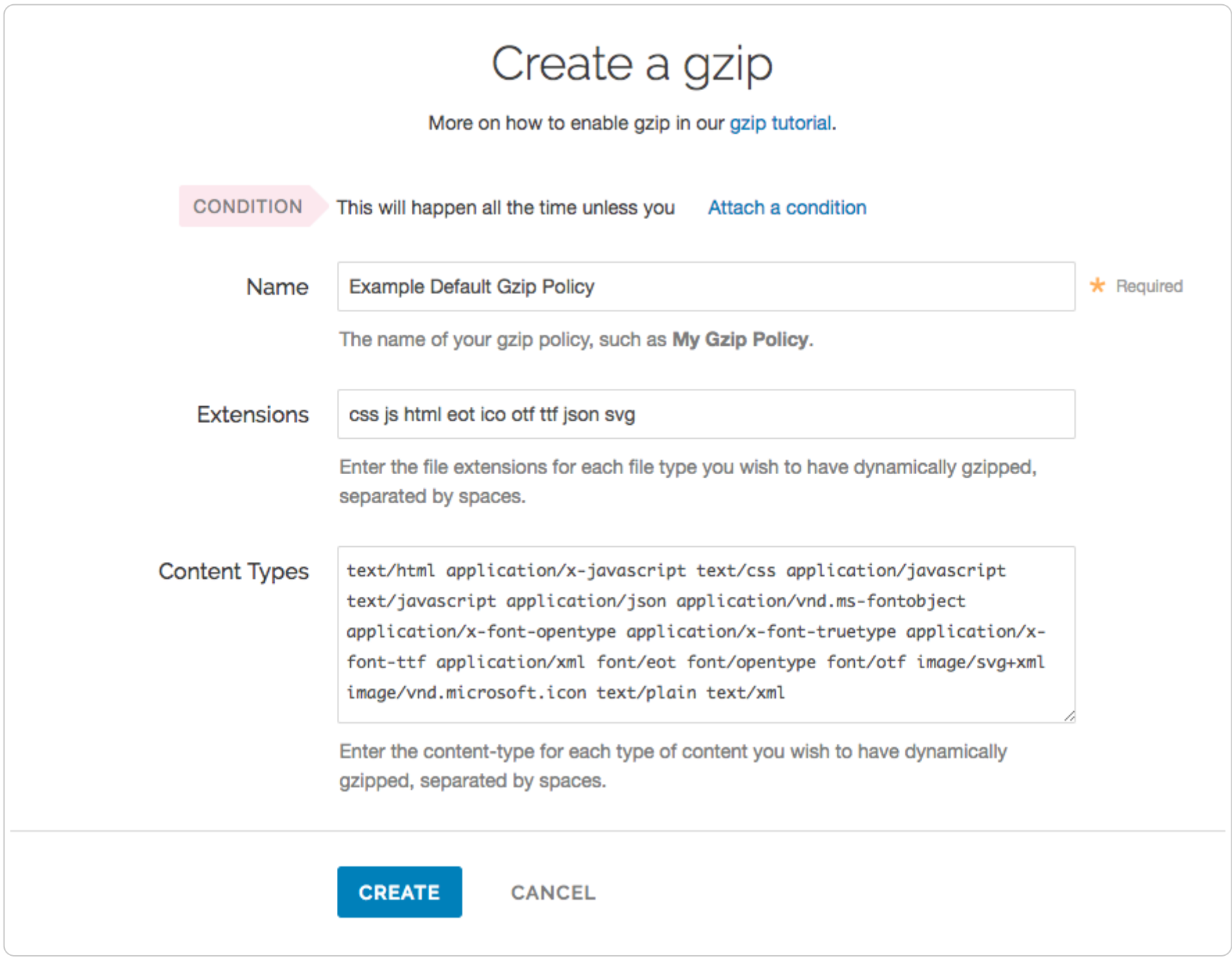
To dynamically gzip cacheable content based on file extension or content-type, follow the steps below to enable the default gzip policy:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type an arbitrary name for your new gzip rule.
   - In the **Extensions** field, type the file extension for each file type to be dynamically gzipped, separated by spaces. Only type the three- or four-letter string representing the file extension.
   - In the **Content Types** field, type the content-type for each type of content you wish to have dynamically gzipped, 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.

Automatic normalization

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. The modified header will be set to a single encoding type, or none, and will reflect the best compression scheme supported by the browser. This includes removing Accept-Encoding values in requests from browsers that advertise support for gzip but whose implementation is broken, such as IE6.

Specifically, we run the following steps on inbound requests:

1. If the `User-Agent` matches a pattern for browsers that have problems with compressed responses, remove the `Accept-Encoding` header
2. Else if the `Accept-Encoding` header includes the string "gzip", set the entire value to the string "gzip"
3. Else if the `Accept-Encoding` header includes the string "deflate", set the entire value to the string "deflate"
4. Else remove the `Accept-Encoding` header

Where this normalization process has changed the header value, the original value is made available in the custom header `Fastly-Orig-Accept-Encoding`.

If a user agent advertises support for brotli, currently we will normalize this to gzip because we do not support brotli encoding at the edge. However, if you are doing brotli encoding at your origin server, you may want to modify our normalization algorithm.

Failure modes with large files

https://docs.fastly.com/en/guides/failure-modes-with-large-files

Varnish is excellent at caching, managing, and delivering small files but historically the handling of large files has been challenging. By default, Fastly limits cached file sizes to 2GB. Using streaming miss increases that limit for files up to 5GB. Fastly's also allows you to enable support for files larger than the 5GB caching limit using VCL snippets. Regardless of which feature you use, there are a few failure modes you may encounter when working with large files, especially those larger than 5GB in size.

Maximum object size limits

If the response from the origin has a `Content-Length` header field that exceeds the maximum object size, Fastly will immediately generate a 503 response to the client unless you tell Fastly to specifically pass those objects directly to the user from the origin 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 maximum object size, we will generate a status 503 response to the client (again, unless you specifically tell Fastly to do otherwise as previously mentioned).

If no `Content-Length` header field is present and Streaming Miss is in effect, 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

https://docs.fastly.com/en/guides/http2-server-push

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.c
ss>; rel=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

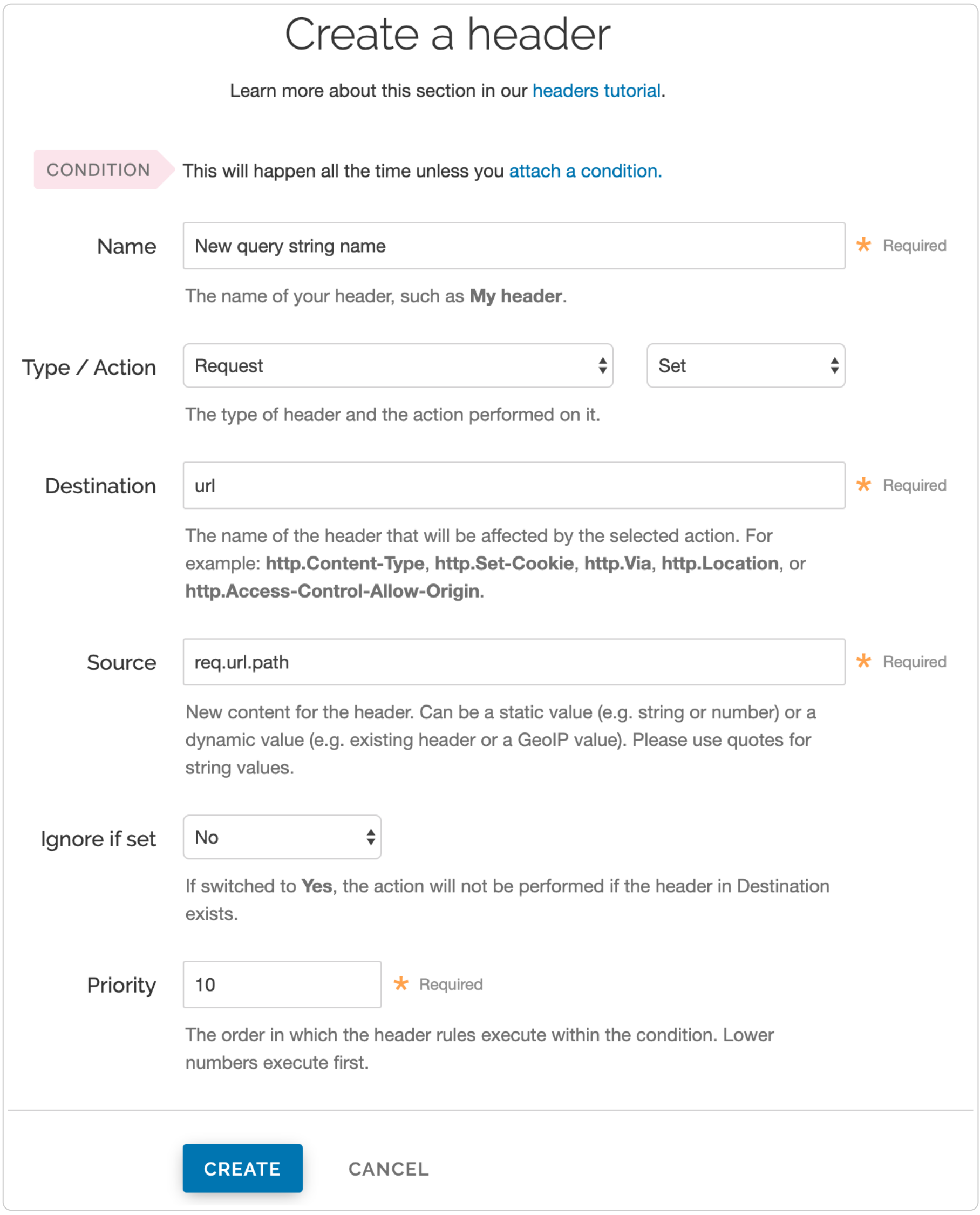
https://docs.fastly.com/en/guides/making-query-strings-agnostic

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 file.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type `url`.
   - In the **Source** field, type `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 Cache Control Tutorial.

Request collapsing

https://docs.fastly.com/en/guides/request-collapsing

This guide describes Fastly's Request Collapsing feature, frequently used when creating advanced service configurations.

NOTE: This guide requires advanced knowledge of Varnish and the VCL language.

The basics

Request Collapsing causes simultaneous cache misses within a single Fastly datacenter to be "collapsed" into a single request to an origin server. While the single request is being processed by the origin, the other requests wait in a queue for it to complete. Two types of Request Collapsing exist:

1. Collapsing on a single cache server
2. Collapsing within the datacenter between cache servers

Each cache server will automatically queue duplicate requests for the same hash and only allow one request to origin. You can disable this behavior by setting `req.hash_ignore_busy` to `true` in `vcl_recv`.

Within a datacenter, not every cache stores every object. Only two servers in each datacenter will store an object: one as a primary and one as a backup. Only those two servers will fetch the object from origin.

How it works

In Fastly's version of Varnish, VCL subroutines often run on different caches during a request. For a particular request, both an edge node and a cluster node will exist (though a single cache can, in some cases, fulfill both of these roles). The edge node receives the HTTP request from the client and determines via a hash which server in the datacenter is the cluster node. If this cache determines it is the cluster node and has the object in cache, it fulfills both the edge node and the cluster node roles.

Certain VCL subroutines run on the edge node and some on the cluster node:

- Edge Node: *`vcl_recv`*, *`vcl_deliver`*, *`vcl_log`*, *`vcl_error`*
- Cluster Node: *`vcl_miss`*, *`vcl_hit`*, *`vcl_fetch`*, *`vcl_error`*

Determining if a cache is an edge or a cluster node

The `fastly_info.is_cluster_edge` VCL variable will be true if the cache currently running the VCL is the edge node and false if it is the cluster node.

Caveats

Keep in mind the following limitations when using the Request Collapsing feature:

1. Any `req.http.*` headers are not transferred from the cluster node back to the edge node. Remember this when writing advanced configurations that use headers to keep track of state. If you set a `req.http.*` header in any of the subroutines that run on the cluster node, expect that the change will not persist on the edge node.
2. A single, slow request to origin can sometimes cause a great many other requests for the same object to hang and fail. Because many requests for a single object are being collapsed down to one, they all succeed or fail based on the request that reaches the origin.

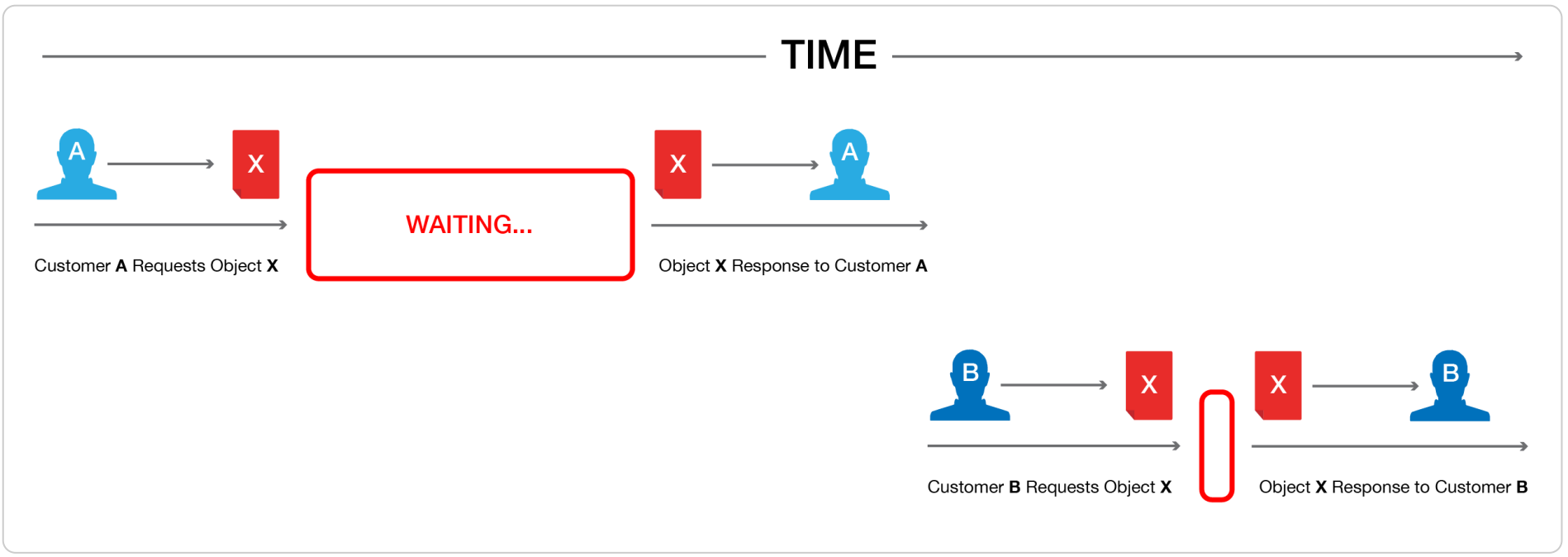
Serving stale content

https://docs.fastly.com/en/guides/serving-stale-content

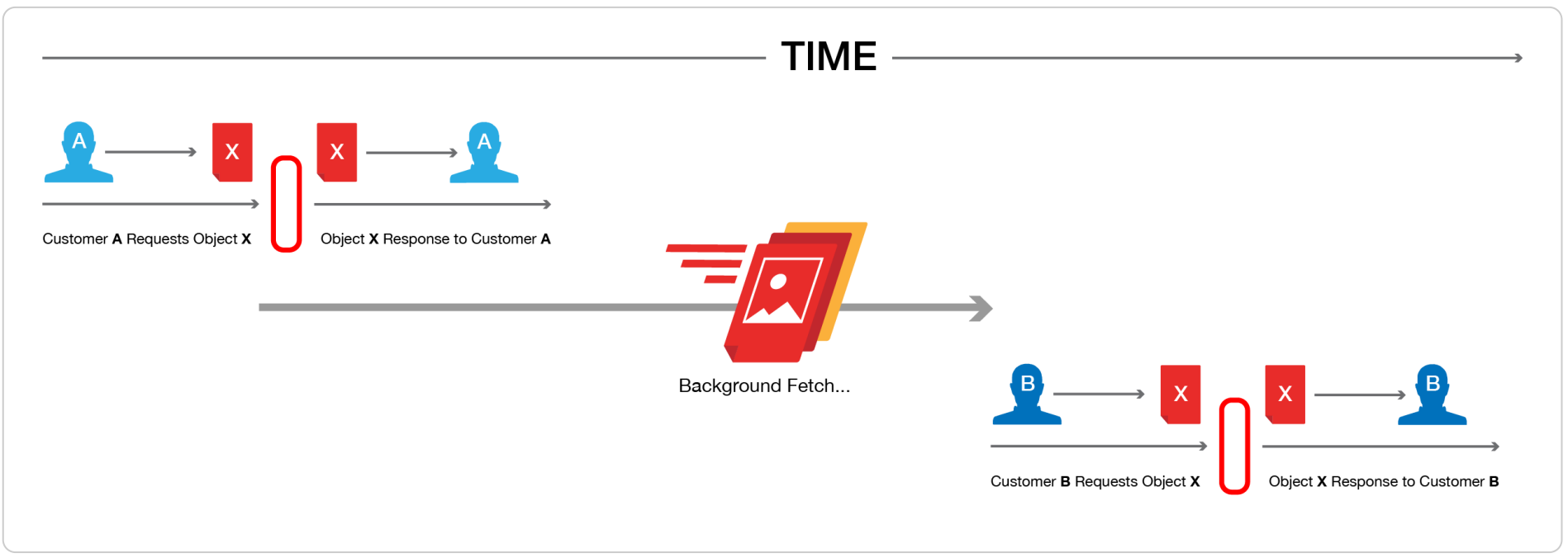
Fastly can optionally serve stale content when there is a problem with your origin server or if new content is taking a long time to fetch from your origin server. For example, if Fastly can't contact your origin server, our POPs will continue to serve cached content when users request it. These features are not enabled by default.

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.



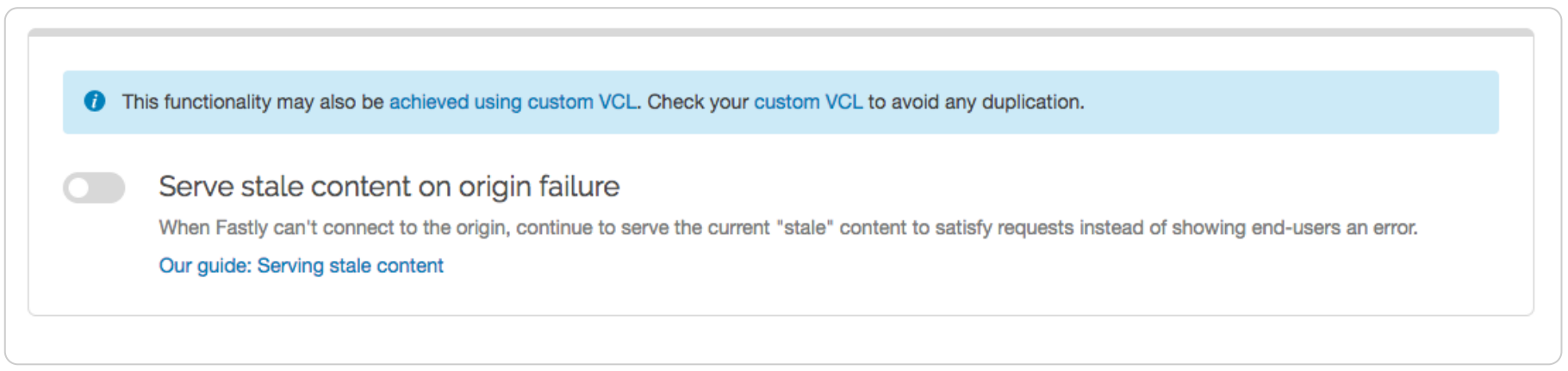
Fastly builds on the behavior proposed in RFC 5861 "HTTP Cache-Control Extensions for Stale Content" by Mark Nottingham, which is under consideration for inclusion in Google's Chrome browser.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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 (req.http.Fastly-FF) {
    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-age)") {
    # 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 origin is offline.</html>"};
    return(deliver);
  }

}

sub vcl_pass {
#FASTLY pass
}

sub vcl_log {
#FASTLY log
}
```

```
8
9
9
0
9
1
9
2
9
3
9
4
9
5
9
6
9
7
9
8
9
9
1
0
0
1
0
1
1
0
2
1
0
3
1
0
4
1
0
5
1
0
6
1
0
7
1
0
8
1
0
9
1
1
0
1
1
1
1
2
1
1
3
1
1
4
1
1
5
1
1
6
1
1
7
1
1
8
1
1
9
1
2
```

```
0
1
2
1
```

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.
- **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 longer than 3700s 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

https://docs.fastly.com/en/guides/streaming-miss

By default, Fastly limits cached file sizes to 2GB. Our Streaming Miss feature allows you to increase that limit for large files up to 5GB in size. Streaming Miss can be performed on any kind of large file, not just streamed video files.

TIP: Consider taking advantage of our Segmented Caching feature for files larger than the 5GB limit offered by Streaming Miss.

How Streaming Miss works

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).

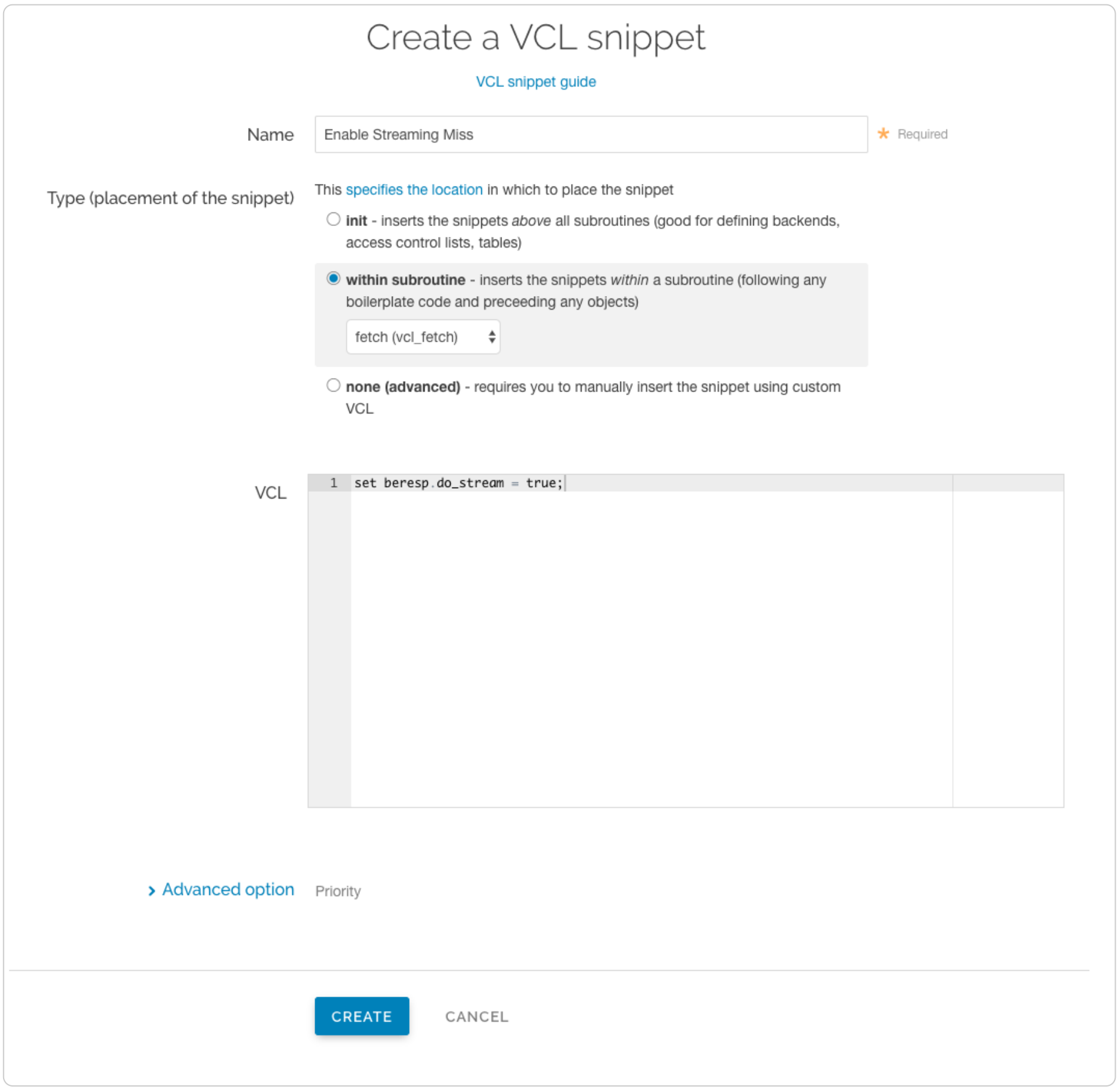
When Streaming Miss is enabled, the maximum cache file size is slightly below 5GB (specifically 5,368,578,048 bytes). Without it, the maximum cache file size is slightly below 2GB (specifically 2,147,352,576 bytes).

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 using a VCL Snippet

Using a VCL Snippet, you can enable Streaming Miss by setting `beresp.do_stream` to true in `vcl_fetch`:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

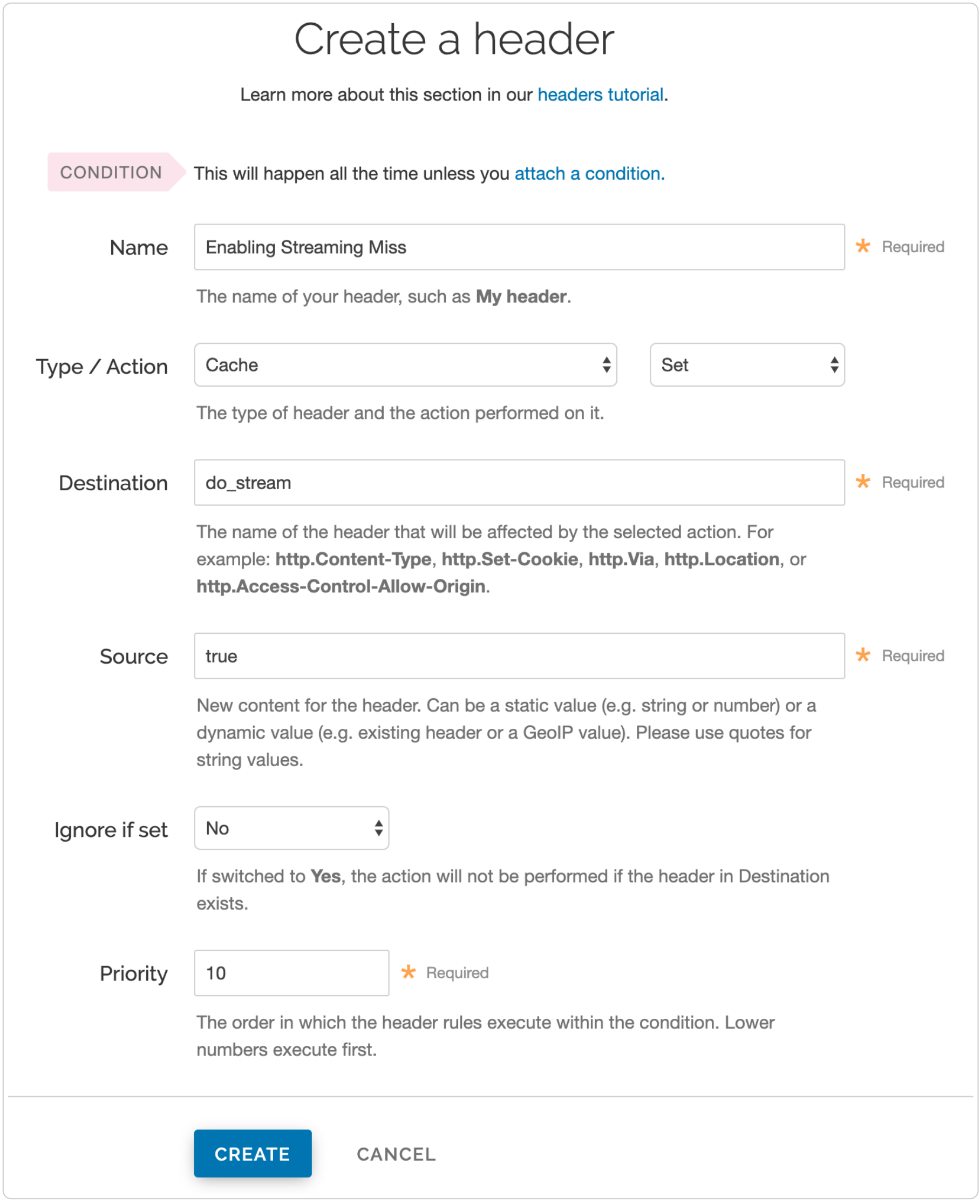


6. In the **Name** field type an appropriate name (e.g., `Enabling Streaming Miss`).
7. From the **Type (placement of the snippets)** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **fetch (vcl_fetch)**.
9. In the **VCL** field, add `set beresp.do_stream = true;`.
10. Click **Create** to create the snippet.
11. Click the **Activate** button to deploy your configuration changes.

Enable Streaming Miss using a header

You also can enable Streaming Miss by creating a new header via the web interface (this also can be controlled with conditions).

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type `do_stream`.
   - In the **Source** field, type `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, type 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.

Load balancing

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

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 datacenters, 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 more information, see 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:

- a **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 datacenter or on-premises location.
- a **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 datacenters 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

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 more information, see 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>/pool -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

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 more information, see 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 'address=<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>/serve
r/<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

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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.

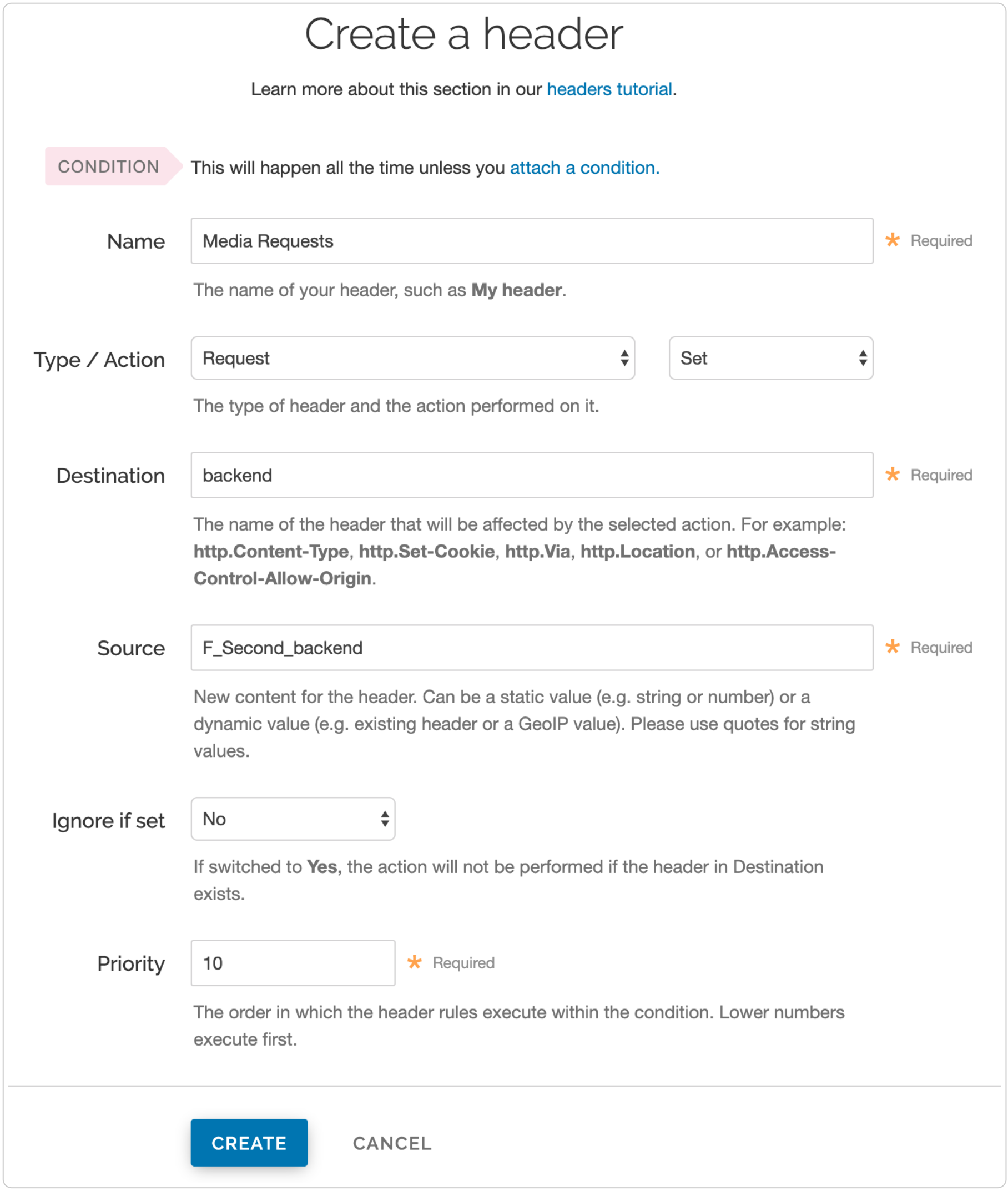
Setting conditions on origin servers

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.

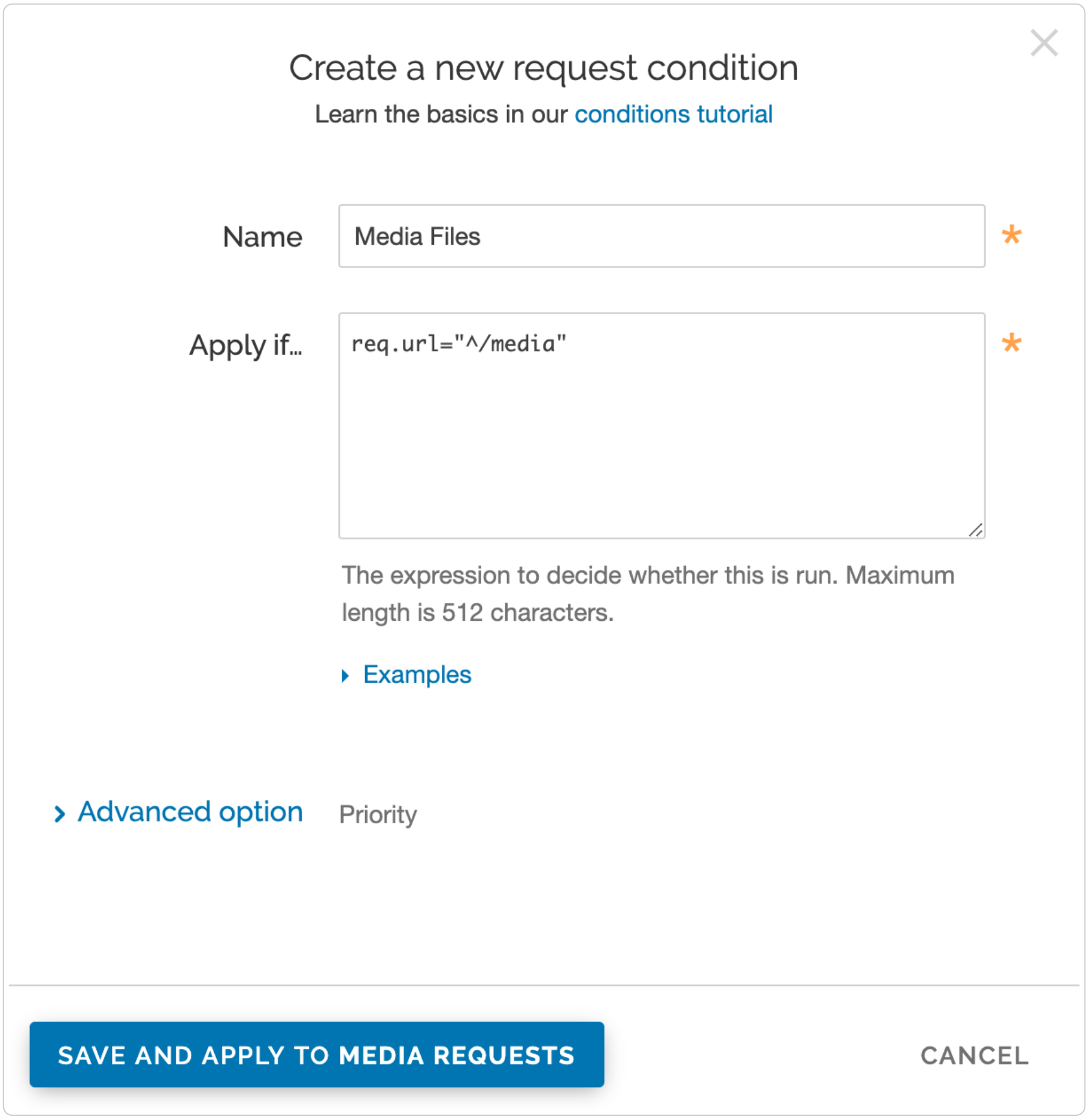
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. In the **Headers** 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, type 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, type the name of the header that will be affected by the action (for example, `backend`).
   - In the **Source** field, type 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, type a descriptive name for the new condition (for example, `Media Files`).
   - In the **Apply if** field, type 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
}
```

Purging

These articles describe how to purge cache.

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

Authenticating URL purge requests via API

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.

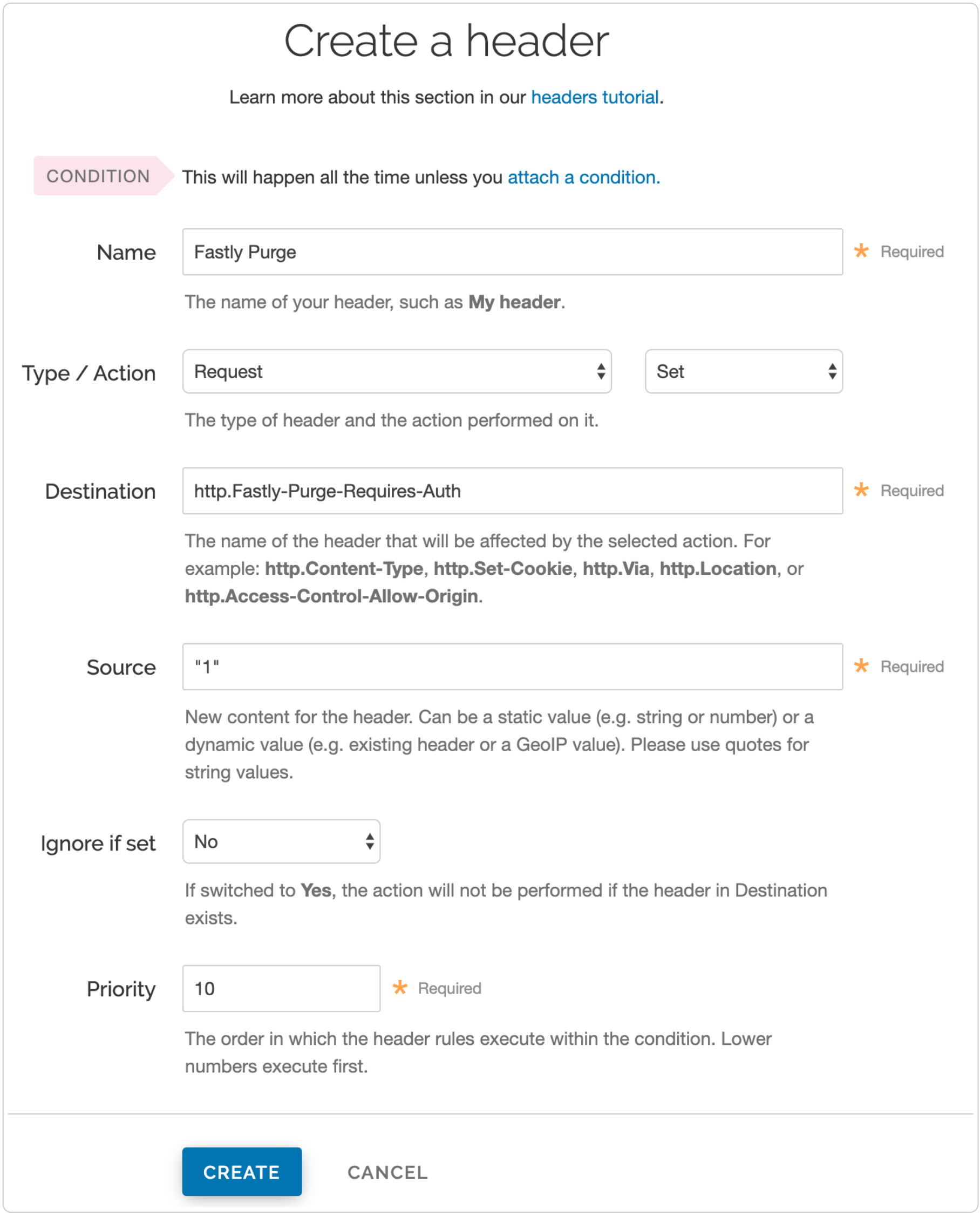
NOTE: All purge requests other than URL purges require authentication by default, as indicated in the API documentation.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

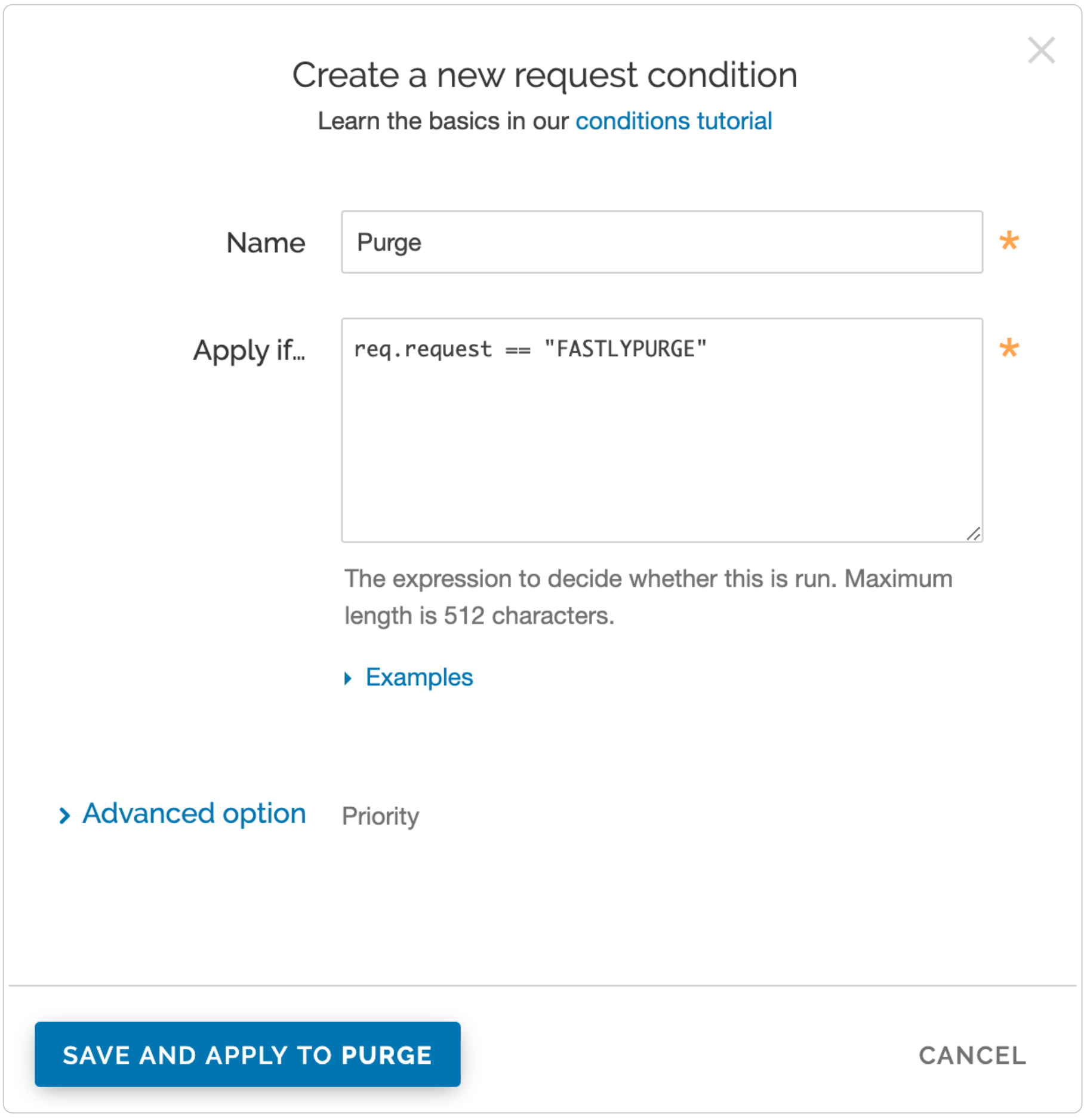


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type the name of your header rule (for example, `Fastly Purge`).
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Fastly-Purge-Requires-Auth`.
   - In the **Source** field, type `"1"`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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.

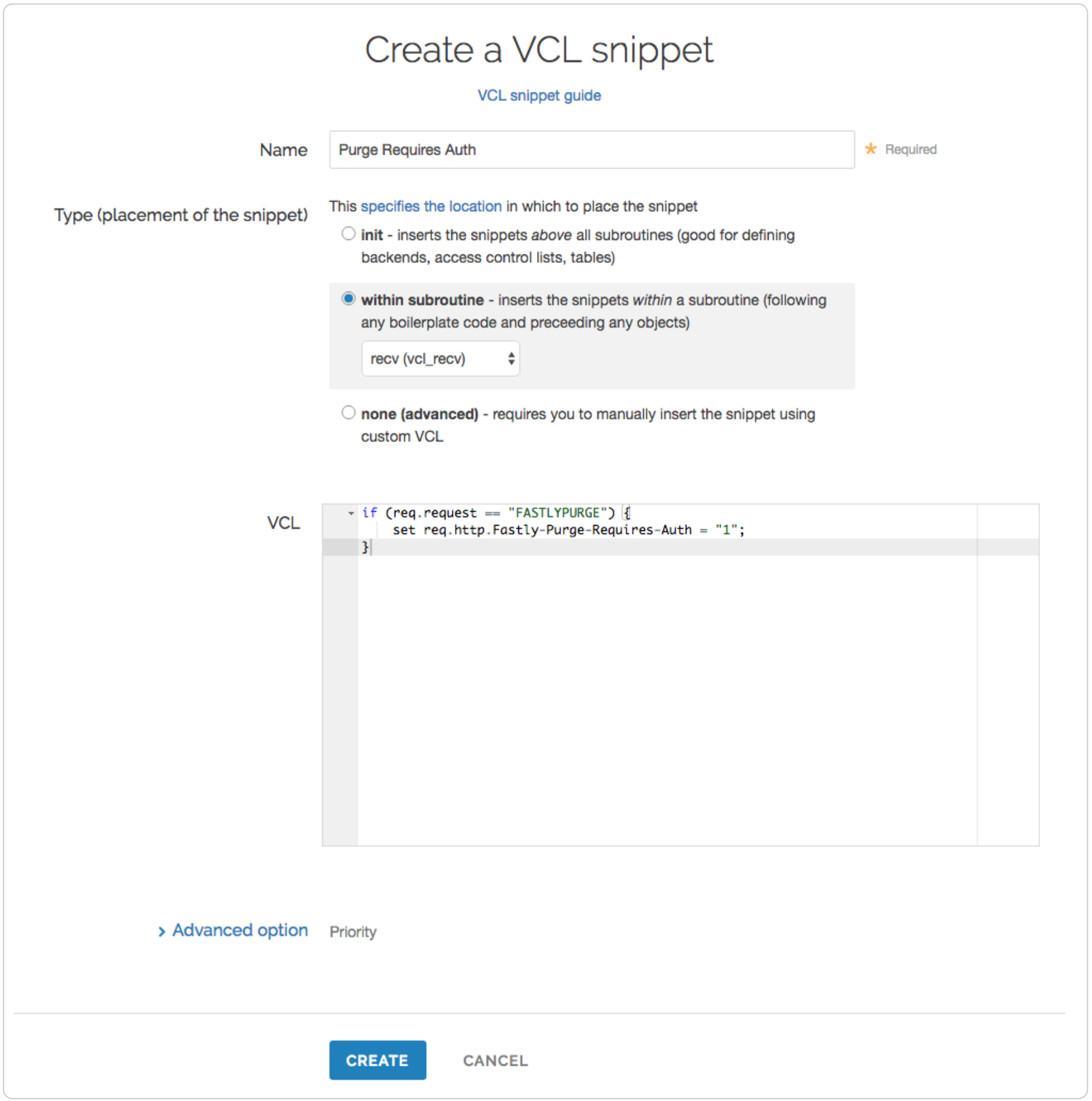


2. Fill out the **Create a new request condition** fields as follows:
   - In the **Name** field, type a descriptive name for the new condition (for example, `Purge`).
   - In the **Apply if** field, type `req.request == "FASTLYPURGE"`.
3. Click the **Save and apply to** button.
4. Click the **Activate** 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:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type an appropriate name (e.g., Purge Require Auth).
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:

```
1  if (req.request == "FASTLYPURGE") {
2      set req.http.Fastly-Purge-Requires-Auth = "1";
3  }
```

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

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, we recommend purging using a POST request with a secure Fastly API URL. This will ensure that your API token in the header is encrypted in transit. The request will look like this: `curl -X POST -H Fastly-Key:FASTLY_API_TOKEN https://api.fastly.com/purge/<your_url_here>`.

Getting started with surrogate keys

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

Efficient cache invalidation is an essential part of keeping your website fast. Purging too much cache using purge all may 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 allow you to selectively purge related content. Using the `Surrogate-Key` header, you can tag a group of objects with a key term, a string of any characters you want, and then use that term to purge multiple pieces of content at once. This process can occur automatically within your application, making it easier to cache and purge content that changes rapidly and unpredictably.

ℹ **NOTE:** 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.

Understanding surrogate keys

After you've signed up for Fastly and added one or more services, you can start examining how your origin server responds to requests. When your origin server responds to an HTTP request for content, it's because Fastly hasn't yet cached that content or the cache has expired. Your server's response to the request will resemble the example shown below. (Note that you can use the curl command to inspect any of your server's responses.)

```
1  HTTP/1.1 200 OK
2  Content-Type: text/html
3  Connection: keep-alive
4  ...
```

To control how your content is served to users and cached by Fastly, you can add to or modify the headers that are included in your origin server's response. The `Surrogate-Key` header is one of the headers that you can add to the response. It allows you to "tag" an object, such as an image or a blog post, with one or more keys. When the object changes, you can reference the key in a purge request to remove the object from the cache.

You can add space-delimited strings to the `Surrogate-Key` header, like this:

```
1  HTTP/1.1 200 OK
2  Surrogate-Key: key1 key2 key3
3  Content-Type: text/html
4  ...
```

This response contains three surrogate keys: `key1`, `key2`, and `key3`. 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:

```
1  GET /blog/ HTTP/1.1
2  Host: www.example.com
3
4  HTTP/1.1 200 OK Content-Type: text/html
5  Content-Length: 1234
6  Surrogate-Key: mainpage template-a
```

```
GET /blog/article/fastly-rocks HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 2345
Surrogate-Key: template-a article-fastly-rocks
```

In this example, there are two objects (`/blog` and `/blog/article/fastly-rocks`) with three keys (`mainpage`, `template-a`, and `article-fastly-rocks`). Two of the keys (`mainpage` and `article-fastly-rocks`) are associated with a single object and a third key (`template-a`) is associated with both objects.

Purging objects with surrogate keys

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` object from the cache. On the other hand, purging the `template-a` key would remove both the `/blog` and `/blog/article/fastly-rocks` objects from the cache.

You can use the Fastly web interface to manually purge objects via key, or you can use our Purge API. If you're using Fastly to cache your API, check out the guide on purging API cache with surrogate keys to learn how surrogate keys can help you.

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. We describe how to use the Fastly web interface in our guide to generating Surrogate-Key headers based on URLs (we have a separate guide for Amazon S3 origins).

However, it is more useful to make your surrogate key associations on your own application server and include them in the HTTP response that you send to Fastly. This way, you can assign exactly the keys that you want on every response.

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 size limitations. Individual surrogate keys may not exceed 1,024 bytes in length, and a `Surrogate-Key` header value (comprising one or more space-separated keys) must not exceed 16,384 bytes in length. If either of these limits is reached while parsing a `Surrogate-Key` header, the key currently being parsed and all keys following it within the same header will be ignored.

Logging purge requests

https://docs.fastly.com/en/guides/logging-purge-requests

If you've set up remote log streaming, you can log URL purge requests by adding the following VCL to the `vcl_recv` subroutine. For example:

```
if (req.method == "FASTLYPURGE") {
  log {"syslog req.service_id <log name> :: "} "event=purge"" service="req.service_id;
}
```

The `log name` must match the name of your logging endpoint in the Fastly web interface.

You can add the VCL via custom VCL or VCL Snippets.

NOTE: Purge all requests are logged in event logs. Key purge requests can't be logged.

Purging API cache with surrogate keys

https://docs.fastly.com/en/guides/purging-api-cache-with-surrogate-keys

Fastly makes it possible for you to cache your API so you can accelerate the performance of your service-oriented architecture. Of course, caching your API is one thing - efficiently invalidating the API cache is another matter entirely. If you've already enabled API caching and implemented API cache control, you've probably run into this problem, which was aptly described by Phil Karlton:

> *There are only two hard things in computer science: cache invalidation and naming things.*

This guide explains how to use the Fastly API to purge your API cache with surrogate keys. Surrogate keys allow you to reduce the complexity of caching an API by combining multiple cache purges into a single key-based purge.

What's a surrogate key?

Surrogate keys allow you to selectively purge related content. Using the `Surrogate-Key` header, you can "tag" an object, such as an image or a blog post, with one or more keys. When Fastly fetches an object from your origin server, we check to see if you've specified a `Surrogate-Key` header. If you have, we add the response to a list we've set aside for each of the keys.

When you want to purge all of the responses associated with a key, issue a key purge request and all of the objects associated with that key will be purged. This makes it possible to combine many purges into a single request. Ultimately, it makes it easier to manage categorically related data.

To learn more about surrogate keys and to see how you can integrate them into your application, see our guide on getting started with surrogate keys.

Example: Purging categories

To see how surrogate keys work in conjunction with an API endpoint, imagine you have an online store and an API endpoint that returns the details of a product. When a user wants to get information about a specific product, like a keyboard, the request might look like this:

```
GET /product/12345
```

If your API is using Fastly and the response is not already cached, Fastly will make a request to your API's origin server and receive a response like this:

```
1  HTTP/1.1 200 OK
2  Content-Type: text/json
3  Cache-Control: private
4  Surrogate-Control: max-age=86400
5  Surrogate-Key: peripherals keyboards
6
7  {id: 12345, name: "Uber Keyboard", price: "$124.99"}
```

You knew that entire product categories would occasionally need to be purged, so you thoughtfully included the `peripherals` and `keyboards` product categories as keys in the `Surrogate-Key` header. When Fastly receives a response like this, we add it to an internal map, strip out the `Surrogate-Key` header, cache the response, and then deliver it to the end user.

Now imagine that your company decides to apply a 10% discount to all peripherals. You could issue the following key purge to invalidate all objects tagged with the `peripherals` surrogate key:

```
PURGE /service/:service_id/peripherals
```

When Fastly receives this request, we reference the list of content associated with the `peripherals` surrogate key and systematically purge every piece of content in the list.

Relational dependencies

Your API can use surrogate keys to group large numbers of items that may eventually need to be purged at the same time. Consider the example presented above. The API for your online store could have surrogate keys for product types, specific sales, or manufacturers.

From this perspective, the `Surrogate-Key` header provides Fastly with information about relations and possible dependencies between different API endpoints. Wherever there's a relation between two different types of resources in an API, there might be a good reason to keep them categorized by using a surrogate key.

Example: Purging product reviews and action shots

To learn how surrogate keys can help with relational dependencies, imagine that your online store wants to allow buyers to post product reviews and "action shots" depicting the products in use. To support these new features, you'll need to change your API. First, you'll need to create a new `review` endpoint:

```
1  GET /review/:id
2  POST /review
```

Next, you'll need to create a new `action_shot` endpoint:

```
1  POST /product/:id/action_shot
2  GET /product/:id/action_shot/:shot_id
```

Since both of the new endpoints refer to specific products, they'll need to be purged when relevant product information changes. Surrogate keys are a perfect fit for this use case. You can implement them by modifying the `review` and `action_shot` to return the following header:

```
Surrogate-Key: product/:id
```

This relates each of the endpoints to a specific product in the cache (where `:id` the product's unique identifier). When the product information changes, your API issues the following surrogate key purge:

```
POST https://api.fastly.com/service/:service_id/purge/product/:id
```

When Fastly receives this request, we purge each of the related endpoints at the same time.

Variations on a theme

You'll also want to consider using surrogate keys if your API has many different endpoints that all derive from a single source. Any time the source data changes, each of the endpoints associated with it will need to be purged from the cache. By associating each of the endpoints with a surrogate key, a single purge can be issued to purge them from the cache when the source changes.

Example: Purging product images

To understand how this works, imagine that your online store has an API endpoint for retrieving product images in various sizes:

```
GET /product/:id/image/:size
```

This endpoint returns an image of the appropriate `:size` (e.g., `small`, `medium`, `large`) for the product of the given `:id`. To save disk space, you opt to have the API generate each specifically sized image from a single source image using an imaging library like ImageMagick. Since the sales and marketing team uses the API to upload new product images, you set up the endpoint to include a surrogate key:

```
Surrogate-Key: product/:id/image
```

When the PUT endpoint for uploading a product image is called, the API sends the following purge request:

```
POST https://api.fastly.com/service/:service_id/purge/product/:id/image
```

When Fastly receives this request, we purge all size variations of the product image.

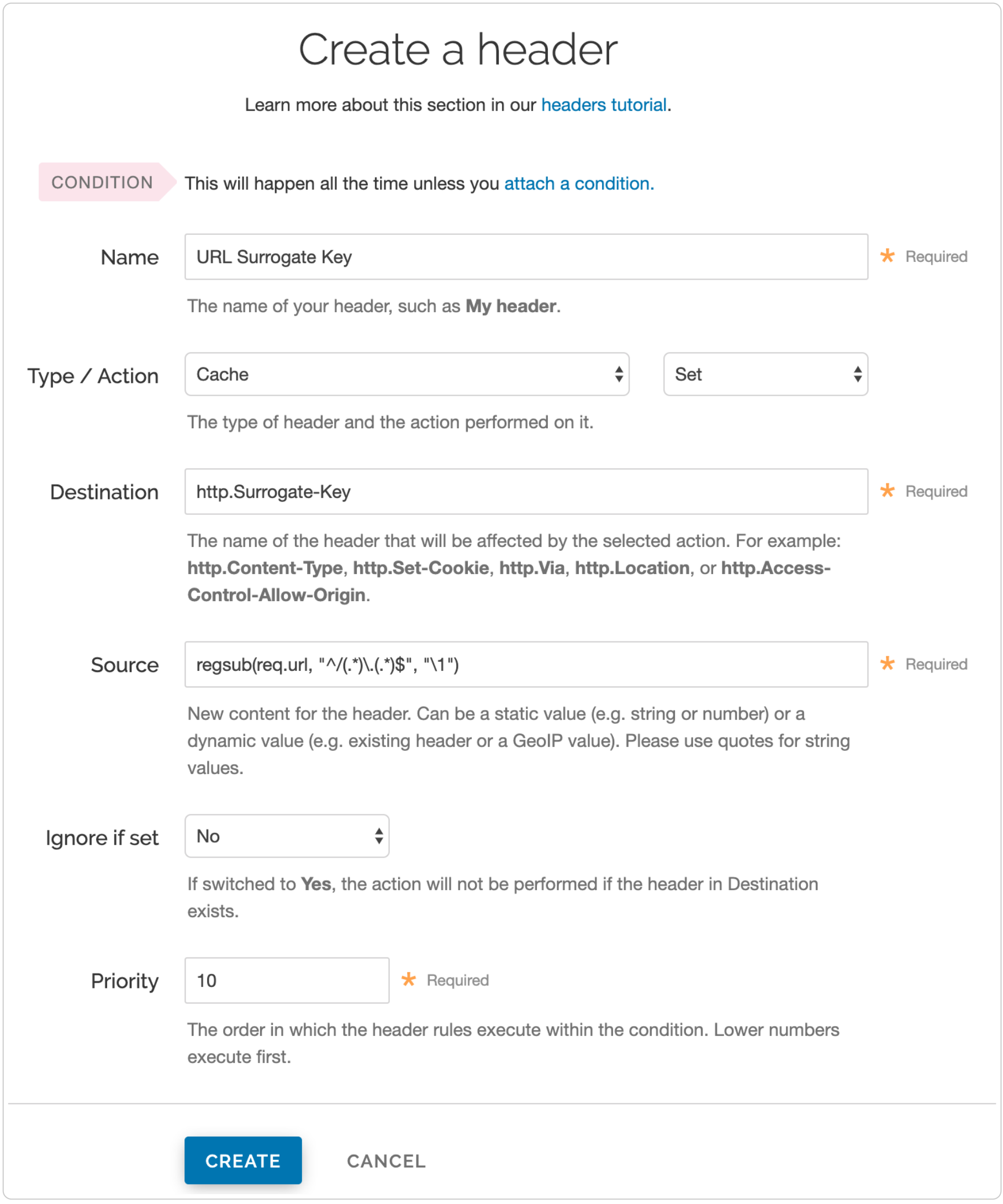
Setting Surrogate-Key headers based on a URL

https://docs.fastly.com/en/guides/setting-surrogate-key-headers-based-on-a-url

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.

Follow these instructions to set Surrogate-Key headers based on a URL:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type `http.Surrogate-Key`.
   - In the **Source** field, type `regsub(req.url, "^/(.*)\.(.*)$", "\1")`. This will accept a URL that looks like `/foo.html` and will create the Surrogate-Key `foo`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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.

Setting Surrogate-Key headers for Amazon S3 origins

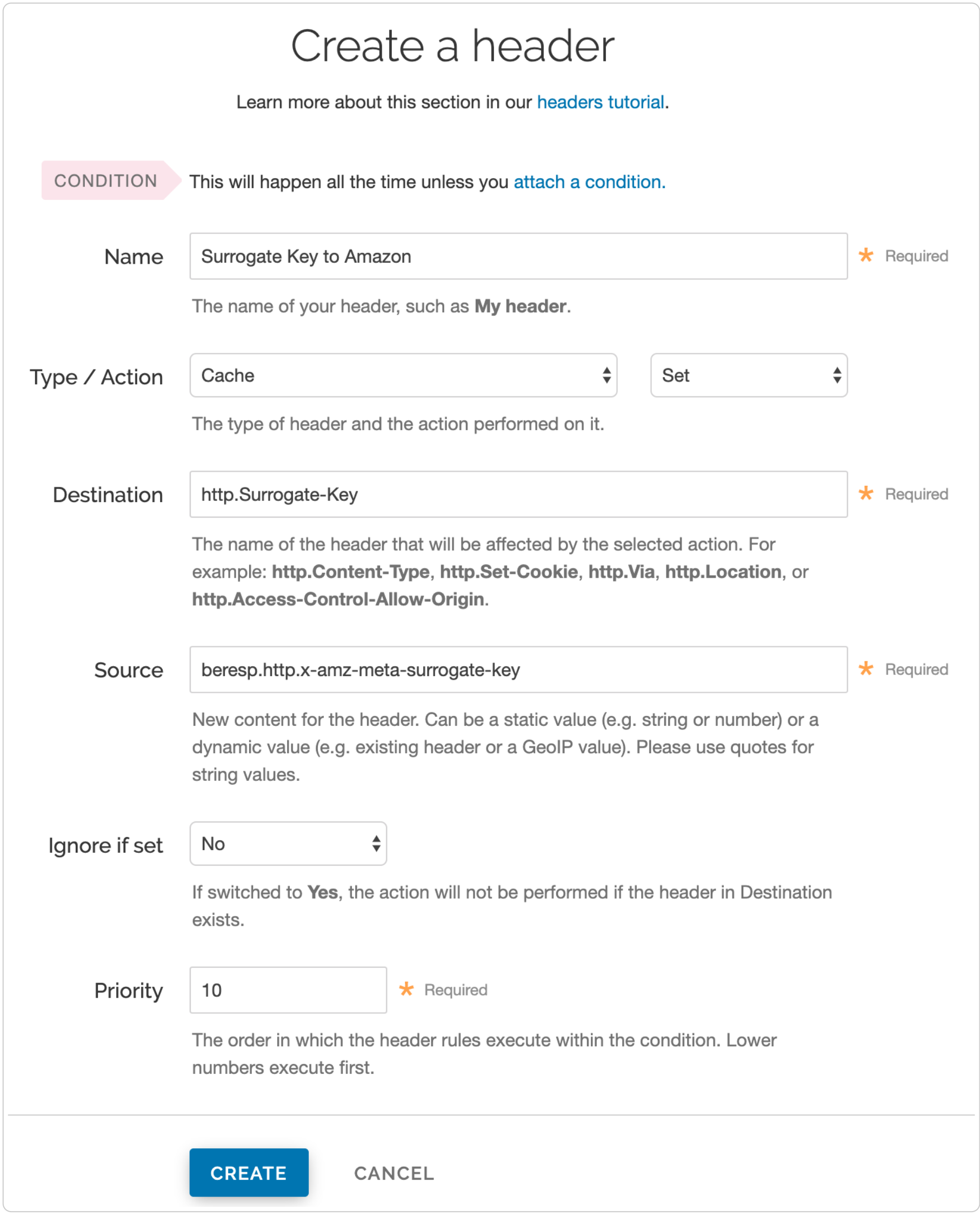
https://docs.fastly.com/en/guides/setting-surrogate-key-headers-for-amazon-s3-origins

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type `http.Surrogate-Key`.
   - In the **Source** field, type `beresp.http.x-amz-meta-surrogate-key`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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.

Single purges

https://docs.fastly.com/en/guides/single-purges

Fastly provides several levels of cache purging. You can purge something as small as a single URL via the "Purge URL" command or as large as all content under a service via the "Purge All" command. You can also selectively purge content via key-based purging using the "Purge Key" command. We also provide a purging feature called Soft Purge that allows you to mark content as outdated (stale) instead of permanently deleting it from Fastly's caches.

TIP: To mark content as outdated instead of permanently deleting it, check out our Soft Purge feature. You may also be interested in our wildcard purging.

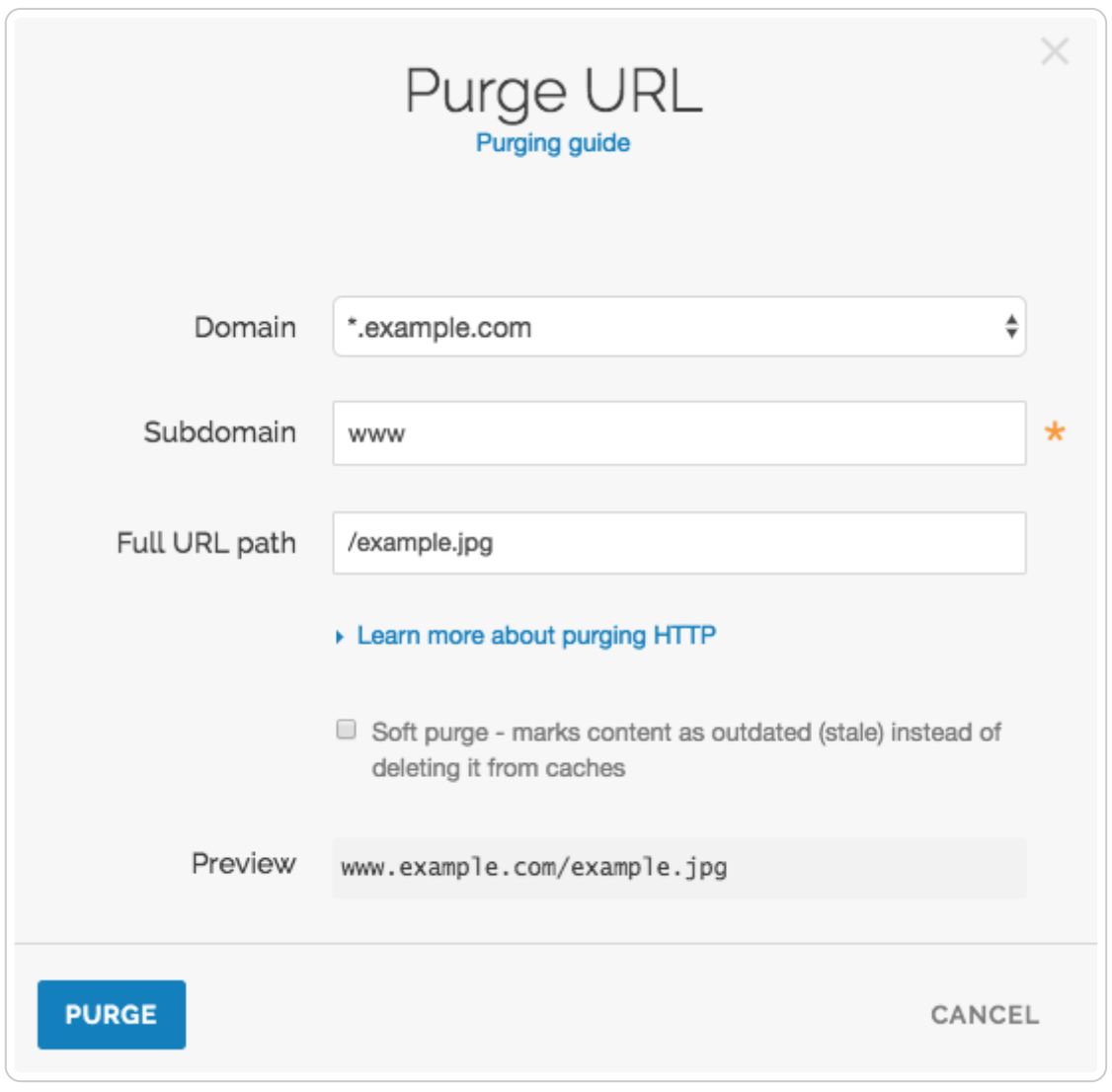
Purging via the user interface

To purge content using the Fastly web interface, choose one of the purging methods below.

Purging a URL

To purge a single URL, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. From the **Purge** menu, select **Purge URL**. The Purge URL window appears.

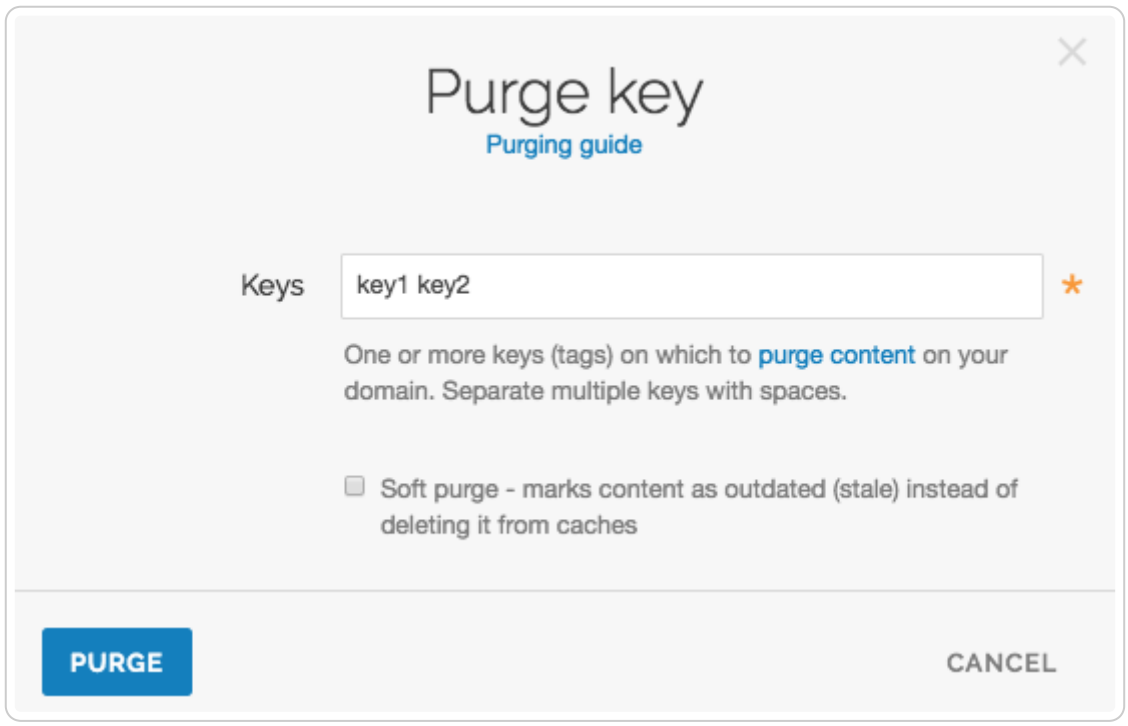


4. From the **Domain** menu, select the domain on which your content resides. If the domain you select is a wildcard domain (e.g., `*.example.com`) the Subdomain field will appear.
5. If the **Subdomain** field appears, type the subdomain to purge for the wildcard domain you've selected (e.g., `www`).
6. In the **Full URL path** field, type 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.

Purging with keys

To purge content with surrogate keys, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. From the **Purge** menu, select **Purge key**. The Purge key window appears.



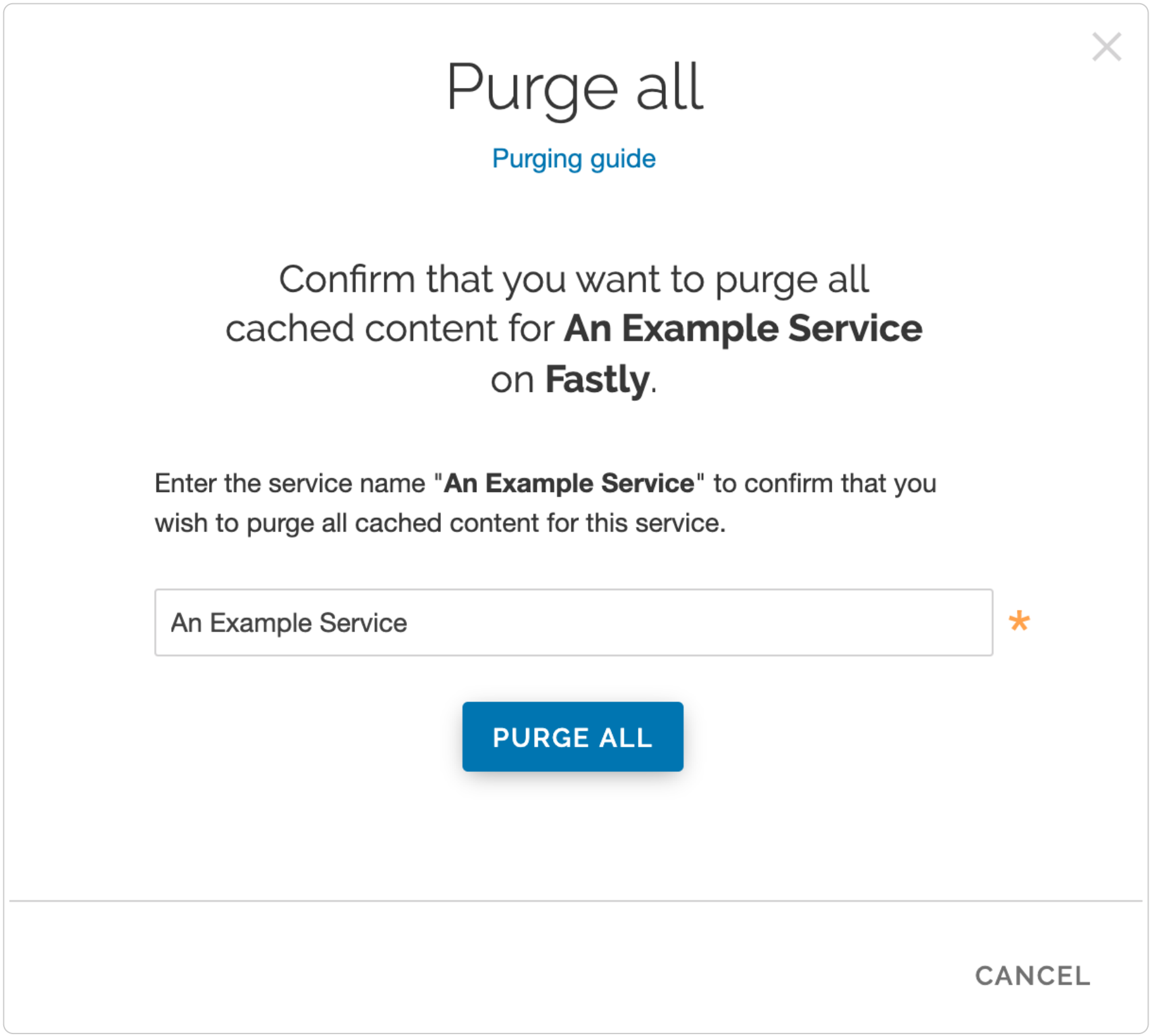
4. In the **Keys** field, type one or more surrogate keys. Use spaces to separate multiple keys.
5. Optionally select the Soft purge checkbox to mark your content as outdated instead of deleting it from cache.
6. Click the **Purge** button.

Purging all content

⚠ **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 cache repopulates.

To instantly purge all content under your service, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
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** button.

Purging via API

The syntax for purging a service through the API can be found in the Purging section of the API documentation.

Soft purges

https://docs.fastly.com/en/guides/soft-purges

Fastly provides a Soft Purge feature that allows you to mark content as outdated (stale) instead of permanently purging and thereby deleting it from Fastly's caches. Objects invalidated with Soft Purge will be treated as outdated (stale) while Fastly fetches a new version from origin. You can purge by URL or by surrogate key using Soft Purge.

Before using Soft Purge, we recommend you implement one of the following revalidation methods:

- 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 implement 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/service/<SID>/purge/<S-Key>
```

Wildcard purges

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.

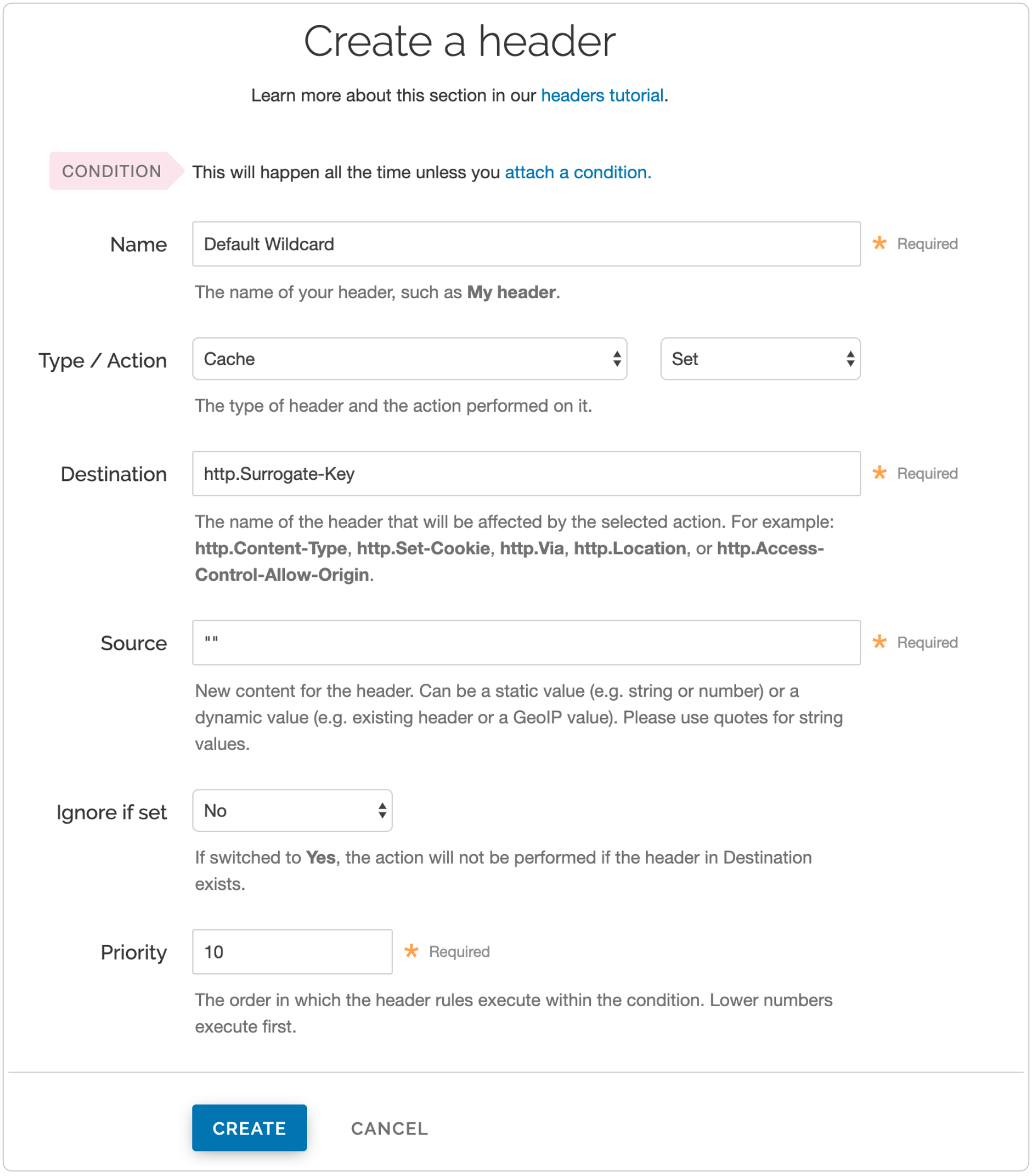
Via the web interface

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

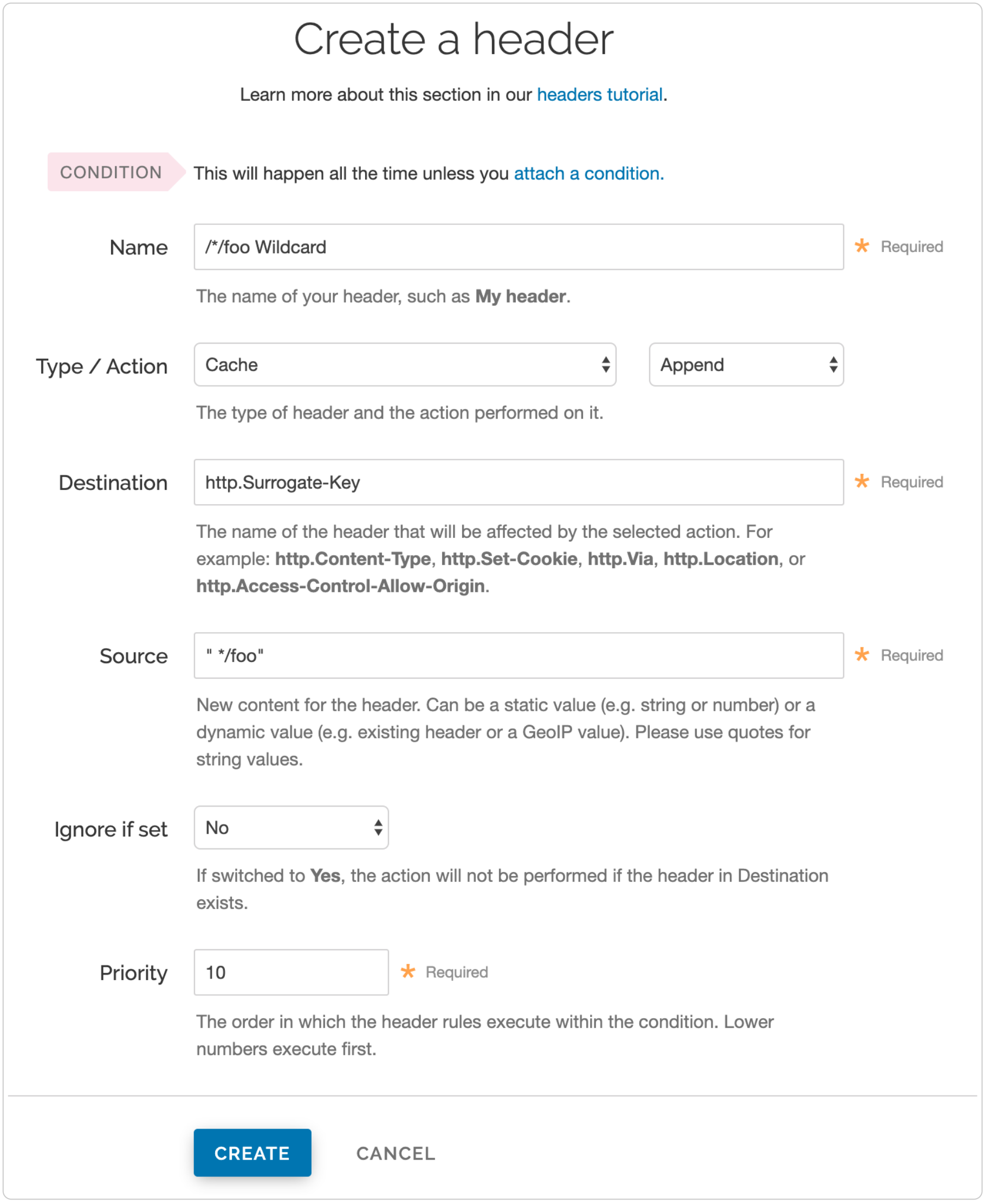


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Default Wildcard`. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache** and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Surrogate-Key`.
   - In the **Source** field, type `""`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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.



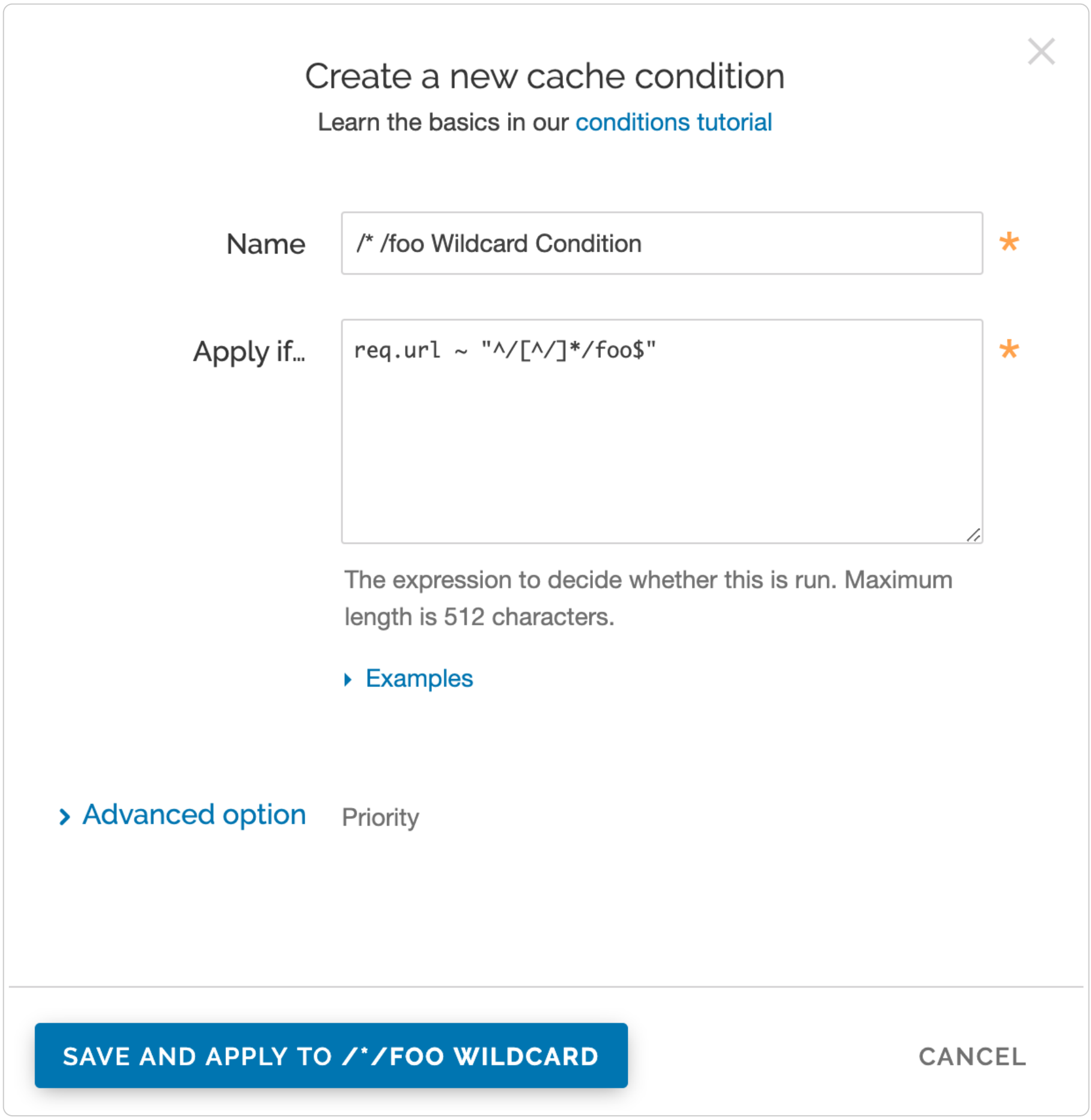
2. Fill out the **Create a header** fields as follows:
   - In the **Description** field, type `/*/foo Wildcard`. This name is displayed in the Fastly web interface.
   - From the **Type** menu, select **Cache**, and from the **Action** menu, select **Append**.
   - In the **Destination** field, type `http.Surrogate-Key`.
   - In the **Source** field, type `" */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, type `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, type `/*/foo Wildcard Condition`.
   - In the **Apply if** field, type `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.

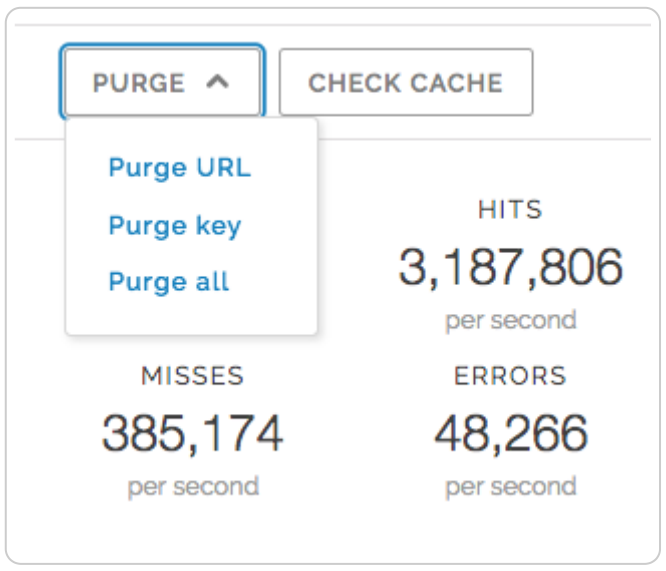
Some other examples for URL wildcard conditions:

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

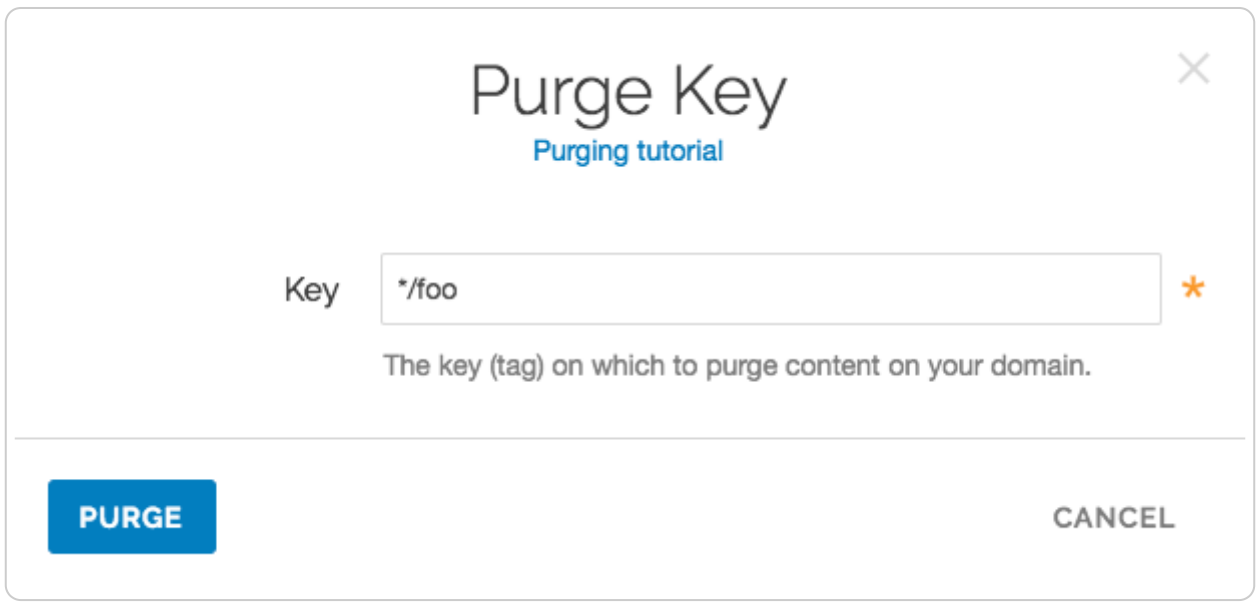
Ready to purge that wildcard? You can do this through the UI using the steps below.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the **Purge** menu, select **Purge Key**.



   The Purge Key window appears.

3. In the **Keys** field, type the surrogate key you want to purge. Continuing with our example, you would type `*/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 1
454101973) (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.

Custom VCL

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

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

Accept-Language header VCL features

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

```
1  set req.http.Normalized-Language =
2    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:

```
1  set req.http.Normalized-Language =
2    accept.language_lookup("en-US:nl", "nl", req.http.Accept-Language);
3  if (!req.http.Normalized-Language) {
4    # User will get Dutch even if he doesn't want it!
5    # (Because none of our languages were acceptable)
6    set req.http.Normalized-Language = "nl";
7  }
```

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

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

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

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. Basic authentication can be implemented using custom VCL or VCL Snippets.

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

Follow the steps below to set up basic authentication for your service:

1. Create an Edge Dictionary with a list of Base64-encoded usernames and passwords. You can include the usernames in plaintext for reference. You can also use the API to create the Edge Dictionary and add dictionary items, and you can use custom VCL as shown below.

```
table customer_keys {
  "Basic am9lOjQzNEAvMzkyIzgyPzk2": "joe",
  "Basic bWlrZTo4MjM0MzNzWjQ0SDZlNw==": "mike"
}
```

   The first value in the key pair is the username and password Base64-encoded. You can generate this in a terminal application as shown below. In this example, the username is `joe`, and the password is `434@/392#82?96`.

```
$ echo -ne joe:434@/392#82?96 | base64
am9lOjQzNEAvMzkyIzgyPzk2
```

   The result (`am9lOjQzNEAvMzkyIzgyPzk2`) is the second half of the first key pair (`Basic am9lOjQzNEAvMzkyIzgyPzk2`).

2. In `vcl_recv`, create a table lookup to authorize customer credentials against those in the table.

```
##table lookup from customer_keys dictionary, plus part in vcl_error
if(! table.lookup(customer_keys, req.http.Authorization) ) {
  error 401 "Restricted";
}

```

3. In `vcl_error`, create your Custom 401 Restricted HTML page.

```
## Start 401 custom code
if (obj.status == 401) {
  set obj.http.Content-Type = "text/html; charset=utf-8";
  set obj.http.WWW-Authenticate = "Basic realm=Secured";
  synthetic {"

 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/1999/REC-html401-19991224/loose.dtd">

 <HTML>
 <HEAD>
 <TITLE>Error</TITLE>
 <META HTTP-EQUIV='Content-Type' CONTENT='text/html;'>
 </HEAD>
 <BODY><H1>401 Unauthorized (varnish)</H1></BODY>
 </HTML>
 "};
  return (deliver);
} # End custom 401 code
```

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.

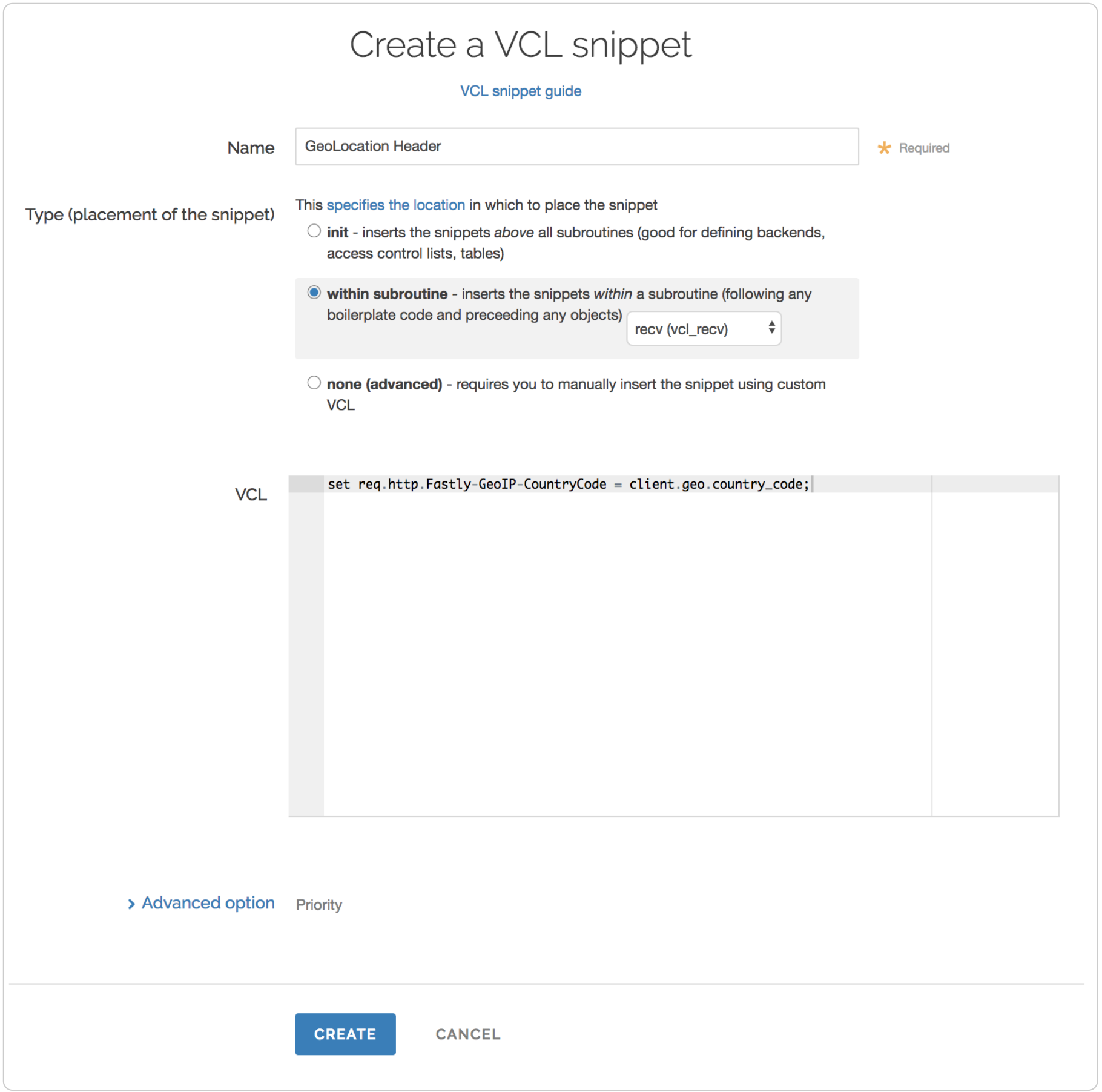
Creating location-based tagging

https://docs.fastly.com/en/guides/creating-location-based-tagging

You can set custom HTTP headers in your varnish configuration (VCL) based on the variables we expose. Use the geolocation features we have built into Varnish to create location-based tagging. We provide a list of geographic information based on a client's IP address. For a complete list of available geolocation variables, read about which geolocation features are accessible via VCL.

In the example below, an HTTP header Fastly-GeoIP-CountryCode is created with the two letter country code of the client's IP address using VCL Snippets.

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type an appropriate name (e.g., GeoLocation Header).
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:

```
set req.http.Fastly-GeoIP-CountryCode = client.geo.country_code;
```

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

Custom responses that don't hit origin servers

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

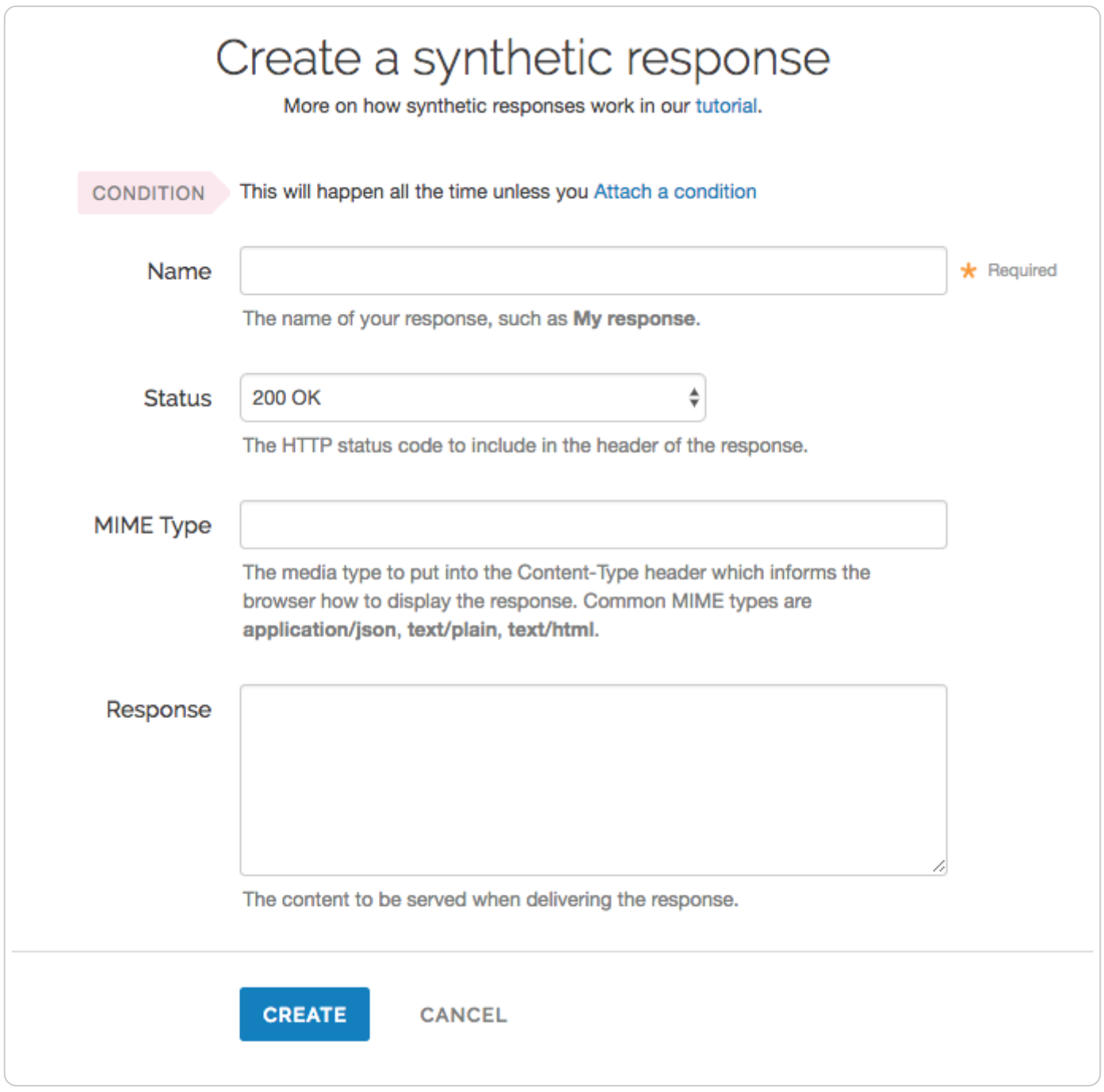
Fastly provides features that allow you to quickly enable and configure responses for a robots.txt file and 404 and 503 errors. For more information, see our guides on creating and customizing a robots.txt file and creating error pages with custom responses.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

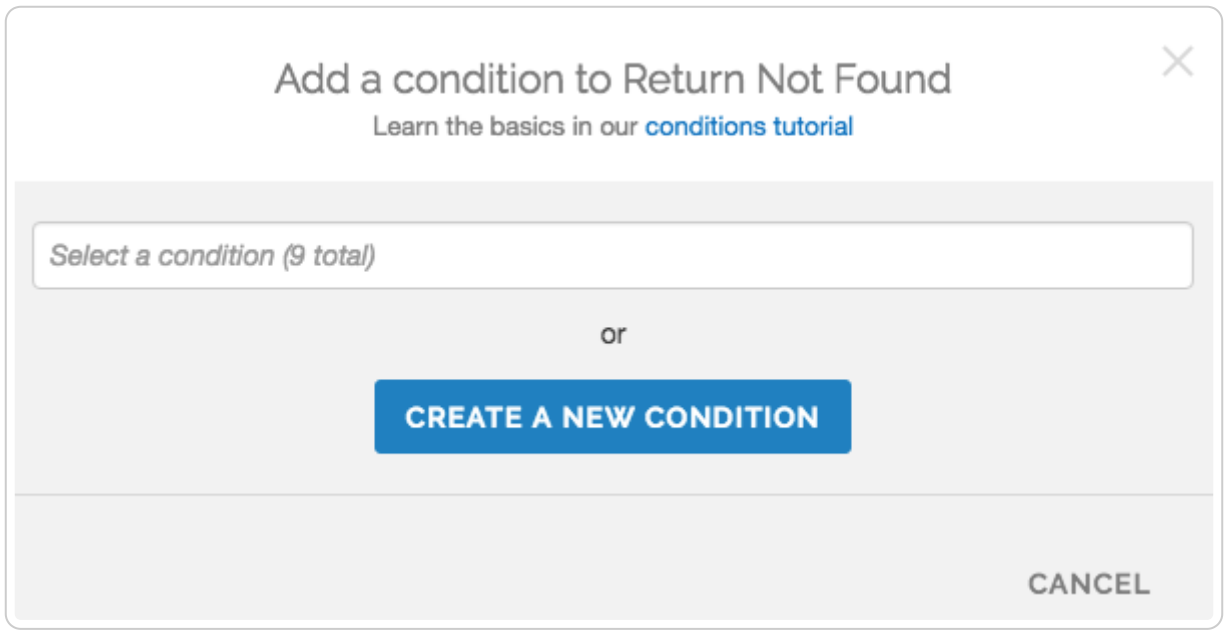


6. Fill out the **Create a synthetic response** fields as follows:
   - In the **Name** field, type a human-readable name for the response. For example `Return Not Found`.
   - From the **Status** menu, select an HTTP code to return to the client. For example, `404 Not Found`.
   - In the **MIME Type** field, type the MIME type of the response. For example, `text/html`.
   - In the **Response** field, type 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.

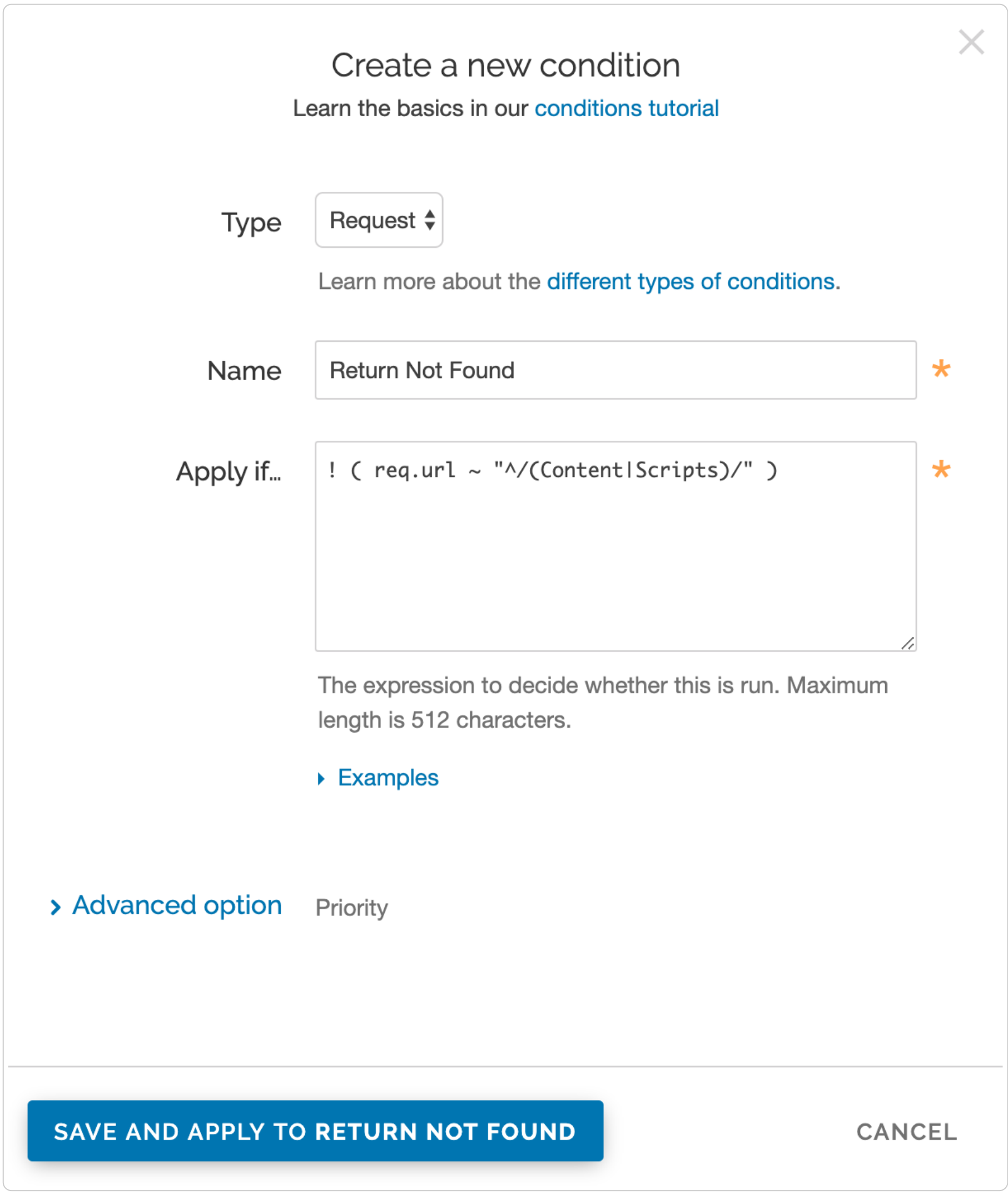
Creating the request condition

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

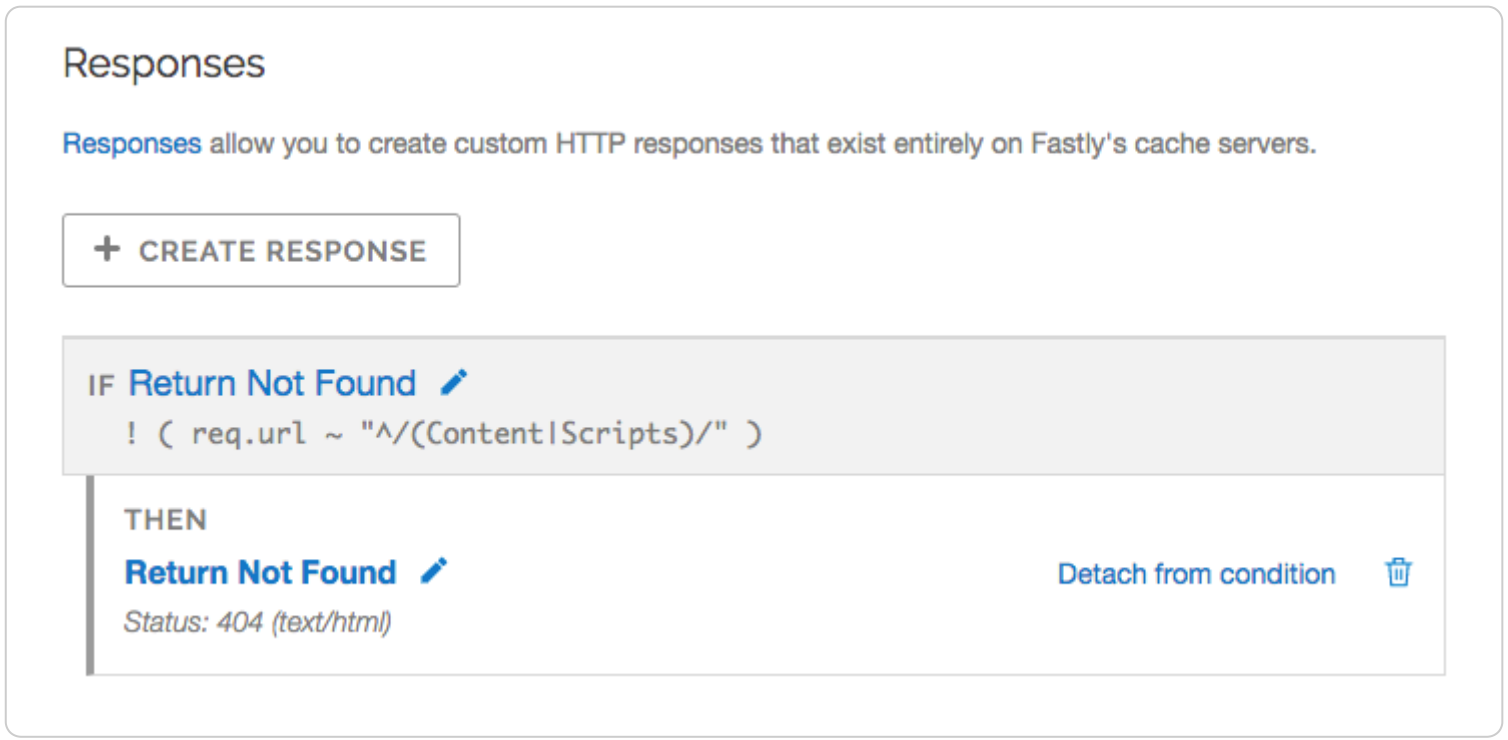
1. Click the **Attach a condition** link next to the response that you just created. The Add a condition window appears.



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, type a human-readable name for the condition. For example, `Return Not Found`.
   - In the **Apply if** field, type 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
```

Delivering different content to different devices

https://docs.fastly.com/en/guides/delivering-different-content-to-different-devices

The easiest way to deliver different content based on the device being used is to rewrite the URL of the request based on what the user agent is. We've written an article that describes how to change the URL based on conditions using our user interface but in pure VCL it would look something like this:

```
1  sub vcl_recv {
2    if (req.http.User-Agent ~ "(?i)ip(hone|od)") {
3      set req.url = "/mobile" req.url;
4    } elsif (req.http.User-Agent ~ "(?i)ipad") {
5      set req.url = "/tablet" req.url;
6    }
7  #FASTLY recv
8  }
```

Obviously the code fragment above doesn't contain a comprehensive list of mobile and tablet devices. Google has an official blog post on detecting Android mobile versus tablet and this VCL fragment from Varnish Software can detect several different types of devices quite reliably, although it doesn't include Windows mobile and tablet, Blackberry Playbook, and the Kindle user agents.

The most comprehensive device detection routine we've seen so far is this one:

```
# based on https://github.com/varnish/varnish-devicedetect/blob/master/devicedetect.vcl
sub detect_device {
  unset req.http.X-UA-Device;
  unset req.http.X-UA-Vendor;

  set req.http.X-UA-Device = "desktop";
  set req.http.X-UA-Vendor = "generic";

  # Handle that a cookie or url param may override the detection altogether
  if (req.url ~ "[&|?]device_force=([^&\s]+)") {
    set req.http.X-UA-Device = regsub(req.url, ".*[&|?]device_force=([^&\s]+).*", "\1");
  } elsif (req.http.Cookie ~ "(?i)X-UA-Device-force") {
    # ;?? means zero or one ;, non-greedy to match the first
    set req.http.X-UA-Device = regsub(req.http.Cookie, "(?i).*X-UA-Device-force=([^;]+);??.*", "\1");
    # Clean up our mess in the cookie header
    set req.http.Cookie = regsuball(req.http.Cookie, "(^|; ) *X-UA-Device-force=[^;]+;? *", "\1");
    # If the cookie header is now empty, or just whitespace, unset it
    if (req.http.Cookie ~ "^ *$") { unset req.http.Cookie; } # "$ # stupid syntax highlighter
  } else {
    if (req.http.User-Agent ~ "(?i)(ads|google|bing|msn|yandex|baidu|ro|career|)bot" ||
      req.http.User-Agent ~ "(?i)(baidu|jike|symantec)spider" ||
      req.http.User-Agent ~ "(?i)scanner" ||
      req.http.User-Agent ~ "(?i)(web)crawler") {
      set req.http.X-UA-Device = "bot";
    } elsif (req.http.User-Agent ~ "(?i)ipad") {
      set req.http.X-UA-Device = "tablet";
      set req.http.X-UA-Vendor = "apple";
    } elsif (req.http.User-Agent ~ "(?i)ip(hone|od)") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "apple";
    # how do we differ between an android phone and an android tablet?
    #   http://stackoverflow.com/questions/5341637/how-do-detect-android-tablets-in-general-useragent
    #   http://googlewebmastercentral.blogspot.com/2011/03/mo-better-to-also-detect-mobile-user.html
    } elsif (req.http.User-Agent ~ "(?i)android.*(mobile|mini)") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "android";
    # android 3/honeycomb was just about tablet-only, and any phones will probably handle a bigger page layout
    } elsif (req.http.User-Agent ~ "(?i)android") {
      set req.http.X-UA-Device = "tablet";
      set req.http.X-UA-Vendor = "android";
    # see http://my.opera.com/community/openweb/idopera/
    } elsif (req.http.User-Agent ~ "Opera Mobi") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "android";
    } elsif (req.http.User-Agent ~ "PlayBook; U; RIM Tablet") {
      set req.http.X-UA-Device = "tablet";
      set req.http.X-UA-Vendor = "blackberry";
    } elsif (req.http.User-Agent ~ "hp-tablet.*TouchPad") {
      set req.http.X-UA-Device = "tablet";
      set req.http.X-UA-Vendor = "hp";
    } elsif (req.http.User-Agent ~ "Kindle/3") {
      set req.http.X-UA-Device = "tablet";
      set req.http.X-UA-Vendor = "kindle";
    } elsif (req.http.User-Agent ~ "Mobile.+Firefox") {
      set req.http.X-UA-Device = "mobile";
      set req.http.X-UA-Vendor = "firefoxos";
    } elsif (req.http.User-Agent ~ "^HTC") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "htc";
    } elsif (req.http.User-Agent ~ "Fennec") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "fennec";
    } elsif (req.http.User-Agent ~ "IEMobile") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "microsoft";
    } elsif (req.http.User-Agent ~ "BlackBerry" || req.http.User-Agent ~ "BB10.*Mobile") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "blackberry";
    } elsif (req.http.User-Agent ~ "GT-.*Build/GINGERBREAD") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "android";
    } elsif (req.http.User-Agent ~ "SymbianOS.*AppleWebKit") {
      set req.http.X-UA-Device = "smartphone";
      set req.http.X-UA-Vendor = "symbian";
    } elsif (req.http.User-Agent ~ "(?i)symbian" ||
      req.http.User-Agent ~ "(?i)^sonyericsson" ||
      req.http.User-Agent ~ "(?i)^nokia" ||
      req.http.User-Agent ~ "(?i)^samsung" ||
      req.http.User-Agent ~ "(?i)^lg" ||
      req.http.User-Agent ~ "(?i)bada" ||
      req.http.User-Agent ~ "(?i)blazer" ||
      req.http.User-Agent ~ "(?i)cellphone" ||
      req.http.User-Agent ~ "(?i)iemobile" ||
```

```
 84         req.http.User-Agent ~ "(?i)midp-2.0" ||
 85         req.http.User-Agent ~ "(?i)u990" ||
 86         req.http.User-Agent ~ "(?i)netfront" ||
 87         req.http.User-Agent ~ "(?i)opera mini" ||
 88         req.http.User-Agent ~ "(?i)palm" ||
 89         req.http.User-Agent ~ "(?i)nintendo wii" ||
 90         req.http.User-Agent ~ "(?i)playstation portable" ||
 91         req.http.User-Agent ~ "(?i)portalmmm" ||
 92         req.http.User-Agent ~ "(?i)proxinet" ||
 93         req.http.User-Agent ~ "(?i)sonyericsson" ||
 94         req.http.User-Agent ~ "(?i)symbian" ||
 95         req.http.User-Agent ~ "(?i)windows\ ?ce" ||
 96         req.http.User-Agent ~ "(?i)winwap" ||
 97         req.http.User-Agent ~ "(?i)eudoraweb" ||
 98         req.http.User-Agent ~ "(?i)htc" ||
 99         req.http.User-Agent ~ "(?i)240x320" ||
100         req.http.User-Agent ~ "(?i)avantgo") {
101         set req.http.X-UA-Device = "mobile";
102       }
103     }
104   }
```

Enabling URL token validation

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.

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

  # extract the token
  set var.token = subfield(req.url.qs, "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;

  # 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"), req.url.path + var.token_expiration), "^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 (0x00) bytes. In Linux, use the command `while (b=$(openssl rand -base64 32) ; echo $b; echo $b | base64 -d | hd | grep " 00 " > /dev/null); do :; done | tail -1`. In macOS, use the command `while (b=$(openssl rand -base64 32) ; echo $b; echo $b | base64 -D | hexdump | grep " 00 " > /dev/null); do :; done | 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 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 (0x00), then that byte and any bytes following it will not be included in the response. See Base64 decoding for more information.

Excluding the token query string from the cache key

If all tokens are dynamic and different, you may want to exclude the token query string from the cache key to avoid potentially affecting the cache hit ratio. To do this, you'll need to add the following code in custom VCL in addition to the example code above:

```
1  /* strip out the token querystring so Fastly does not vary the cache. */
2  set req.url = querystring.filter(req.url, "token");
```

Guide to VCL

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

Fastly's Edge Cloud services use the Fastly Varnish Configuration Language (VCL). 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 user interface when you use custom VCL, but keep in mind that custom VCL always takes precedence over any VCL generated by the user 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.

Fastly VCL Extensions

In addition, Fastly has included a number of extensions to VCL that won't be covered by any other documentation. Specifically:

Extension	Description

Extension	Description
accept-language features	Provides functions to parse and normalize the `Accept-Language` header.
cryptographic and hashing functions	Supports Hash-based Message Authentication Code (HMAC), a message authentication code that uses a cryptographic key in conjunction with a hash function.
date- and time-related features	Supports the default VCL "now" variable that provides the current time as an RFC 850 formatted date (e.g., Tuesday, 29-Apr-14 08:41:55), as well as several new functions that allow you to have more flexibility when dealing with dates and times.
Geolocation features	Provides the ability to search a geolocation database for a given host or IP address, and return information about the country, city or Internet Service Provider (ISP) for that IP address.
local variables in VCL	Supports variables for storing temporary values during request processing
randomness features	Supports the insertion of random strings, content cookies, and decisions into requests.
size-related variables	Supports reporting variables that offer insight into what happened in a request.
TLS and HTTP/2 variables	Supports the use of variables and functions related to TLS and HTTP/2.
miscellaneous features and variables	Provides miscellaneous VCL extensions not easily grouped into other categories.

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 Varnish and VCL

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

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 `Set-Cookie` response from origin was `value1=123value123; testValue=asdf_true; staff_user=true;`, you could isolate the `staff_user` value using this logic:

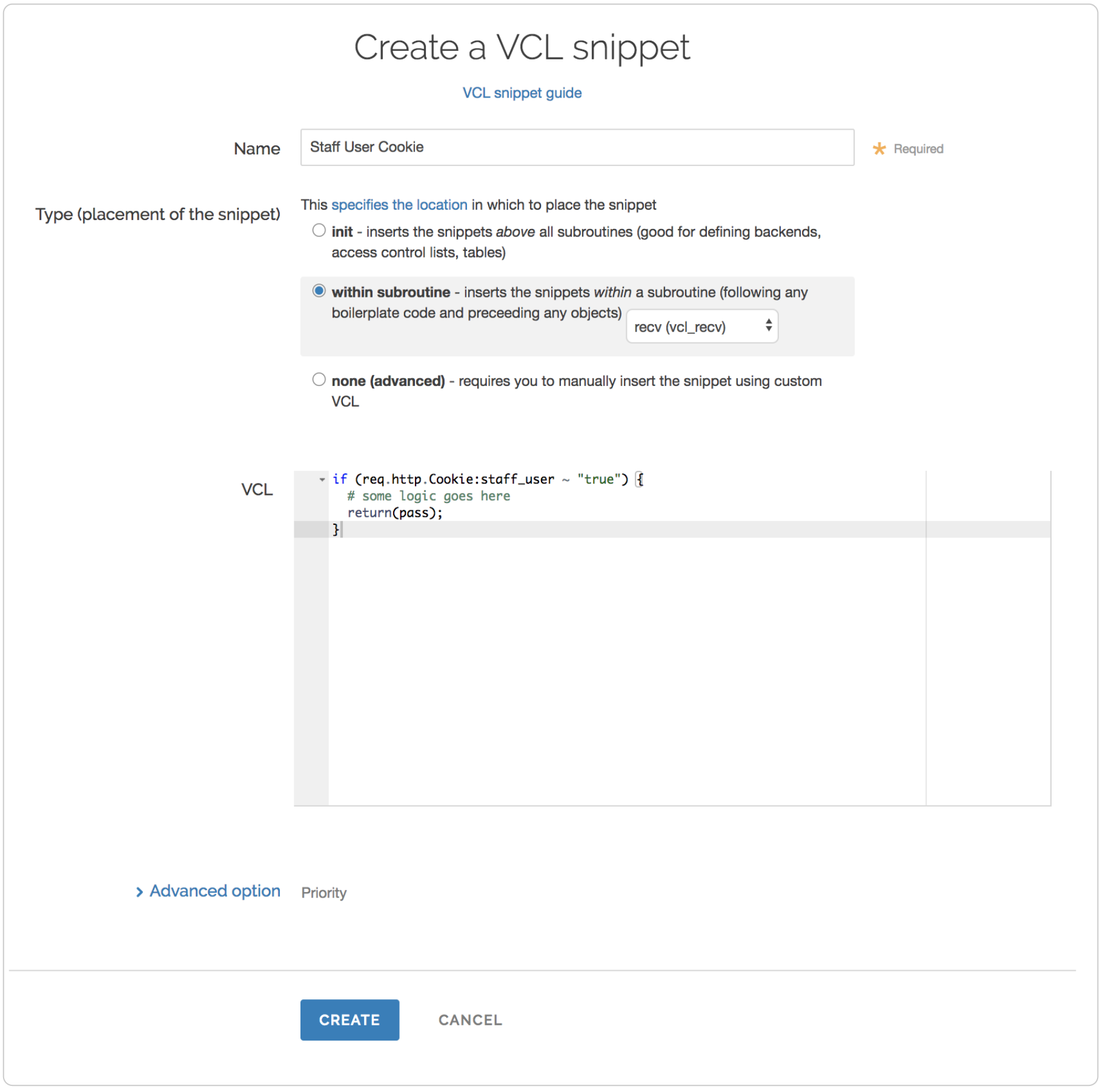
```
beresp.http.Set-Cookie:staff_user
```

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

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.



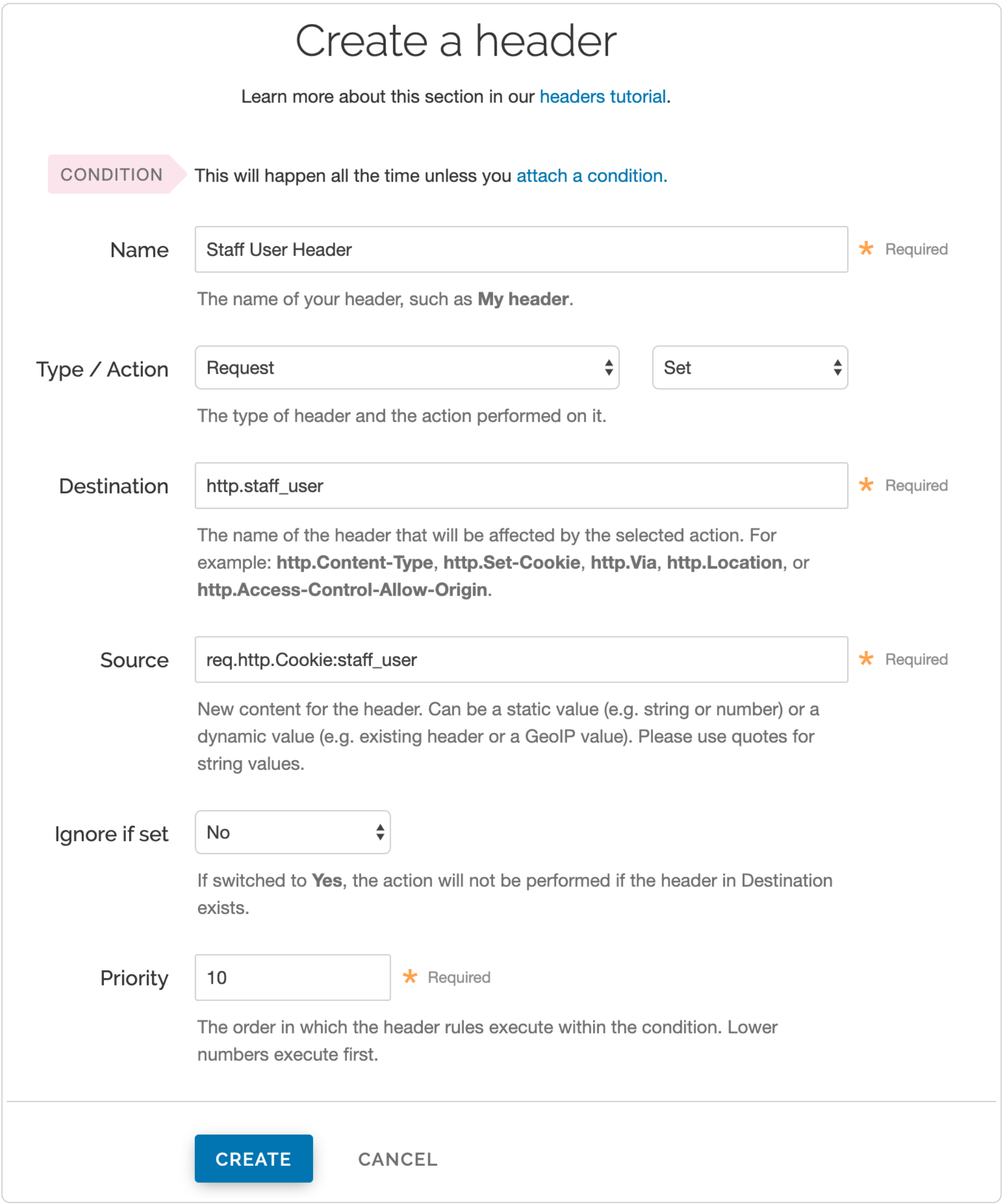
6. In the **Name** field, type an appropriate name (e.g., `Staff User Cookie`).
7. From the **Type** controls, select **within subroutine**.
8. From the **Select subroutine** menu, select **recv (vcl_recv)**.
9. In the **VCL** field, add the following condition:

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

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.

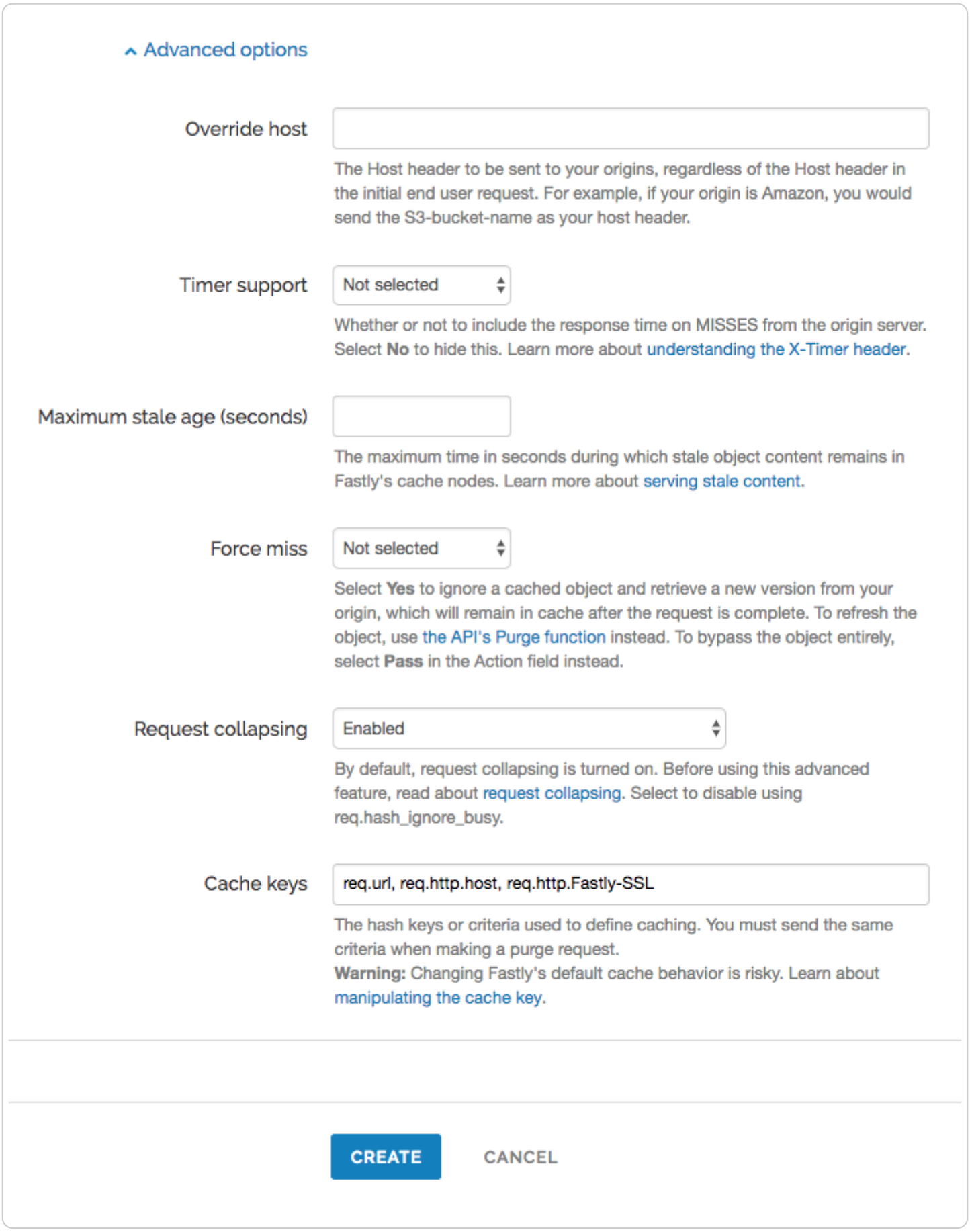
Explicitly setting the cache key

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 user 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

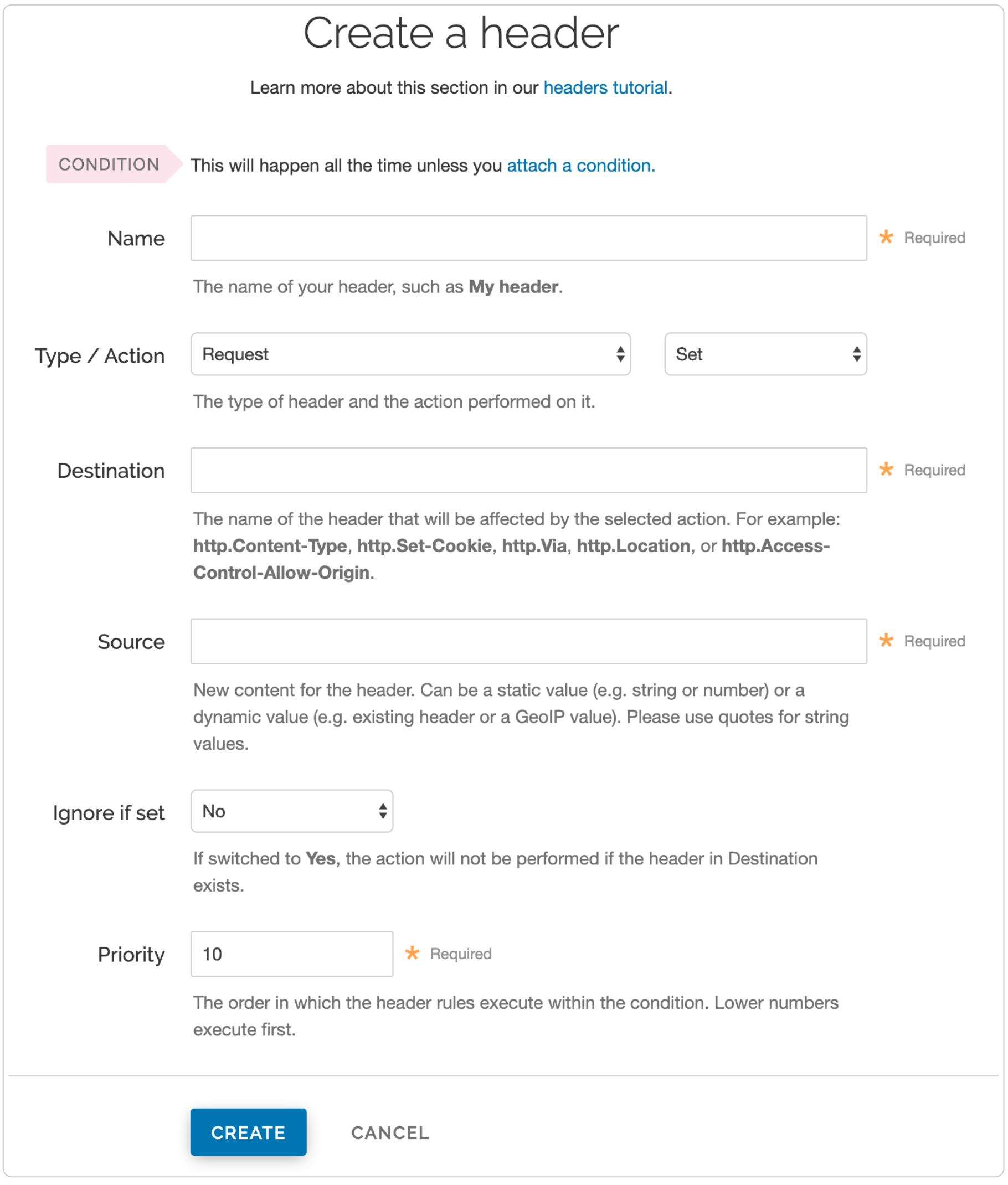
You can use a cookie as a cache key or just check for the presence of a cookie set to a specific value by controlling its request conditions. Both methods are simple and shown in the steps below.

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

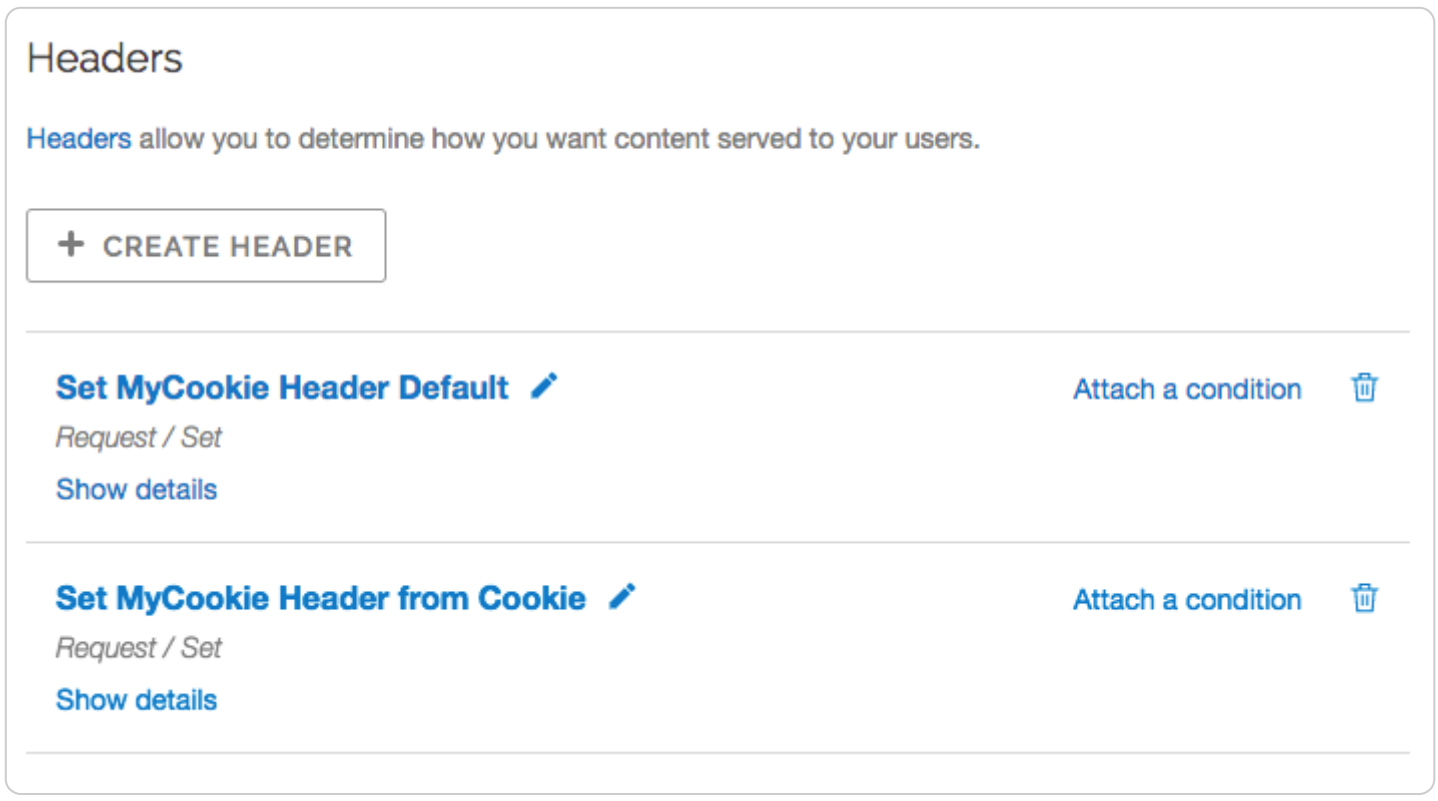
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a header page appears.



6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Set MyCookie Header Default`.
   - From the **Type** menu, select **Request**, and from the **Action** menu select **Set**.
   - In the **Destination** field, type `http.X-MyCookie`.
   - In the **Source** field, type `"0"` (with quotes).

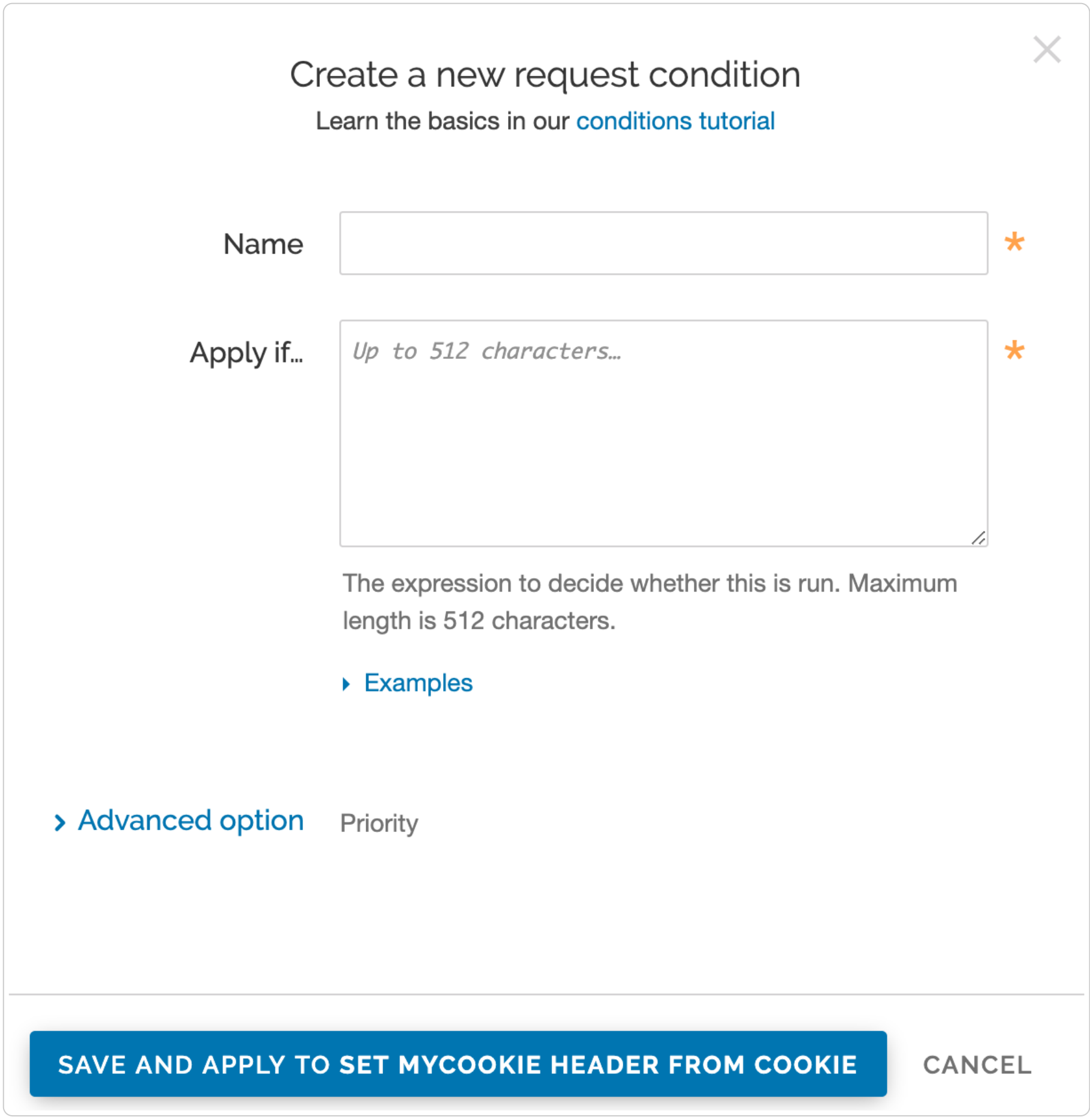
- Leave the **Ignore if set** menu set to the default, **No**.
- In the **Priority** field, type a number representing the order in which the header rule should execute. The default is set to `10` for new headers.

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, type `Set MyCookie Header from Cookie`.
   - From the **Type** menu, select **Request**, and from the **Action** menu select **Set**.
   - In the **Destination** field, type `http.X-MyCookie`.
   - In the **Source** field, type `req.http.cookie:mycookie`.
   - Leave the **Ignore if set** menu set to the default, **No**.
   - In the **Priority** field, type a larger number than the priority of previous header you just created. For example, if you left the default priority set to `10`, type `20`.
9. Click the **Create** button. The second header appears in the Headers area of the Content page.

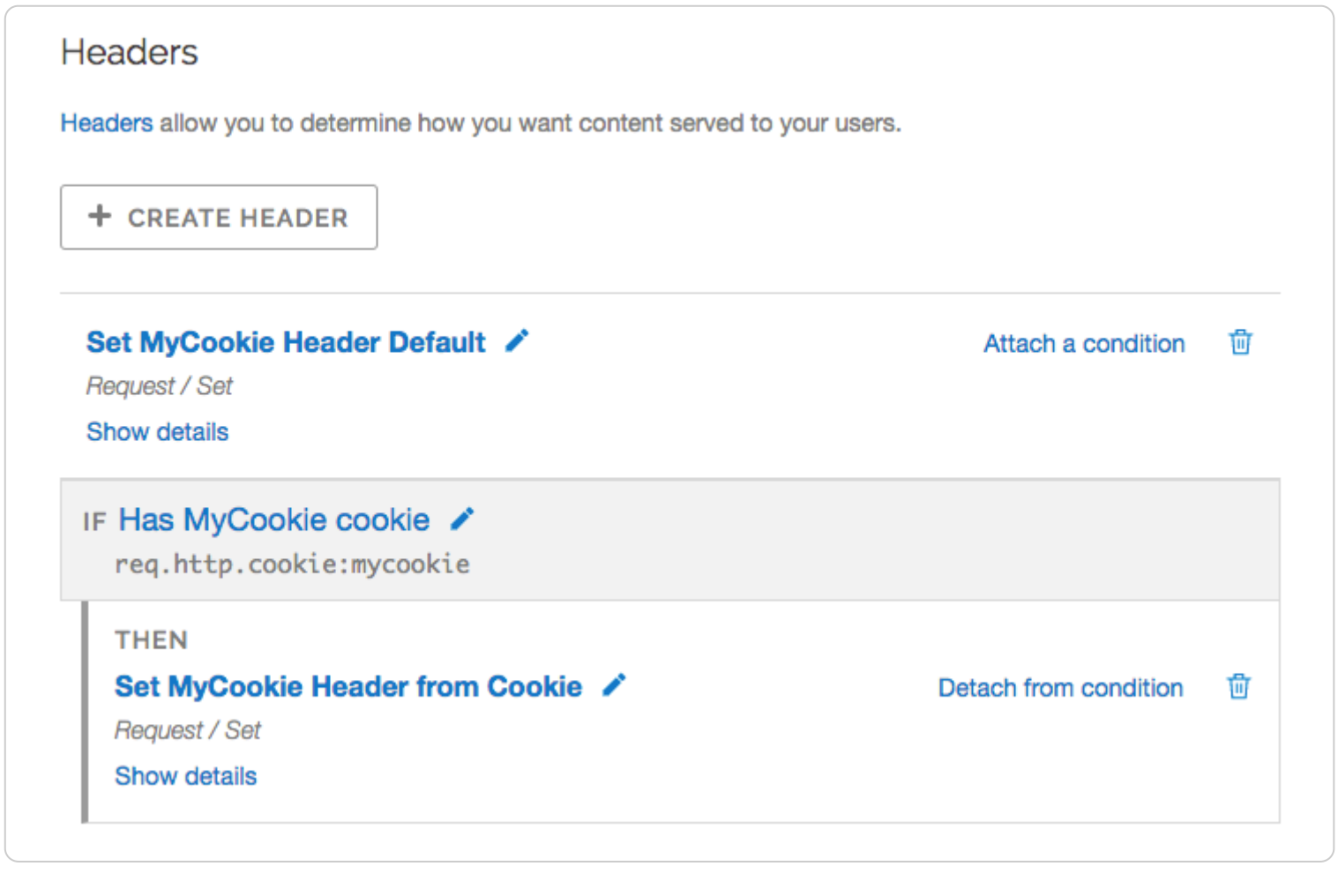


Attach conditions to the new headers

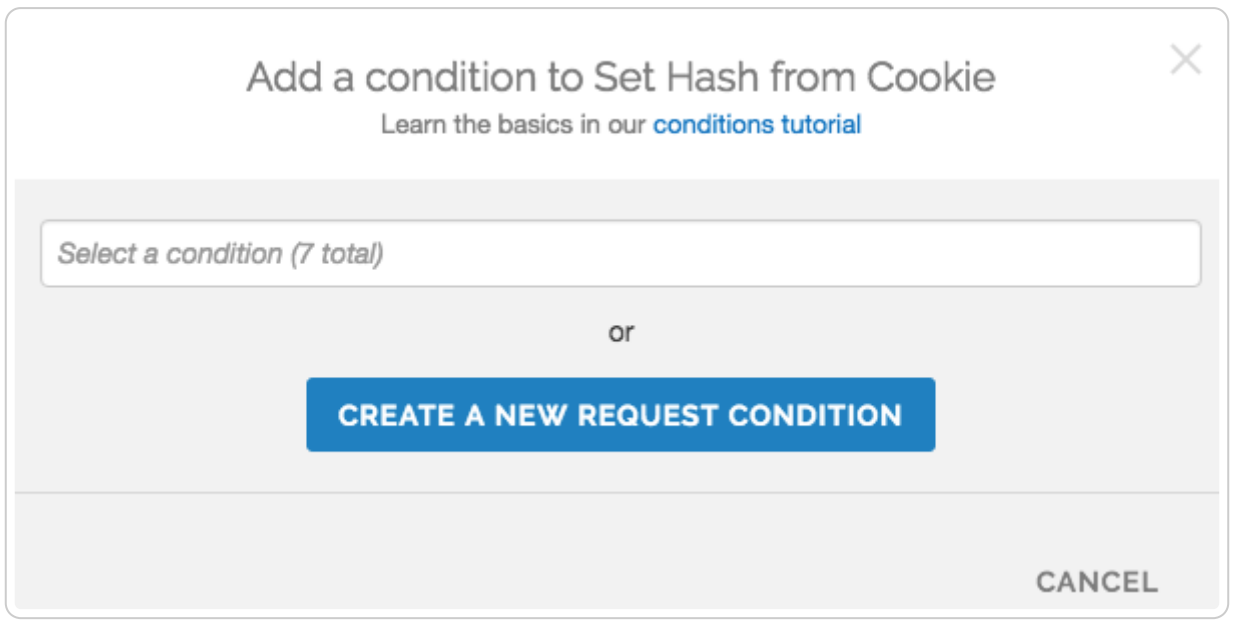
1. Click the **Attach a condition** link next to the `Set MyCookie Header from Cookie` header. The add a condition window appears.
2. Click the **Create a new request condition** button. The Create a new request condition window appears.



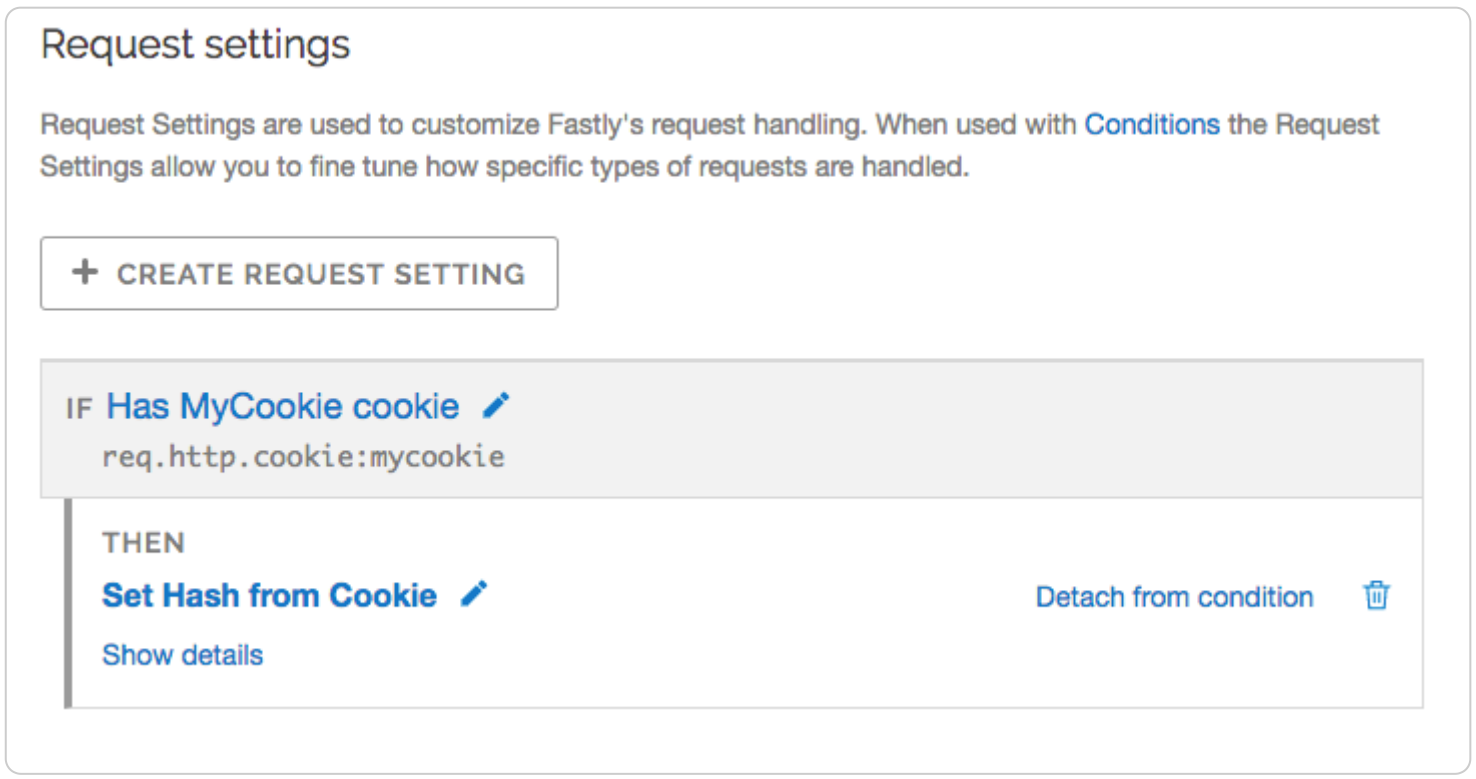
3. Fill out the fields of the **Create a new request condition** page as follows:
   - In the **Name** field, type `Has MyCookie cookie`.
   - In the **Apply if** field, type `req.http.cookie:mycookie`.
4. Click the **Save and apply to** button. The Headers area now displays the condition that must be 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, type `Set Hash from Cookie`.
8. Click the **Advanced options** link. The Advanced options appear.
9. In the **Cache keys** field, type `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`.

IP geolocation variables: Migrating to the new dataset

https://docs.fastly.com/en/guides/migrating-geolocation-variables-to-the-new-dataset

Fastly's IP geolocation variables are now based on a new IP geolocation dataset. Following Fastly's feature retirement policy, we'll continue to support variables that use the older version of the geolocation dataset until all of our customers have had time to migrate their service configurations to the newer version. As you migrate your configurations, keep the following important considerations in mind.

Namespaces differ between versions

The old version of the IP geolocation variables exist in the `geoip` namespace. The new version of these variables exist in the `client.geo` namespace and the Autonomous System (AS) variables exist in the `client.as` namespace.

Results for IPv6 addresses will only be returned for `client.geo` and `client.as` namespaces.

Geolocation data may be different

The data returned for a given IP address may be different between the dataset versions, especially at the city level. While it's possible to migrate configurations by replacing the older `geoip.*` namespace with `client.geo.*`, we recommend you carefully review any business logic that may rely on this data, especially if it's implemented in VCL or if the values are exposed via HTTP headers or real-time streaming logs.

In particular, understand that:

- The IP geolocation datasets are sourced from different vendors, each with different conventions for textual values. For example, `client.geo.city` and `client.geo.country_name` in the new dataset exist as lowercase ASCII values whereas the values returned for the same fields in older dataset are mixed case.
- The `client.geo.region` field contains ISO 3166-2 region codes but the `geoip.region` field contains FIPS 10-4 region codes. The FIPS 10-4 standard was withdrawn in 2014.

Overriding which IP address the geolocation features use

https://docs.fastly.com/en/guides/overriding-which-ip-address-the-geolocation-features-use

By default geolocation lookup is based on the IP address of the user. In some cases, such as with traffic through proxies, this type of lookup doesn't work properly. In particular, users of Opera Mini always browse through a proxy and the true IP address appears in the X-Forwarded-For header. Similarly, the Amazon Silk browser can optionally come through a proxy, indicated via the User

Agent string. In cases like these, the X-Forwarded-For header will contain a comma-separated list of IPs instead of just one IP. Attempting a geolocation lookup on anything other than a single IP will result in a lookup failure, so you need to ensure the lookup is done on the end-user's IP only.

To work around this and to account for both the Opera Mini and Amazon Silk browsers, you would use code like this in `vcl_recv`:

```
if ((req.http.X-OperaMini-Features || req.http.User-Agent ~ " Silk-Accelerated=true$") && req.http.X-Forwarded-For
){
  set client.geo.ip_override = regsub(req.http.X-Forwarded-For, ",.+$", "");
}
```

which tells Fastly to use only the first IP in the X-Forwarded-For header as the value for the IP address. If it is not available, then the code will fall back to using the IP address of the client.

Finally, just in case there's some scenario or browser we haven't anticipated, you can also override based on an arbitrary header:

```
set client.geo.ip_override = req.http.Custom-IP-Override;
```

Setting this variable will force the geolocation information to be reloaded.

Response Cookie handling

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

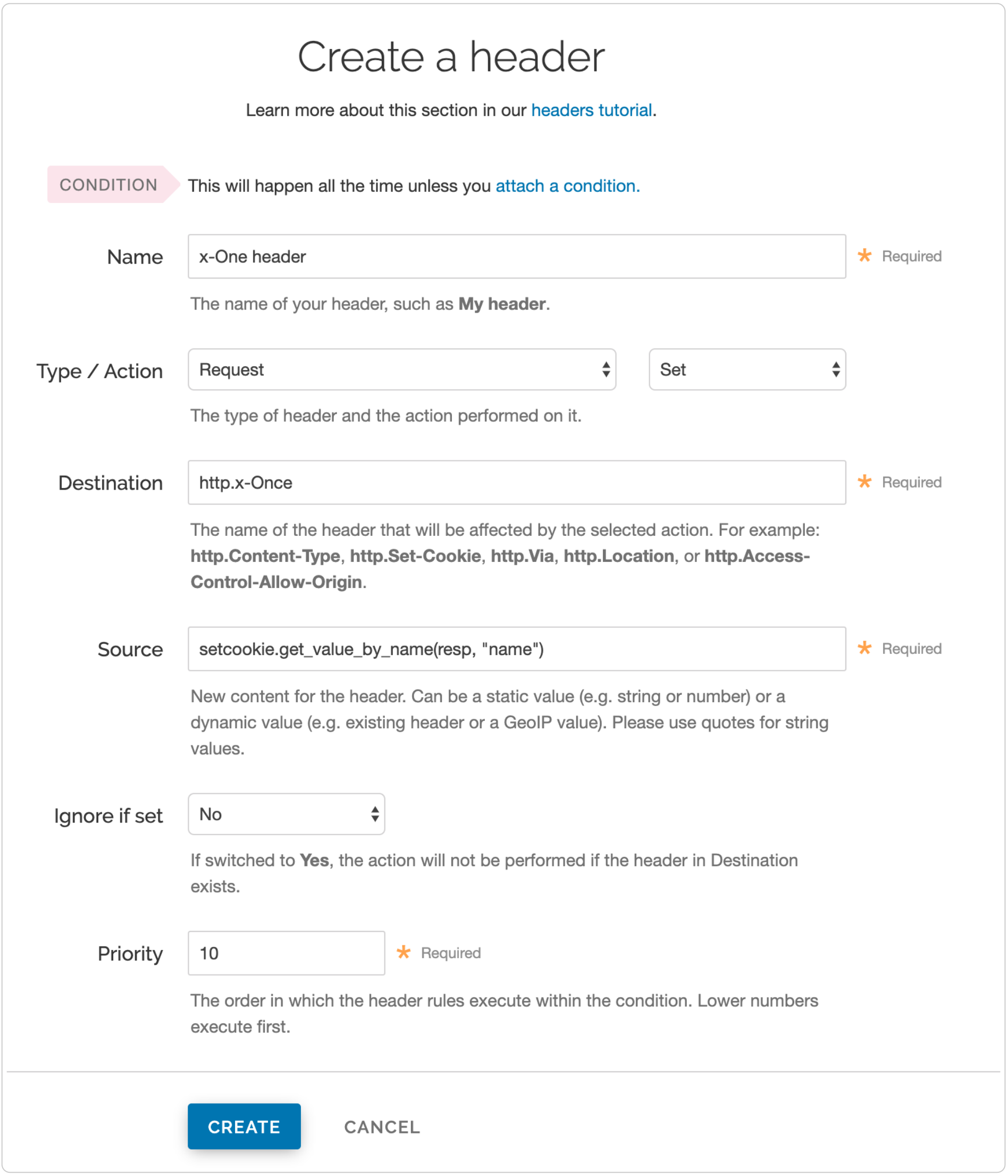
```
HTTP/1.1 200 OK
Cache-Control: max-age=60
Content-Type: text/html; charset=utf-8
Content-Length: 80806
Accept-Ranges: bytes
Date: Tue, 11 Aug 2015 19:00:04 GMT
Age: 123
Connection: keep-alive
Set-Cookie: one=a; httponly; secure
Set-Cookie: two=b or not to b; httponly
```

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:



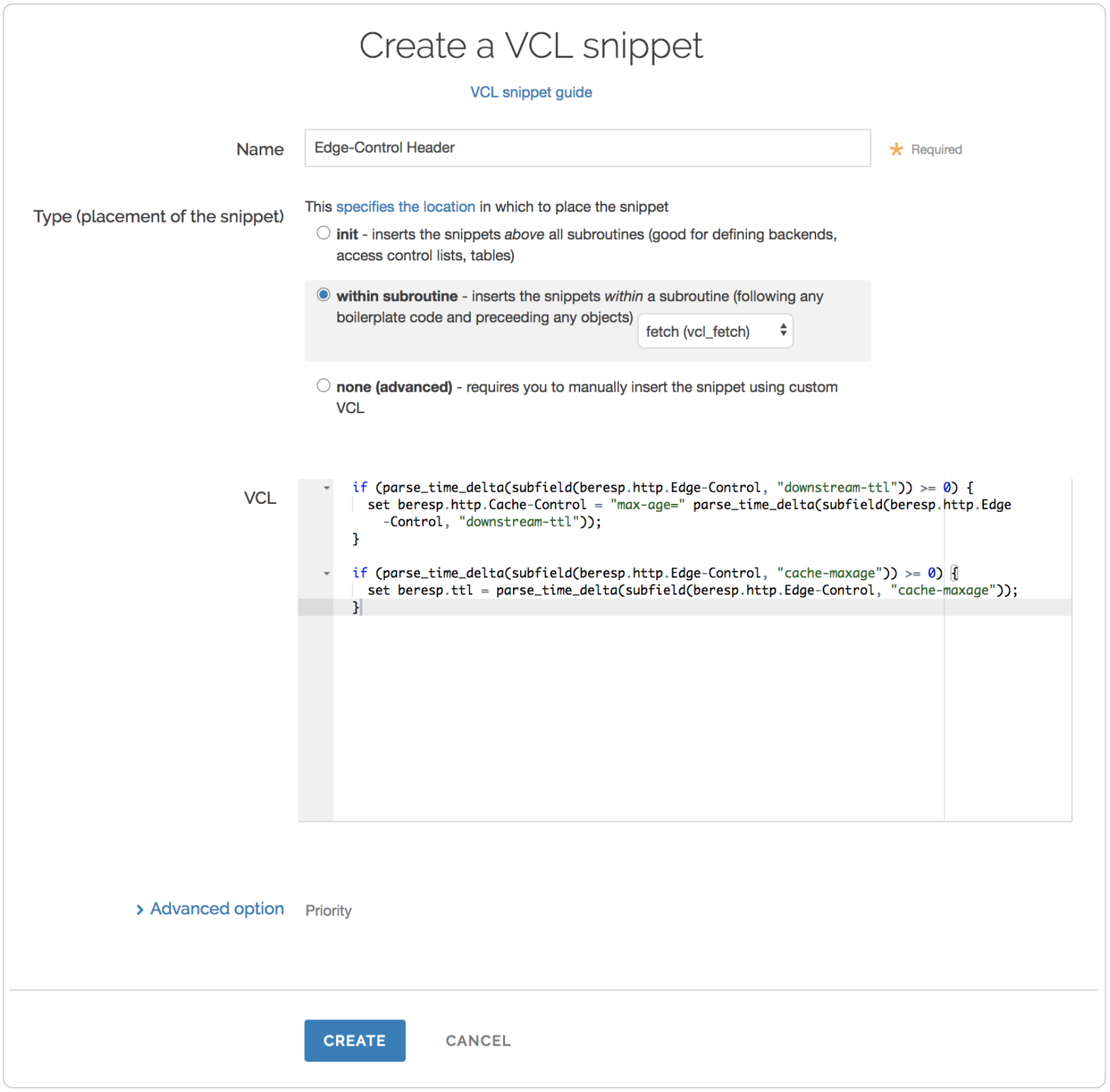
Support for the Edge-Control header

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type an appropriate name (e.g., `Edge-Control Header`).
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(subfield(beresp.http.Edge-Control, "downstream-ttl")) >= 0) {
  set beresp.http.Cache-Control = "max-age=" parse_time_delta(subfield(beresp.http.Edge-Control, "downstream-
ttl"));
}

if (parse_time_delta(subfield(beresp.http.Edge-Control, "cache-maxage")) >= 0) {
  set beresp.ttl = parse_time_delta(subfield(beresp.http.Edge-Control, "cache-maxage"));
}
```

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

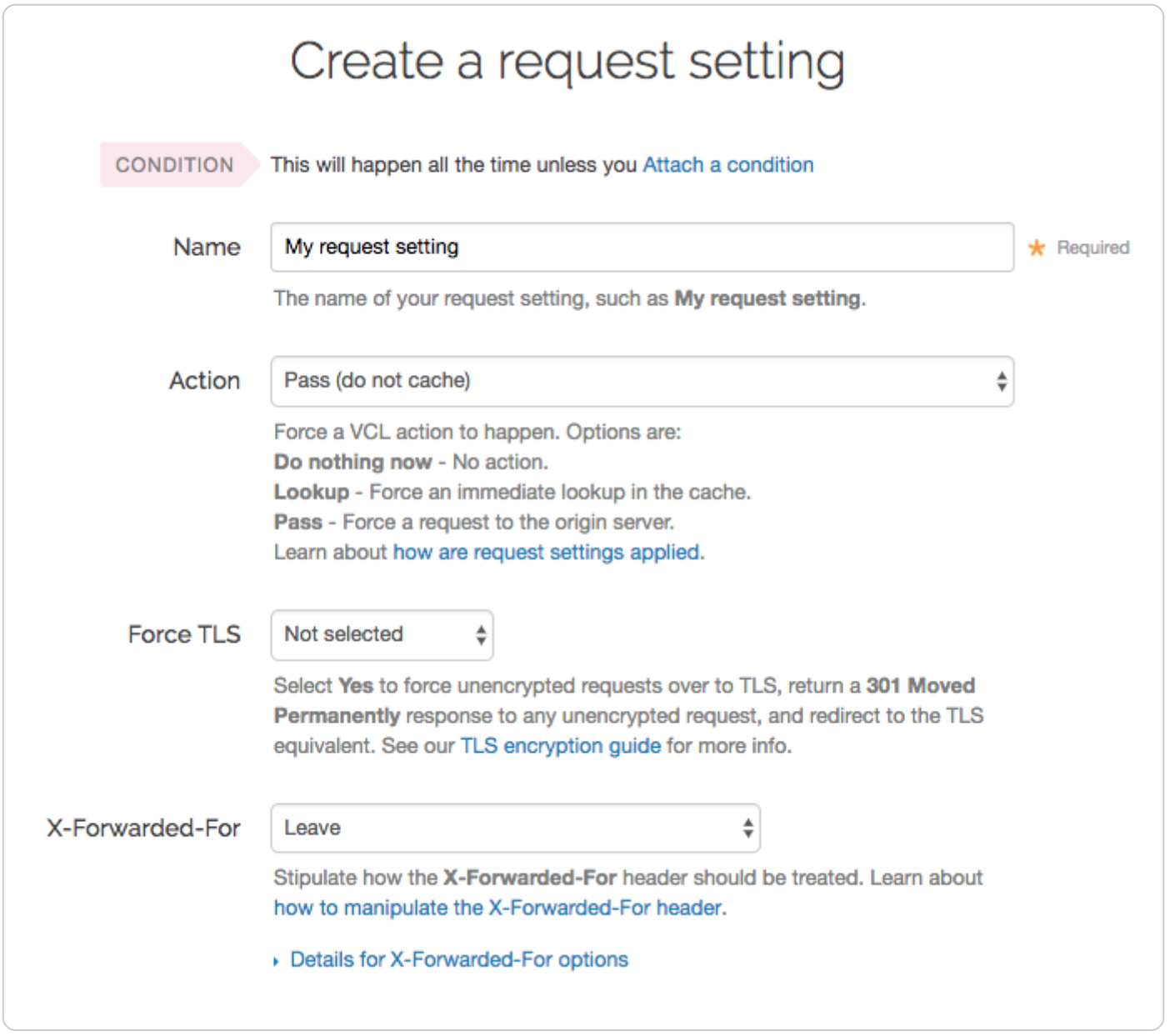
The subfield function parses the Edge-Control field for subfields, and the `parse_time_delta` function converts time values like "7m" into a number of seconds. You can then use that number of seconds to populate `beresp.ttl` (the TTL of the object on the Fastly edge) or you can use it to construct a 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

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)`.

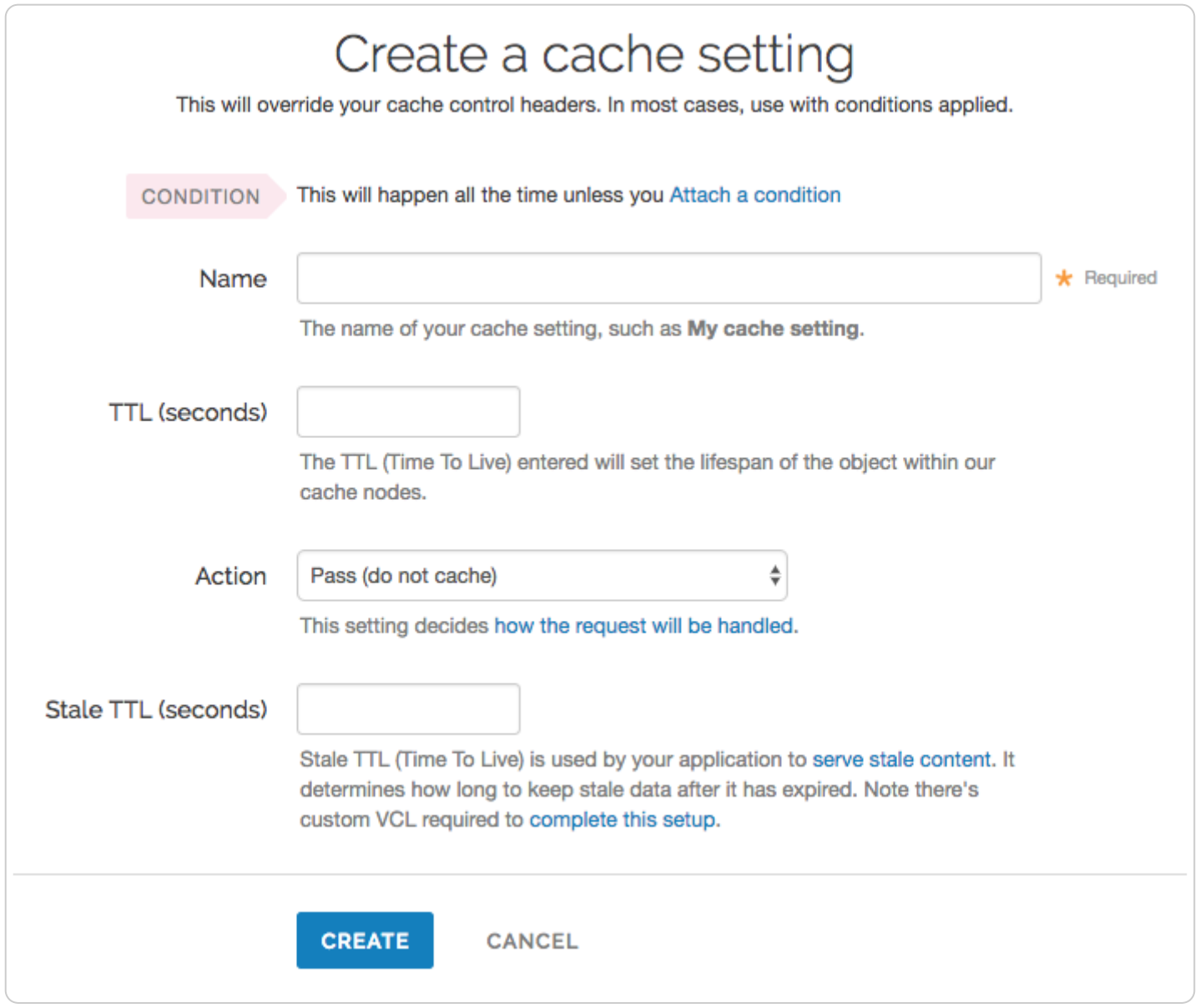
Using a request setting



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. It is different when 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.

Using edge side includes (ESI)

https://docs.fastly.com/en/guides/using-edge-side-includes

You can implement basic edge side includes (ESI) through Fastly using custom VCL. Fastly supports the following ESI elements:

- include
- comment
- remove

We don't support the following ESI language elements:

- inline
- choose | when | otherwise
- try | attempt | except
- vars
- ESI Variables

VCL regular expression cheat sheet

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`. See our Miscellaneous VCL features guide for more information.

```
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`.

Image optimization

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

https://docs.fastly.com/en/guides/about-fastly-image-optimizer

The Fastly Image Optimizer manipulates and transforms your images in real time and caches optimized versions of them. When an image is requested from your origin server, the Fastly Image Optimizer 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.

Getting started

To use the Fastly Image Optimizer, start by contacting sales to request access. Be sure to include the Service ID of the service for which image optimization should be enabled.

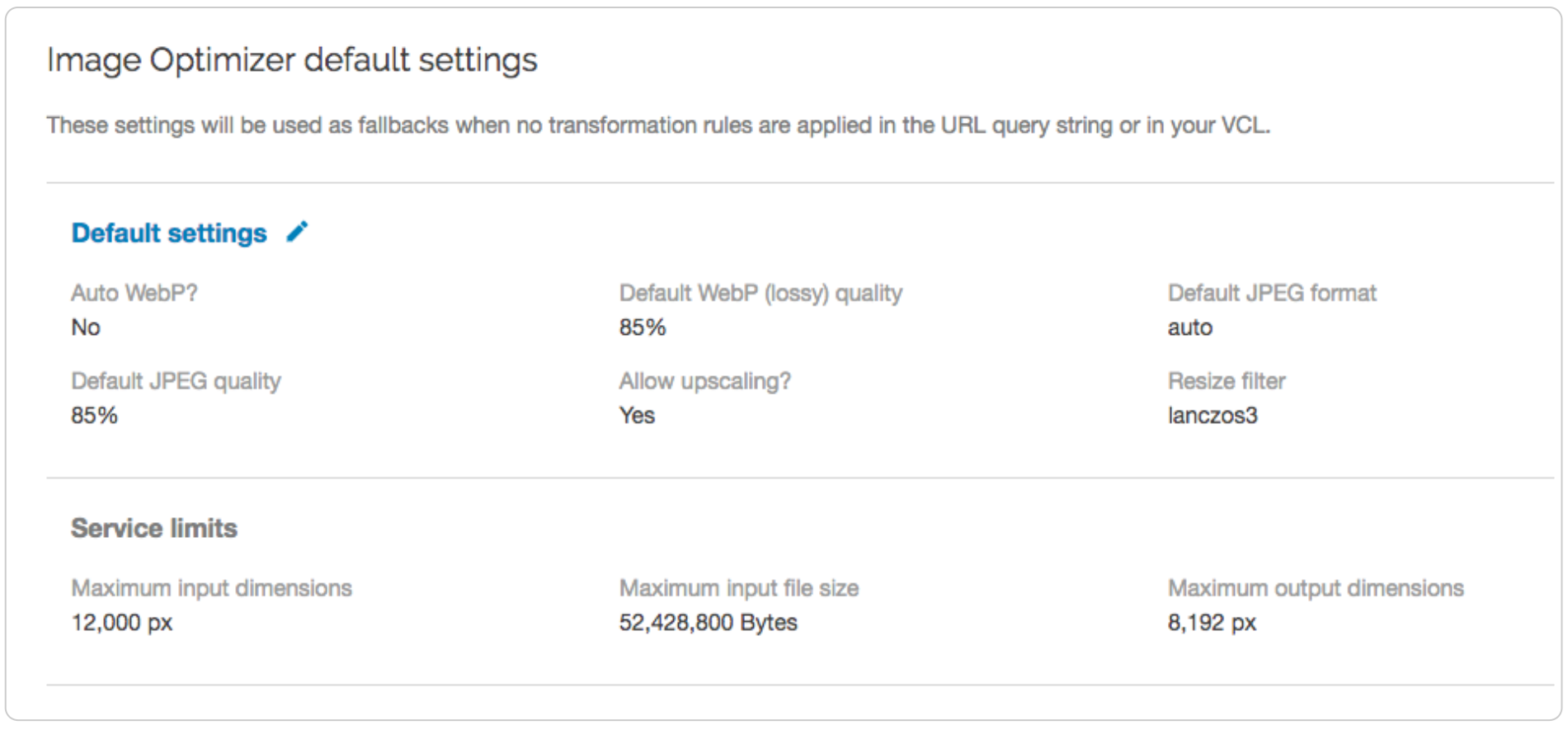
Setting up image optimization

Once we've enabled image optimization for your service, you can set up image optimization by following the steps in our guide to setting up image optimization. This is a four-step process:

1. Add a header.
2. Create a request condition.
3. Enable shielding.
4. Test an image.

Configuring basic 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. See our instructions on reviewing and editing the default image settings for more information.



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.

You can change existing URLs in your source by adding one or more Fastly URL API query string parameters directly to your site's HTML. You can also change them programmatically. For more information, see our section on changing image settings other than the defaults.

Using custom VCL with image optimization

If you're using custom VCL with the Fastly Image Optimizer, we recommend using the image optimization VCL boilerplate. This boilerplate is specially designed to work with image optimization. It also fixes several potential issues that can arise when using image optimization with our default VCL boilerplate.

Debugging

Running into problems? See our section 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.

Automating optimization

https://docs.fastly.com/en/guides/automating-optimization

Encode to a JPEG and enable WebP automatic format selection to supported browsers

`https://www.fastly.io/image.jpg?format=jpeg&auto=webp`

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.

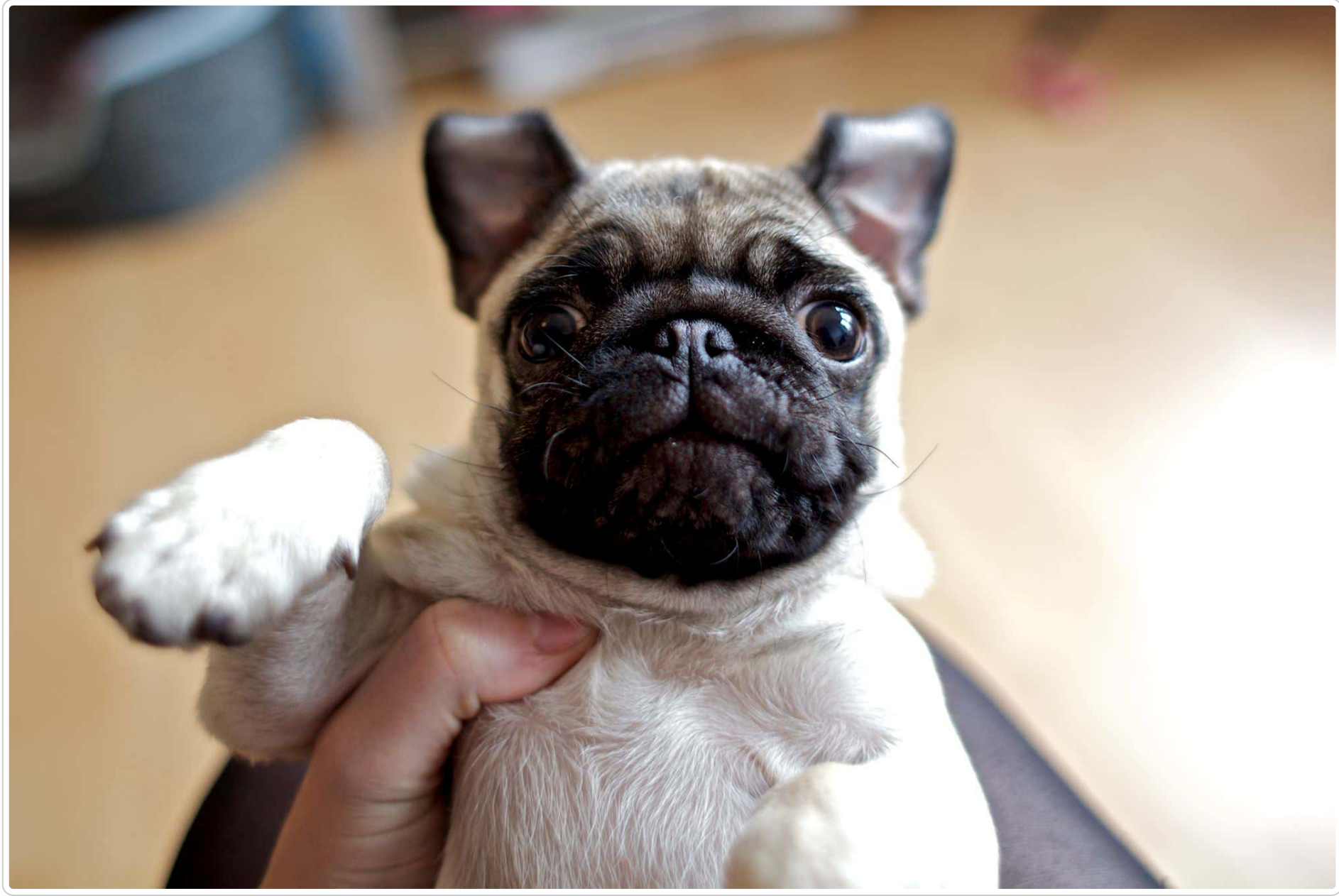
Controlling image quality

https://docs.fastly.com/en/guides/controlling-image-quality

Deliver an image at a specific level of quality

Output an image with a specific compression level of 60 (from a total of 0 to 100, where 85 is the default for lossy images when IO is enabled).

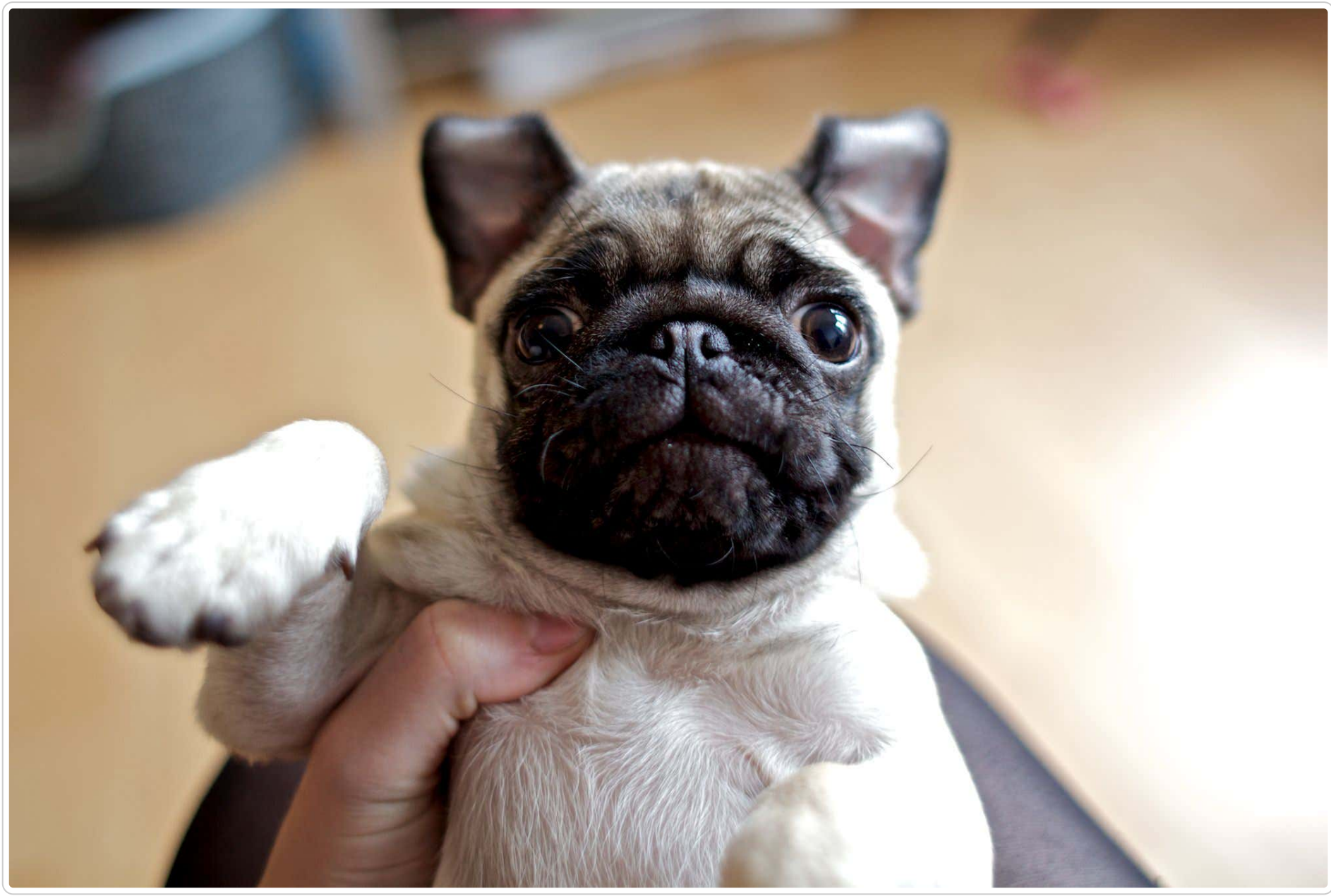
`https://www.fastly.io/image.jpg?quality=60`



Transcode and deliver an image at a specific level of quality

Convert the image format to jpg and output an image with a specific compression level of 60.

```
https://www.fastly.io/pug.png?format=jpg&quality=60
```



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.

Cropping images

https://docs.fastly.com/en/guides/cropping-images

Region crop

Crop the image to 150px by 100px.

`https://www.fastly.io/image.jpg?crop=150,100`

Aspect ratio crop

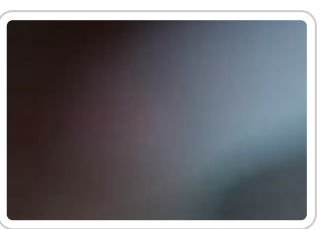
Crop the image to an aspect ratio of 16:9.

`https://www.fastly.io/image.jpg?crop=16:9`

Region crop and sub region

Crop the image to 150px by 100px and also select 50px as the starting sub region x coordinate and 50px as the sub region y coordinate.

`https://www.fastly.io/image.jpg?crop=150,100,x50,y50`



Aspect ratio crop and offset

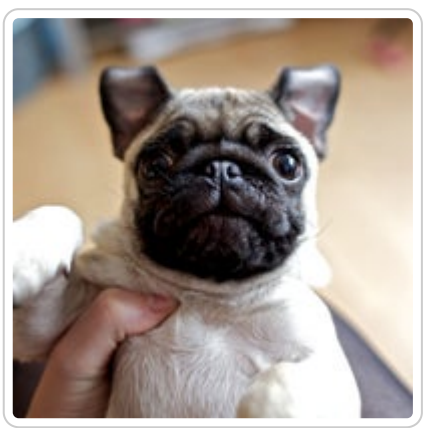
Crop the image square and offset the x-axis 25% and the y-axis 50%.

`https://www.fastly.io/image.jpg?crop=1:1,offset-x0.25,offset-y0.50`

Aspect ratio crop (with width)

Crop the image square and resize the width to 200px.

```
https://www.fastly.io/image.jpg?crop=1:1&width=200
```



Smart cropping (with trim and width)

Smart crop the image square, trim all edges by 30% and resize the width to 200px.

```
https://www.fastly.io/mountaineer.jpg?trim=0.30&width=200&crop=1:1,smart
```

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.

Image optimization VCL boilerplate

https://docs.fastly.com/en/guides/image-optimization-vcl-boilerplate

If you use the Fastly Image Optimizer (IO) with custom VCL, you should consider using the IO VCL boilerplate. This boilerplate is specially designed to work with IO. It also fixes several potential issues that can arise when using IO with our default VCL boilerplate.

IO VCL boilerplate

Before using the IO VCL boilerplate, review the Image Optimizer documentation and the instructions in our custom VCL guide.

```
sub vcl_recv {
#FASTLY recv
  if (req.method != "HEAD" && req.method != "GET" && req.method != "FASTLYPURGE") {
    return(pass);
  }

  # Enable IO for image file-types
  if (req.url.ext ~ "(?i)^(?:gif|png|jpe?g|webp)$")  {
    set req.http.X-Fastly-Imageopto-Api = "fastly";
  }

  return(lookup);
}

sub vcl_fetch {
#FASTLY fetch

  # Unset headers which reduce cacheability for images
  if (req.http.X-Fastly-Imageopto-Api) {
    unset beresp.http.Set-Cookie;
    unset beresp.http.Vary;
  }

  # Check origin caching headers and override / apply defaults
  if (beresp.http.Expires || beresp.http.Surrogate-Control ~ "max-age" || beresp.http.Cache-Control ~ "(s-maxage|max-age)") {
    # Keep origin TTL
  } else {
    # Apply a default where origin does not provide TTL
    if (beresp.status == 200) {
      set beresp.ttl = 604800s; # 7 days
      set beresp.http.Cache-Control = "max-age=604800, public";

      # Apply a longer default TTL for images
      if (req.http.X-Fastly-Imageopto-Api) {
        set beresp.ttl = 2592000s; # 30 days
        set beresp.http.Cache-Control = "max-age=2592000, public";
      }

    } else {
      # Apply short TTL for non-200 responses
      set beresp.ttl = 60s;
    }
  }

  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
}

sub vcl_pass {
#FASTLY pass

}

sub vcl_log {
#FASTLY log
}
```

```
4
7
4
8
4
9
5
0
5
1
5
2
5
3
5
4
5
5
5
6
5
7
5
8
5
9
6
0
6
1
6
2
6
3
6
4
6
5
6
6
6
7
6
8
6
9
7
0
7
1
7
2
7
3
7
4
7
5
7
6
7
7
7
8
7
9
8
0
```

Customizing the IO VCL boilerplate

Read the information in this section before modifying the IO VCL boilerplate.

Shielding

You must use shielding with IO. When a request is received at the edge for a particular variation of an image that isn't cached, the shield passes the request along to the image processors, which pass the request to your origin for the original image. Shielding is important because original images and image variations are cached at the shield. Without shielding enabled, more requests are passed directly to your origin.

Limiting IO passthrough to images

The simplest way to prevent non-image files passing through to IO is to limit application of the header by image file extension.

```
sub vcl_recv {
  if (req.url.ext ~ "(?i)^(?:gif|png|jpe?g|webp)$") {
    set req.http.X-Fastly-Imageopto-Api = "fastly";
  }
  return(lookup);
}
```

If the origin doesn't have a file extension or doesn't have valid file extensions, you'll need to determine validity using another method. One common approach is identifying images by path:

```
sub vcl_recv {
  if (req.url.path ~ "/images/") {
    set req.http.X-Fastly-Imageopto-Api = "fastly";
  }
  return(lookup);
}
```

Another approach is dedicating the entire service to image assets.

X-Fastly-Imageopto-Api header

The `X-Fastly-Imageopto-Api` header must be applied unconditionally for IO requests at both edge and shield. We unset this by default to prevent the header from being spoofed. Applying this header at the edge only (wrapping with `if (!req.http.Fastly-FF)`) can result in unexpected behavior. The cache key is constructed differently based on whether IO is enabled or not, so only applying the header at the edge will create a scenario where the same assets reside under different cache keys at the shield and the edge.

Query string passthrough

By default, any query string parameters which don't exist as part of our IO API are stripped in `master vcl_recv` to protect your origin. Because additional query string parameters form part of the cache key, for each query string variation, there is an additional branch of image variations.

With query string passthrough disabled (default), the following will occur:

```
Fastly:    ?width=100&something=else
Fastly IO: ?width=100
Origin:    [none]
```

With query string passthrough enabled, the following will occur:

```
Fastly:    ?width=100&something=else
Fastly IO: ?width=100&something=else
Origin:    ?something=else
```

Enabling query string passthrough presents an attack vector for your origin. For this reason, query string passthrough is only allowed via explicit opt-in, using the following header:

```
set req.http.X-Fastly-Imageopto-Api = "fastly; qp=*";
```

Default TTL

The standard Fastly VCL boilerplate applies a default TTL of 3600s. Ideally, image content should have a greater longevity. When using IO, there's a more severe consequence for a low TTL. It doesn't just mean a cache invalidation and pull from origin. It also invalidates all image variations which results in reprocessing and therefore increased miss latency.

Purging optimized images

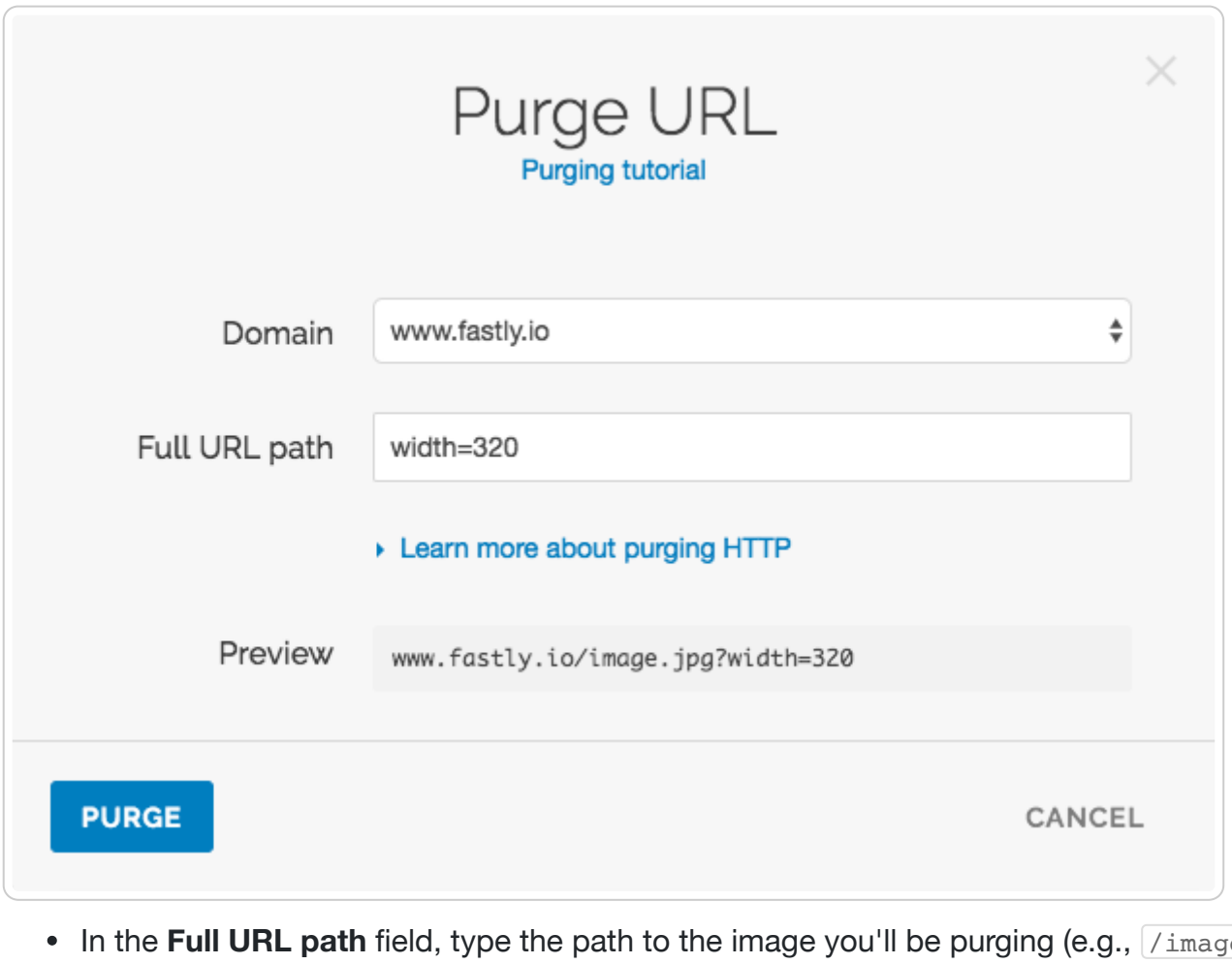
https://docs.fastly.com/en/guides/purging-optimized-images

Instant Purging removes an image from Fastly caches immediately so it can be refreshed from your origin servers.

Purging images via the user interface

- Log in to the Fastly web interface and click the **Configure** link.
- From the service menu, select the appropriate service.
- From the **Purge** menu, select **Purge URL**. The Purge URL window appears.

Purge an individual image



- In the **Full URL path** field, type the path to the image you'll be purging (e.g., `/image.jpg?width=320`). The Preview field displays the URL that will be purged.
- Click the **Purge** button.

Purge all transformed image variations

- In the **Full URL path** field, type the path to the image removing all Fastly Image Optimizer API query string parameters. (e.g., `/image.jpg`). The Preview field displays the URL that will be purged.
- Click the **Purge** button.

Purging images via API

The syntax for purging a service through the API can be found in the Purging section of the API documentation.

Purge an individual image via API

To purge an individual image URL, type the path to the image you want to purge.

For example: `curl -X PURGE https://www.fastly.io/image.jpg?width=320`

Purge all transformed image variations via API

To purge all transformed image variations belonging to a specific image, remove all the Fastly Image Optimizer API query string parameters.

For example: `curl -X PURGE https://www.fastly.io/image.jpg`

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.

Reorienting images

https://docs.fastly.com/en/guides/reorienting-images

Reorient the image right

`https://www.fastly.io/image.jpg?orient=r`

Flip the image horizontally

`https://www.fastly.io/image.jpg?orient=2`

Flip the image horizontally and vertically

`https://www.fastly.io/image.jpg?orient=3`

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.

Resizing images

https://docs.fastly.com/en/guides/resizing-images

Pixel width resize

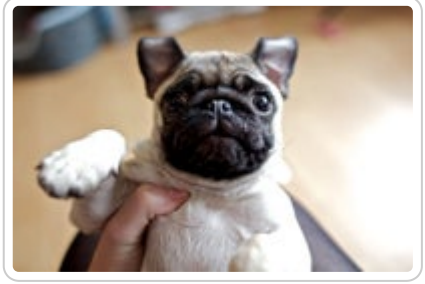
Resize the width to 200px.

```
https://www.fastly.io/image.jpg?width=200
```

Percentage height resize

Resize the height to 10% of the input image.

```
https://www.fastly.io/image.jpg?height=0.10
```



Percentage width resize over 100%

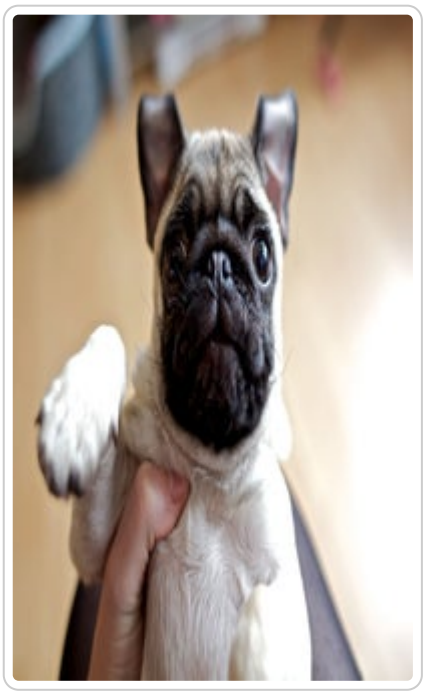
Resize the width to 150% of the input image.

```
https://www.fastly.io/image.jpg?width=150p
```

Disproportionate resize

Disproportionally resize to a width of 200px and a height that is 25% of the original.

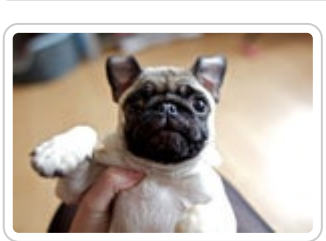
```
https://www.fastly.io/image.jpg?width=200&height=0.25
```



Resize to bounds

Resize the image to fit within the bounds of 150px in width by 150px in height.

```
https://www.fastly.io/image.jpg?width=150&height=150&fit=bounds
```



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.

Serving images

https://docs.fastly.com/en/guides/serving-images

By adding the transformation URL API query string parameters to your existing image URLs, images can be resized, cropped, rotated, compressed, and transcoded into different image formats for increased performance. Image transformations can be applied programmatically and on-demand, eliminating the need to batch process or maintain multiple copies of an image to support different sizes and characteristics of device viewing your image content.

Example transformation

Resize an image to 200px wide.

```
<img src="https://www.fastly.io/image.jpg?width=200"/>
```

Transformation order

Although the URL API parameters can be specified in any order, we normalize the transformation sequence within our system to the following order:

1. `trim`
2. `crop`
3. `orientation`
4. `width` `height` `dpr` `fit` `resize-filter` `disable`
5. `pad` `canvas` `bg-color`
6. `overlay`
7. `brightness` `contrast` `saturation`
8. `sharpen`
9. `blur`
10. `format` `quality`

Supported input and output image formats

The source image can be any of the following image formats:

`GIF`, `PNG`, `JPEG`, `WEBP`

The optimized output image can be any of the following image formats:

`GIF`, `PNG`, `JPEG`, `WEBP`

Input and output limits

- The maximum input image file size is 50 Megabytes.
- The maximum input image dimensions are 12,000x12,000 pixels.
- The maximum output image dimensions are 8,192x8,192 pixels (8K Ultra HD).
- The maximum number of frames an animated GIF can contain is 1,000.

Default quality level

If no `quality` parameter is present for `jpg`, `pjpg`, or `webp` the output image will be returned at the default value set in the Image Optimizer User Interface.

Meta data removal

To optimize your images for delivery, all metadata (for example, EXIF, XMP, or ICC) is removed to reduce file size. If an image contains an ICC profile, the data is applied directly to the image to ensure color output is correct. If the image doesn't contain an ICC profile, a default profile is added.

WebP image support

WebP images can be delivered to supported browsers by adding the `auto=webp` parameter or by applying the **Auto WebP** control in the Image Optimizer user interface.

Image upscaling

Image upscaling is disabled by default and discouraged from use because it increases the file size and delivery time of the image to the user with no improvement of visual quality. We recommend handling upscaling on the client-side (e.g., web browser, css, native app) by setting the width and height attributes of the image.

Debugging

To debug images served from the Image Optimizer, the following HTTP headers will be present in the response, depending on the response's result.

fastly-io-info

The `fastly-io-info` header is added to every successfully optimized image. The header values can be used to compare the image file's size, dimensions, and format of the origin image against the optimized edge image. The header contains a string made up of 6 key and value pairs (i.e., `ifsz=3076875 idim=4000x3000 ifmt=png ofsz=83179 odim=893x670 ofmt=jpeg`). The following table defines each key:

Key	Description
`ifsz`	Input image file size
`idim`	Input image dimensions
`ifmt`	Input image format
`ofsz`	Output image file size
`odim`	Output image dimensions
`ofmt`	Output image format

fastly-io-warning

The `fastly-io-warning` header is added when issues with the source image are encountered that are not fatal enough to cause the image to error.

fastly-io-error

The `fastly-io-error` header is added to the following error scenarios:

- Image exceeds maximum dimensions
- Image could not be parsed
- Not a supported image format
- Unsupported Content-Encoding
- Gzipped body exceeds maximum length
- Gzipped body could not be decoded
- Invalid status
- Response is pass
- Response is not cacheable

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.

Serving responsive images

https://docs.fastly.com/en/guides/serving-responsive-images

The Fastly Image Optimizer allows you to generate optimized images for use in responsive websites. The examples below describe several common use cases to get you started implementing responsive images.

Adaptive device pixel ratios

Deliver a fixed-width image that can adapt to varying `device-pixel-ratios`.

```
<img srcset="https://www.fastly.io/image.jpg?width=320&dpr=1.5 1.5x,
             https://www.fastly.io/image.jpg?width=320&dpr=2 2x"
     src="https://www.fastly.io/image.jpg?width=320"/>
```

Learn about `srcset` browser support and specification.

Art direction

Use the HTML5 `<picture>` tag to deliver different image crops at different browser viewport sizes.

```
<picture>
  <source srcset="https://www.fastly.io/image.jpg?width=600&crop=16:9" media="(min-width: 600px)"/>
  <img src="https://www.fastly.io/image.jpg?width=320&crop=1:1"/>
</picture>
```

Learn about `<picture>` browser support and specification.

Type-switching

Use the best file format for the browser and allow graceful fallback for non-supporting formats.

```
<picture>
  <source type="image/webp" srcset="https://www.fastly.io/image.webp"/>
  <source type="image/png" srcset="https://www.fastly.io/image.png"/>
  <img src="https://www.fastly.io/image.jpg"/>
</picture>
```

Learn about `<picture>` browser support and specification.

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.

Setting up image optimization

https://docs.fastly.com/en/guides/setting-up-image-optimization

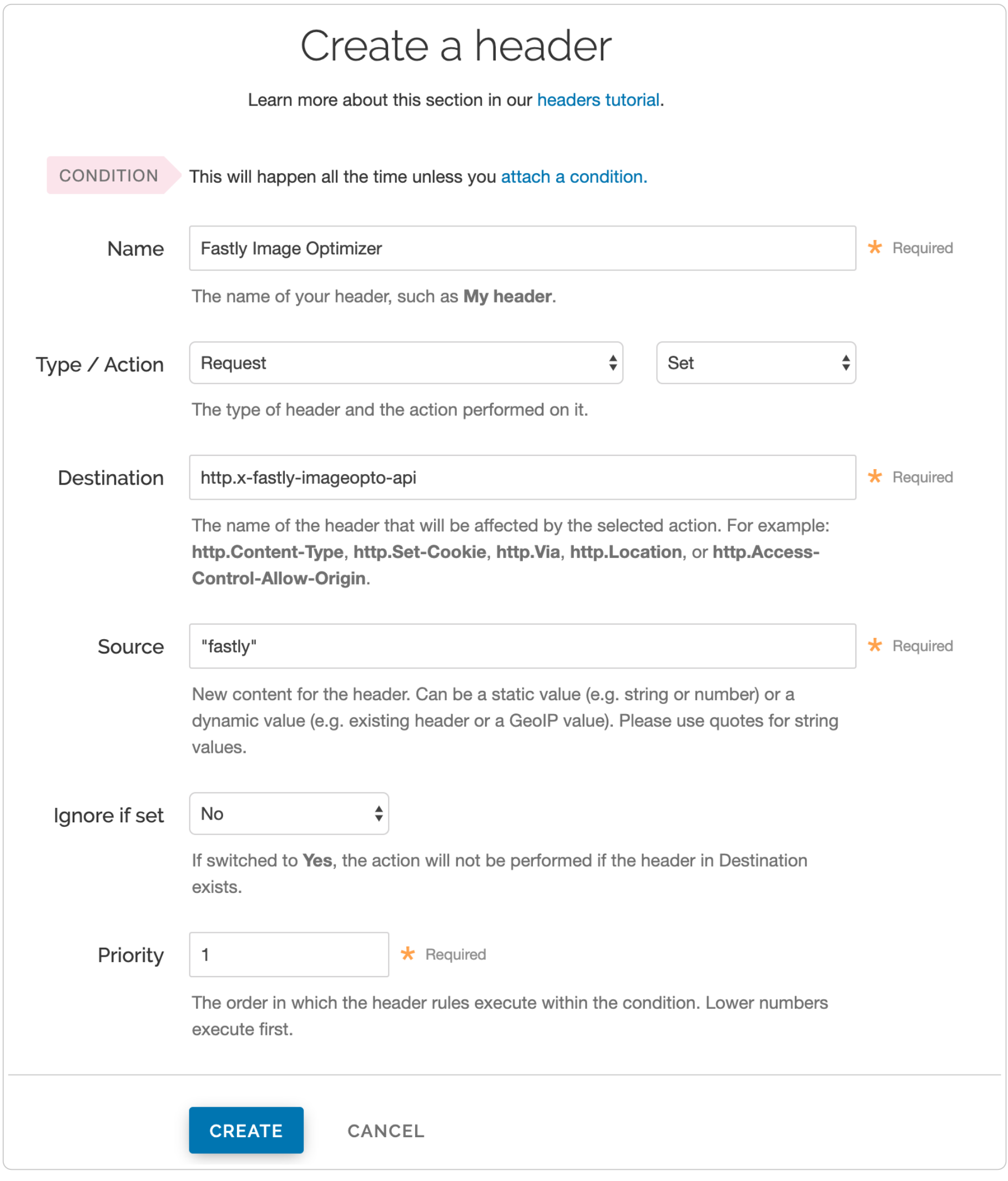
> ⚠ **WARNING:** Only send image content through the Fastly Image Optimizer. Non-image content can't be optimized by the Image Optimizer but will still be counted and charged as an Image Optimizer request, which may cost you more.

To use the Fastly Image Optimizer, start by contacting sales to request access. Be sure to include the Service ID of the service for which image optimization should be enabled. Then, set up image optimization by following the steps below.

Add the Fastly Image Optimizer header

Once image optimization has been activated on 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type `Fastly Image Optimizer`.
- From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
- In the **Destination** field, type `http.x-fastly-imageopto-api`.
- In the **Source** field, type `"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 type `"fastly; qp=*"` instead.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `1`.

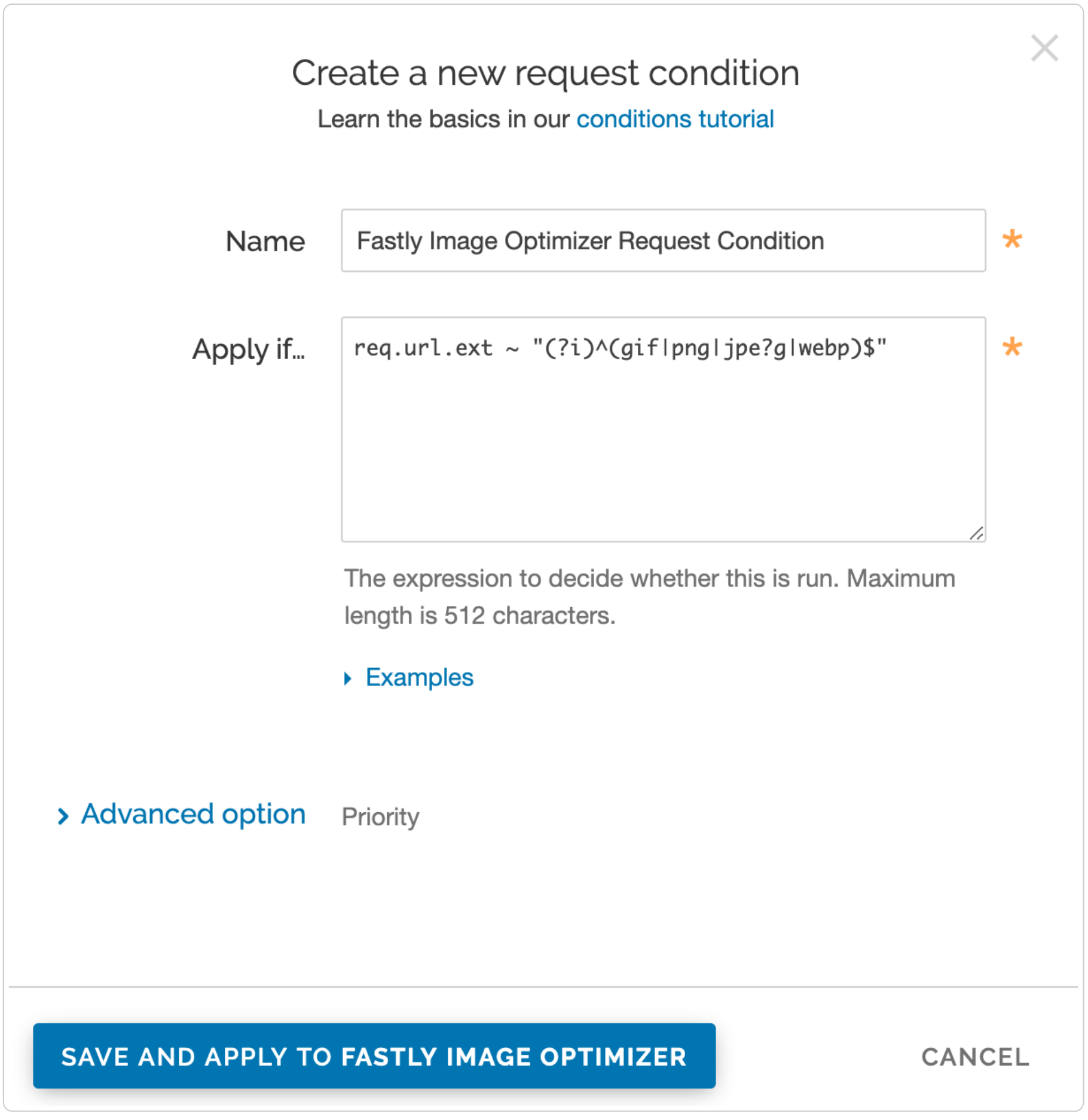
7. Click **Create** to create the new header.

> ★ **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, type a descriptive name for the new condition (for example, `Fastly Image Optimizer Request Condition`).
- In the **Apply if** field, type 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.

Enable shielding

To reduce cache miss latency and ensure long-lived connections, you must enable shielding for your origin. The shield location should be as geographically close to your image's origin as possible.

Our guide to enabling shielding provides more information on how to set this up. 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.

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
```

Review and edit the default image settings

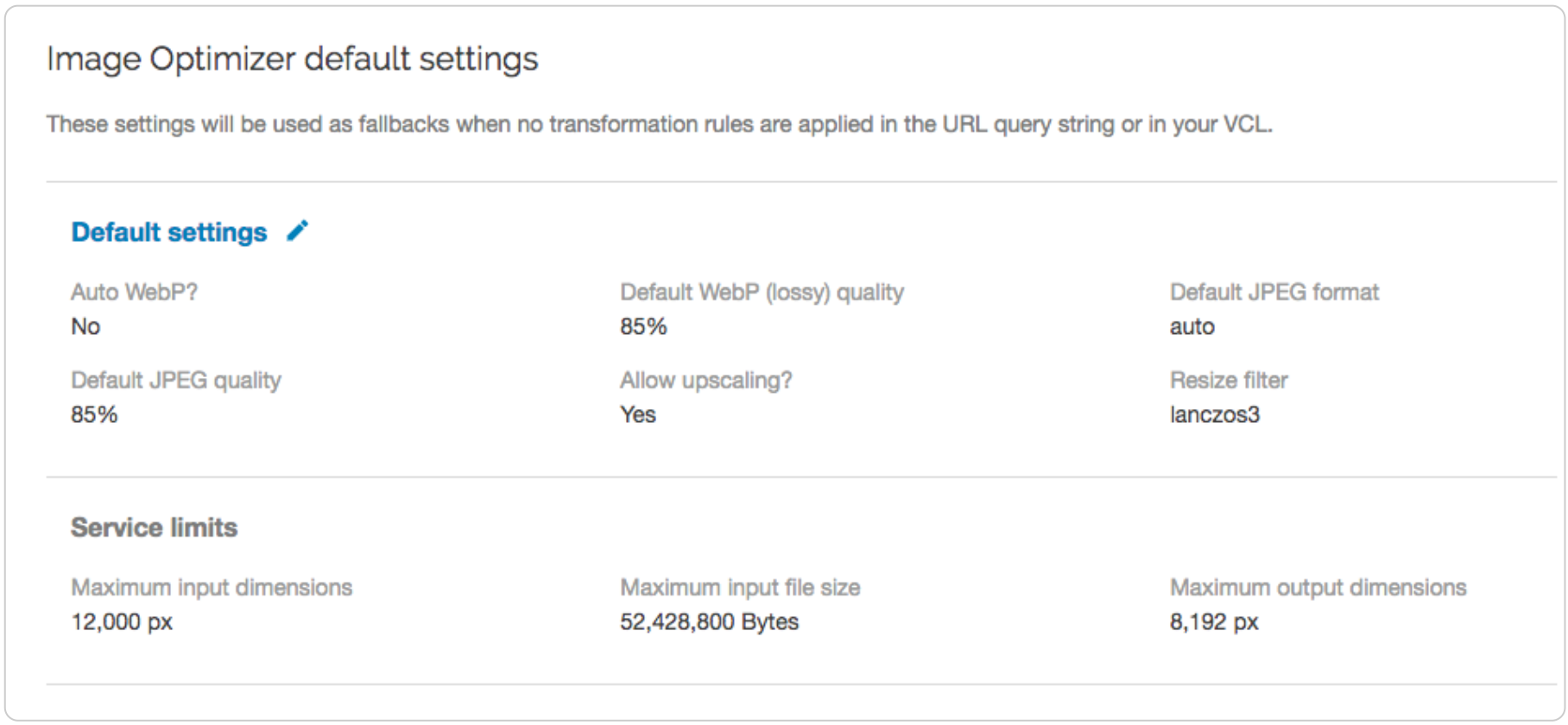
Fastly applies specific image optimization settings to all images by default.

Changing default image settings in the web interface

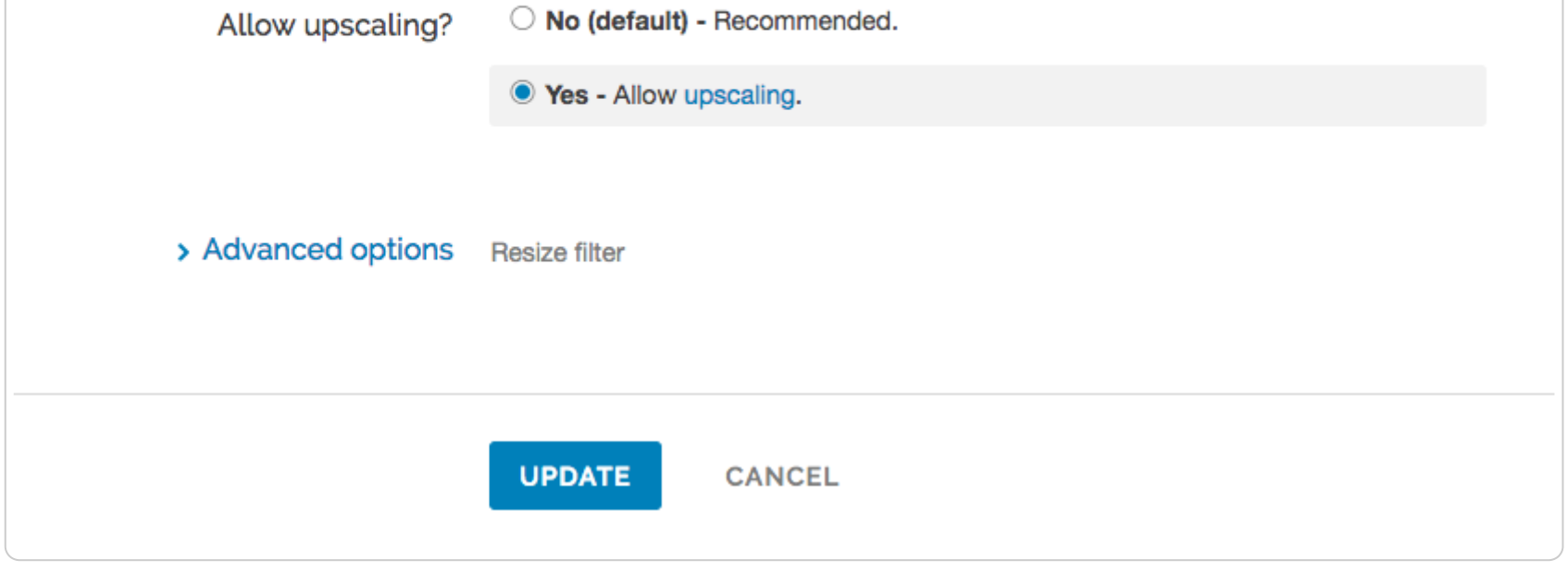
The Fastly web interface provides the easiest way to review the default optimization settings in a single location. You can use the web interface to make changes to these settings as well. Changes to other image settings, however, including most image transformations, require issuing API calls.

To review and edit the default image settings via the web interface, follow the steps below:

1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

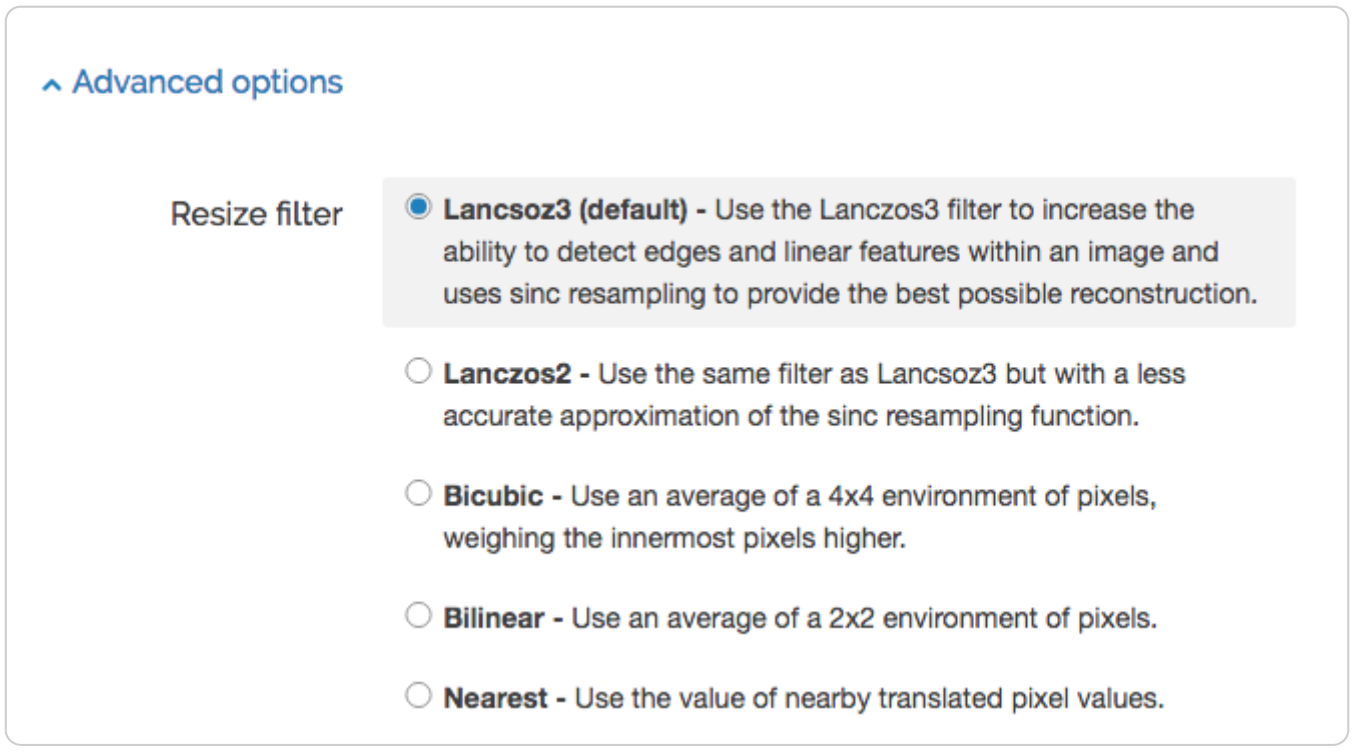


6. Adjust the **Edit default settings** as follows:
   - From the **Auto WebP** controls, leave the settings at their default or select **Yes** to convert images to the WebP format in browsers that support it. When you use the default setting, **No**, Fastly uses the image file type instead.
   - In the **Default WebP (lossy) quality** field, leave the settings at their default or type 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 type 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. 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**.

Changing image settings other than the defaults via API calls

The Fastly web interface only allows you to change the most basic settings of image optimization and transformation. For more complex changes to settings beyond these defaults, you must change your existing image URLs by adding Fastly API query string parameters to them. For example, if your image source existed at `http://www.example.com/image.jpg`, you would need to add `?<PARAMETER=VALUE>` to create the proper query string structure for Fastly to transform the image in some way.

You can change existing URLs in the source by adding one or more Fastly URL API query string parameters directly to your site's HTML. You can also change them programmatically. For more information about how to do this, see our guides and API documentation as follows:

- Our image optimization examples demonstrate some of the most common image transformations you can add to your URLs, like cropping and resizing. 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.
- Our guide to serving images provides additional details you should know before you start adding Fastly image transformation URL API query string parameters to your existing image URLs. It specifically discusses the transformation order of parameters when you specify more than one parameter at a time (e.g., `?<PARAMETER1=VALUE&PARAMETER2=VALUE>`).
- 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.

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.

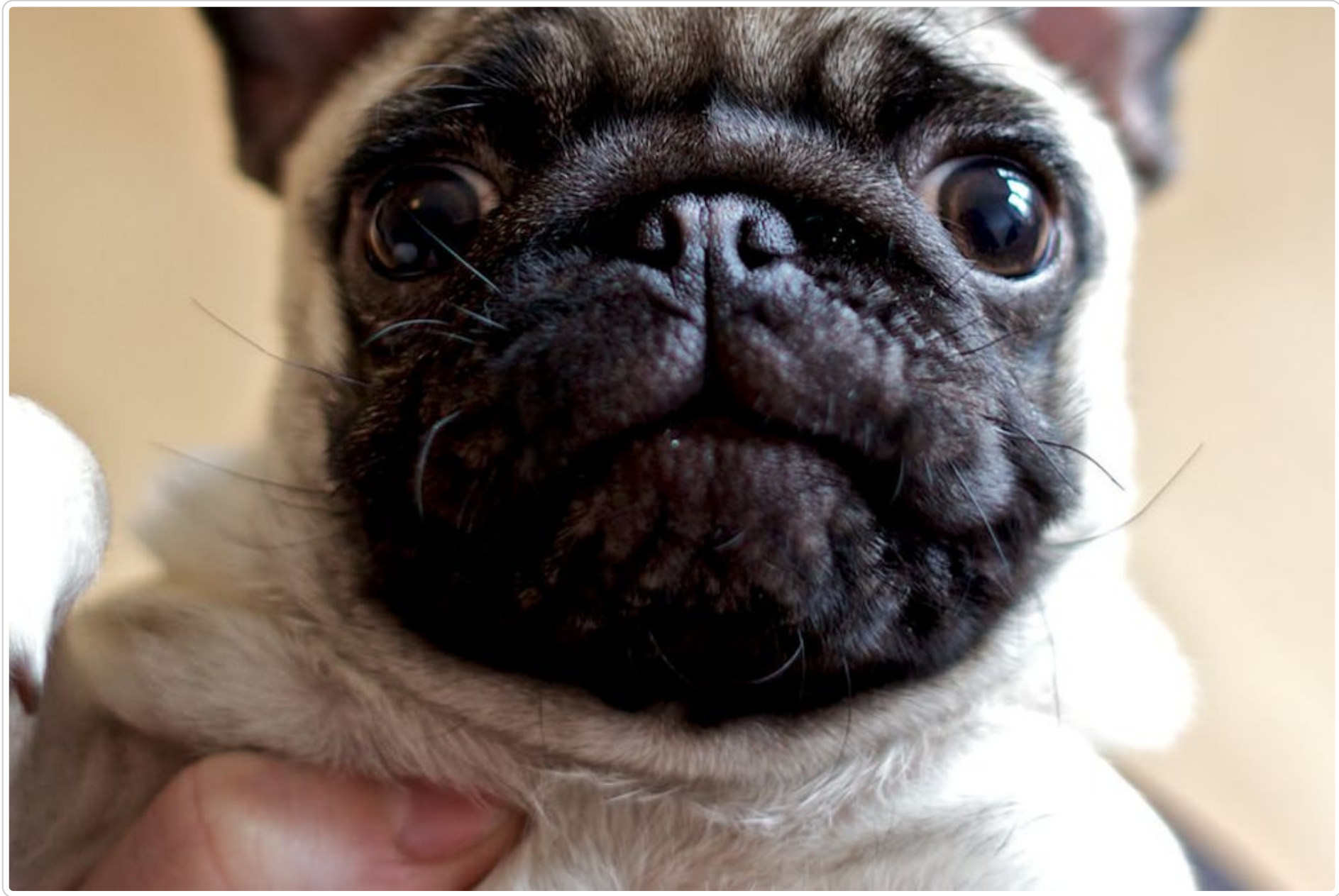
Trimming images

https://docs.fastly.com/en/guides/trimming-images

Trimming all edges by the same percentage

Trim all edges by 25%.

`https://www.fastly.io/image.jpg?trim=0.25`



Trimming parallel edges the same percentage

Trim top and bottom edge 25px, right and left edge 50px.

```
https://www.fastly.io/image.jpg?trim=25,50
```

Trimming all edges a different percentage

Trim top edge 25px, right edge 50px, bottom edge 75px and left edge 100px

`https://www.fastly.io/image.jpg?trim=25,50,75,100`

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.

Video

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

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

Directory path example	Description
`/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

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type a human-readable name for the condition.
   - In the **Apply if** field, type `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

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.

Advanced TCP optimizations

You can enable TCP optimizations between cache servers and clients to improve response time, specifically metrics like video startup times (a.k.a., "time-to-first-frame") and re-buffering percentages. Consider implementing the following optional, TCP-related configurations to improve a client's experience.

Change the default value for client.socket.cwnd

Set the TCP socket initial congestion window (a.k.a., initial CWND) to 30. The default value is 10. To do this, add the following VCL to your service using VCL Snippets:

```
sub vcl_deliver {
#FASTLY deliver

  # increase init cwnd for only client requests
  if (!req.http.Fastly-FF && client.requests == 1) {
    set client.socket.cwnd = 30;
  }

  return(deliver);
}
```

Enable the experimental BBR congestion algorithm

Enable the BBR TCP congestion control algorithm. 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

Access Control Lists

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

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.

How 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.

How to use ACLs

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 event logs. 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

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

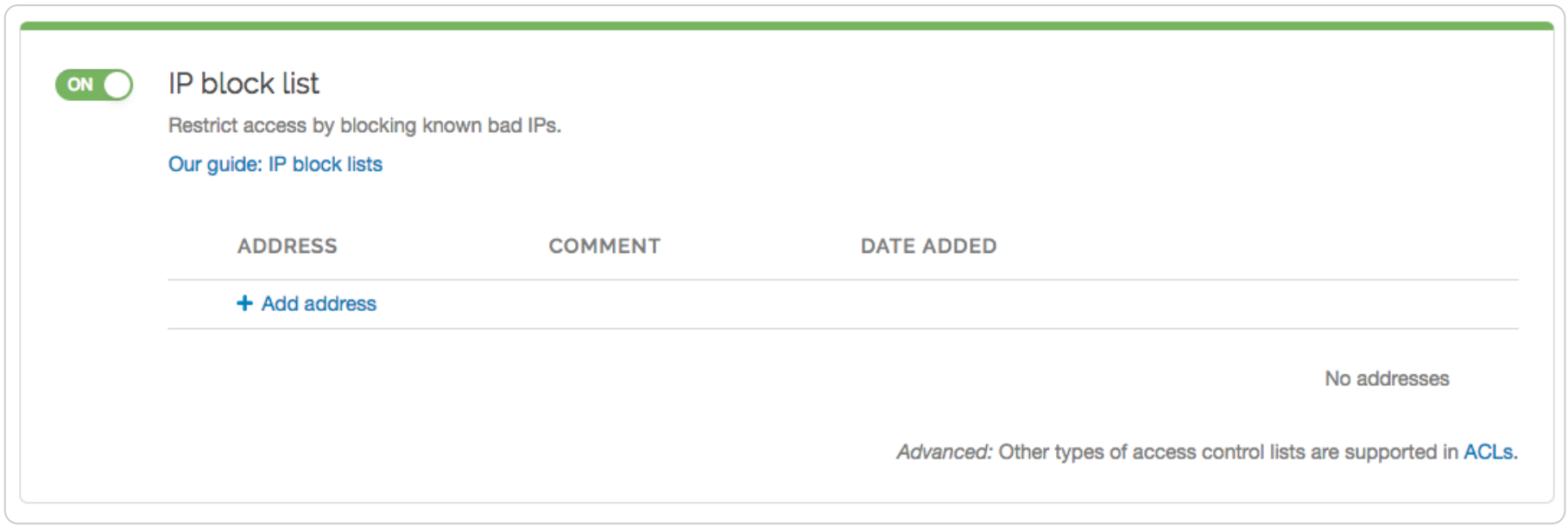
https://docs.fastly.com/en/guides/using-the-ip-block-list

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. Click the **IP block list** switch to **On**.

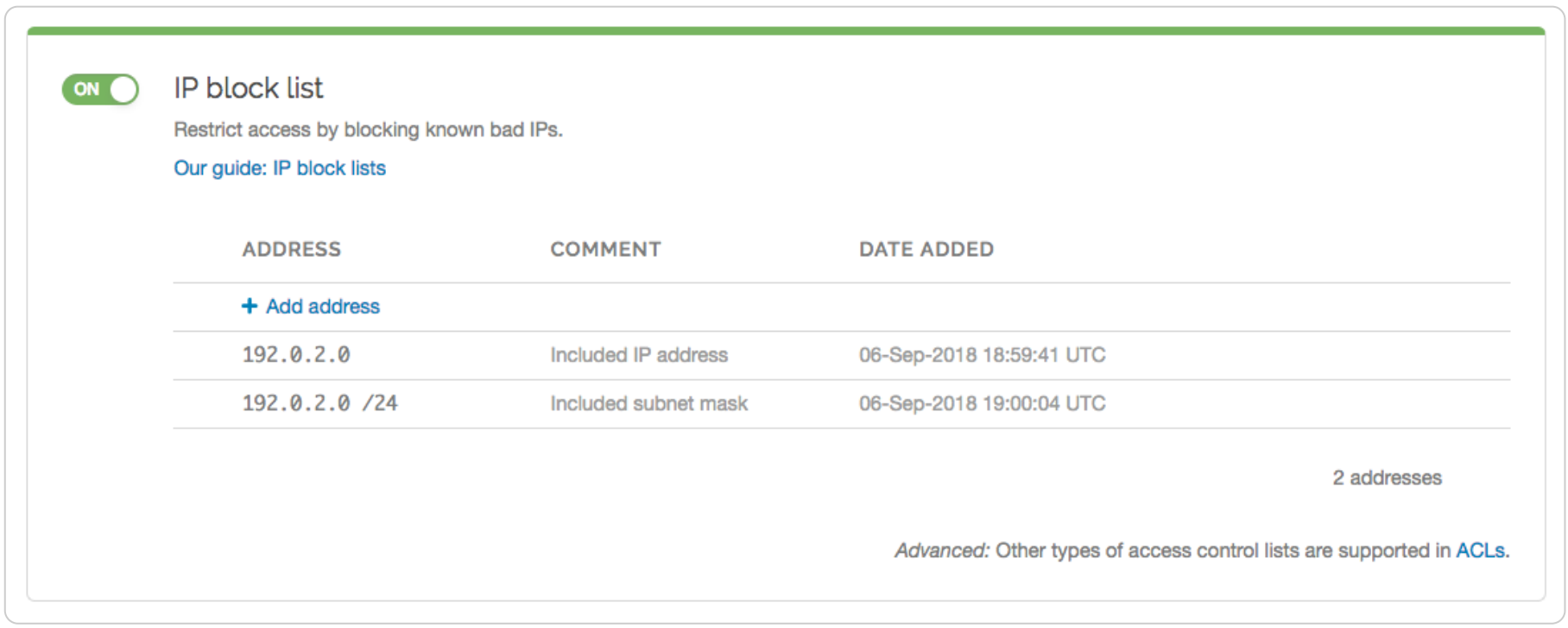


6. Click the **Activate** button to deploy your configuration changes.

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, type 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, type 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

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 ID 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_n
umber>/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>/a
cl/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>/a
cl
```

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

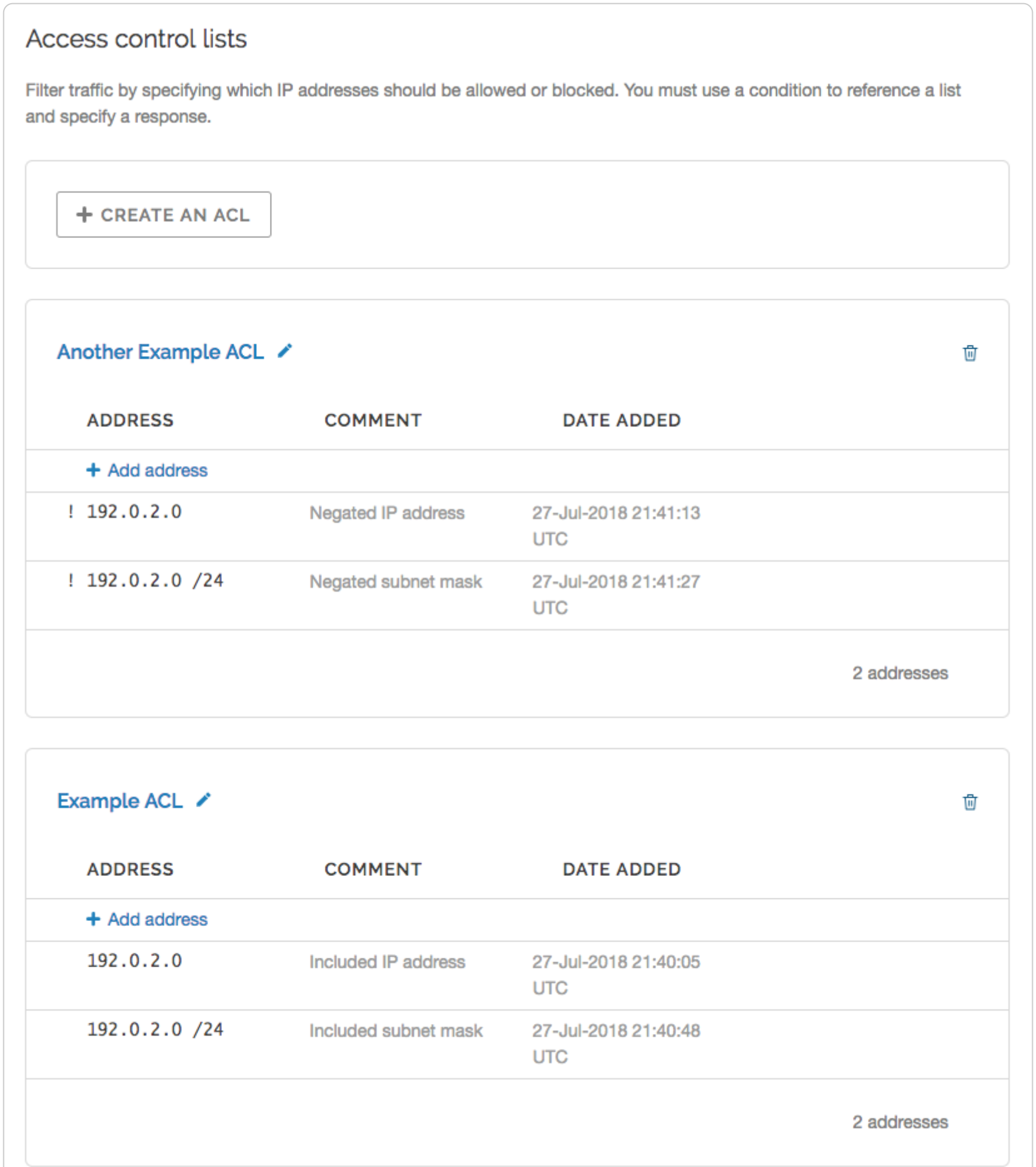
https://docs.fastly.com/en/guides/working-with-acls-using-the-web-interface

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
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.

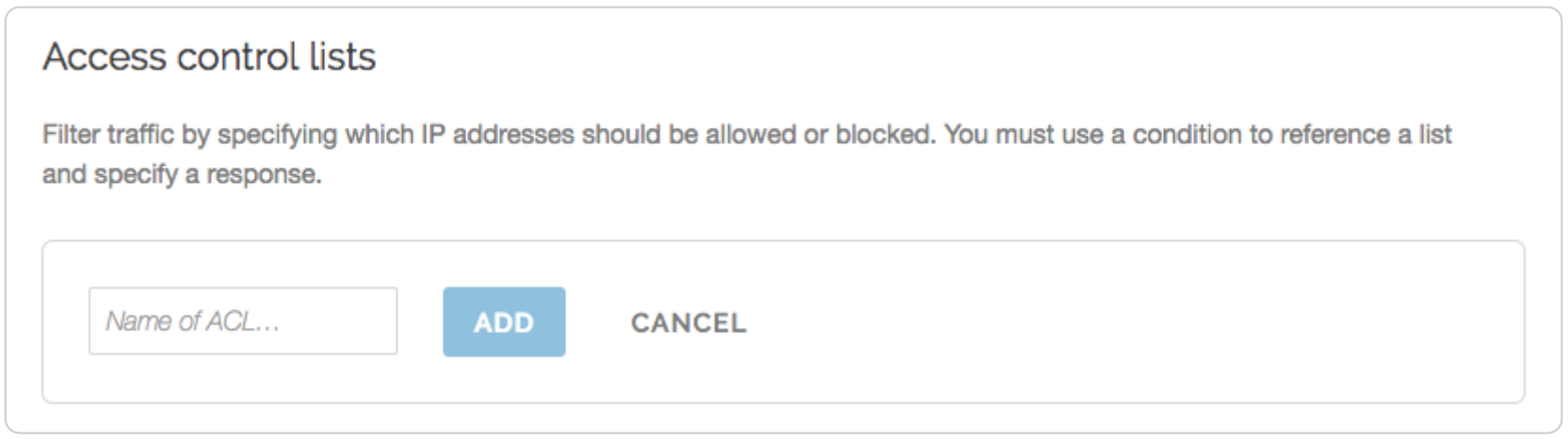
Creating an ACL

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Data** link. The Data page appears.
5. Click **Create an ACL**. The ACL container name field appears.

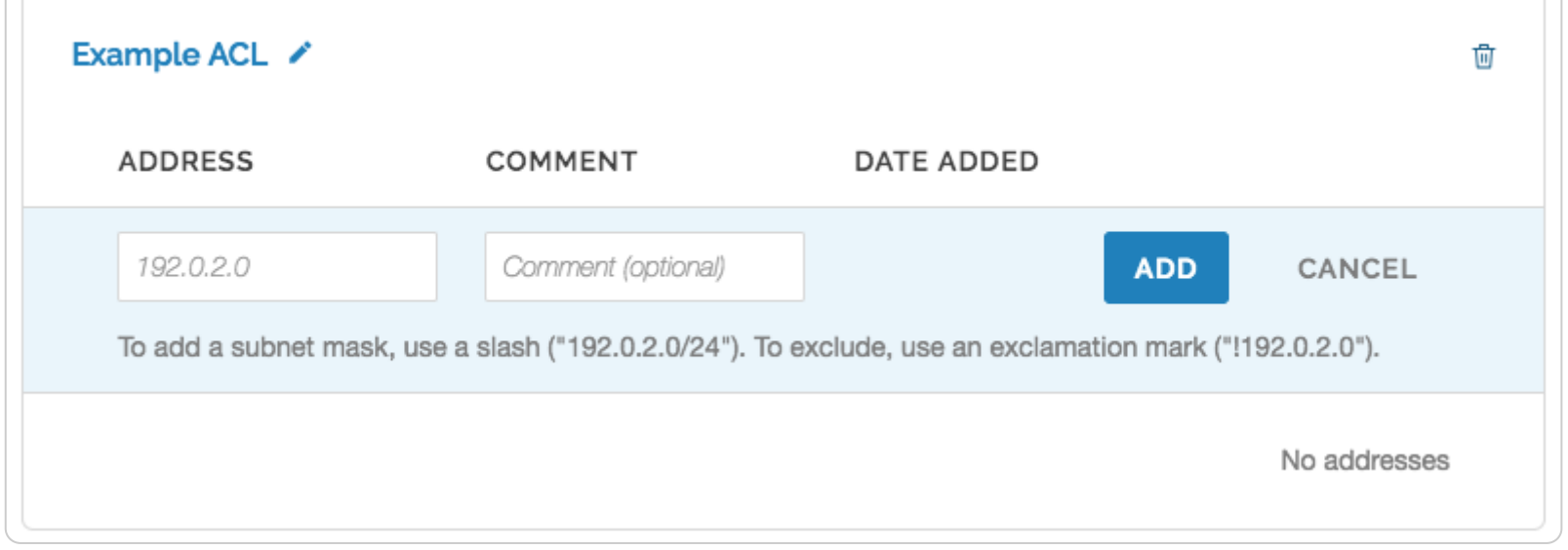


6. In the **Name of ACL** field, type a descriptive name for the ACL (e.g., `Example ACL`).
7. Click the **Add** button. The empty ACL container you created appears.
8. Click the **Activate** button to deploy your configuration changes to the service version you're editing.

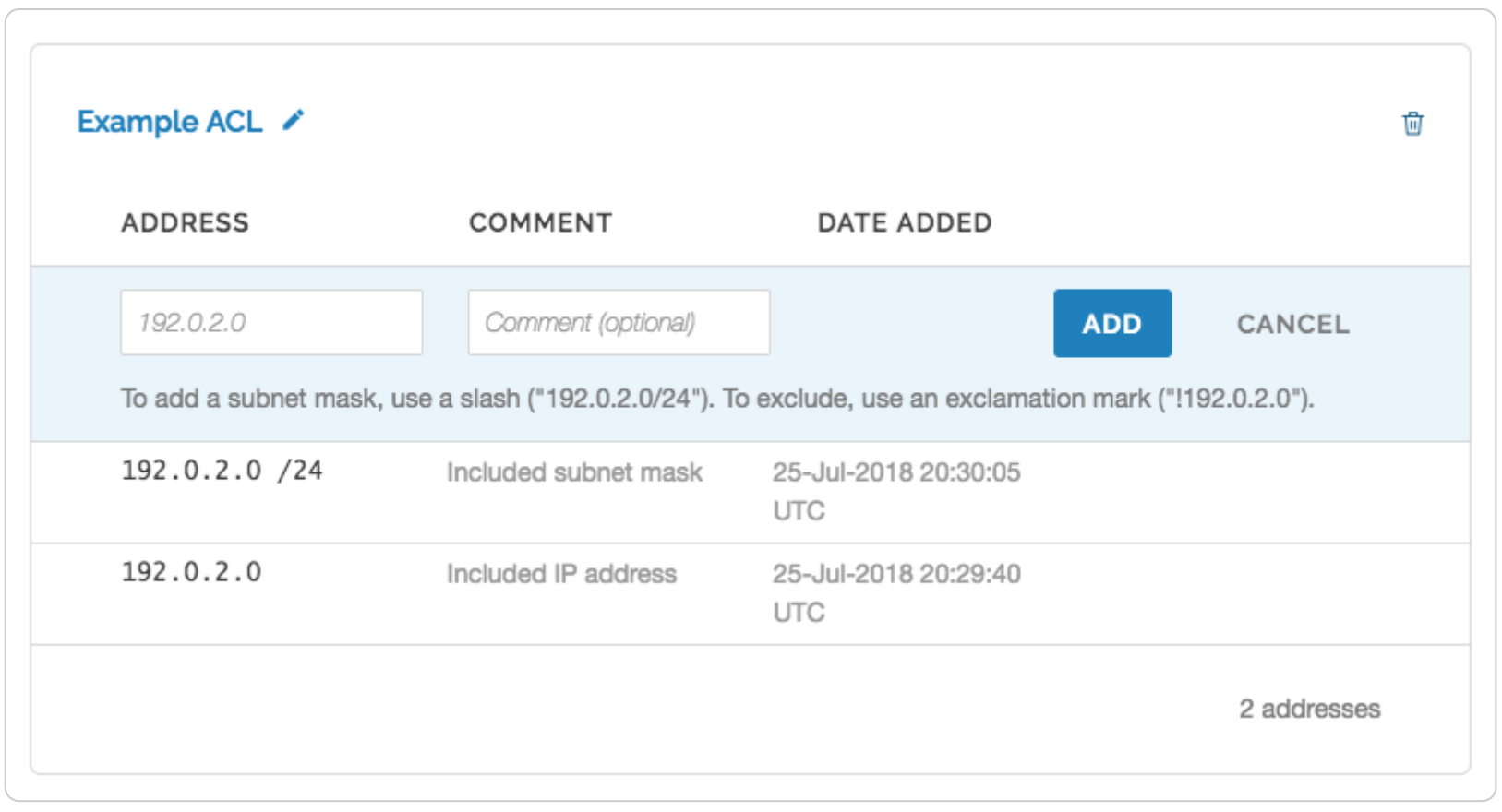
Creating an ACL entry

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, type 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, type 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.

Monitoring and testing

These articles provide information about monitoring and testing the security of services behind Fastly.

https://docs.fastly.com/en/guides/security#_monitoring-and-testing

Monitoring account activity with event logs

https://docs.fastly.com/en/guides/monitoring-account-activity-with-event-logs

Event logs keep track of events related to your services, account, and users. You can use event logs to determine which changes were made and by whom. For example, you can use them to:

- see who activated the most recent version of your service
- review who logged in to your account via the web interface
- learn which users have two-factor authentication enabled
- view recent service configuration setting changes

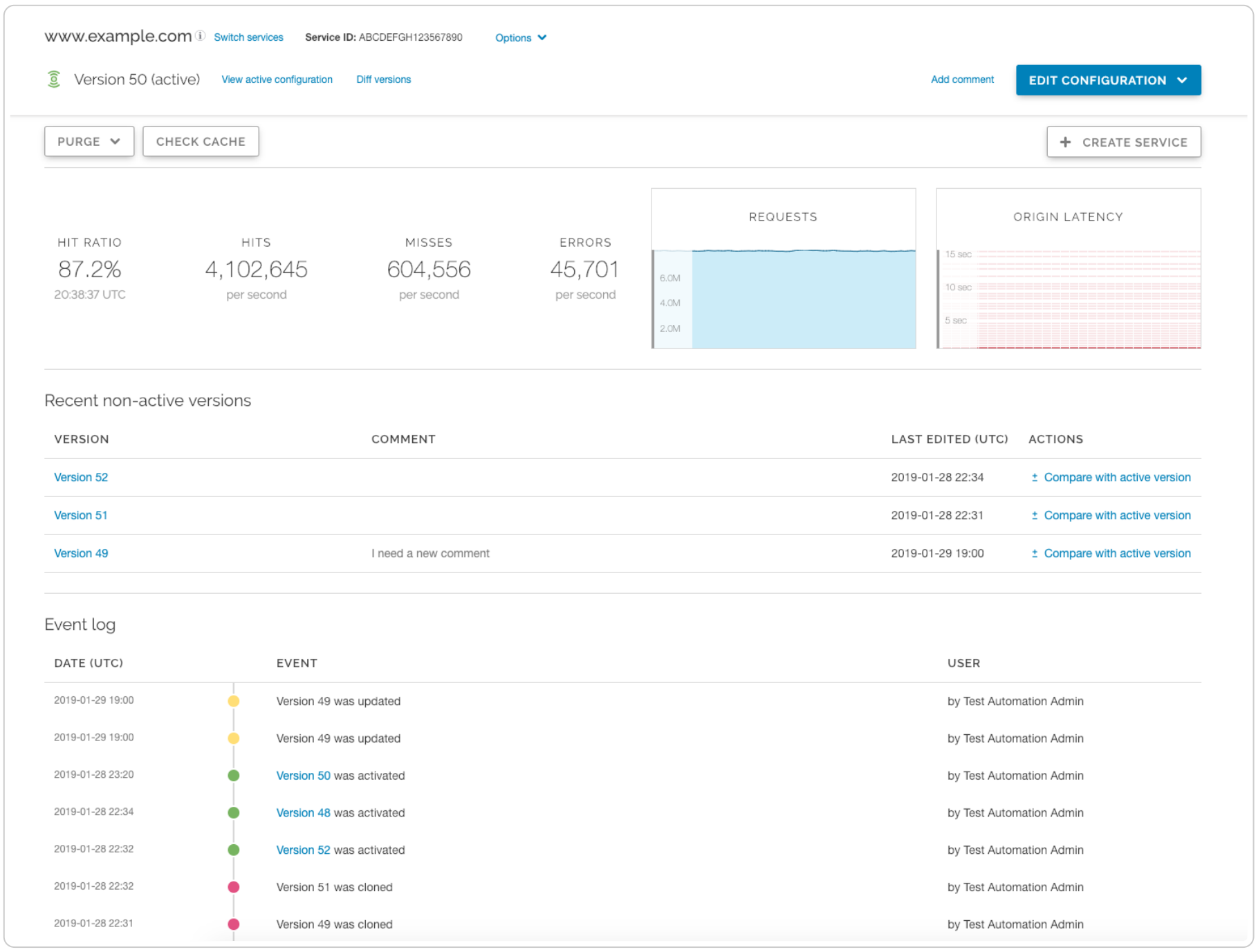
Use the Fastly API to retrieve a service's event logs or view a limited subset of those logs via the web interface.

Accessing event logs via the web interface

The web interface displays the last 20 service-related events for the selected service. Events related to users and accounts are not displayed in the web interface.

Follow these instructions to access the event logs for a service:

1. Log in to the Fastly web interface and click the **Configure** link.
2. The most recent service-related events are displayed near the bottom of the page, in the Event log area.



NOTE: Event log data is currently retained indefinitely.

Accessing event logs via the API

The `/events` API endpoint can be used to retrieve a service's event logs. 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.

Penetration testing your service behind Fastly

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.

Securing communications

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

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. You can access the list via URL:

https://api.fastly.com/public-ip-list

You can then automate the API call (for example, by running a script as a cron job) to request the list of IPs to detect when the IP ranges change.

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

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:didComple
teWithError:.
================================================================================

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
---
```

```
84
85  ---
86  TLSv1.0 with PFS disabled
87
88  Result : PASS
89  ---
90
91  ================================================================================
92
93  Configuring TLS exceptions with PFS disabled and insecure HTTP allowed for www.fastly.com
94
95  ---
96  TLSv1.2 with PFS disabled and insecure HTTP allowed
97
98  Result : PASS
99  ---
10
 0  ---
10  TLSv1.1 with PFS disabled and insecure HTTP allowed
 1
10  Result : PASS
 2  ---
10
 3  ---
10  TLSv1.0 with PFS disabled and insecure HTTP allowed
 4
10  Result : PASS
 5  ---
10
 6
10
 7
10
 8
10
 9
11
 0
```

Security measures

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

Security measures

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.

- Security program
- Technology compliance

NOTE: 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 event logs

- PerimeterX Bot Defender
- Streaming logs
- Securing communications
- TLS

Security program

https://docs.fastly.com/en/guides/security-program

Fastly's security program includes safeguards that help protect your data as it moves through the Fastly service. Information about these safeguards is organized by category. Our technology compliance guide describes additional safeguards we maintain.

Authentication and authorization

User account assignment. We assign individual user accounts to personnel who access Fastly systems and devices. These assignments help us monitor and enforce accountability of user activity.

User-level privileges. Our systems and devices enforce user roles or similar measures to control the extent of access we grant individual users.

Multi-factor authentication. We enforce multi-factor authentication to better secure our computing resources from unauthorized logins.

Application security

Secure software development. We provide annual training to Fastly developers to help identify and prevent common software vulnerabilities, including the OWASP Top 10. Developer code undergoes peer review prior to deployment, and internal security engineers and third-party security validators periodically analyze code for software components with higher potential security risk.

Web application security review. A third party assesses the security of the Fastly web application annually. We address findings from this assessment according to the risk they pose to the security of the Fastly service.

Network and infrastructure security

Network security reviews. We regularly perform vulnerability scans and third-party penetration tests on the Fastly network. We review and address findings from these activities to help maintain the security of our network.

Configuration standards. We document and follow configuration standards to maintain secure systems and network devices. These standards include business justification for used ports, protocols, and services, as well as the removal of insecure default settings.

Vulnerability and patch management. To maintain awareness of potential security vulnerabilities, Fastly monitors 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.

Encryption

Secure data transmission. The Fastly service supports TLS configurations to encrypt connections both externally to end users and backend origin servers, as well as internally within the Fastly network.

Encryption key management. We maintain technology and procedures to secure private keys throughout their lifecycle.

Key storage and access security. We store private keys in encrypted repositories, and we restrict key storage access to personnel who support our key management processes.

Datacenter and physical security

Physical access restrictions. Our datacenters are fully enclosed with perimeter protection such as fences, gates, and mantraps to prevent unwanted entry. Only authorized people (including datacenter personnel, our employees, and contractors) may enter and move within a datacenter.

Datacenter access management. We ensure movement within our datacenters is monitored via onsite safeguards such as security guard assignment, facility access logging and review, and video surveillance. Additionally, we periodically review and adjust the list of personnel who may enter our datacenters.

Secure asset installation. We install computer and network hardware in locked cages and racks. Only authorized individuals may physically access this equipment.

Environmental safeguards. Our datacenters compensate for environmental disruptions with systems that control backup power, temperature and humidity, and fire suppression.

Business continuity and operational resilience

Service failover. If any of our points of presence (POPs) experience issues serving content, we can redirect traffic to a neighboring POP without interrupting the delivery of content to end users.

Internet redundancy. Our datacenters have connections with multiple internet service providers. We do not rely on any single carrier to serve content to end users.

Service monitoring. We monitor multiple internal and external reporting channels to detect service-related issues. Personnel are available 24x7x365 to confirm and respond to disruptions of the Fastly service.

Communication and reporting. We update impacted customers using various communication methods (such as status.fastly.com), depending on an incident's scope and severity.

Security incident management

Incident response plan. 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 the Fastly service.

Incident response team. Representatives from key departments help address security-related incidents we discover. These personnel coordinate the investigation and resolution of incidents, as well as communication with external contacts as needed.

Breach notification. Fastly will notify affected customers within 48 hours of validating an unauthorized disclosure of customer confidential information.

Logging and monitoring

Log analysis. We aggregate and securely store Fastly internal system activity. Monitoring these logs helps us discover and investigate potential security issues.

Change and configuration monitoring. We use multiple monitoring and alert mechanisms to enhance the visibility of technology changes and help ensure adherence to our change management process.

Intrusion detection. We maintain mechanisms to detect potential intrusions at the network and host level. Our Security department inspects and responds to events these detection measures discover.

Customer and end user data management

Cache data and configurations. Customers manage which content is cached, where, and for how long by setting policies that control that content. See our introduction to caching for more information. We may directly access or modify customer accounts or configurations to provide our services, prevent or address service or technical issues, as required by law, or as customers expressly permit. For the same reasons, we may also access or modify equipment, systems, or services that manage customer content.

Client IP addresses. As part of our caching network's general interaction with the internet, Fastly independently collects anonymized and aggregated client IP address information on a limited basis to provide and improve its services. Client IP addresses are retained 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 (such as 503 "Service Unavailable" errors), and are discarded thereafter.

Subscriber IP addresses. Fastly independently collects the IP addresses of users who access their services within the Fastly web interface or through the API. We make these IP addresses available to customers through our event log functionality. If customers define origin servers or syslog endpoints with IP addresses, we save those IP addresses as part of their configurations. We may retain IP addresses from event logs or configurations indefinitely. Dynamically-resolved origin IP addresses may be retained for up to two business days, or up to seven days if those addresses are associated with transmission errors (such as 503 "Service Unavailable" errors), and are discarded thereafter.

IP addresses and security monitoring. Fastly may retain indefinitely any non-anonymized, non-aggregated client or subscriber IP addresses associated with suspicious activity that may pose a risk to the Fastly network or our customers, or that are associated with administrative connections to the Fastly service.

Content request data. Content enters, transits, and departs our network in response to 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.

Subscriber log streaming. Subscribers may stream syslog activity, including end user IP addresses, to a remote endpoint for analysis and use. Fastly does not retain subscriber syslog activity, except as described above.

Cloud infrastructure security and compliance program

The use of third-party cloud infrastructure to host Fastly products that deliver content or process requests requires us to address certain aspects of our security and technology compliance programs differently from when Fastly directly manages the infrastructure.

Datacenter and physical security. For cloud infrastructure we use, Fastly relies on datacenter space under the control of the cloud infrastructure providers. These providers may have physical access to assets that contain data from Fastly services. As part of our third-party security review process, we confirm that these providers maintain appropriate physical security measures to protect their datacenter facilities.

Business continuity and operational resilience. We deploy cloud-hosted products in multiple infrastructure regions or zones to help maintain those services when operational issues occur. If failure of a service occurs within a single availability zone, Fastly will automatically attempt to use cloud nodes in another zone.

Encryption. Fastly leverages in-transit and at-rest encryption to help secure data sent between Fastly and the cloud infrastructure provider or to secure data that resides on cloud infrastructure. Because we use at-rest encryption features offered by infrastructure providers, those providers may also hold the private encryption keys. As part of our third-party security review process, we confirm that these providers maintain secure encryption key management processes.

Technology compliance

https://docs.fastly.com/en/guides/technology-compliance

Fastly's technology compliance program includes safeguards that help protect your data as it moves through the Fastly service. Information about these safeguards is organized by category. Our security program guide describes additional safeguards we maintain.

Governance

Information security roles and responsibilities. We have formally assigned information security duties to Fastly personnel. Our Chief Security Officer and Security organization work with other departments to safeguard sensitive information related to the Fastly service.

Policies and procedures. Our policies and procedures help us maintain security in our systems, processes, and employee practices. Fastly's Security organization formally reviews these policies and procedures at least annually.

Risk management. We integrate risk assessment activities with various processes to identify and address information security risk to the company and customer data on our network.

Vendor security oversight. Fastly performs 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

Employee background screening. 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.

Confidentiality agreement. Our employees formally agree to safeguard the sensitive information they may view, process, or transmit as part of their job functions.

Security awareness training. 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.

Data privacy

Privacy policy. Our privacy policy describes how we collect, use, and protect the personal information of customer personnel using our websites or configuring services in the Fastly web interface. We certify our privacy policy and practices with TRUSTe.

Personal data transfer. The Fastly services by default do not process personal data. However, our service can be configured or used at the direction of the customer to process personal data. Our guide about our terms provides additional information about data privacy compliance related to the processing of personal data.

Technology change management

Change management process. We follow a defined set of procedures to develop and deploy technology changes. These changes include updates to software, configurations, and devices that support the Fastly service.

Testing. We test technology changes at various stages of development, and we confirm those tests are successful before completing a deployment into the Fastly service.

Change approval and notification. As part of our deployment process, we prepare, approve, and communicate change notices to maintain awareness among personnel who manage the Fastly network and systems.

Post-implementation review. We confirm the success of changes after their deployment. Should we experience issues during implementation, we also maintain procedures to revert changes.

Identity and access management

User requests and approval. We document and approve requests for user access to the Fastly network. Our security administrators confirm appropriate documentation is in place before granting requested user rights.

Access modification. We promptly update or remove an employee's access to the Fastly network to match that employee's current job function or employment status.

User access review. We periodically inspect access privileges to make sure our personnel have appropriate access to Fastly systems and data.

TLS

These articles describe how to set up TLS certificates with Fastly services.

https://docs.fastly.com/en/guides/security#_tls

Domain validation for TLS certificates

https://docs.fastly.com/en/guides/domain-validation-for-tls-certificates

When you purchase one of Fastly's TLS options, our partner certification authority (GlobalSign) must verify you control the domains requested and that you authorize us to request a certificate service on your behalf. You can choose:

- DNS text record verification (preferred)
- Email verification
- URL verification

Regardless of the verification method you use, be sure to follow our instructions to begin the TLS ordering process.

DNS text record verification

We provide you with a unique DNS TXT record you need to add for the zone origin ("@") for each of your domains. The text of this entry will change depending on the certificate to which each domain is added. The meta tag will be formatted similar to one of the following (where the `{META TAG}` will change depending on the certificate):

- `@ IN TXT "globalsign-domain-verification={META TAG}"`
- `@ IN TXT "_globalsign-domain-verification={META TAG}"`

We will provide you with the appropriate text record listed above. Consult the documentation for your registrar or DNS provider for more information about how to add the record. This text record must be wholly separate from other text records. A prepended, inserted, or appended record will not work.

Email verification

GlobalSign will give Fastly a list of acceptable email addresses to which they can send a validation email. Generally these email addresses will be the following:

- admin@example.com
- administrator@example.com
- hostmaster@example.com
- postmaster@example.com
- webmaster@example.com

For entries requested for a subdomain, each of those addresses `@subdomain.example.com` will also work (e.g., `admin@subdomain.example.com`).

We will send you the list of acceptable email address. You will need to tell us which email address to use. GlobalSign will then send a verification email to the email address you specify. Once you receive the verification email, you will need to click on a link in that email and follow the instructions to complete the validation.

URL verification

We provide you with an HTML meta tag you need to add to a specifically named web page served at the requested domain or apex domain you're adding. Use the format `http://<REQUESTED APEX OR SUBDOMAIN>/.well-known/pki-validation/gsdv.txt` where `<REQUESTED APEX OR SUBDOMAIN>` is the domain being added to the certificate. The meta tag will be formatted similar to one of the following (where the `{META TAG}` text will change depending on the certificate):

- `<meta name="globalsign-domain-verification" content="{META TAG}" />`
- `<meta name="_globalsign-domain-verification" content="{META TAG}" />`

We will provide you with the appropriate meta tag listed above. This text must be served from the actual requested domain or root domain. For example, if you add the domain `www.example.com` to the certificate, GlobalSign will specifically query `http://www.example.com` or `http://example.com` during the verification process. The verification tag must be served from whatever resource is returned from that URL. GlobalSign will not follow redirects or request a file on that domain, such as `http://www.example.com/verify.html` or `http://www.example.com/index.html`.

Assisted TLS domain validation

To provide uninterrupted TLS services to your origin, Fastly automatically revalidates domains using the HTTP based validation method. Validation happens automatically at regular intervals prior to certificate renewal and does not require any action by you. As long as you maintain your DNS pointing to Fastly we will perform assisted TLS validation to avoid any potential interruption to your service.

If you do not want assisted TLS validation enabled, contact support@fastly.com for additional options.

Enabling HSTS through Fastly

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.

> **NOTE:** HSTS only takes effect *after* a site has been visited on a trusted HTTPS connection. It doesn't replace the need to have redirects from your HTTP site.

Prerequisites

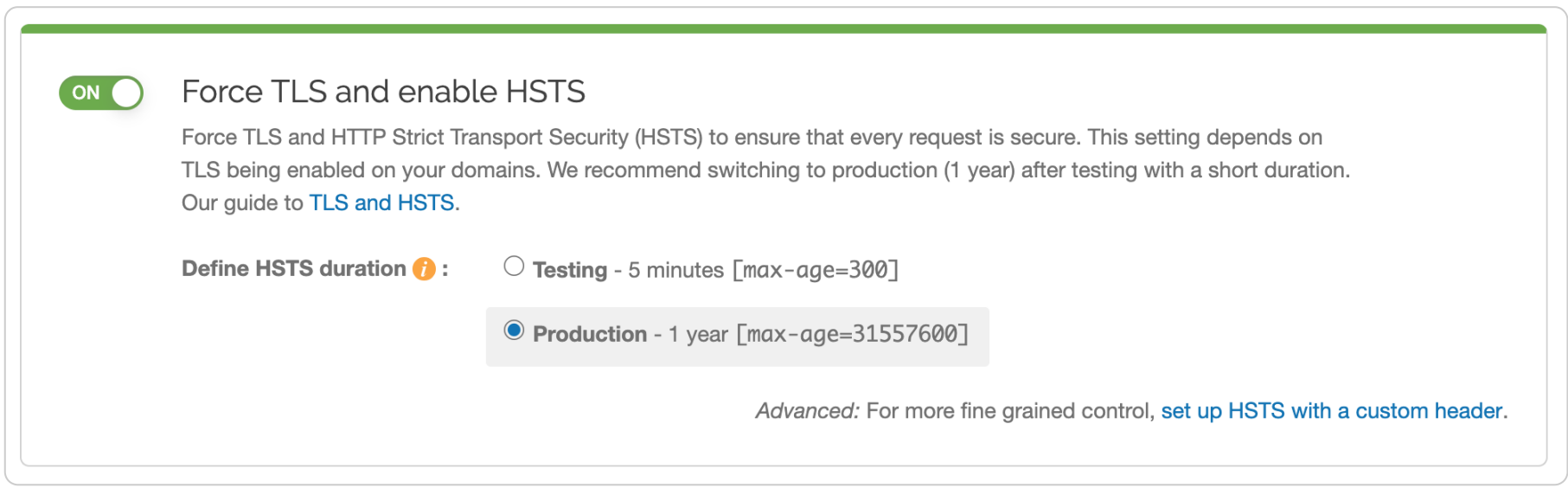
These instructions assume that you've set up TLS service with Fastly.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Settings** link. The Settings page appears.
5. 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.

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

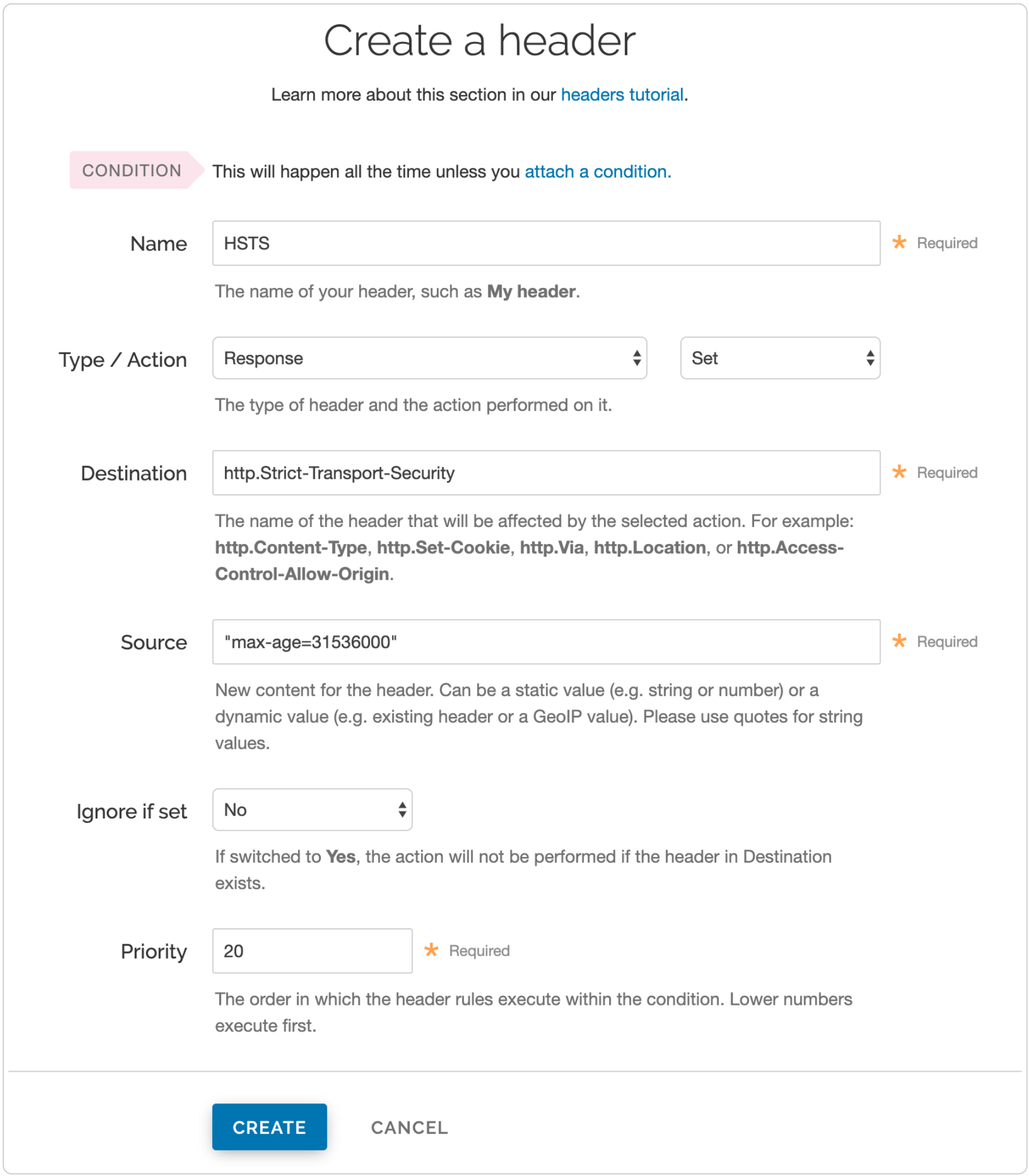
> ⚠ **WARNING:** You may experience problems if you enable this setting along with the override host setting. Instead of enabling the override host setting, create a new request setting and specify the override host in the advanced options.

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.

> ℹ **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, type 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, type `http.Strict-Transport-Security`.
- In the **Source** field, type `"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

Forcing a TLS redirect

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/foo.jpeg would redirect to **https**://www.example.com/foo.jpeg.

WARNING: Requests can still happen over HTTP first even if you force a TLS redirect using these instructions. To keep this from happening, enable HTTP Strict Transport Security (HSTS).

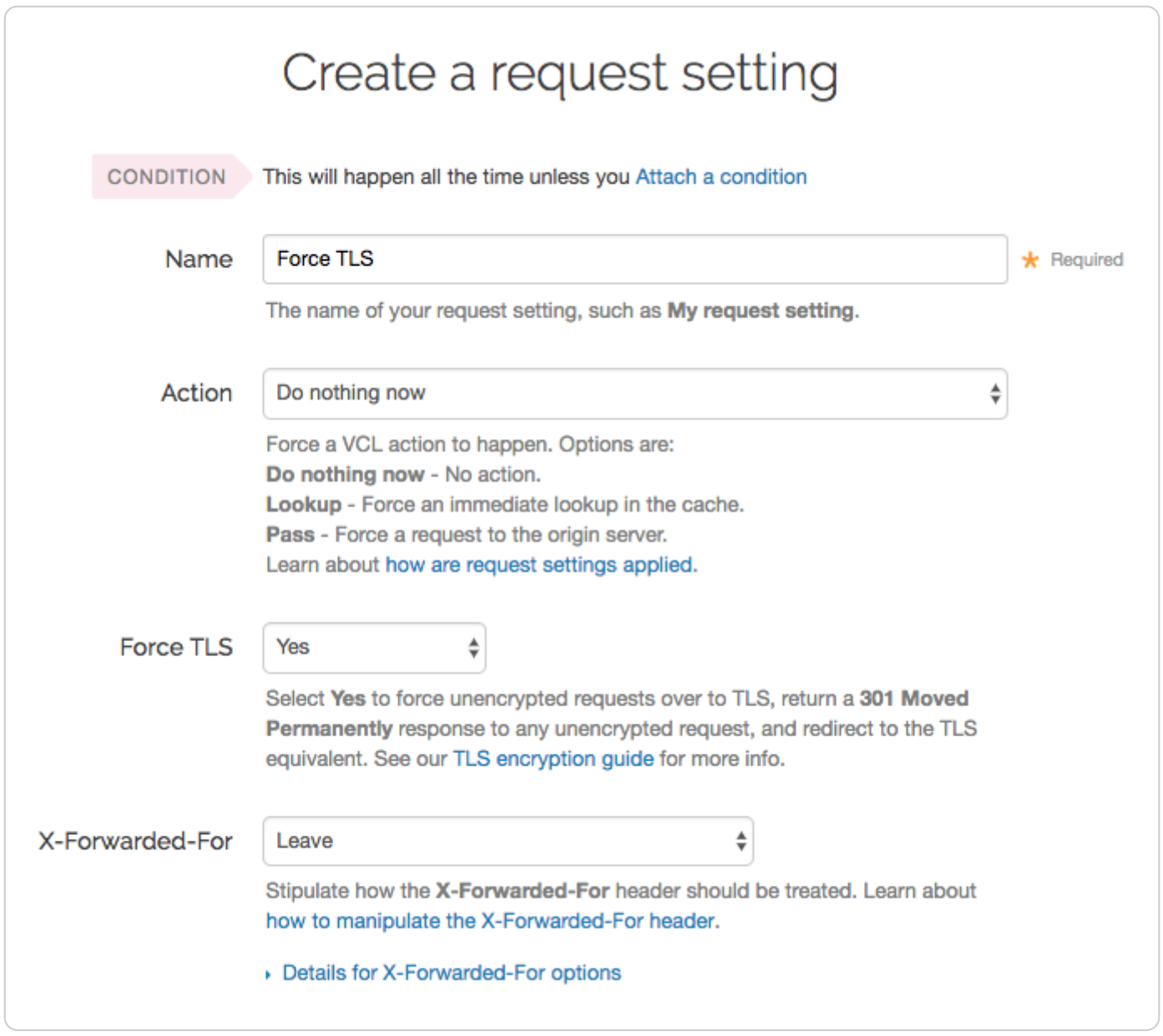
Prerequisites

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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.

Managing domains on TLS certificates

https://docs.fastly.com/en/guides/managing-domains-on-tls-certificates

The Fastly web interface allows you to add and manage domains on one of Fastly's shared TLS certificates. For example, to serve HTTPS traffic for a single website you can add a single domain like www.example.com. However, if you add a wildcard domain, like `*.example.com`, you will be able to serve HTTPS traffic on any related subdomain, like `api.example.com` and `docs.example.com`.

Before you begin

Be sure you understand your TLS options:

- **If you don't have a TLS certificate**, you can add a domain to one of Fastly's shared certificates. Simply complete the steps for adding a TLS domain described in this guide. You'll automatically be billed for this service monthly.
- **If you already have a TLS certificate or if you require a dedicated certificate**, contact support@fastly.com to purchase one of Fastly's hosted TLS certificate options.

Also, when you are managing your TLS domains, keep the following in mind:

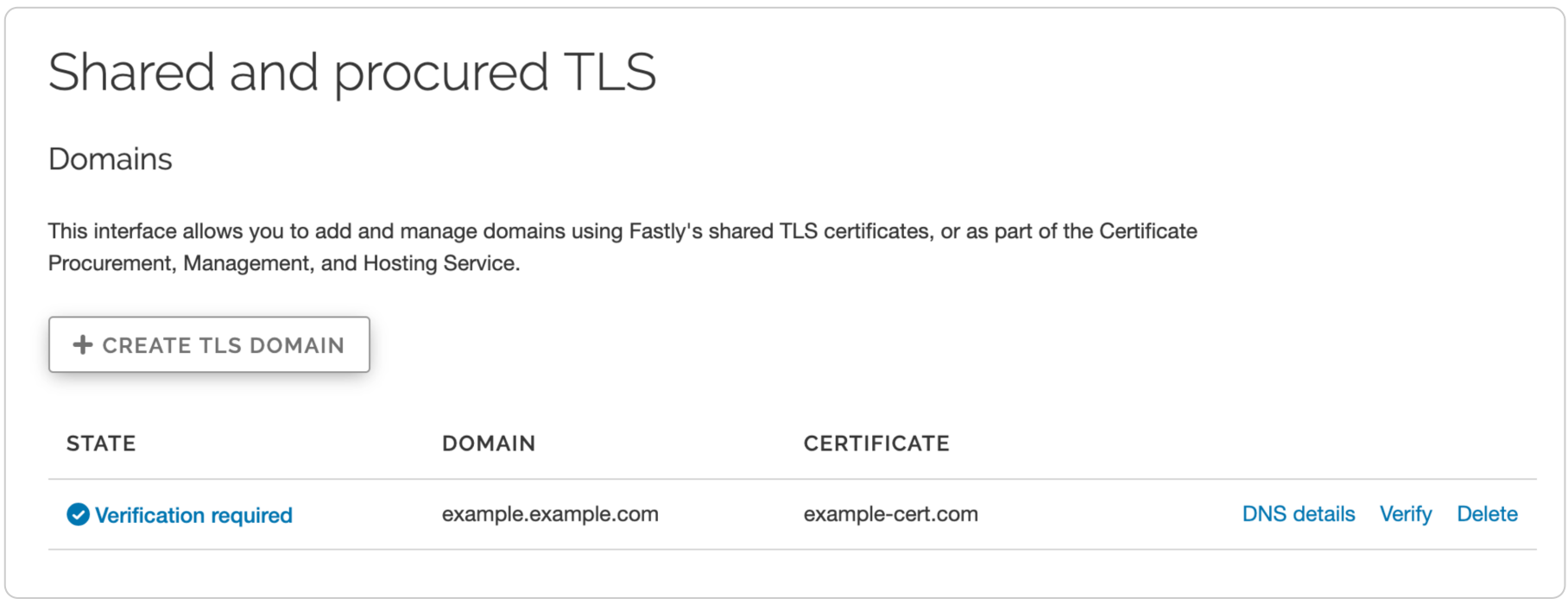
- **You can only manage certificates with a paid Fastly account.** If you're currently using a developer trial account, switch to a paid account first.

- **You must be assigned the role of superuser or have been granted permission to manage account-level TLS.** Only users with TLS management capabilities can manage domains on certificates.
- **You can add up to a total of five TLS domains.** If you require more than five domains, contact support@fastly.com.
- **Each domain you add to a Fastly shared certificate increases your monthly bill.** You'll be automatically charged for each addition the first full month in which it gets used.

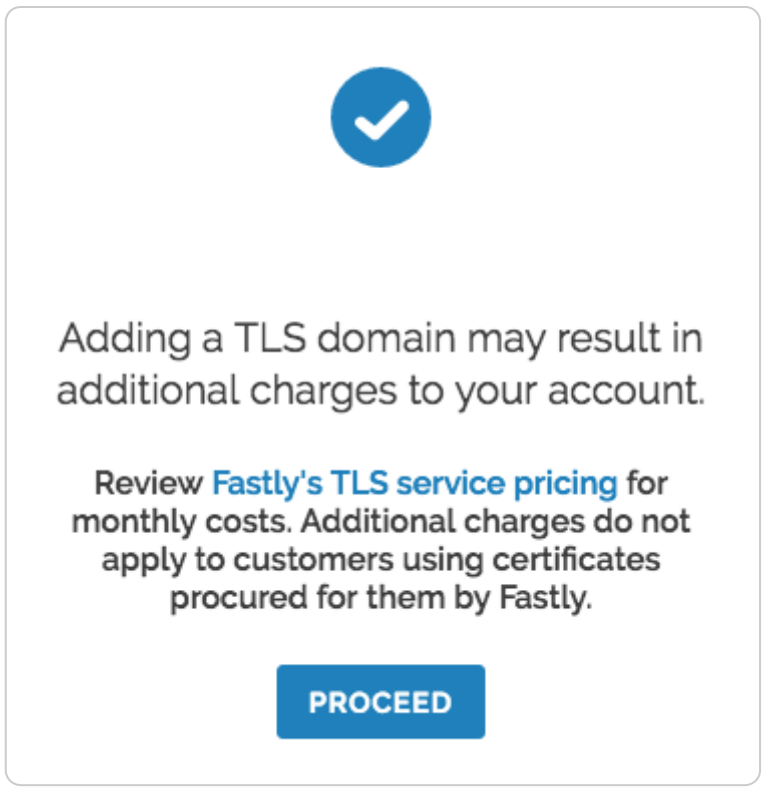
Creating a TLS domain

To create a TLS domain, follow these steps:

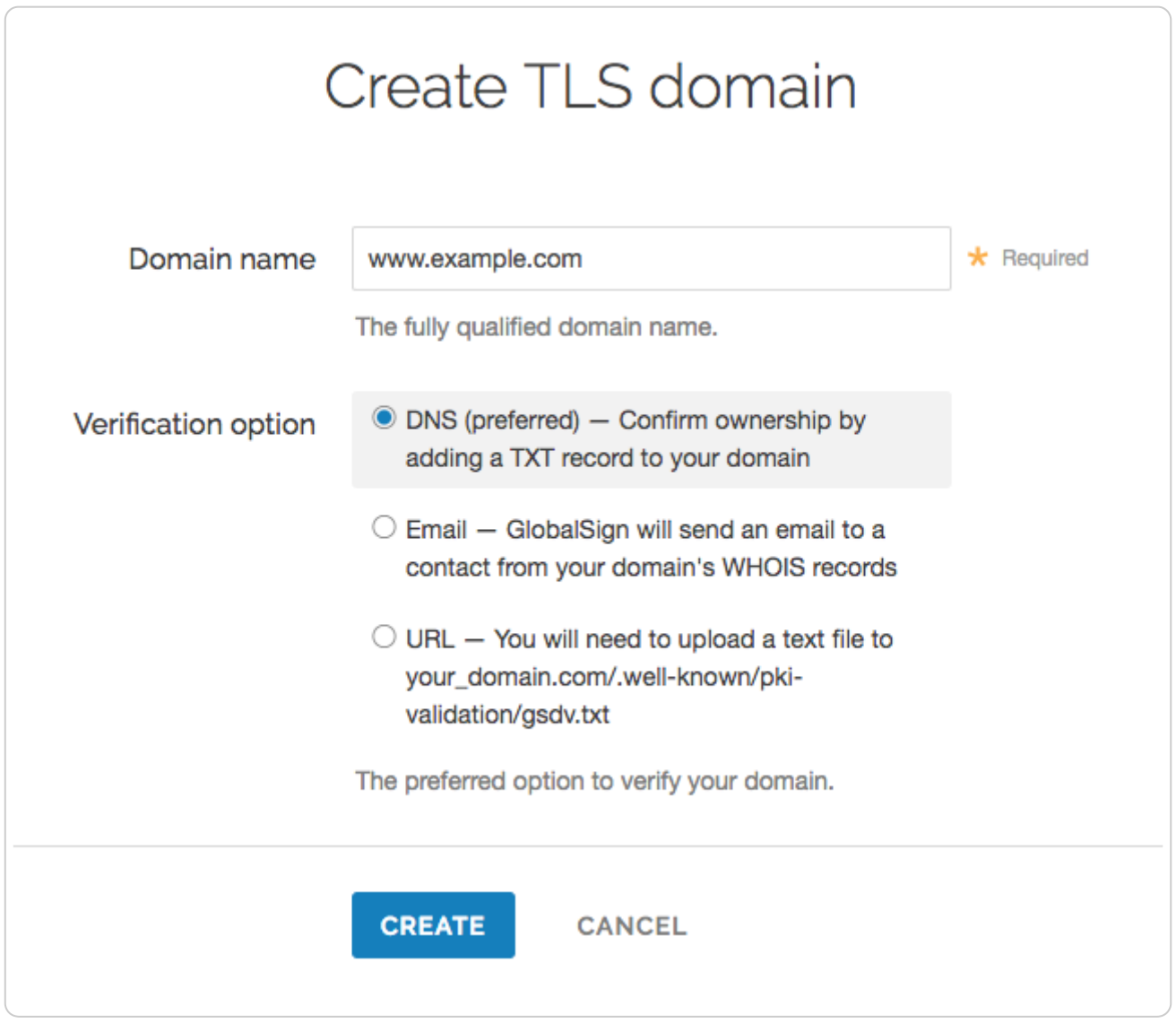
1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Shared and procured TLS** link. The Shared and procured TLS page appears.



3. In the **Domains** area, click the **Create TLS Domain** button. A billing increase notification appears.



4. Click **Proceed**. The Create TLS domain page appears.



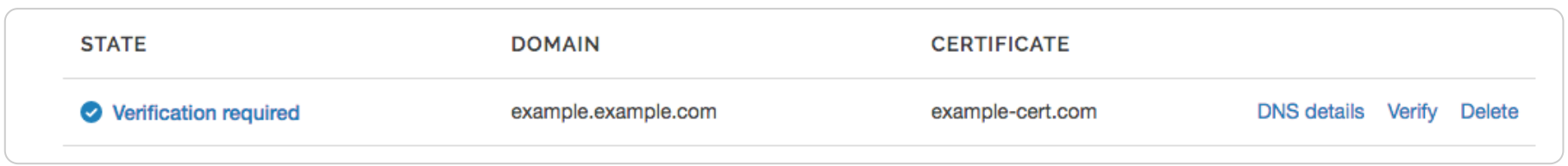
5. Fill out the **Create TLS domain** form as follows:
   - In the **Domain name** field, type the fully qualified domain name to be added to the selected TLS certificate (e.g., `www.example.com` or `*.example.com`).
   - If the **Certificate** menu appears, select the certificate on which to create the domain. This menu only appears if you've previously arranged for Fastly to procure a certificate on your behalf.
   - From the **Verification option** controls, select the method you prefer to use for domain ownership verification. The DNS verification method will be used by default unless you select another option.
6. Click the **Create** button. The request is sent to Fastly for creation and appears as a row in a table in the Domains area of the Transport Layer Security page.

TIP: The table in the Domains area always reflects the current state of your request during processing. You'll need to review that state as you verify domain ownership and when you connect your service to your TLS domain. Always review the state of your request before contacting support if you suspect trouble.

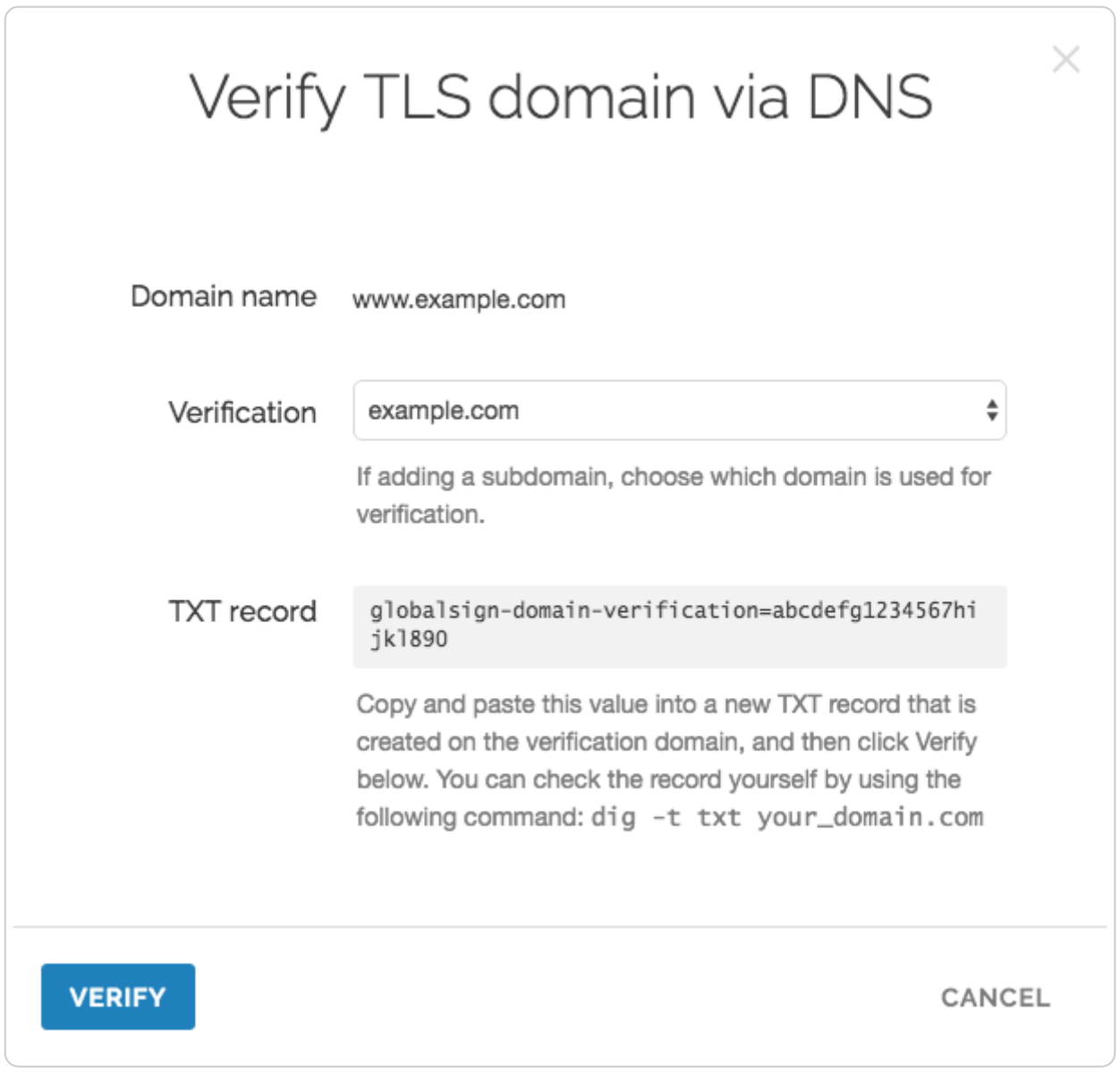
Verifying domain ownership

Any time you request addition of a domain to a certificate, you must verify you own the domain. This helps us ensure no one else is using your domain without your permission. To verify domain ownership, follow these steps:

1. On the **Shared and procured TLS** page, look in the **Domains** list for the TLS domain name you created and review the **State**.



2. When the **State** changes to **Verification required** (usually only a few minutes after Fastly receives your request), click the **Verify** link. The Verify TLS domain window appears.



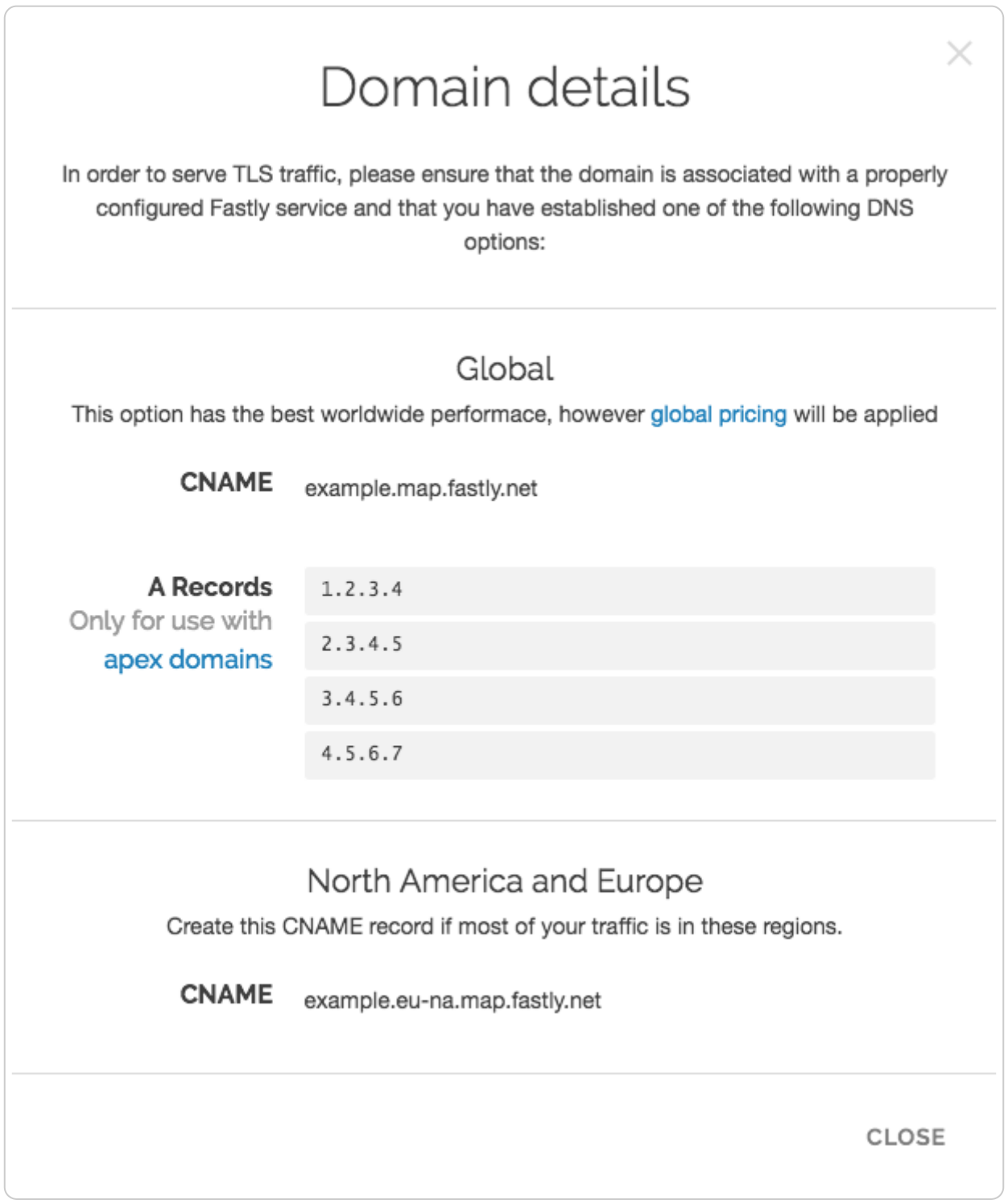
3. Depending on the verification method you selected, do one of the following:
   - Verify the domain via DNS. You'll need to validate domain ownership by adding a DNS TXT record for your domain with your DNS provider.
   - Verify the domain via email. You'll need to validate domain ownership by clicking the link that GlobalSign emails to the contact you've designated for your domain's WHOIS records.
   - Verify the domain via URL. You'll need to validate domain ownership by uploading a text file to a specifically named web page served at the domain you're adding.
4. Click the **Verify** button after you've completed the domain verification steps. This is Fastly's cue to add your domain to the certificate.

Within a few minutes of verification, you'll see the **State** change to **Issued**. This means the domain has begun propagating throughout Fastly's cache nodes and you're ready to connect a service. Within 60 minutes, the domain should be live and Fastly will begin the monthly billing process for these specific TLS certificate services.

Enabling TLS for your service

Once you've verified your domain ownership, you need to connect a service to your TLS domain. Follow these steps:

1. On the **Shared and procured TLS** page, look in the **Domains** list for the TLS domain name you verified and review the **State**.
2. When the domain's **State** changes to **Issued**, click the **DNS details** link. The Domain details page appears.



3. Use the information on the **Domain details** page to update the CNAME record or A Record for your domain with your DNS provider.

> ★ **TIP:** Once you've updated the CNAME or A record for your domain with your DNS provider, we suggest adding that domain to a new or existing service if you haven't already done so.

Deleting a TLS domain

> ❗ **IMPORTANT:** Before you delete a TLS domain, we strongly recommend first modifying or deleting any DNS records pointing to the Fastly hostname associated with it. Follow the instructions on your DNS provider's website.

To delete a TLS domain, 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 **Shared and procured TLS** link. The Shared and procured TLS page appears.
3. In the **Domains** area, find the domain to be deleted and click the **Delete** link that appears to the right of the domain name on the same line. The deletion confirmation window appears.
4. In the **Re-enter domain name** field, type the domain name to be deleted.
5. Click the **Confirm and Delete** button. The request to remove the domain from the SAN certificate will be sent. This is Fastly's cue to remove the domain from the certificate.
6. Watch the **State** for the submitted domain. Once the domain's state changes to **Removed**, the domain has been removed from the certificate and Fastly will discontinue charging you for these specific TLS certificate services.

Understanding domain states

The **State** column on the **Shared and procured TLS** page changes to reflect the current stage of processing for all domain requests.

State	Description

State	Description
Request initiated	We've sent your domain request to our partner certification authority.
Phishing check	Our partner certification authority is performing extra domain ownership verification on this request.
Verification required	The domain request is complete. Your domain ownership verification is now required.
Verifying	Your domain ownership verification is being confirmed by our partner certification authority.
Email verification sent	Our partner certification authority has sent you a domain ownership verification email that requires action on your part.
Issuing	The domain ownership verification was successful and now awaits final issuing before being added to your certificate.
Issued	The domain was successfully added to the certificate. It may take up to 60 minutes to become active.
Removing	Your request to remove a domain from a certificate is being processed.
Removed	A domain was successfully removed from the certificate.

NOTE: Domains that do not get issued due to an error will be automatically removed after 3 weeks of inactivity. You can also manually remove domains if they get stuck in an error state in order to begin the verification process again.

Serving HTTPS traffic using certificates you manage

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 enable 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 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 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.

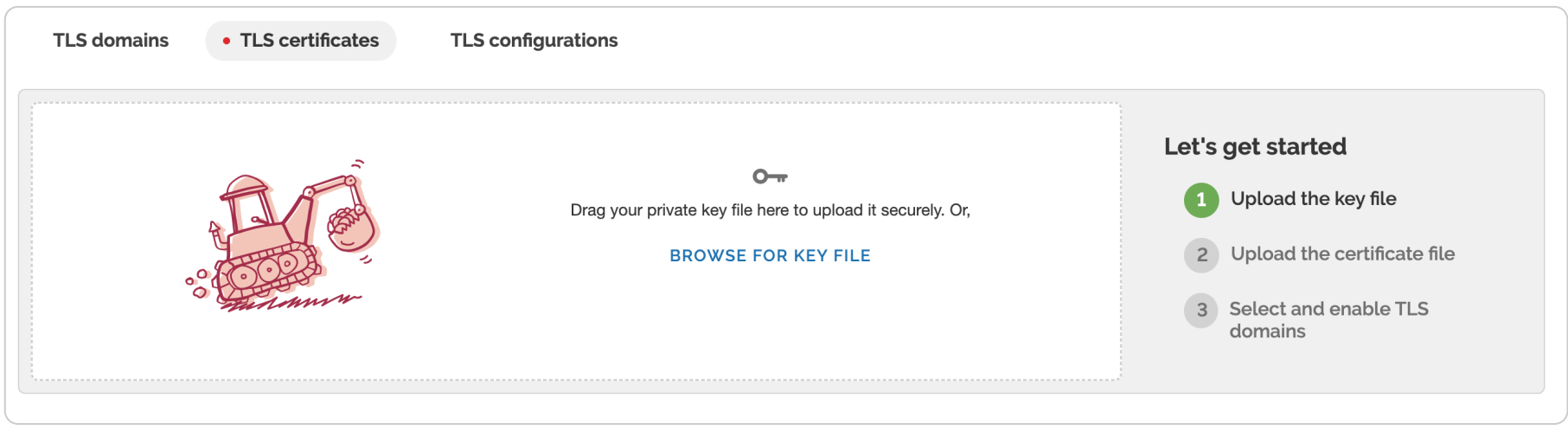
- Uploaded certificates require clients to support TLS v1.2 and Server Name Indication (SNI) by default. To discuss how you can use settings other than these defaults, contact sales@fastly.com. The ability to use custom settings may require you to use a dedicated Fastly IP address pool, which must be 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.

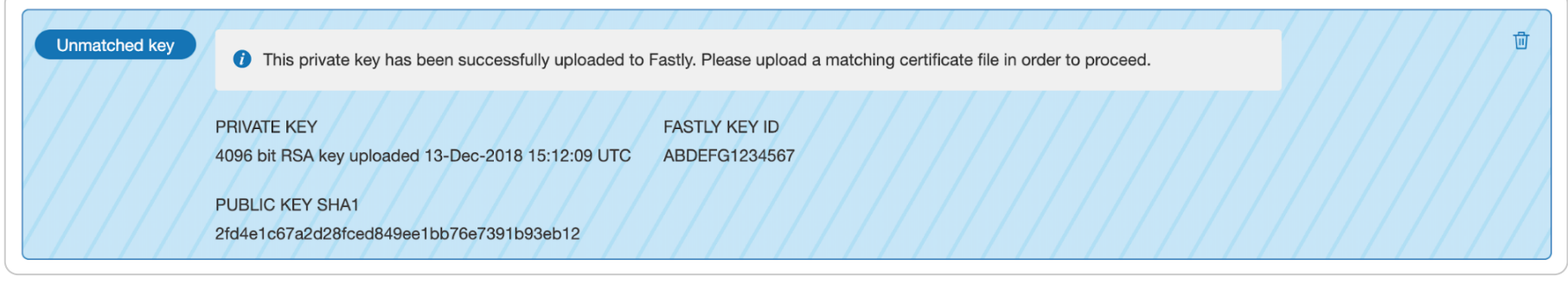
> **IMPORTANT:** The key file upload tool currently only accepts 2048-bit RSA keys. If you require longer key lengths, contact sales@fastly.com.

1. Log in to the Fastly web interface and click the **Configure** link.
2. Click the **HTTPS and network** tab. 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.
3. Click the **Secure another domain** button. The Enter domain window appears.
4. Click the **I want to bring my own certificate and private key** link. The TLS certificates tab appears displaying the secure upload controls.



5. 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.



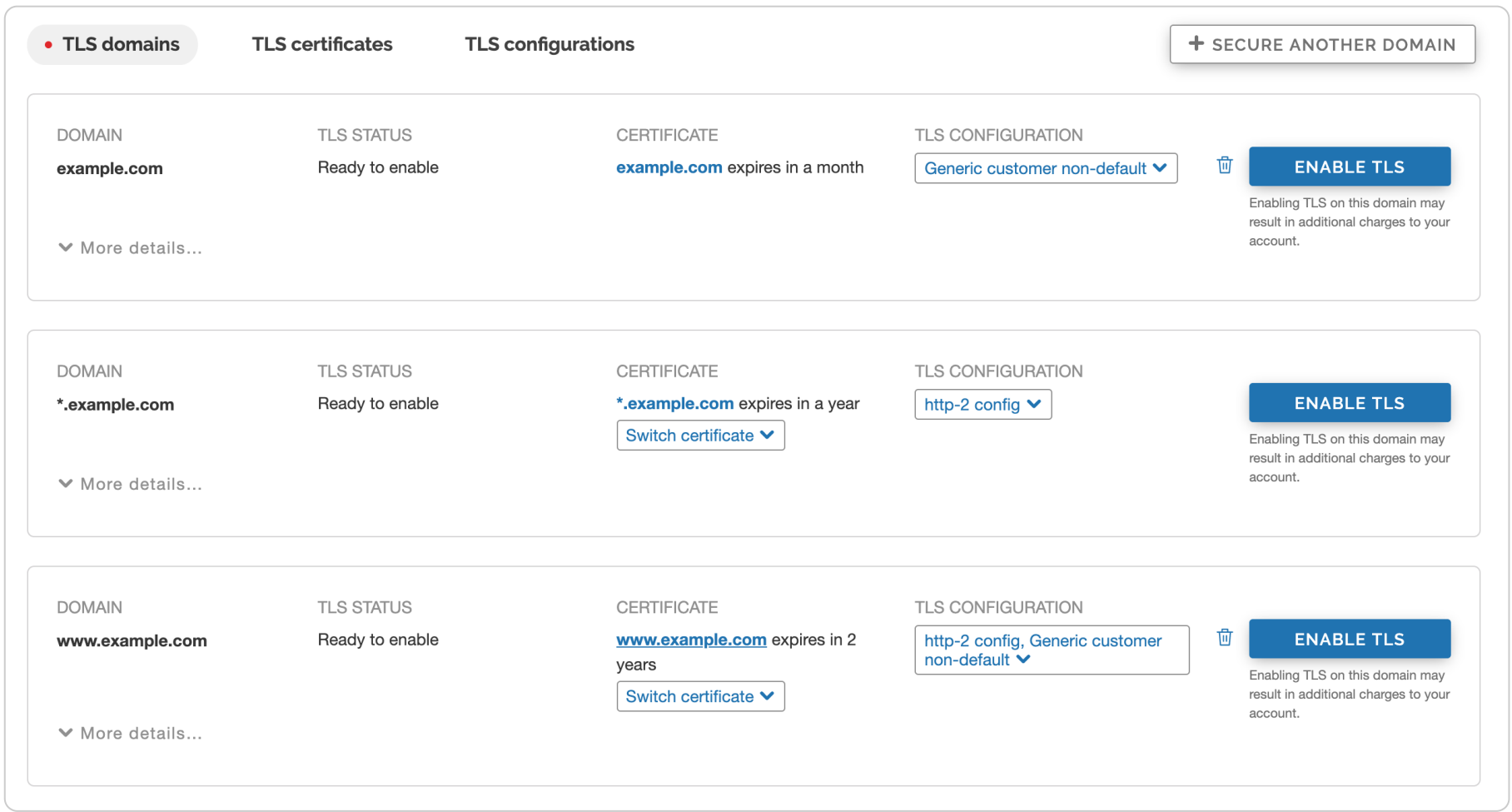
> **TIP:** Unmatched keys are private keys that have no matching TLS certificate. If you have multiple private keys, you can identify each by the unique SHA1 hash. Private keys can only be deleted if they are in the unmatched state.

6. 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.

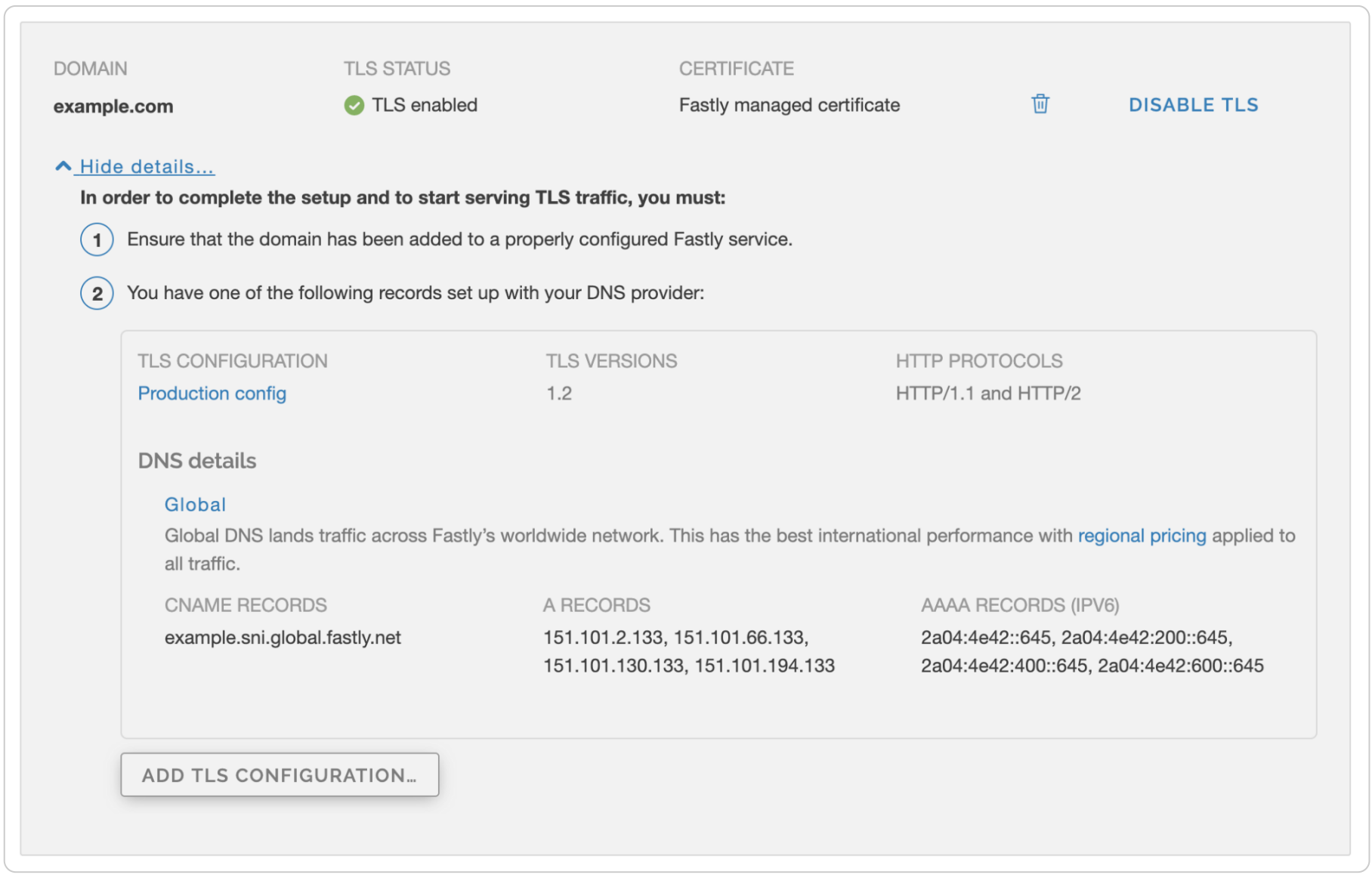
Enabling TLS on a domain

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 enable 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 TLS ready.



2. Click the **Enable TLS** button to the right of the appropriate domain. 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.
3. Click **More details** to view the TLS configuration associated with the domain. Details about your domain's TLS configuration appear.

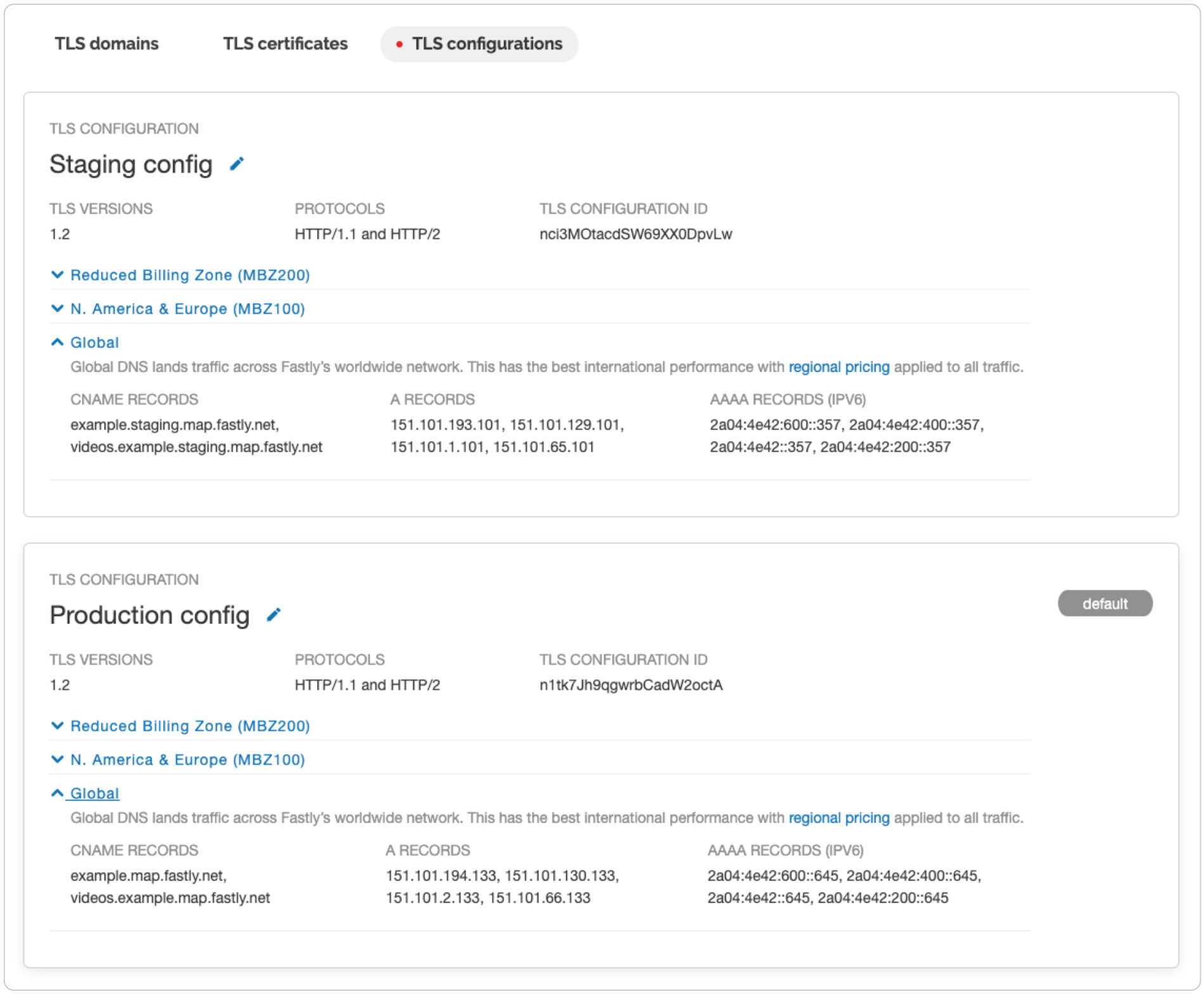


4. Use the DNS details displayed in this section to configure your DNS records so that a TLS connection can be established using your certificate.
   - For TLS on an apex domain (e.g., `example.com`), you'll need to create a series of A records with your DNS provider.
   - For subdomains and wildcard 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

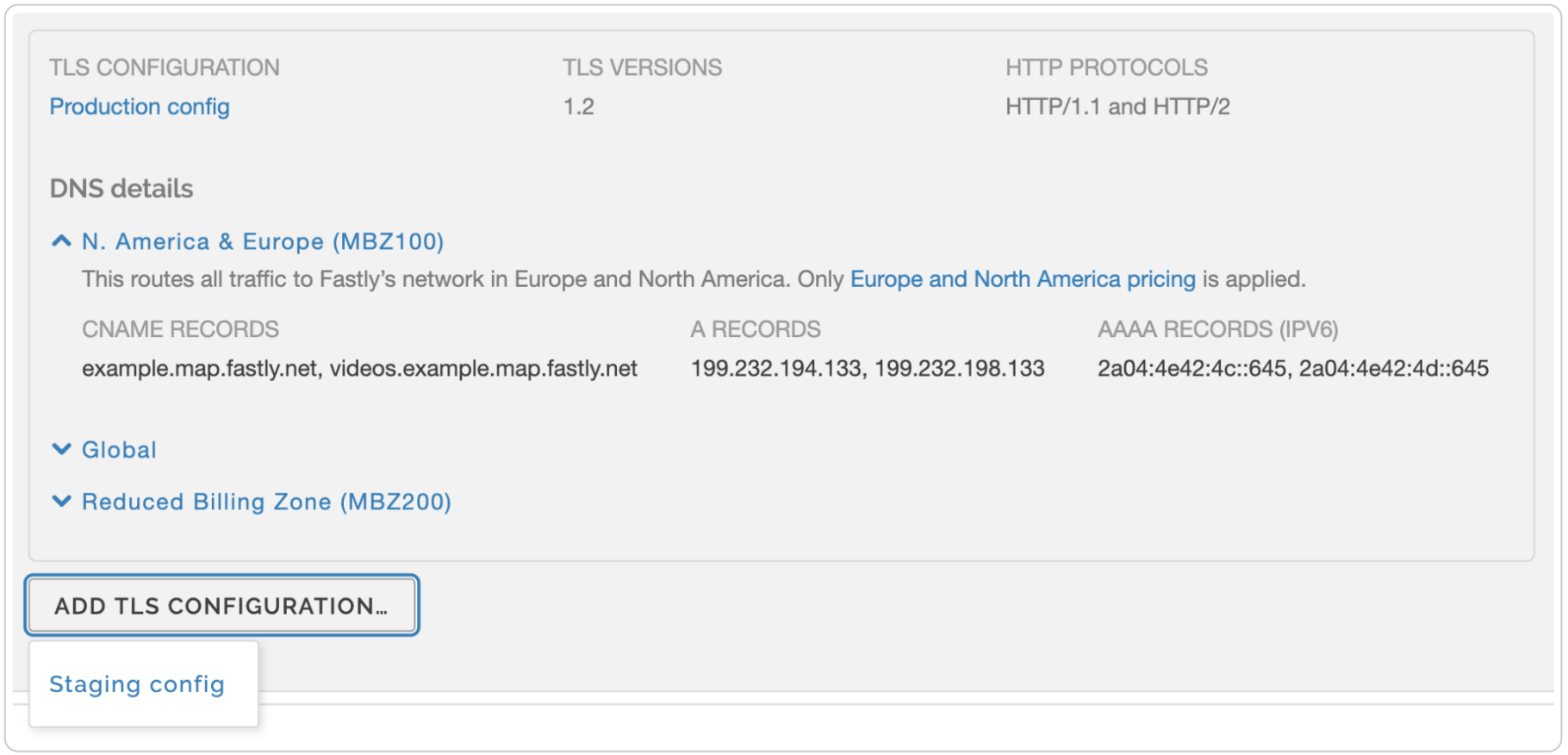
TLS configurations are a collection of TLS settings that include the supported versions of TLS and HTTP, along with networking and handshaking options that clients will use. For accounts with more than one TLS configuration, the default configuration has a label in the right corner of the card.



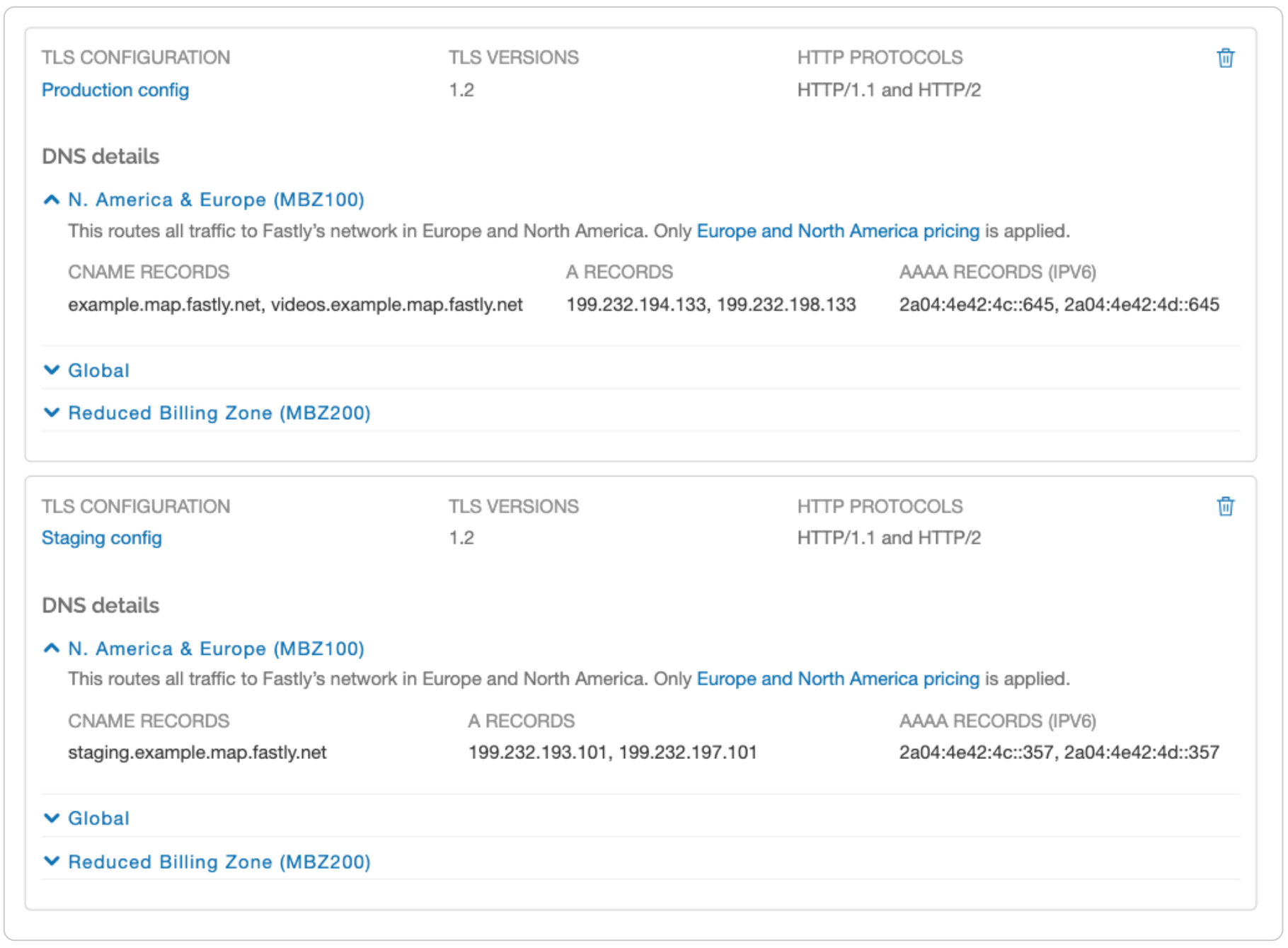
TIP: TLS configuration names are editable by clicking the pencil icon next to the name.

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. Click the **More details** link on the appropriate domain card.
3. From the **Add TLS configuration** menu at the bottom of the card, select the new configuration from the available options.



Once the configuration is selected, the TLS configurations listed are applied to the domain. Each TLS configuration is active and available at their respective IP addresses and DNS records.



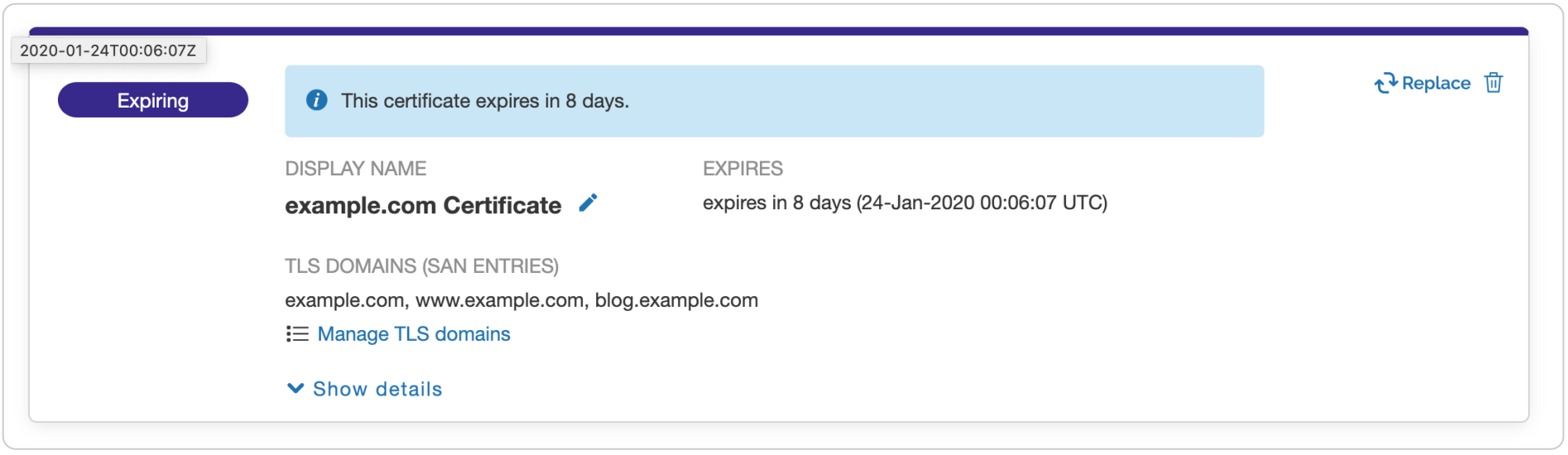
4. Use the **DNS details** under the desired TLS configuration to update your DNS records.
5. Confirm the new DNS records have propagated across the internet (this can take up to 48 hours), then delete the old TLS configuration by clicking the trash can icon.

NOTE: For HTTP/1.1, be sure to enable TLS for each of your domains in the web application. If you upload a certificate with multiple SANs, each domain must be explicitly enabled if you want to secure these domains on Fastly.

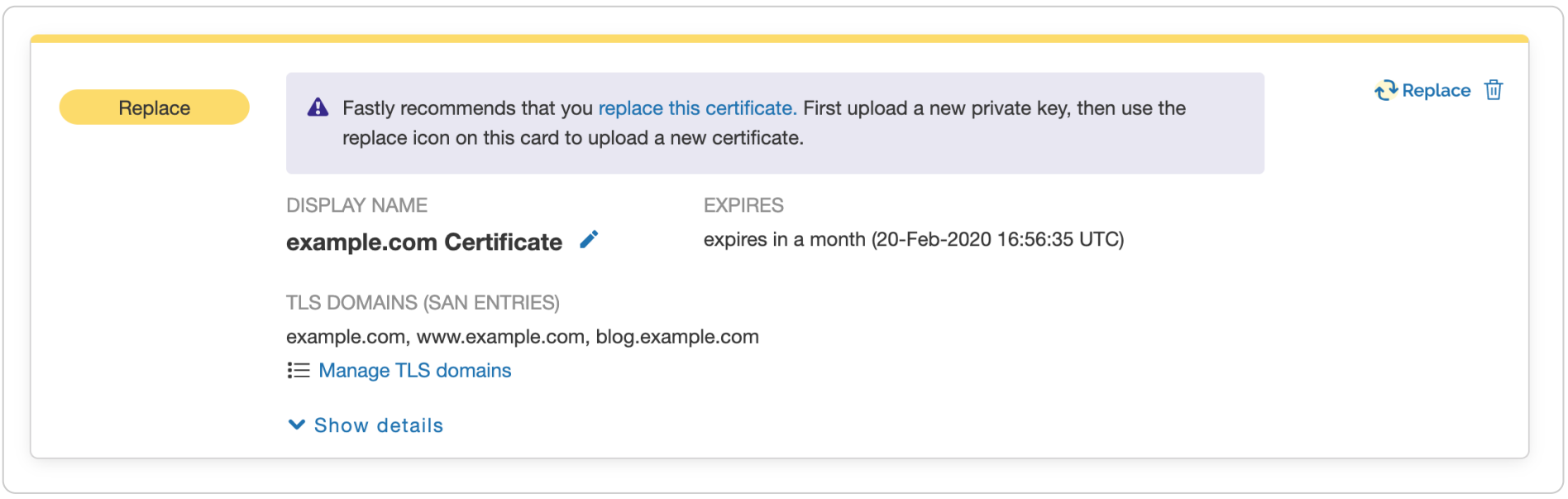
Exceptions may apply in the case of HTTP/2 if your browser coalesces secure connections and has previously received a TLS certificate from an earlier handshake. In this case, some browsers may reuse an existing secure connection to Fastly if its certificate has a matching SAN entry.

Replacing a certificate

The TLS certificates page warns you when a certificate is nearing its expiration date.



There may be situations where Fastly identifies certificates that should be replaced. These certificates will be clearly marked.



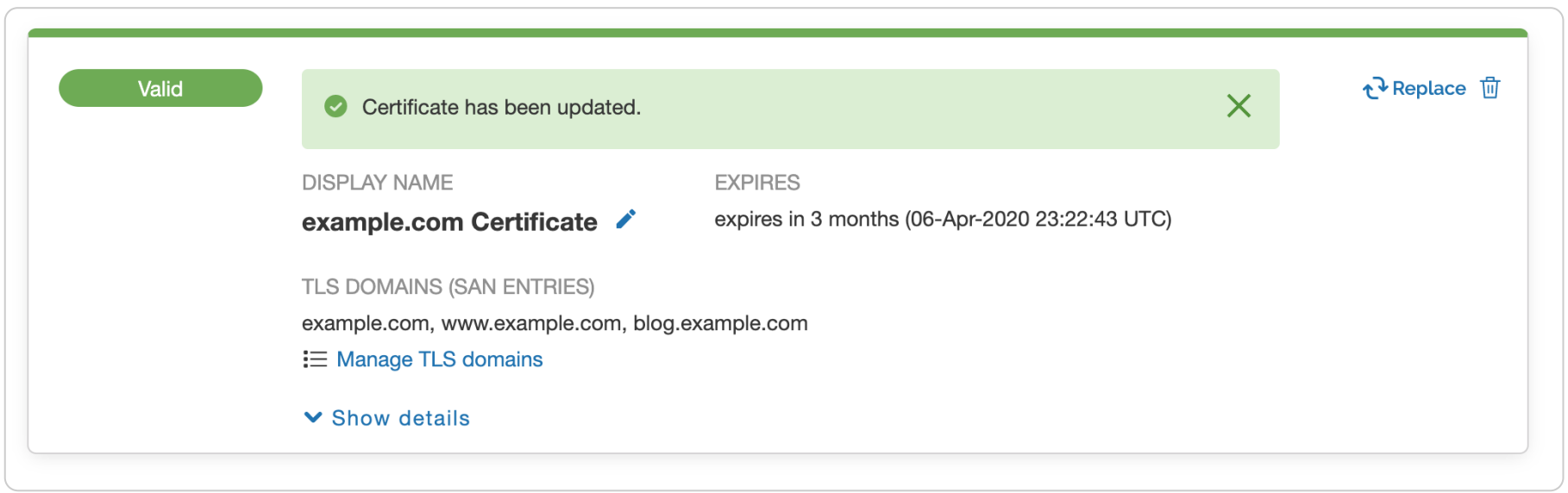
Fastly allows you replace a certificate with a new one at any time.

Replacing certificates when there are no removed domains

To replace a TLS certificate 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, upload it.
3. Find the certificate you're replacing on the **TLS certificates** tab in the web interface.

4. Click the word **Replace** 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.

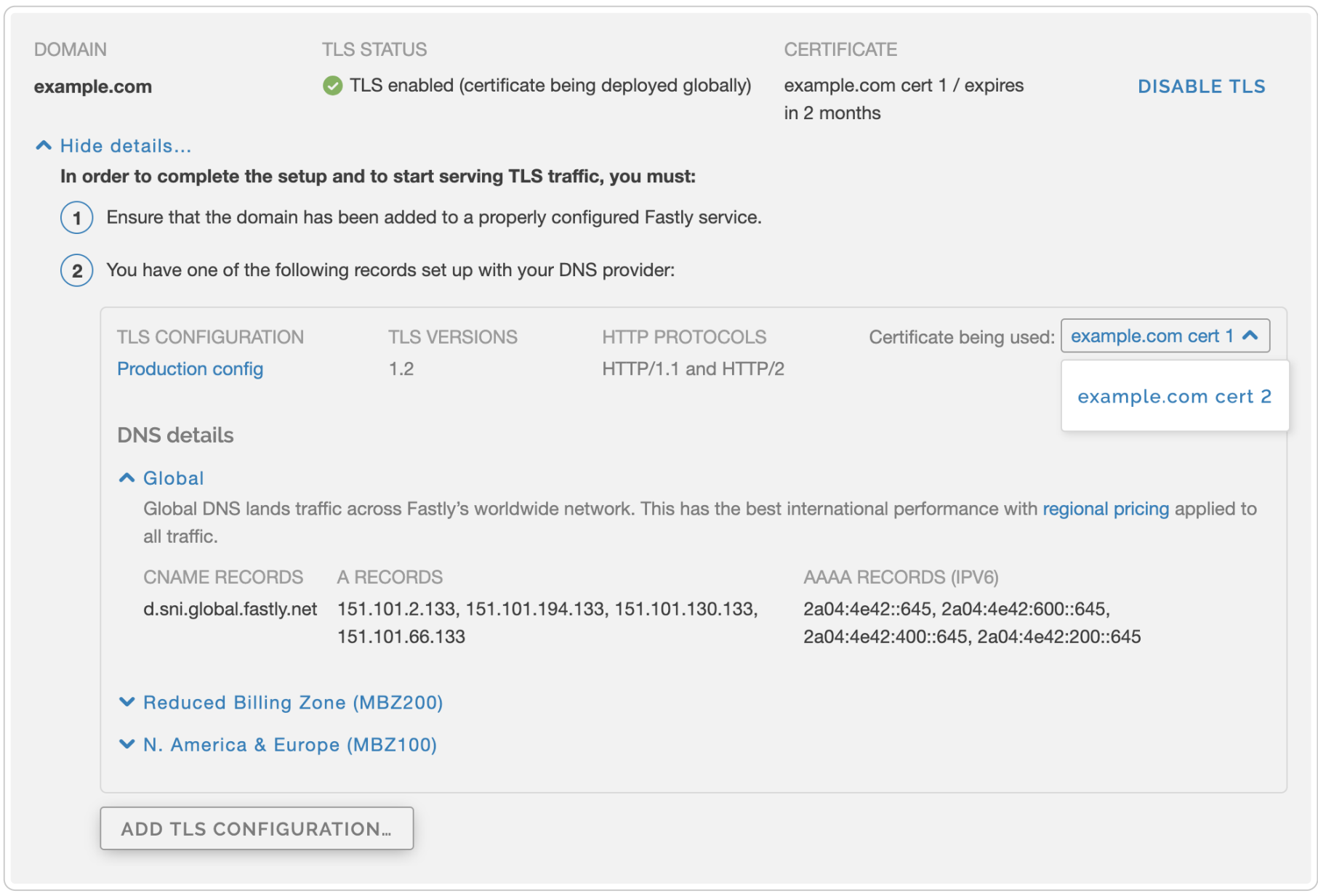


All domains actively serving TLS traffic on the old certificate will be automatically transitioned to the updated certificate within a matter of minutes. Any new domains will need to be manually enabled by following the steps for enabling TLS on a domain.

Replacing 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 enabled on the certificate you're replacing.
3. Click **More details** on the domain card to see the active TLS configurations.



4. From the **Certificate being used** menu, select a new certificate. A confirmation message appears.
5. Click the **Use this certificate** button on the confirmation message.
6. Manually activate any new domains that have been added to the updated certificate by following the steps for enabling TLS on a domain. Certificates for newly activated domains can take between 20 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.

Disabling TLS and deleting certificates and private keys

Once a domain has TLS enabled, you have the option to disable TLS via the Disable TLS link listed on the TLS domains page. Once disabled, Fastly will no longer serve TLS traffic on the selected domain.

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 users on the account with the TLS management permission. 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

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.

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 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.

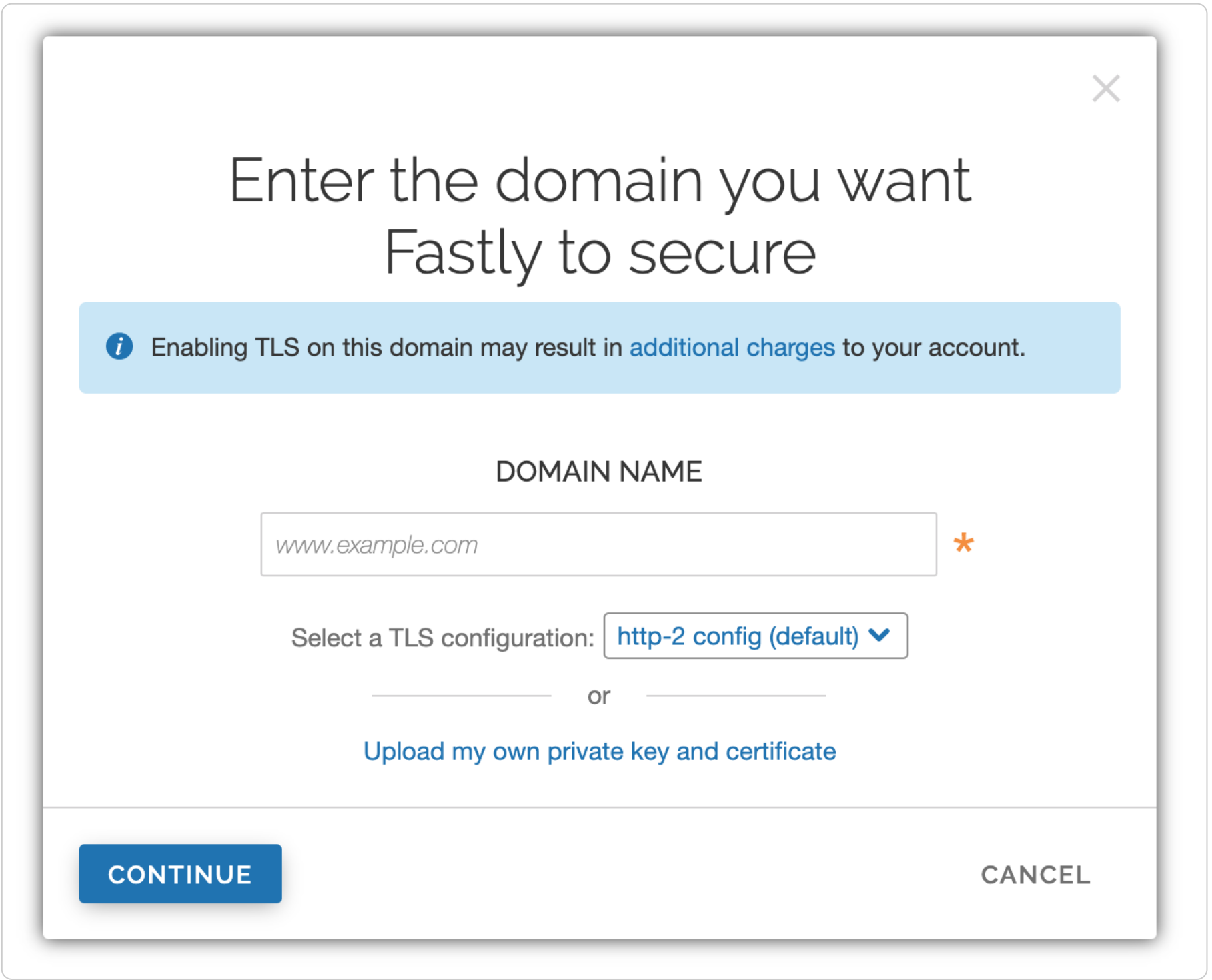
In addition to these prerequisites, be sure you understand the following limitations:

- Fastly TLS comes with a 50 certificate limit. To discuss how to raise this product's certificate limit, contact sales@fastly.com.
- Fastly generates one certificate per domain.
- 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

To set up TLS for an HTTPS domain, follow the steps below.

1. Log in to the Fastly web interface and click the **Configure** link.
2. Click the **HTTPS and network** tab. 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.
3. Click the **Secure another domain** button. The Enter domain window appears.



4. If you have your own TLS certificates and private keys, click the **I want to bring my own certificate and private key** link, and follow the guide to uploading and deploying your own certificates instead of this one.
5. In the **Domain name** field, enter an apex domain (e.g., `example.com`), a subdomain (e.g., `www.example.com` or `api.example.com`), or a wildcard domain (e.g., `*.example.com`).
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 TLS configuration menu, 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 the **Add domain** button. The TLS domains page appears with a series of cards displayed, each listing a single domain, including the domain you just added along with any other domains and their current TLS and certificate statuses.

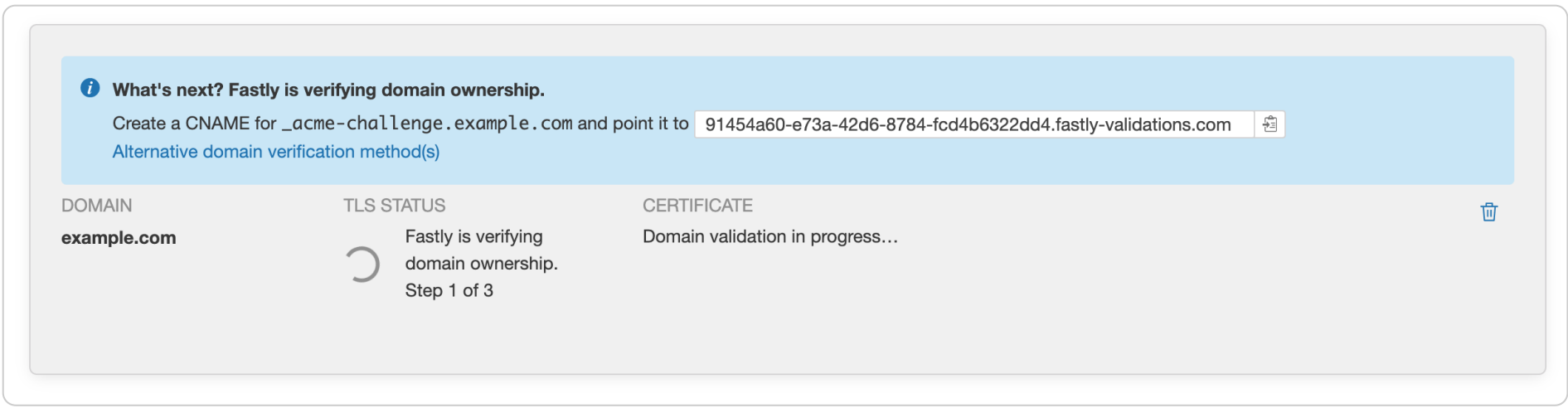
Verifying domain ownership

To begin serving HTTPS traffic, Fastly needs to verify that you control any domain you've added to the web interface. Fastly offers two options to verify apex domains and subdomains: the ACME DNS challenge type and the ACME HTTP challenge type. Each requires you to make specific DNS changes. Wildcard domains require the DNS 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.

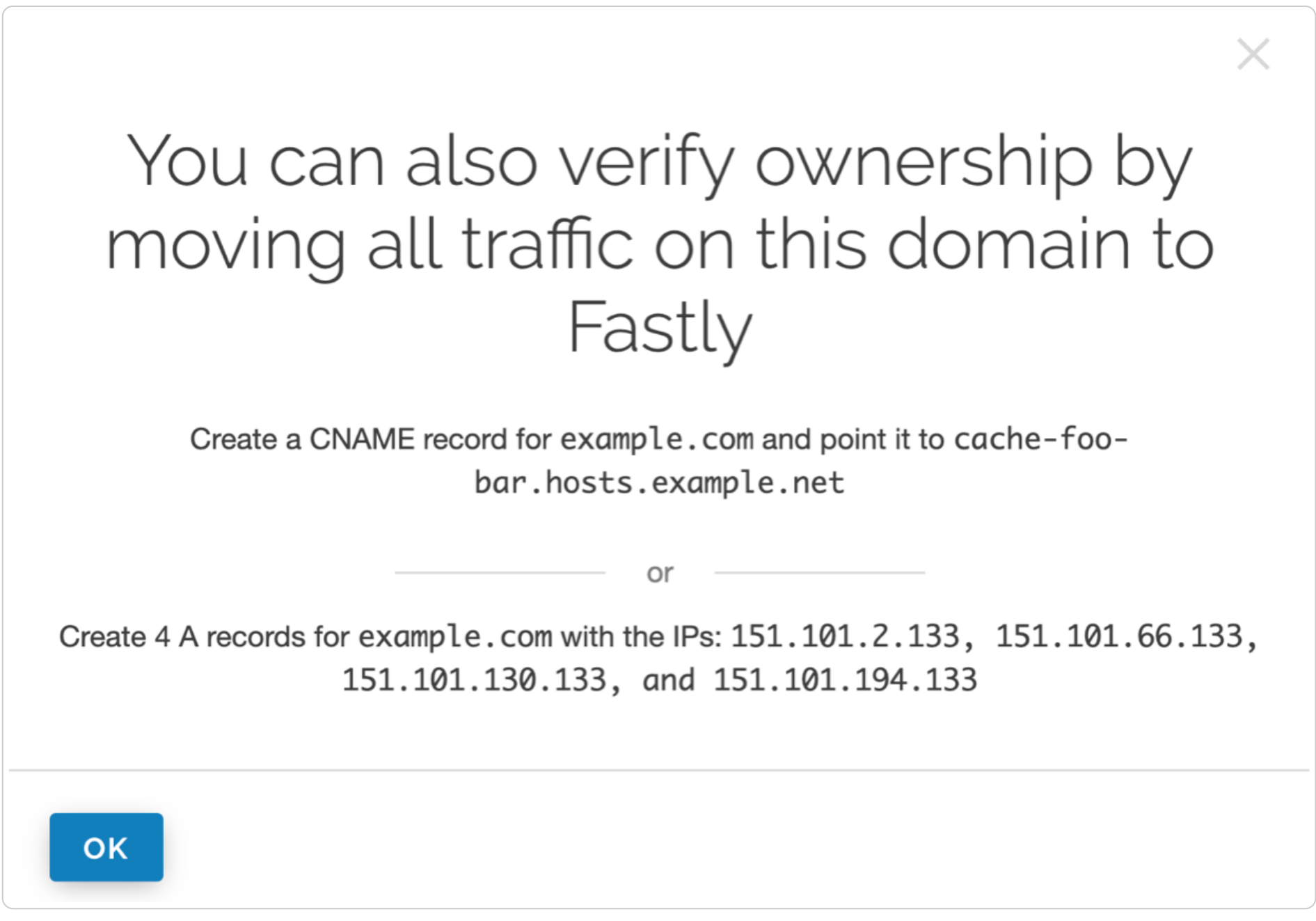
The steps to create the CNAME record will vary depending on your DNS provider's control panel interfaces. Refer to your DNS provider's documentation for exact instructions on how to do this. Your CNAME record must use the format `_acme-challenge.DOMAIN_NAME` (e.g., `_acme-challenge.www.example.com`) 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

An alternative 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 the DNS challenge). 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:



Choose the verification alternative that suits your needs:

- 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

In both cases, once set up, production traffic will immediately begin flowing through Fastly.

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 by updating your 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 believes the DNS records correctly point at Fastly and a certificate has been requested 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
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 CAA record on your domain that is blocking the certification authority from issuing certificates. This Certification Authority Authorization (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

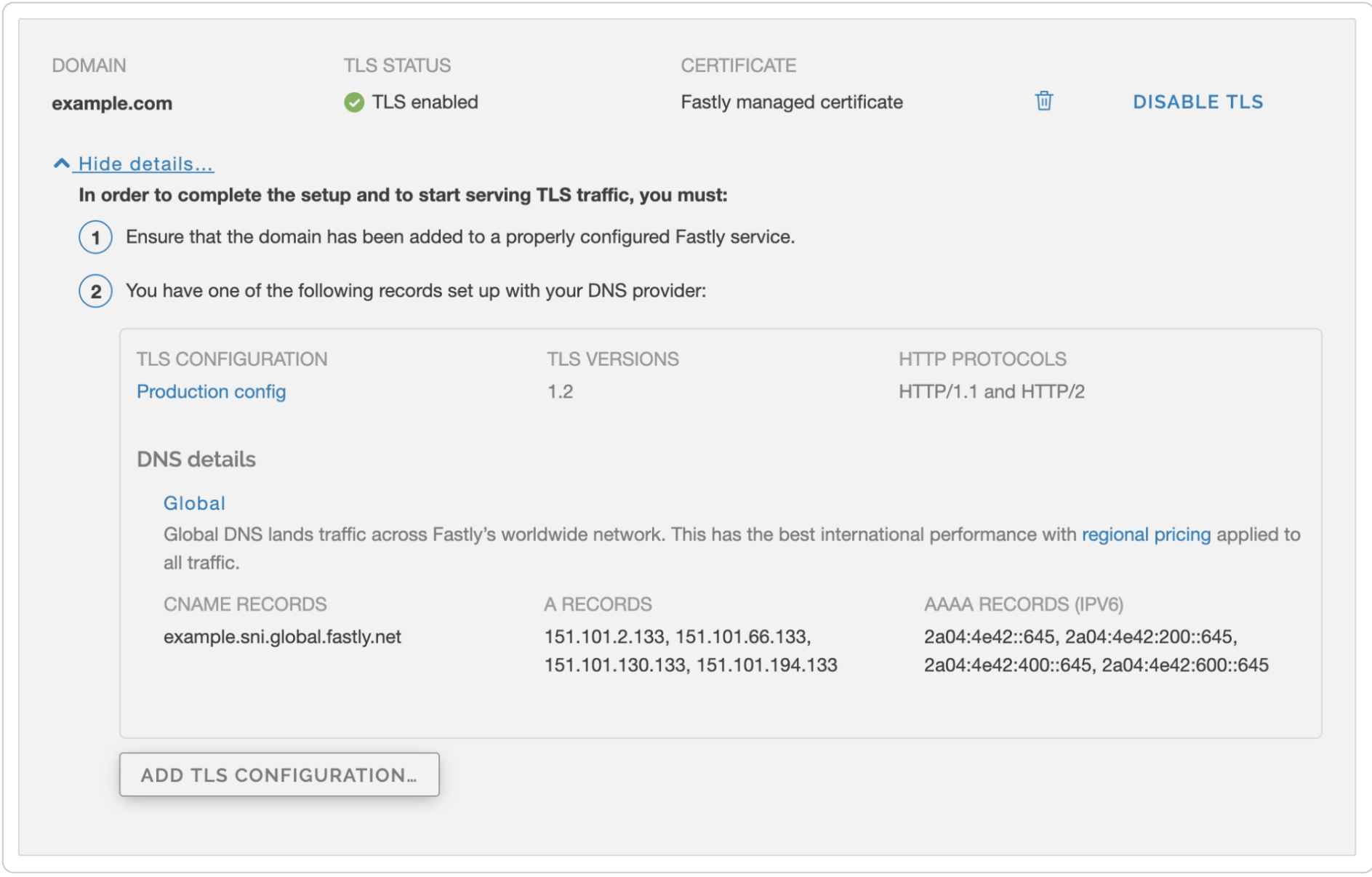
If the domain is stuck in the `Waiting for a response from CA` state, this is likely a temporary issue with the Certification Authority. Be sure to allow up to an hour in this state before contacting support@fastly.com for assistance. If this is a new certificate request, you can also try deleting the domain and starting again.

TLS enabled but certificate not deployed everywhere

If the domain displays the `TLS enabled` 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. All DNS details (CNAME, A records, and optionally AAAA records) can be found by clicking **More details** to view the TLS configuration associated with the domain. If you used the HTTP challenge method to verify domain ownership, you're already pointing traffic at the certificate.

If you have custom network maps or multiple, dedicated IPs, 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.

Disabling TLS and deleting a TLS domain

Once a domain has TLS enabled, you have the option to disable TLS via the Disable TLS link listed on the TLS domains page. Once disabled, Fastly will no longer serve TLS traffic on the selected domain. Fastly will attempt to renew a certificate for a disabled domain. To prevent this renewal process, you must delete the domain after you disable it. Fastly will not renew certificates for deleted domains.

Certificate management and 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.

Thirty days before it is due to expire, 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.

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.

Migrating domains from shared certificates to Fastly-managed certificates

To migrate a domain on Fastly's shared certificates over to the Fastly TLS product, start by following the Setting up TLS for a domain instructions. Be sure to use the ACME DNS challenge using the `_acme-challenge` subdomain for verification, to keep production traffic pointing at the shared certificate.

Once your domain enters the TLS enabled state, wait at least 10 minutes and then verify that the certificate is globally deployed before changing over your DNS records to point away from the active shared certificate. You can do this using the following command in a command line application:

```
openssl s_client -connect TLS.CONFIG.CNAME.RECORD:443 -servername your.domain.com  | openssl x509 -noout -text | grep 'Subject: CN'
```

If the result includes your TLS hostname, this means that Fastly has successfully generated and deployed a managed certificate to our network, and it is safe to update your DNS records with your DNS provider.

If you are currently using a shared certificate with a wildcard or subdomain, your CNAME record will likely end in `shared.global.fastly.net`. For apex domains, you will likely be using four A records. Provided the certificate is present on our network, you can now use the DNS records outlined in the Pointing DNS to serve HTTPS traffic instructions to point your domain to your new certificate, and away from the shared certificate.

After changing the DNS records for the domain with your DNS provider, check to see if the changes have propagated to a local DNS resolver by using the following command:

```
dig your.domain.com +short
```

TIP: Depending on your time to live (TTL) settings, you may need to allow up to 72 hours for DNS records to propagate across the internet.

As soon as your DNS information has been updated worldwide, visit your domain in a browser and check to see which certificate is served. Once you confirm that the new certificate is being served across clients, follow the instructions for deleting a TLS domain to remove the domain from the shared certificate.

Setting up free TLS

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`).

Before you begin

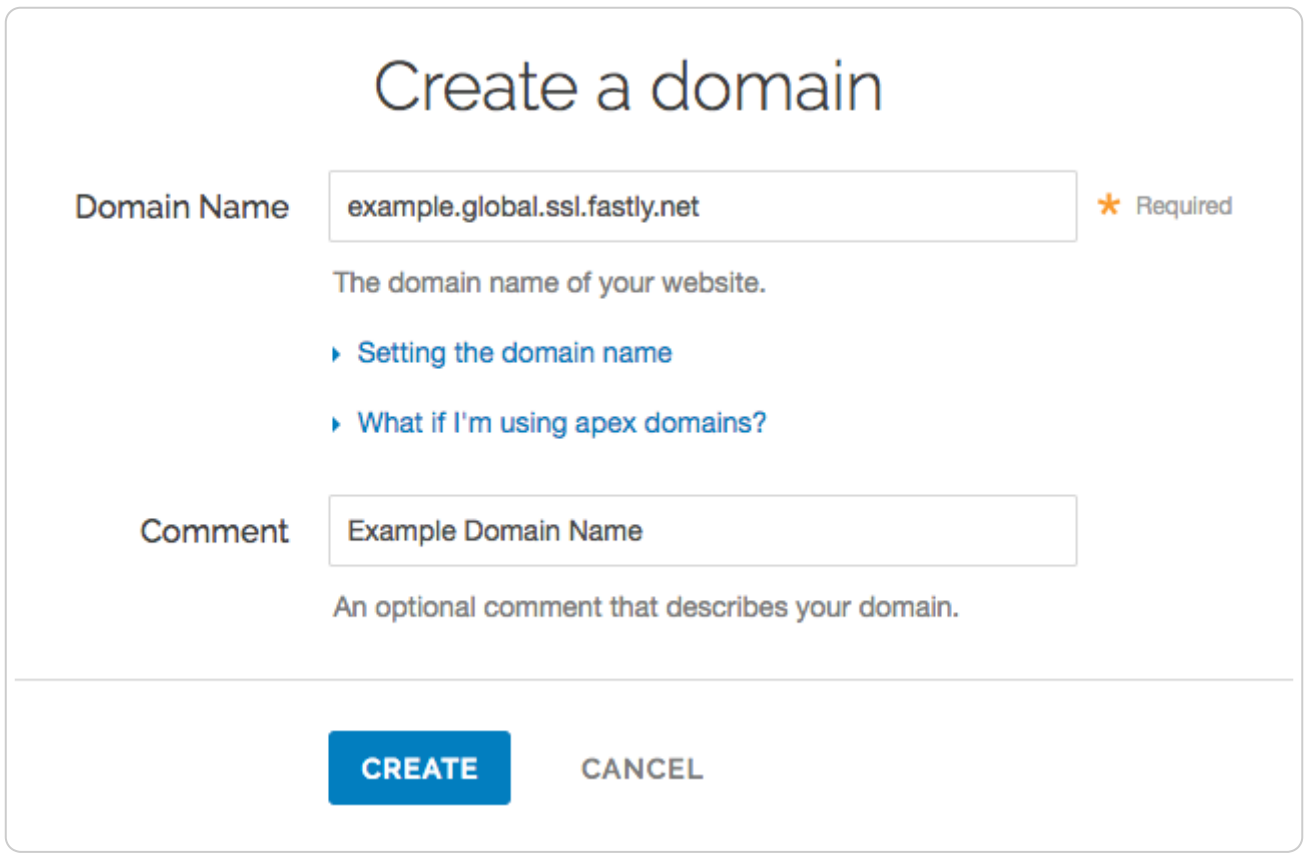
Before you begin setting up free TLS, understand the following:

- Free TLS uses a shared domain name and may not be suitable for a production environment if the domain name you use matters. For that, you'll need a paid TLS option.
- When using free TLS, you cannot DNS alias your own domain (for example, `www.example.org`) to the shared domain. If you do, a TLS name mismatch warning will appear in the browser. The only way to avoid the mismatch error is to order a paid TLS option.
- When using free TLS, all traffic is routed through Fastly's entire global network. If you need the ability to route traffic through specific POPs, order a paid TLS 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Create domain** button. The Create a domain page appears.



5. Fill out the **Create a domain** fields as follows:
   - In the **Domain Name** field, type `<name>.global.ssl.fastly.net`, where `<name>` is 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, type 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 key and certificate replacement

https://docs.fastly.com/en/guides/tls-key-and-certificate-replacement

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

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. Fastly offers a number of ways to deploy TLS certificates across our edge network.

This guide describes how to replace the keys and certificates used to terminate TLS for domains that have already been configured within the Fastly system. If you generate your own keys and certificates and transfer them to Fastly to install, contact support@fastly.com to see if you qualify for this interface.

Prerequisites

To upload new private keys and replace TLS certificates using the web interface, you will need:

- a Fastly user account assigned the role of superuser, or assigned a user role with added TLS management permission

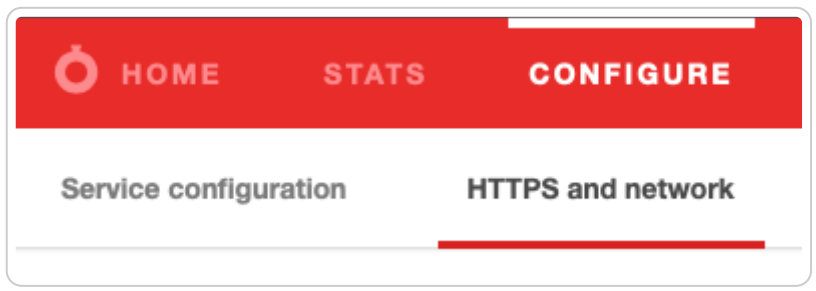
- a valid X.509 TLS certificate from a trusted certification authority (CA)
- a matching 2048-bit RSA private key

Known Issues

- The web interface does not accept 4096-bit keys. If you have such a key you must use Fastly's certificate uploading tool to upload your key.
- When replacing a certificate, the SAN entries of the new certificate must either be an exact match or contain a superset of entries when compared with the existing certificate. All existing domains will be automatically enabled on the new certificate.

Accessing the HTTPS and network settings page

To access the HTTPS and network settings page, log in to your Fastly account, click the **Configure** link, and then click the **HTTPS and network** tab.

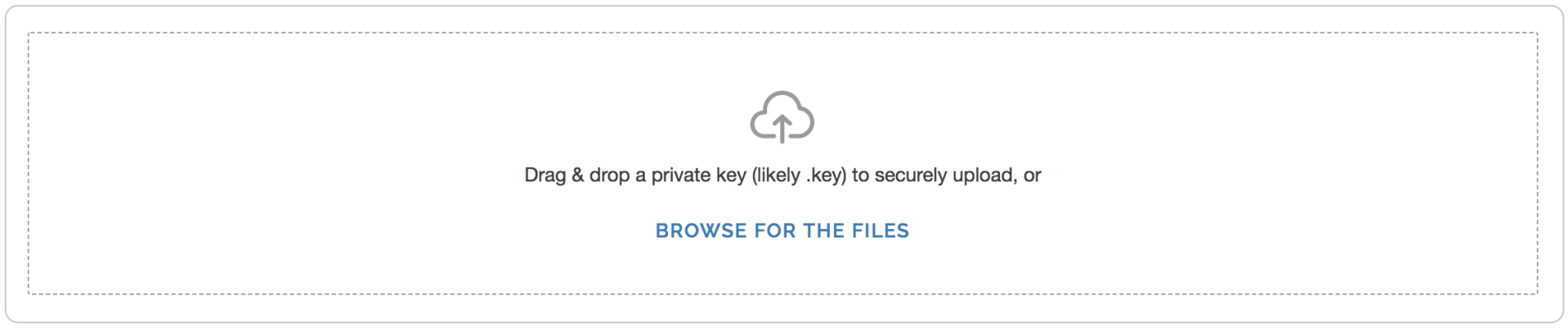


This brings you to the **TLS certificates** page, which lets you view your certificates and private keys, and allows you to upload new keys and replace your existing certificates.

Replacing a key and certificate

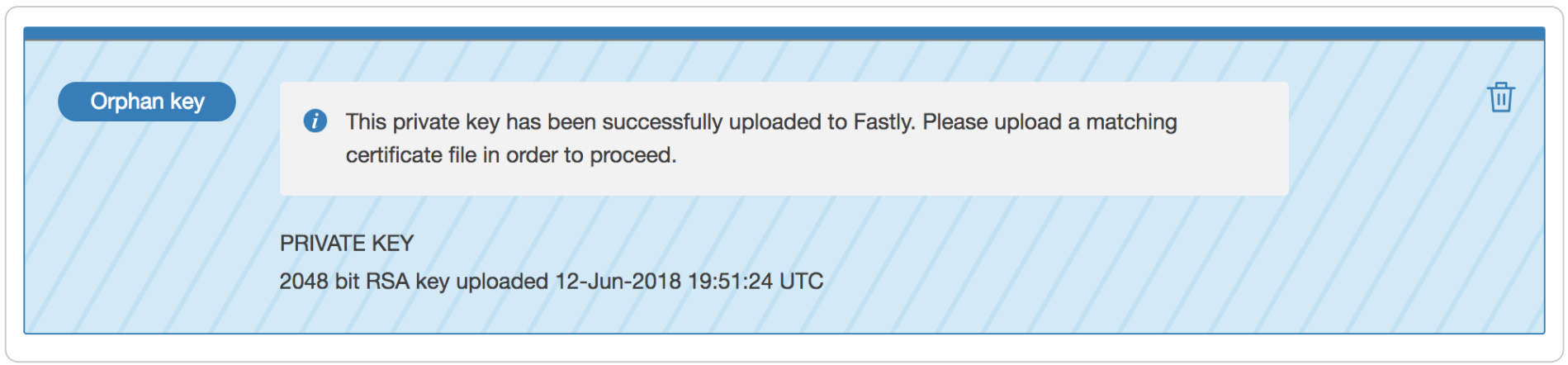
To upload the new key and replace the certificate used to terminate TLS for a domain, first you must generate a new key and certificate with your preferred certification authority. When regenerating a new certificate, you must specify the exact same list of SAN entries as the existing certificate. The TLS certificates page will provide you with information on all of your current certificates and the SAN entries of each of those certificates. If you need to modify the SAN entries for a particular certificate, or if you need to add a brand new certificate for a new set of domains, contact support@fastly.com for assistance.

In order to replace a TLS certificate you will first need to upload the matching private key that was used to generate the new certificate.



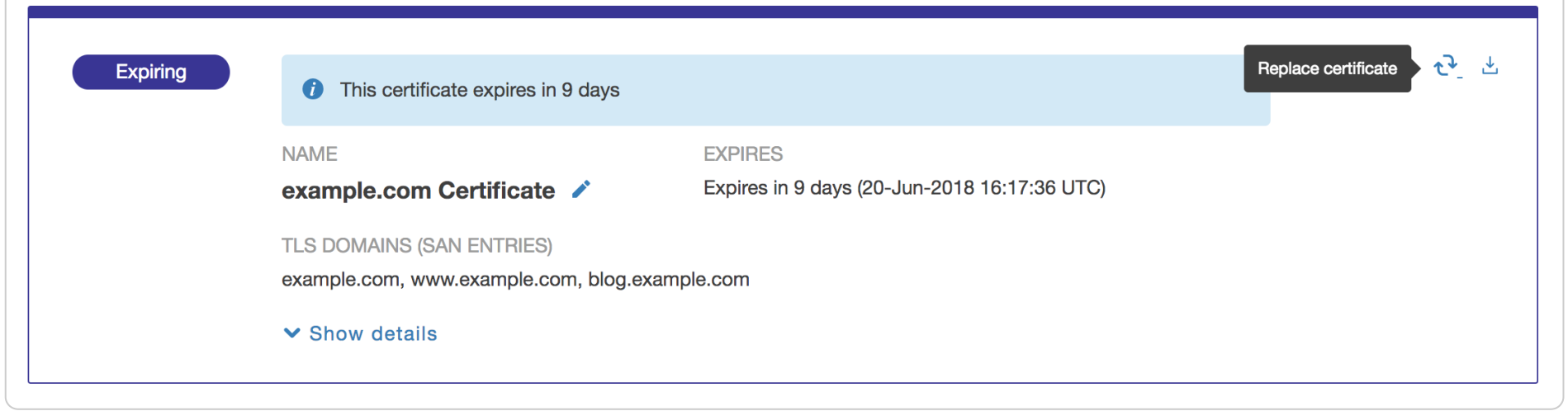
On the **TLS certificates** page there is a drag-and-drop area that you can use to drop your private key file. Alternatively you can browse your file system for the private key. This upload tool currently only accepts 2048-bit RSA keys. If you require longer key lengths, contact support@fastly.com. Valid private keys will automatically upload to Fastly upon being dropped on the page, or after being selected from the file picker.

Upon successfully uploading a private key, the **TLS certificates** page will display the key with the label, **Orphan key**. This refers to a private key that has no matching TLS certificate. If you have multiple private keys, you will be able to identify each by a unique upload date time. Private keys can only be deleted if they are in the orphan state.



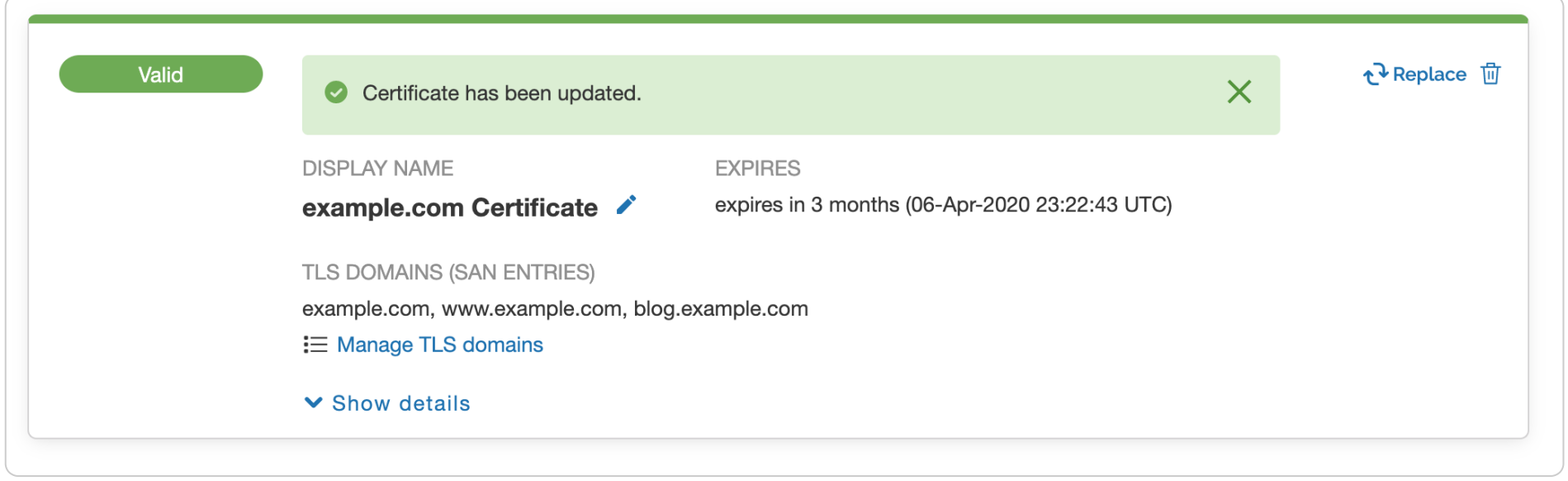
Once you have uploaded the new private key, you will be able to replace the TLS certificate. Find the certificate in the list of certificates. In the example below we show a certificate that is nearing expiration. You will see the Replace icon at the top-right corner. Clicking this icon brings up a file-picker that can be used to select the new certificate. The certificate you select should be

PEM-formatted and must contain all the same SAN entries as the current TLS certificate, either as an exact matching list, or as a superset. You can select either a file containing the full certificate chain or a file containing just the leaf certificate. The intermediate certificates will be automatically backfilled when just the leaf certificate is uploaded.

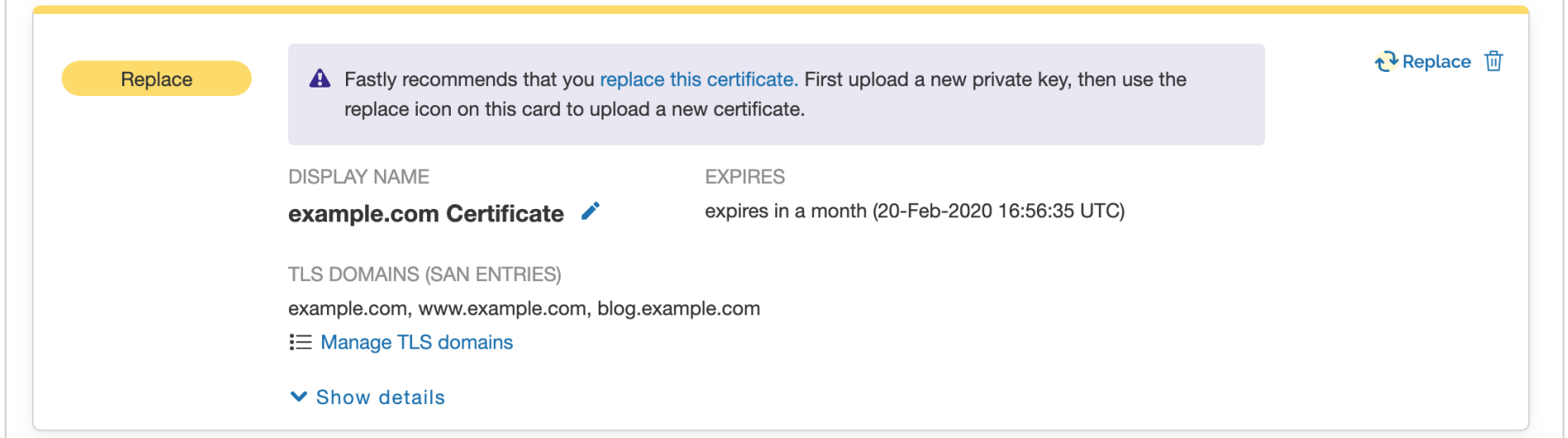


After selecting the new certificate, a success message will be displayed and the certificate information will be updated. When a certificate is replaced, it will be automatically deployed and all domains actively serving TLS traffic on the old certificate will be automatically transitioned to the updated certificate within a matter of minutes.

NOTE: If the new certificate is not being used to serve TLS traffic within 1 hour, contact support@fastly.com for assistance.



There may be situations where Fastly identifies certificates that should be replaced. These certificates will be clearly marked.



TLS termination

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:

```
1    if (req.http.Fastly-SSL) {
2      set resp.http.X-Is-SSL = "yes";
3    }
```

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:

```
1    if (!empty( $_SERVER['HTTP_FASTLY_SSL'])) {
2      $_SERVER['HTTPS'] = 'on';
3    }
```

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.

Web Application Firewall

These articles provide information about the Fastly Web Application Firewall (WAF) security product.

https://docs.fastly.com/en/guides/security#_web-application-firewall

About the Fastly WAF dashboard

https://docs.fastly.com/en/guides/about-the-fastly-waf-dashboard

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 following pages:

- WAF summary
- WAF audit log
- All WAF services

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

Accessing the Fastly WAF dashboard

To access the Fastly WAF dashboard, follow the steps below:

1. Log in to the Fastly web interface. The All services page appears.

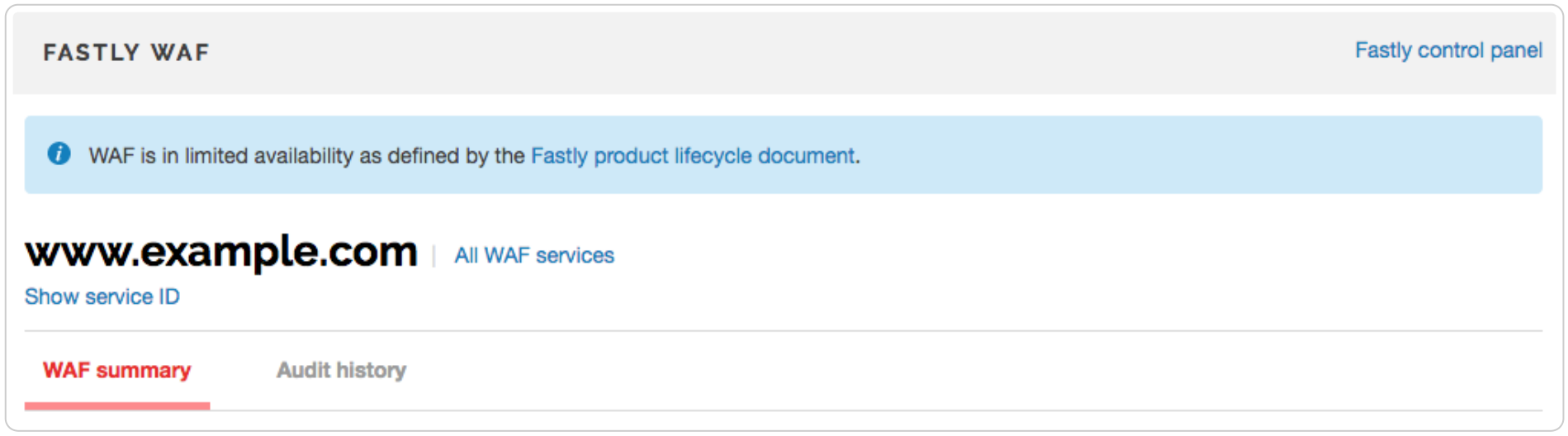
2. Find your Fastly service in the list, and then click the **WAF** link. The WAF summary page appears.

If you have hundreds of services, you might want to jump to the All WAF services page for an overview of all your WAFs.

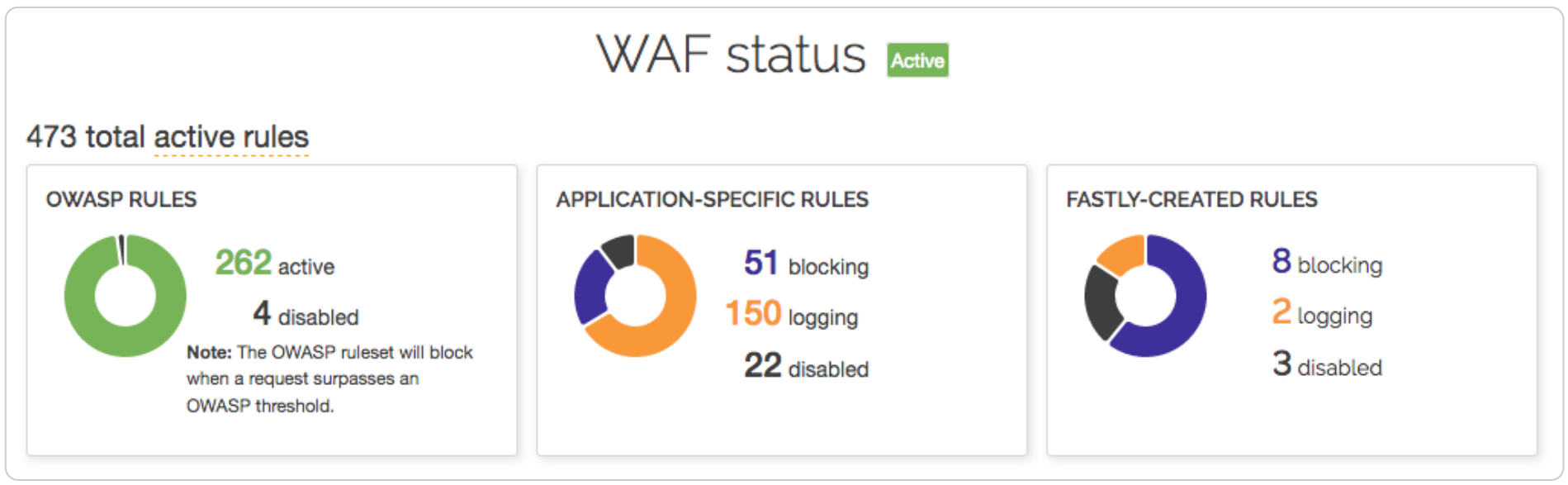
NOTE: To access the Fastly WAF dashboard, you must sign up for a Fastly account and purchase the Fastly WAF. Contact our sales team to get started.

About the WAF summary page

The WAF summary page displays the status of your WAF. The top of the page provides links to the All WAF services page, WAF audit log page, and Fastly control panel.



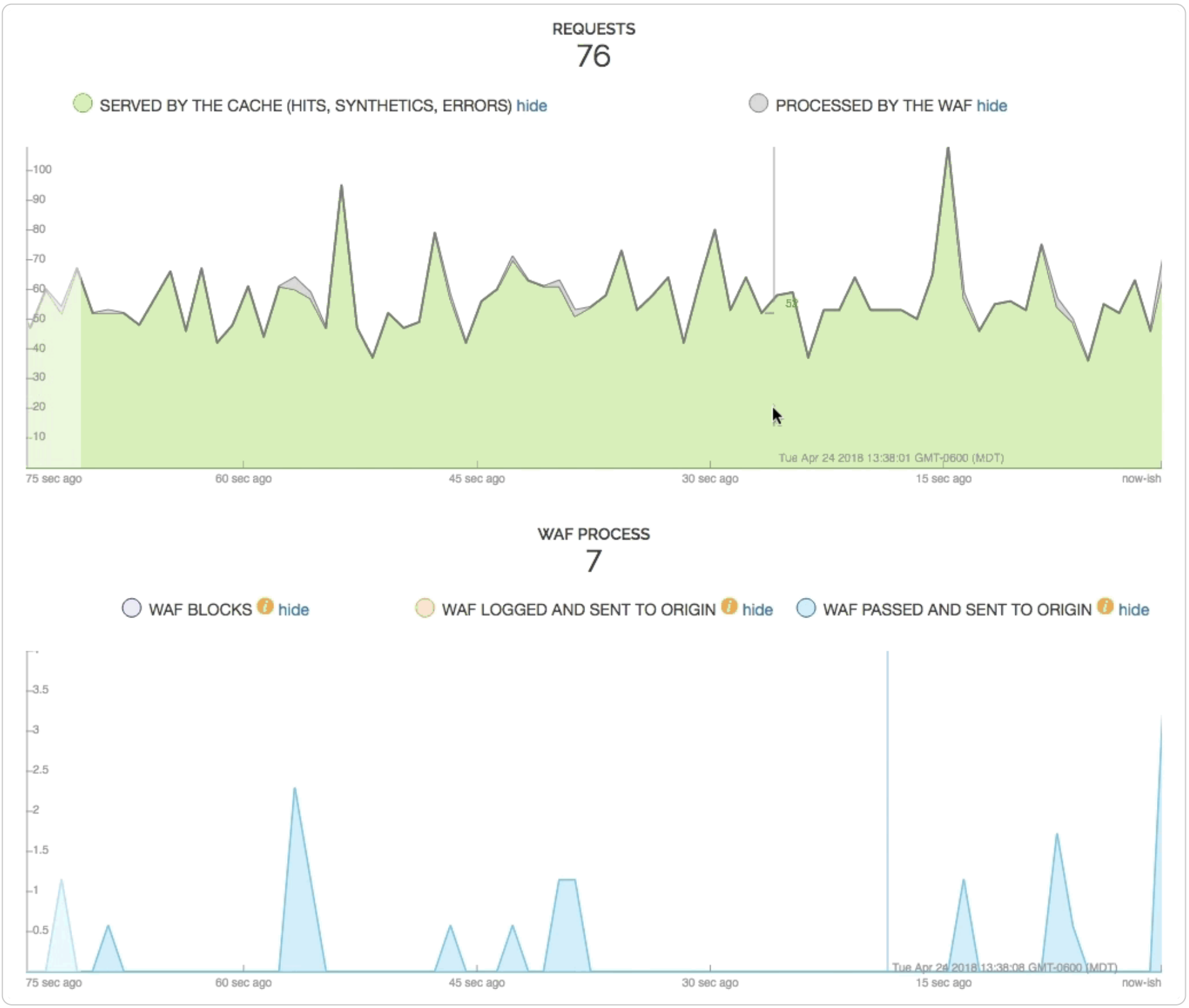
The WAF status section indicates whether the WAF is currently active. To be considered active, the WAF must not be disabled and must have at least one rule status set in either logging or blocking mode. You can see the total number of active rules. This number includes OWASP rules set to "active" and strict match rules set to blocking or logging. The charts show 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.

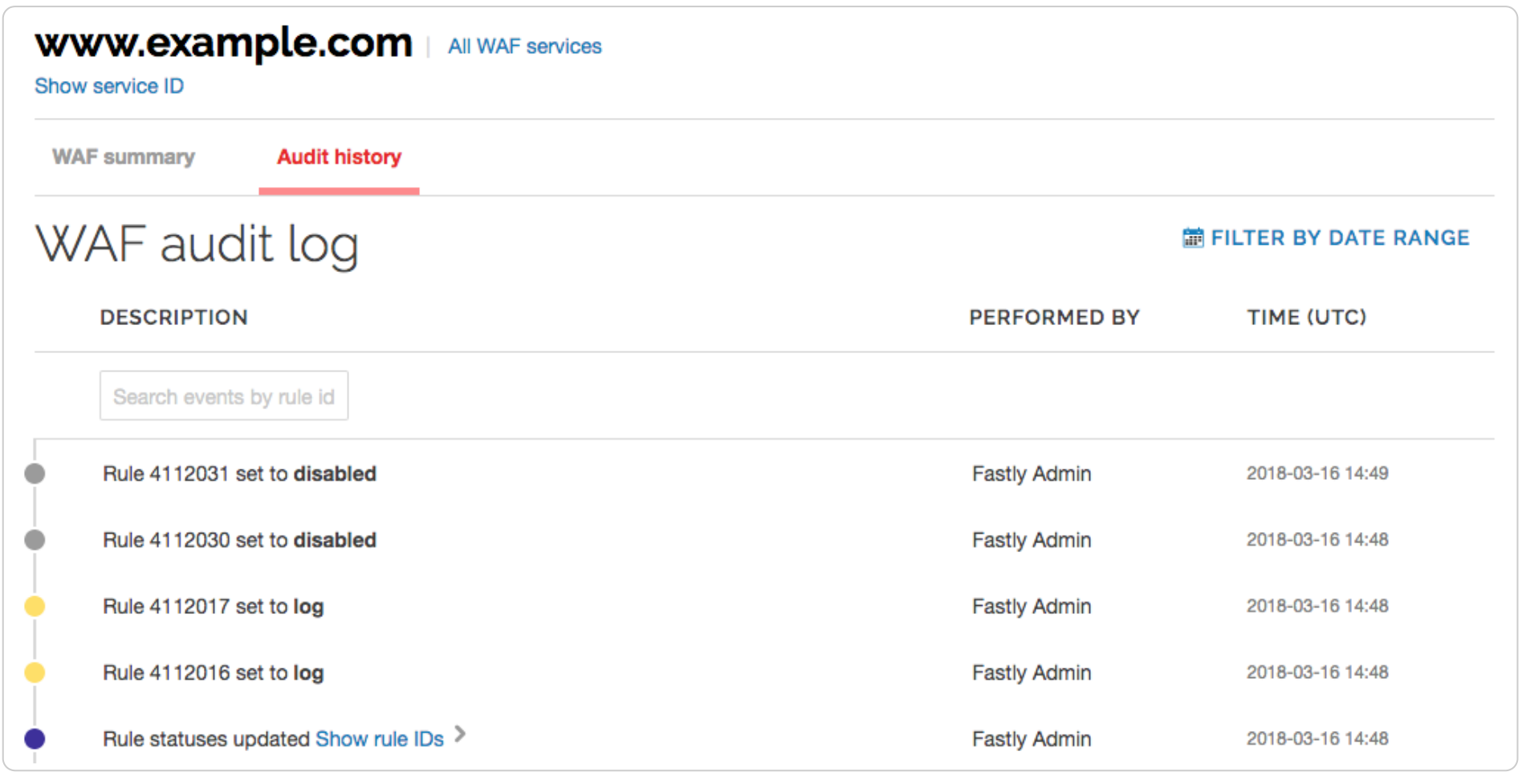
You can exclude certain data from the graphs by clicking the **hide** link next to a data label. Clicking this link will hide that value in the graph's display.

TIP: The Fastly WAF only executes on traffic sent to the origin server.



About the WAF audit log page

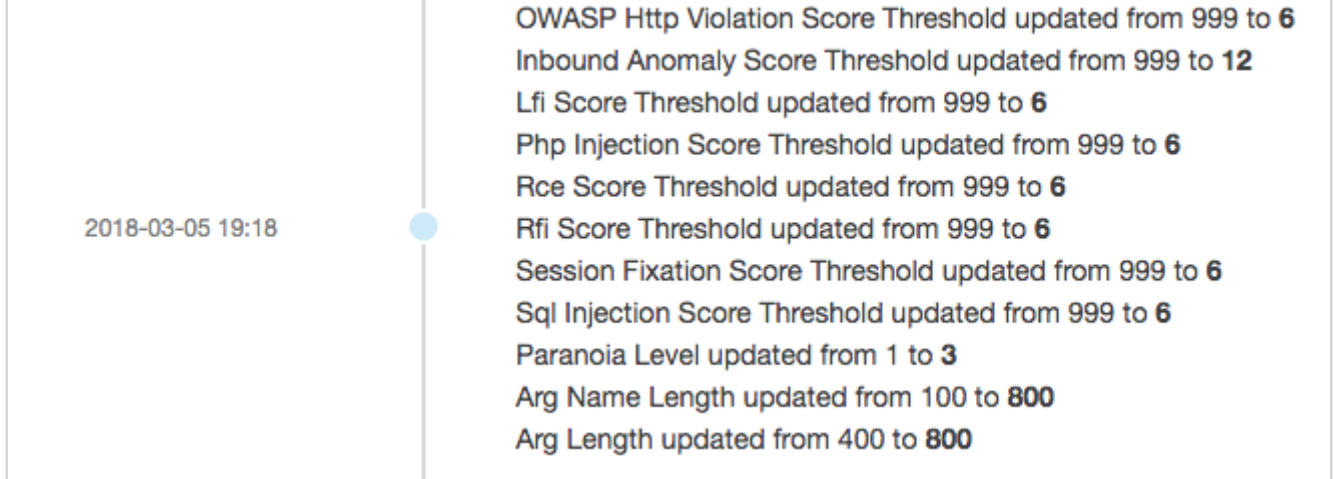
The WAF audit log page displays all configuration changes made to your WAF. You can use this page to determine who made certain types of configuration changes to the WAF, and when the changes were made. The line items indicate when rules were set to log or block, when they were updated, and whether they 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

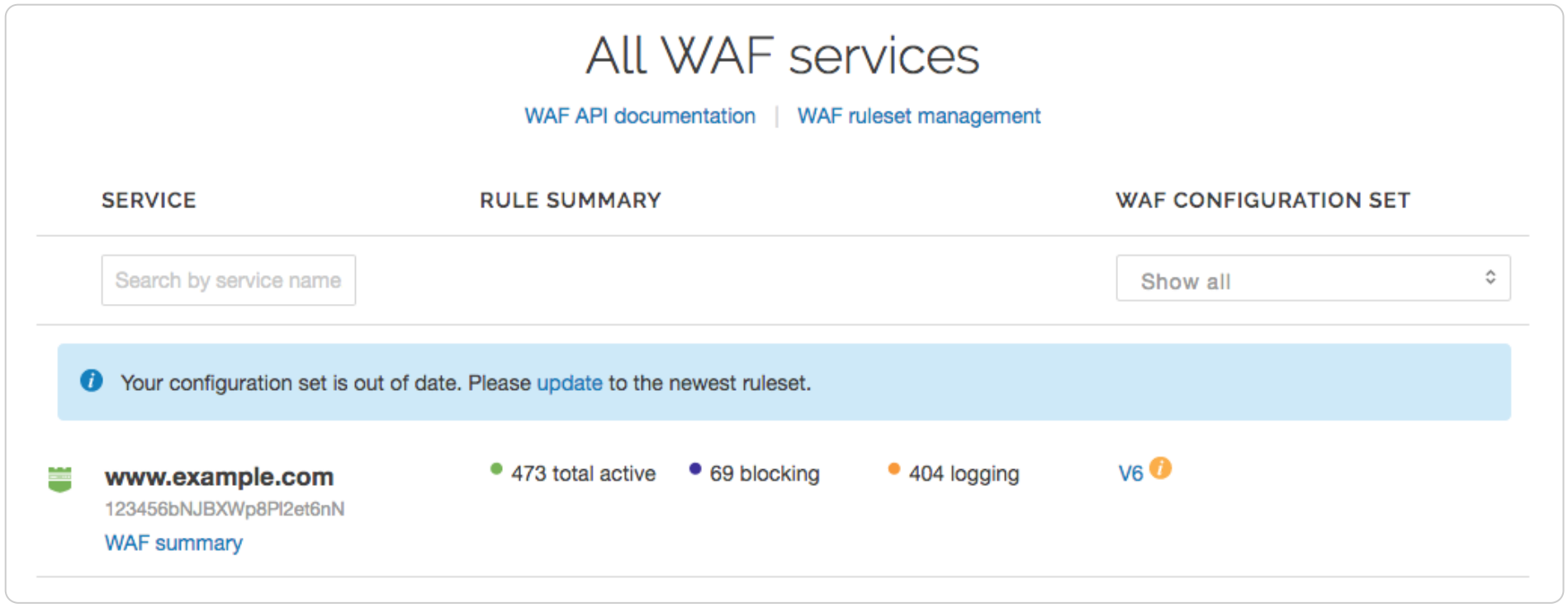
You may see OWASP properties referenced on the WAF audit log page. The table below contains a list of all available properties and their descriptions. The properties shown here reflect changes made by altering the settings in the OWASP object.

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.

OWASP property	Description
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.

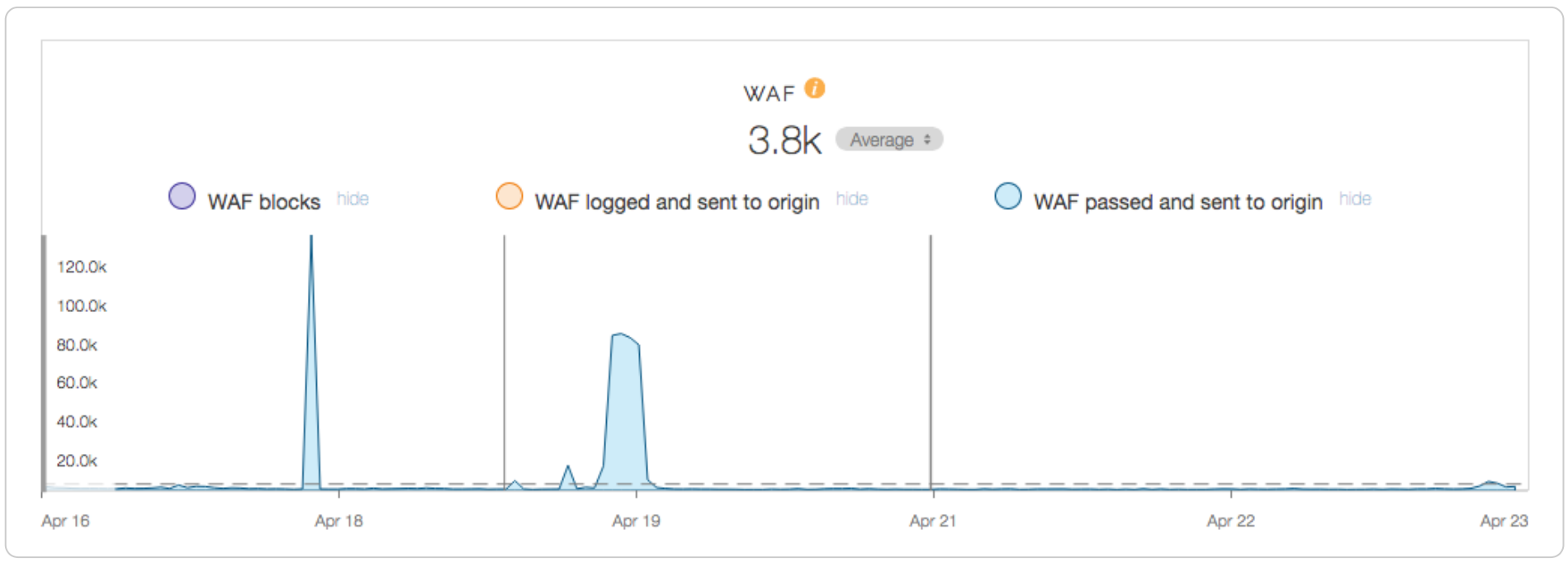
About the All WAF services page

You can use the All WAF services page to monitor all of the WAFs deployed within your services. This page shows which of your services have WAFs, which WAFs are enabled, how many rules are enabled and disabled per WAF, and which configuration sets the WAFs are using. If a configuration set is out of date, a message is displayed alerting you it's time to update to the latest rule set.



About the WAF stats

The WAF stats graph appears on the Stats page. For the selected service, this graph shows blocked traffic that was stopped by the WAF based on rules, logged traffic that triggered rules but was sent to the origin, and passed traffic that didn't trigger rules and was sent to the origin.



Creating a custom WAF error page

https://docs.fastly.com/en/guides/creating-custom-waf-error-page

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

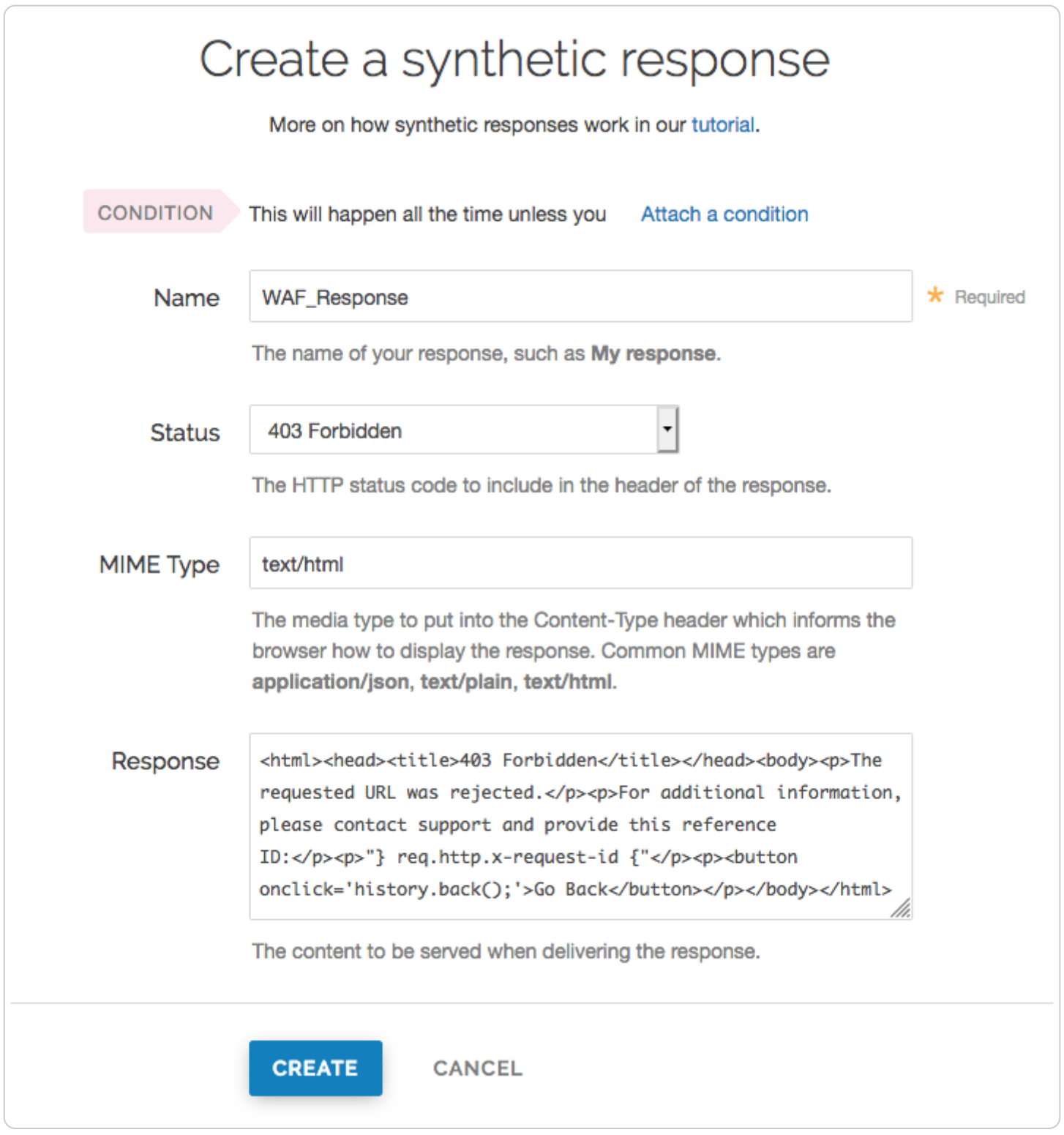
To create a dynamic VCL Snippet for the transaction ID, make the following API call in a terminal application:

```
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=rec
v&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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type `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 if desired, but care should be taken to minimize the probability that multiple requests within a desired 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. Alternatively, 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

https://docs.fastly.com/en/guides/fastly-waf-logging

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.

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

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.

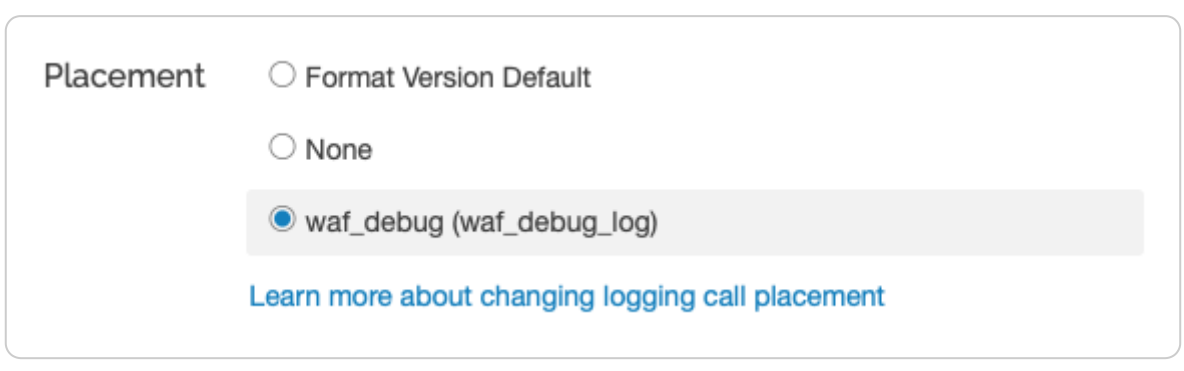
waf_debug_log

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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/<yo
ur Fastly service ID>/version/<version_id>/logging/<logging_integration>/<logging_name>' --data-binary '{"placemen
t":"waf_debug"}'
```

- `waf_debug_log` accepts the logging format via the UI 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 serv
er.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

https://docs.fastly.com/en/guides/fastly-waf-rule-set-updates-maintenance

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.

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

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

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)	
3vnl3cwPda9Q3WYCDRuGW	v10 2018-09-05	• 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	• 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
6LG4xleIDKWLblCJczGpi9	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:

```
1  curl -s -H Fastly-Key:<your Fastly API token> -H Accept:application/vnd.api+json \
2       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 WA
F 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/<you
r 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

https://docs.fastly.com/en/guides/managing-fastly-waf

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.

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

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_stat
us", "attributes":{ "status": "block"}}}' -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/
<your Fastly 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", "attribute
s": {"name": <tag>, "status": "block"}}}' -H 'Content-Type: application/vnd.api+json' https://api.fastly.com/service/
<your Fastly 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: FA
STLY_API_TOKEN" https://api.fastly.com/service/<service_id>/wafs/<waf_id>/owasp -d '{"data": {"attributes":{"inbound_
anomaly_score_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
-Type: 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.

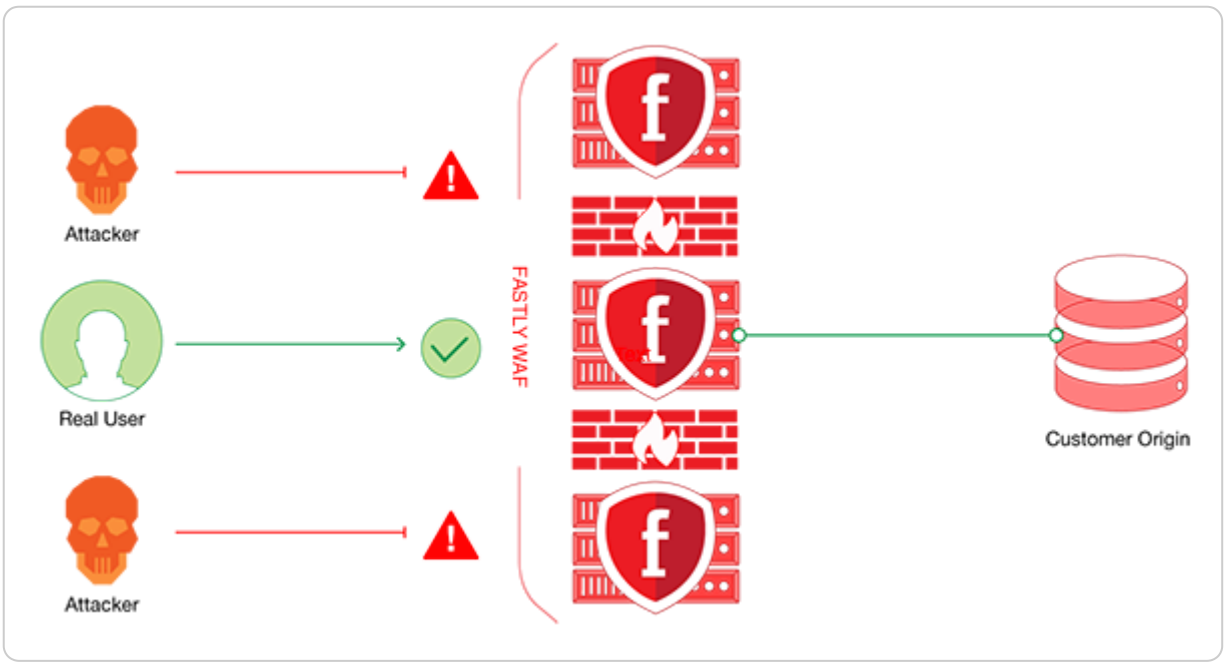
Web Application Firewall (WAF)

https://docs.fastly.com/en/guides/web-application-firewall

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. Contact our sales team to get started.

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 specially for 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_shield`) 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. 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/<your Fastly service ID>/version/<version_id>/condition/Waf_Prefetch -H
"Fastly-Key: FASTLY_API_TOKEN" -H "Content-Type: application/json" -d '{"statement":"!req.backend.is_shield && !(clie
nt.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.

Limitations

All WAF products that exist today have several limitations:

- **False positives:** Any WAF can mistake good traffic for bad. This is why we strongly recommend that you monitor your logs for a minimum of two weeks before blocking traffic. You don't want start blocking traffic with rules that are generating false positives.
- **DNS configuration:** A WAF only works when traffic is directed through it. It cannot protect against malicious requests that are sent to domain names or IP addresses that are not specified in your WAF configuration.
- **Effective rules:** A WAF is only as effective as the provisioned rules. You can add, remove, or modify rule modes using the API or rule management web interface. You can specifically inspect the Fastly WAF rule set using the WAF API.
- **Custom application vulnerabilities:** If attackers discover a vulnerability unique to your application or the technologies you use, and your WAF configuration does not have a rule to protect against exploits for that particular vulnerability, it will not be able to protect your application in that instance. Customer support can work with you to add additional rule sets to help protect against these types of attacks. If you need more protection than the rule sets provide, customer support can work with you to create custom VCL to help block malicious requests.
- **Inspection of HTTP and HTTPS traffic only:** A WAF only inspects HTTP or HTTPS requests. It will not process any TCP, UDP, or ICMP requests.

Security products note

> ❗ **IMPORTANT:** To ensure your web application only receives traffic from your WAF-enabled Fastly service, we strongly recommend you configure TLS client authentication for that service and allowlist Fastly's assigned IP ranges.

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. Subscribers should maintain appropriate security controls on all web applications and origins, and the use of Fastly's security products do not relieve subscribers of this obligation. Subscribers should test and validate the effectiveness of Fastly's security services to the extent possible prior to deploying these services in production, and continuously monitor their performance and adjust these services as appropriate to address changes in the Subscriber's web applications, origin services, and configurations of the other aspects of the Subscriber's Fastly services.

Integrations

These articles describe how non-Fastly services interoperate with Fastly.

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

Logging endpoints

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 S3

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.

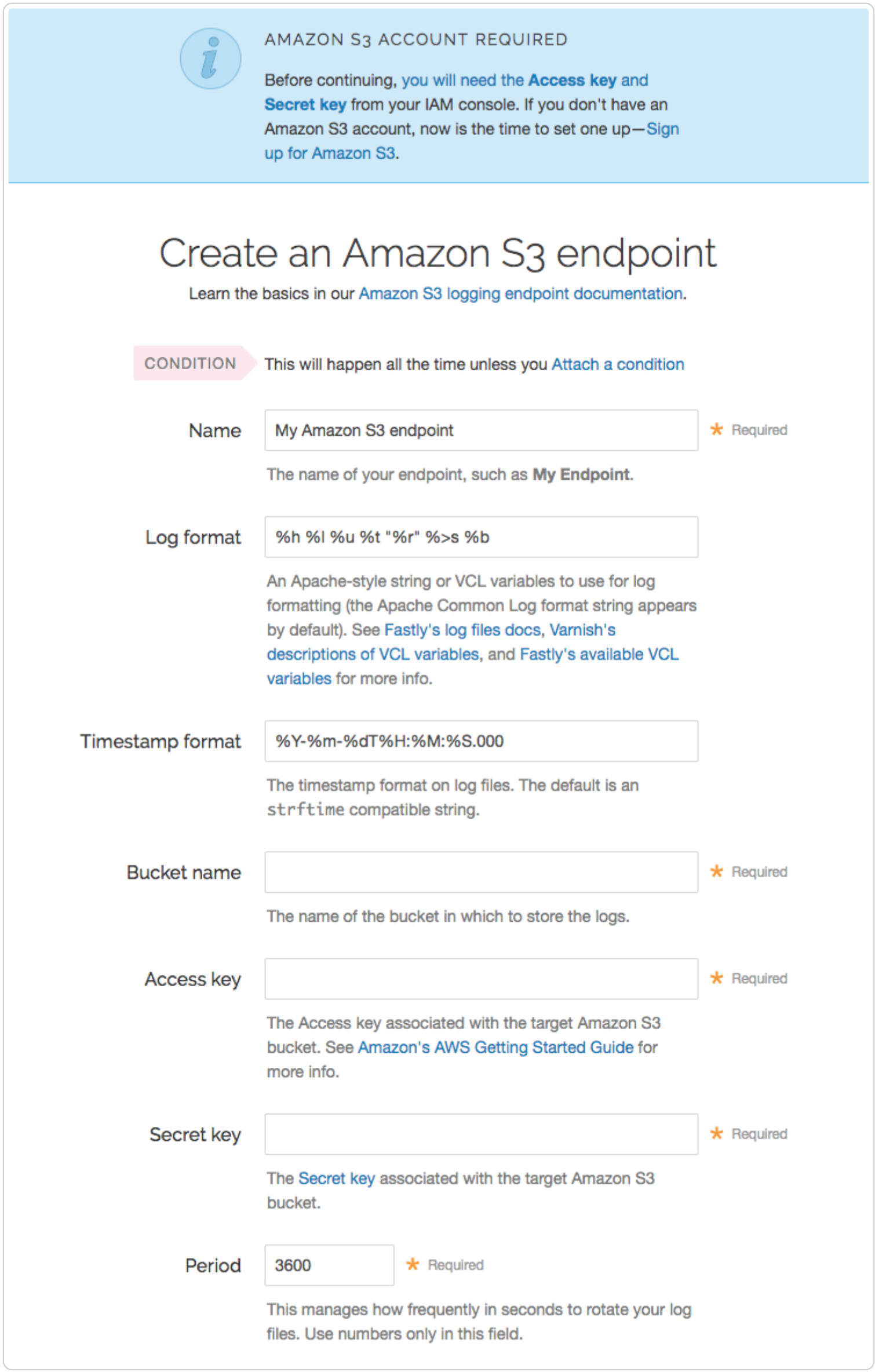
Prerequisites

Before adding Amazon S3 as a logging endpoint for Fastly services, we recommend creating an Identity and Access Management (IAM) user in Amazon S3 specifically for Fastly. Grant the user `ListBucket`, `GetObject`, and `PutObject` permissions for the directory in which you want to store logs. For more information, see Amazon's Getting Your Access Key ID and Secret Access Key page.

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.

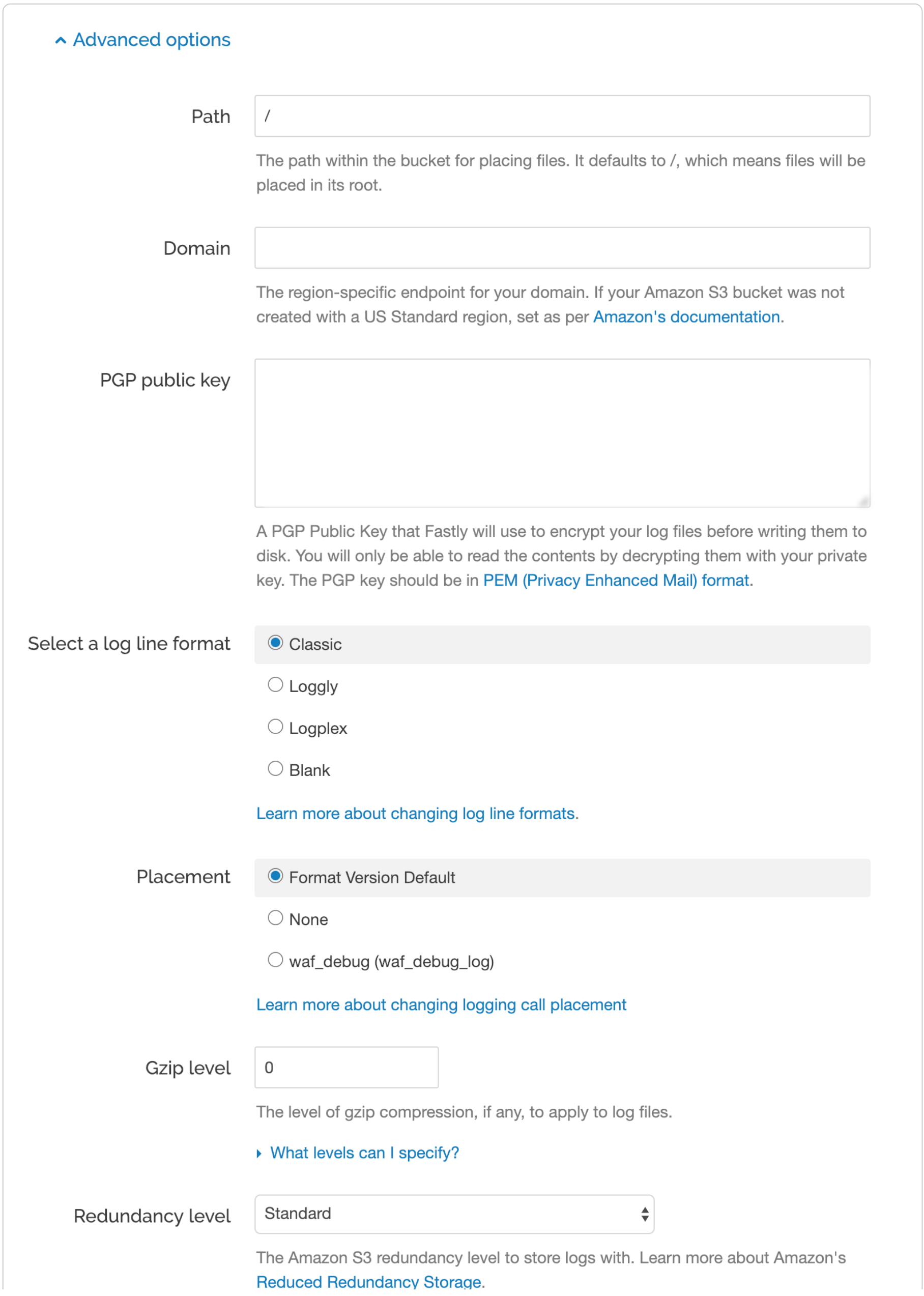


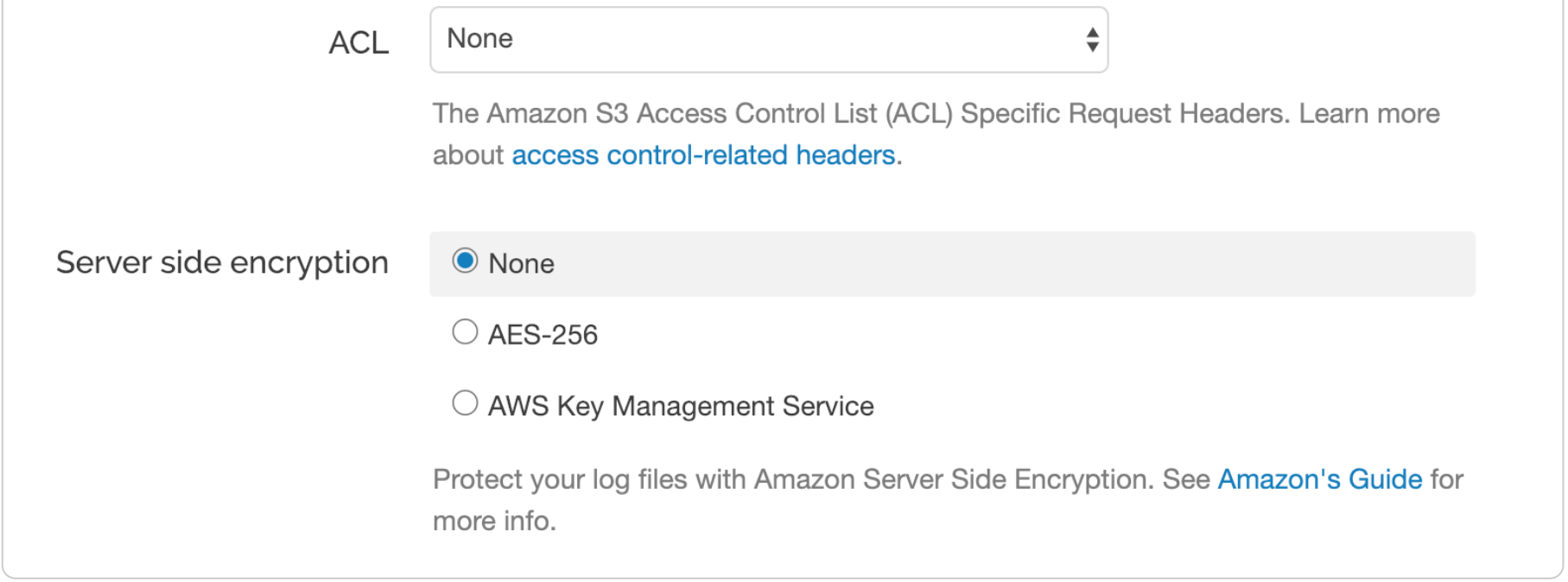
3. Fill out the **Create an Amazon S3 endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
   - In the **Timestamp format** field, optionally type 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, type the name of the Amazon S3 bucket in which to store the logs.
   - In the **Access key** field, type the access key associated with the Amazon S3 bucket. See Amazon's documentation on security credentials for more information.
   - In the **Secret key** field, type the secret key associated with the Amazon S3 bucket. 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.

- In the **Period** field, optionally type 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 type 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 type 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. If you want 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 type 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 **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
   - In the **Gzip level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).
   - 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. Our discussion of format strings also 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 Amazon S3, the Amazon S3 website and API do not make files available for access until after their upload is complete.

Log streaming: Microsoft Azure Blob Storage

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 example.

Here is an example of a SAS token that provides write permissions to a blob:

```
sv=2018-04-05&st=2018-04-29T22%3A18%3A26Z&sr=b&se=2020-04-30T02%3A23%3A26Z&sp=w&sig=Z%2FRHIX5Xcg0Mq2rqI3OlWTjEg2tYkboXr1P9ZUXDtkk%3D
```

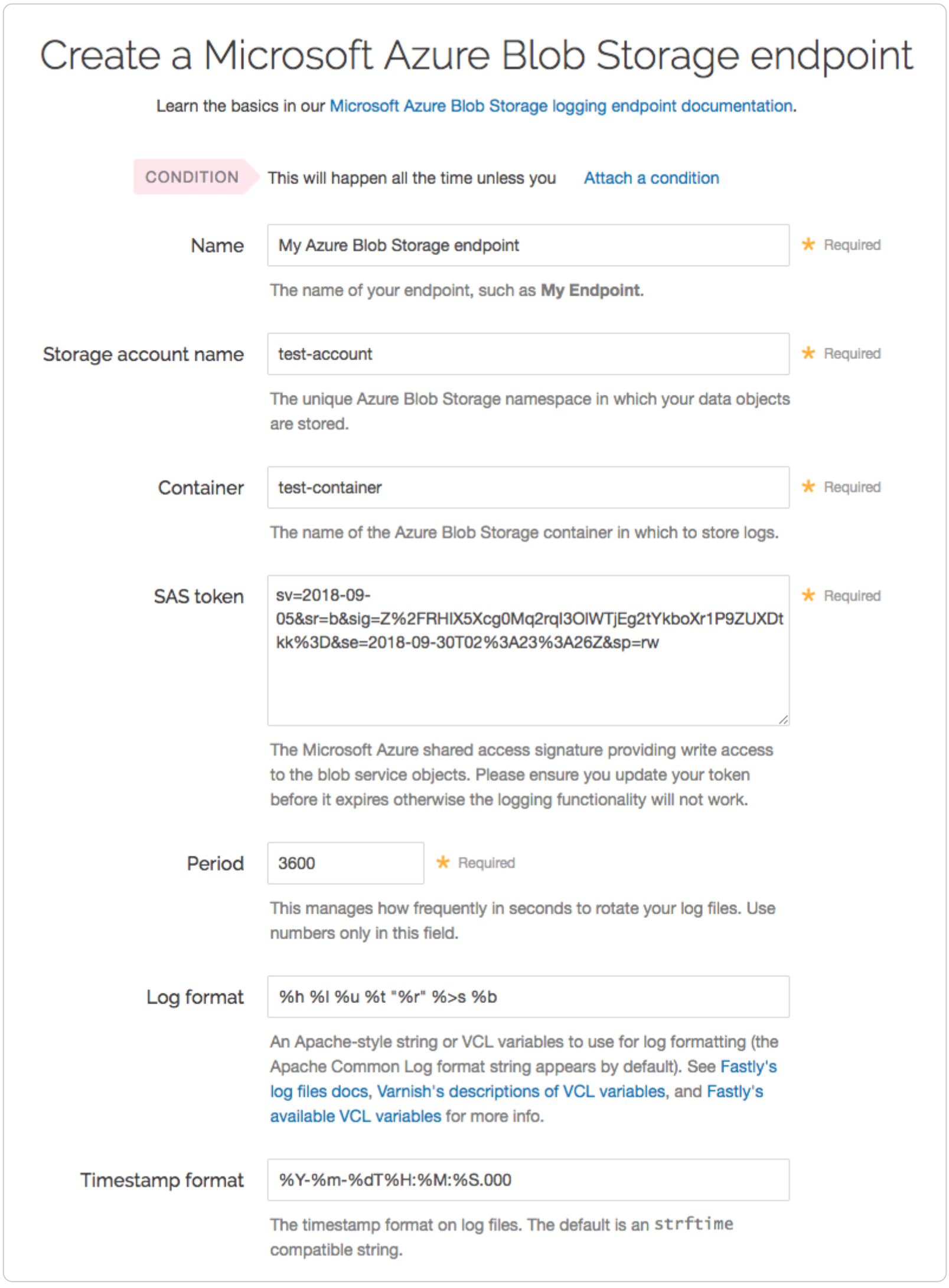
The table breaks down each part of the token to understand how it contributes to the SAS:

Element	Example	Description
`sv`	`sv=2018-04-05`	Storage services version.
`sr`	`sr=b`	Store resources for which this token has access. We require blob (b).
`st`	`st=2018-04-29T22%3A18%3A26Z`	The start time of the token, specified in UTC.
`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.

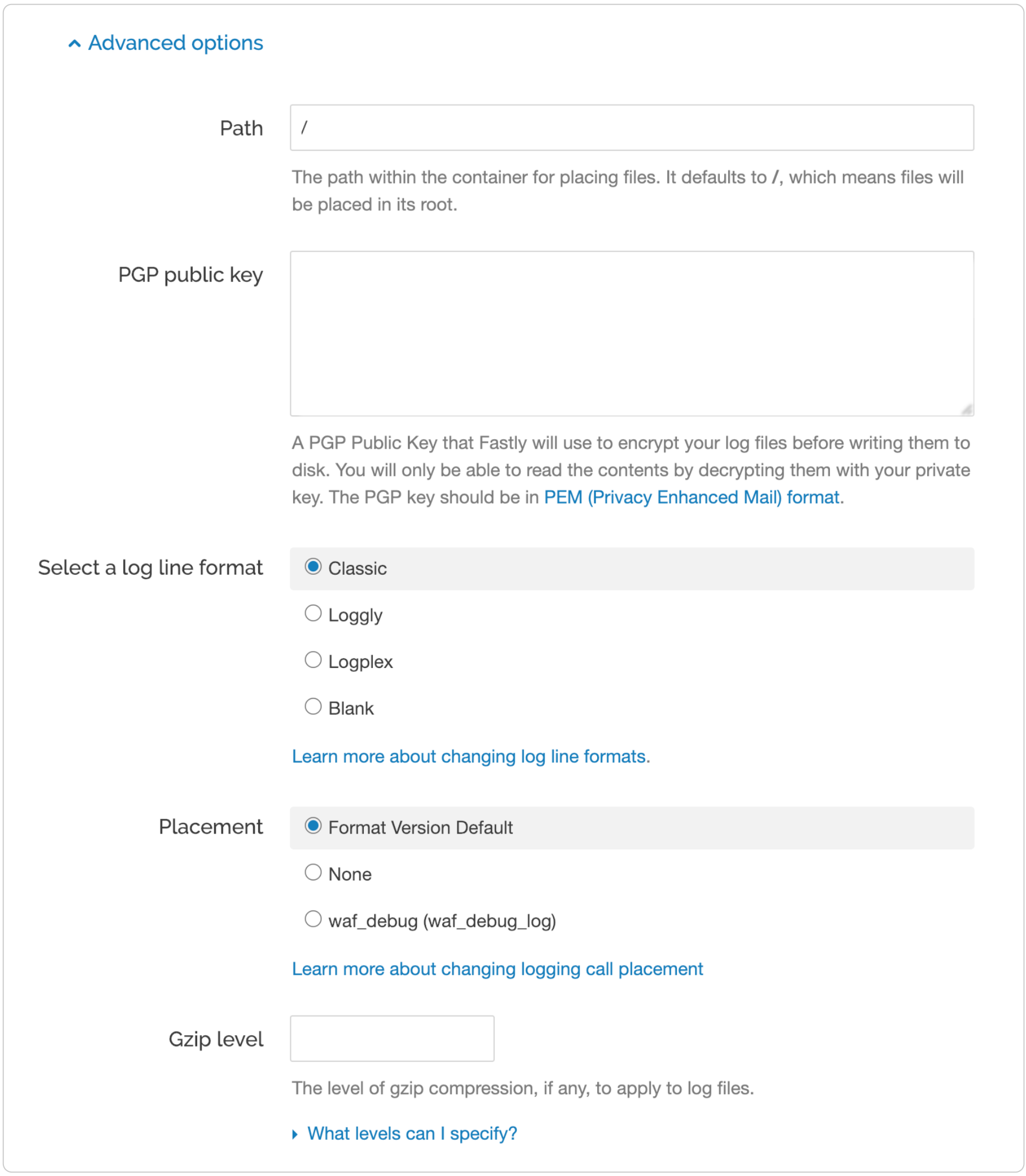


3. Fill out the **Create a Microsoft Azure Blob Storage endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Storage account name** field, type the unique Azure namespace in which your data objects will be stored.
   - In the **Container** field, type 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, type the token associated with the container.

     > **TIP:** Ensure you update your token before it expires otherwise the logging functionality will not work.

   - In the **Period** field, optionally type an interval (in seconds) to control how frequently your log files are rotated. This value defaults to `3600` seconds.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings for more information.
   - In the **Timestamp format** field, optionally type 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 type the path within the container to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the container's root path. Our guide on changing where log files are written provides more information.
   - In the **PGP public key** field, optionally type 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 **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
   - In the **Gzip level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).

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

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.

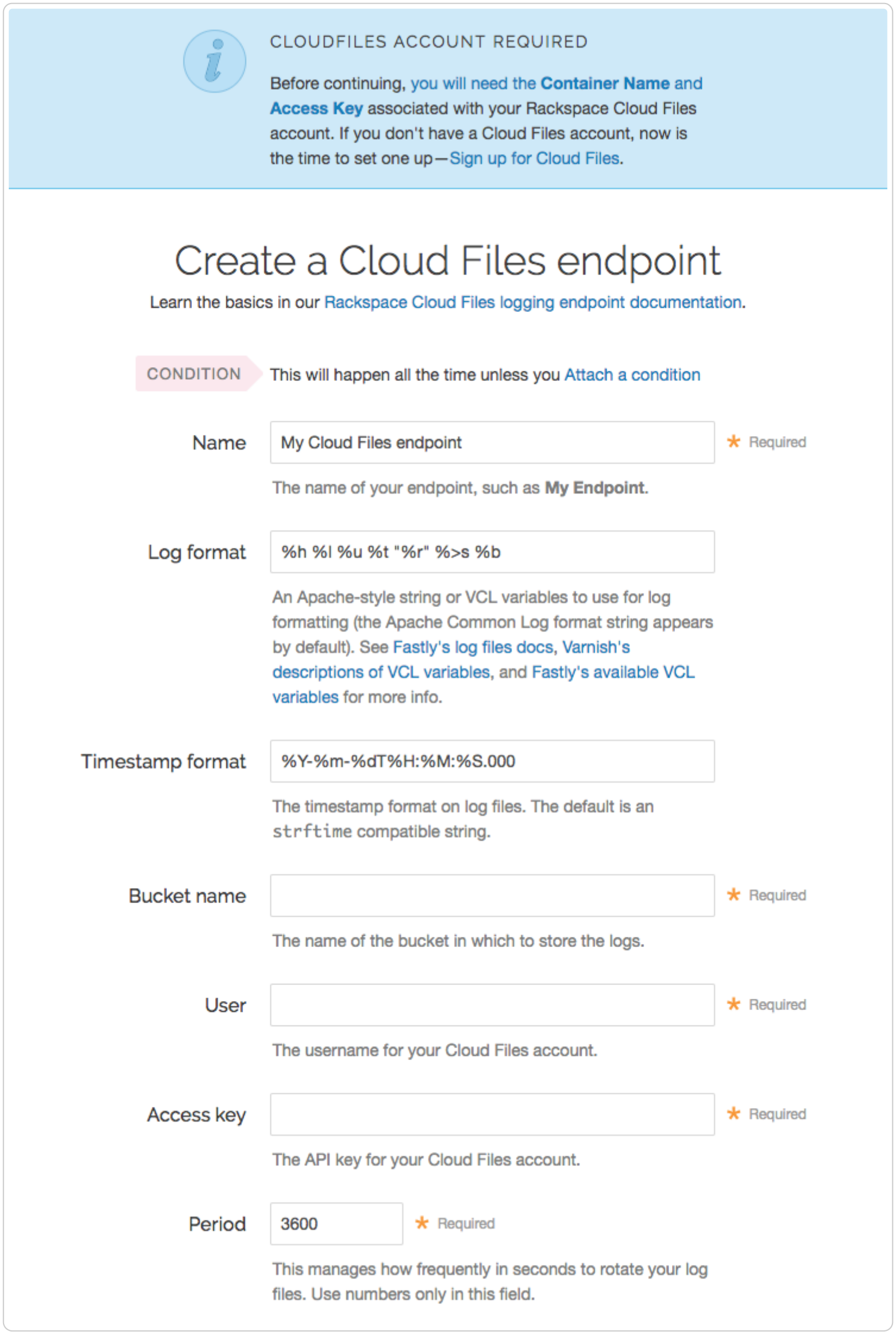
Now that you've created the Cloud Files user and found the API key, you can set up a Cloud Files container.

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**.

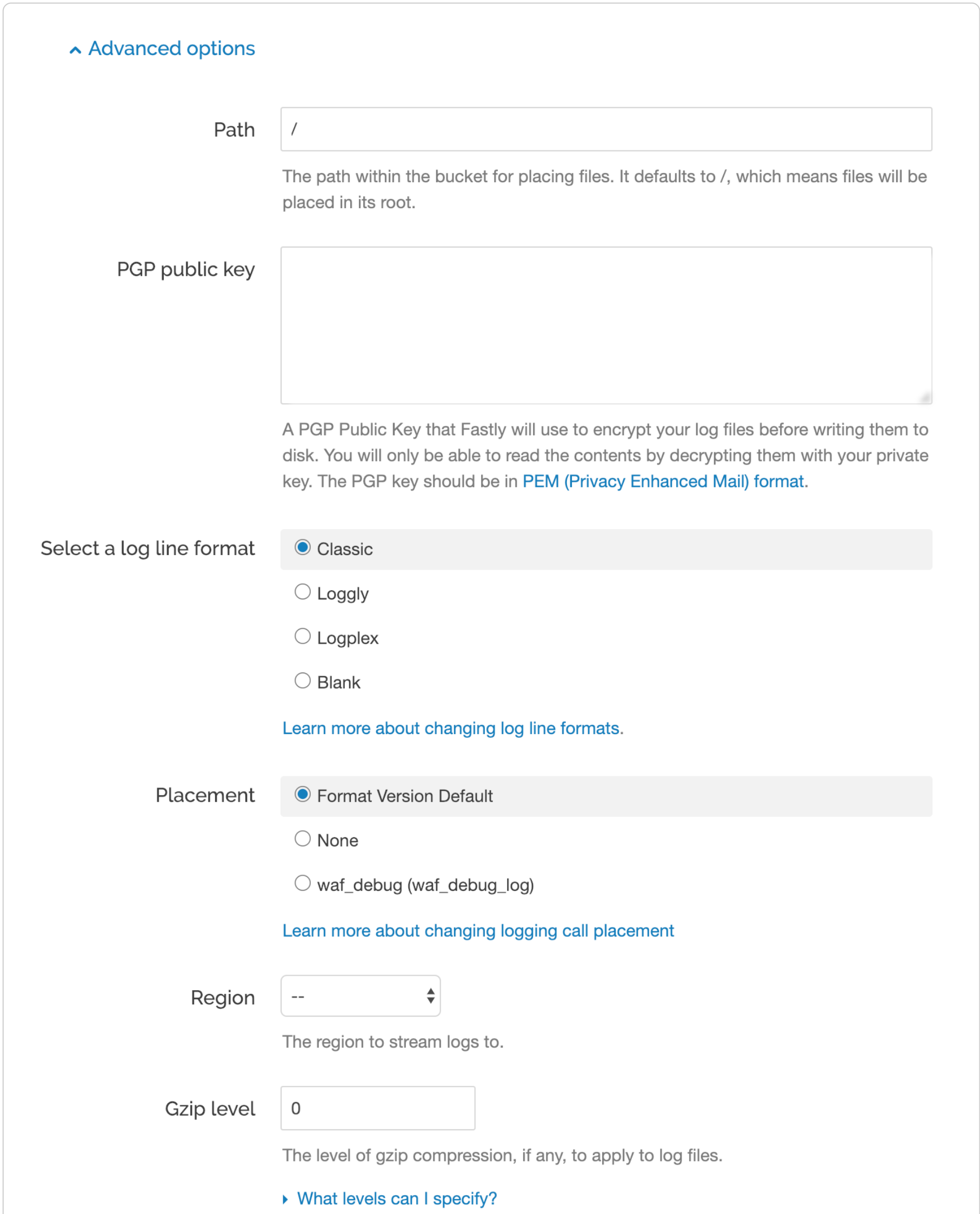
Adding a Cloud Files logging endpoint

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, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings for more information.
   - In the **Timestamp format** field, optionally type 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, type the name of the Cloud Files container in which to store the logs.
   - In the **User** field, type the username of the Cloud Files user you created above.
   - In the **Access key** field, type the API key of your Cloud Files account.
   - In the **Period** field, type an interval (in seconds) to manage how frequently in seconds to rotate your log files. 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 type the path within the container to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the container's root path. Our guide on changing where log files are written provides more information.
   - In the **PGP public key** field, optionally type 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 **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.

- From the **Region** menu, select the region to stream logs to.
- In the **Gzip level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).

6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: Datadog

https://docs.fastly.com/en/guides/log-streaming-datadog

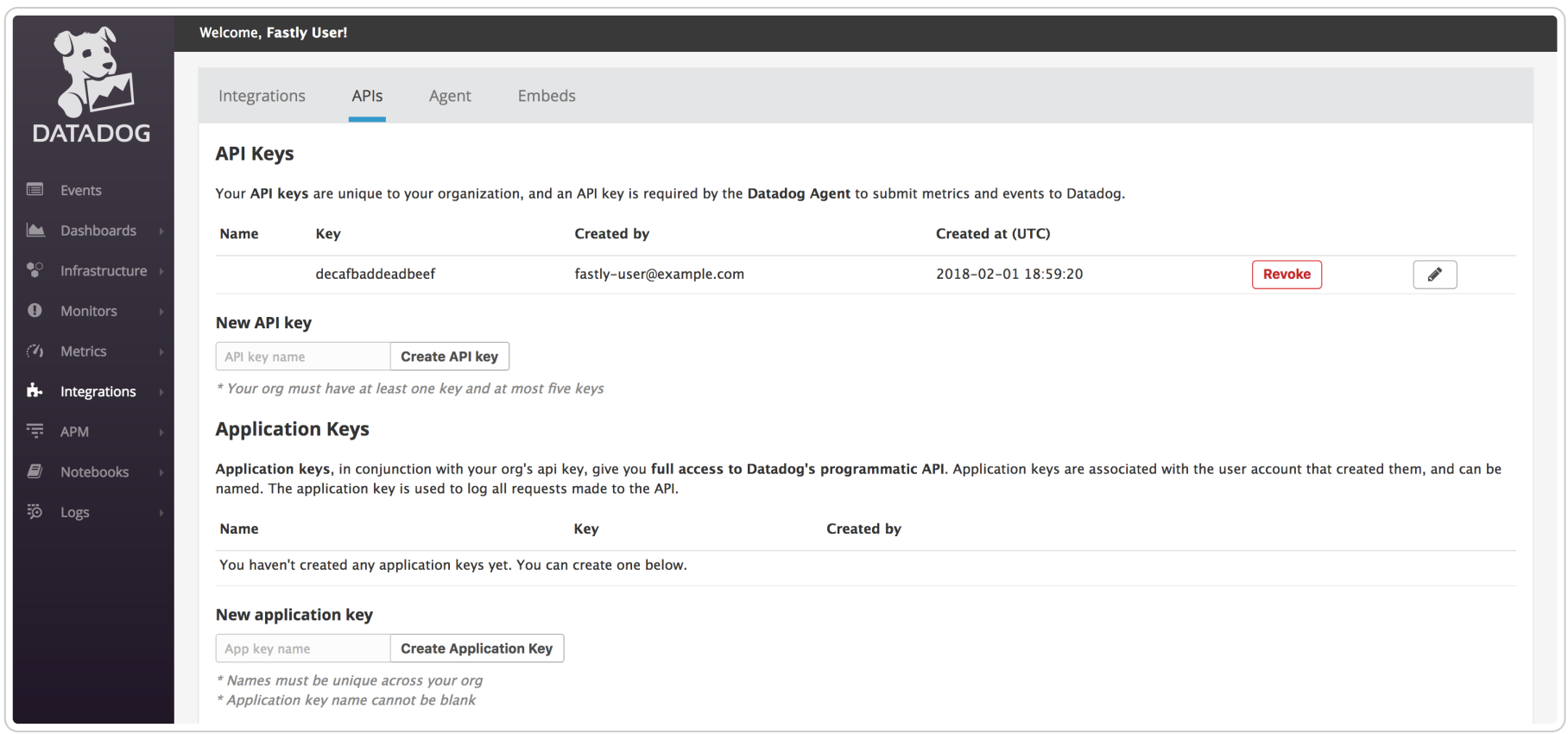
Fastly's Real-Time Log Streaming feature can be configured to send logs in a format readable by Datadog. Datadog is a cloud-based monitoring and analytics solution that allows you to see inside applications within your stack and aggregate the results.

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 (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, type a human-readable name for the endpoint.
   - In the **Log format** field, set the format to be the Datadog API key you noted earlier, followed by a space and then your log string in JSON format. Be sure to add the JSON as a single line to ensure proper parsing. We've described the use of this format below with additional suggestions.
   - In the **Syslog address** field, enter the appropriate information in the domain and port fields according to where your Datadog account is registered. Enter either `intake.logs.datadoghq.com` (for the US) or `tcp-intake.logs.datadoghq.eu` (for the EU) in the domain field before the colon, and then enter either `10516` (for the US) or `443` (for the EU) in the port field after the colon.

- Leave the **Token** field blank.
- 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 the appropriate information according to where your Datadog account is registered. Enter either `intake.logs.datadoghq.com` (for the US) or `tcp-intake.logs.datadoghq.eu` (for the EU). This is the hostname Fastly will use to verify the syslog server's certificate.
- Leave the **TLS CA certificate** field blank.

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 `blank`. Our guide on changing log line formats provides more information.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
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 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": "%{Fastly-Orig-Host}i",
  "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,
  "socket_ploss": %{client.socket.ploss}V
}
```

Using logging formats other than JSON

The log format you specify doesn't have to be JSON. If you use other formats however, you'll need to parse the log manually instead. When selecting a format other than JSON, choose the format that best suits your needs (such as Apache Common Log Format) and then use Datadog's Grok Parser to extract the fields you want.

Fastly's guide to logging formats provides more information about using custom log formats with streaming logs. We also provide additional request and response variables for use with logging beyond the standard logging directives. Our guide to useful logging variables describes these in more detail.

When you use a logging format other than JSON, consider using Datadog's logging pipelines feature, which allows you to create a filtered subset of incoming logs. For example, you could use the User-Agent Parser to identify details like web browser, operating system, and device type and model, or the URL Parser to identify details like protocol, domain, and query string. Simply add these parsers to a Datadog pipeline and adjust the mapping as necessary.

Log streaming: DigitalOcean Spaces

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.

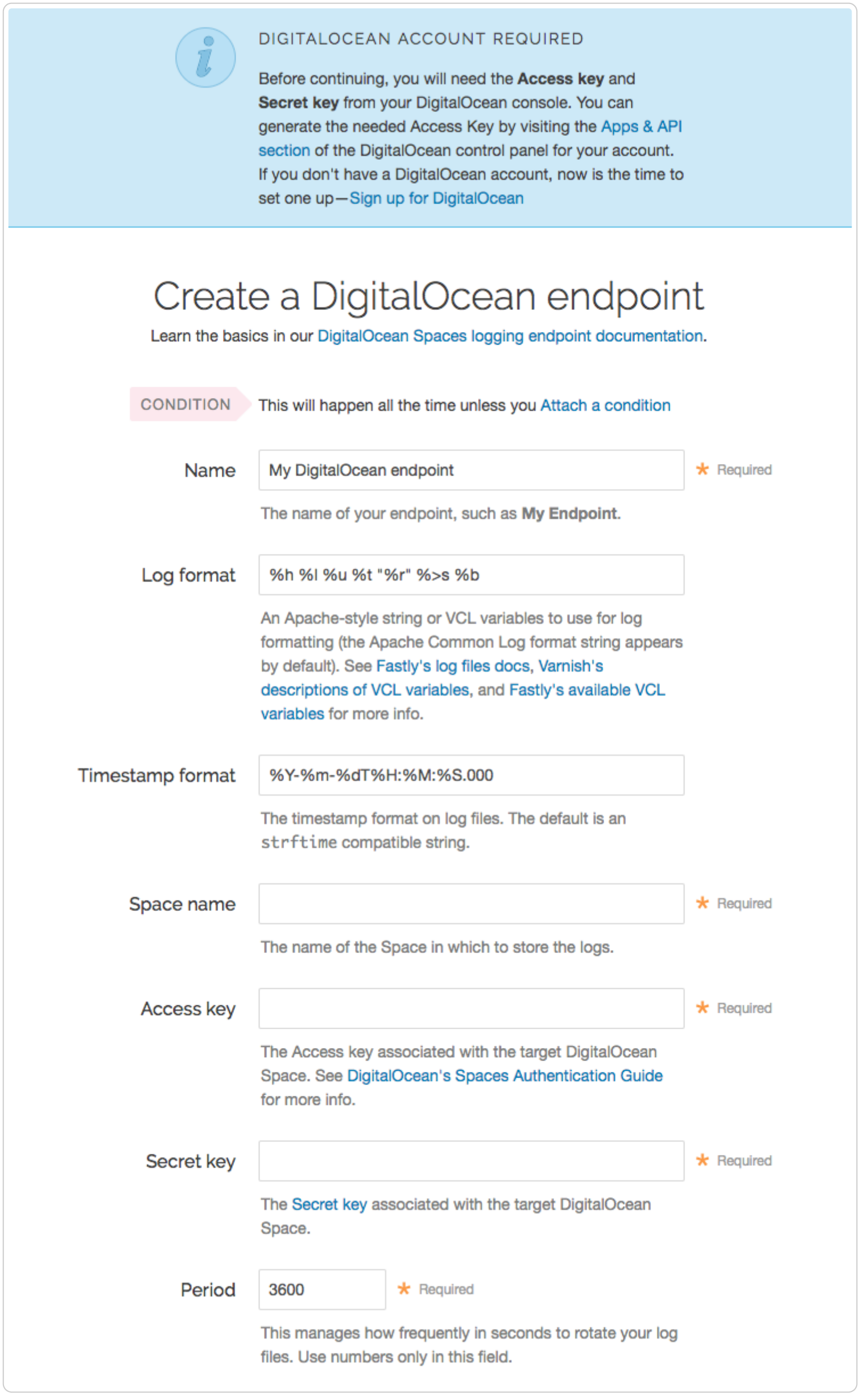
Prerequisites

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:

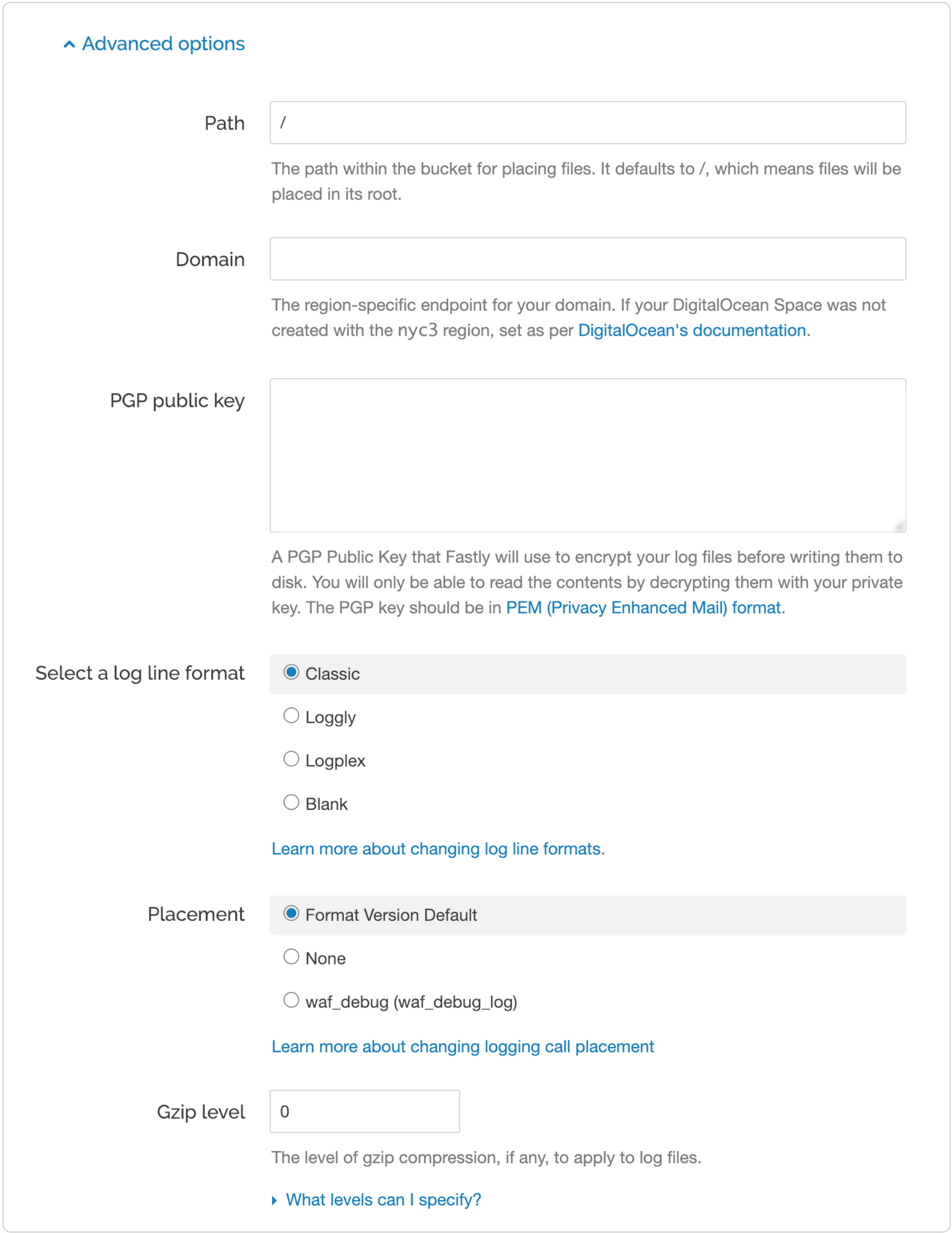
1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Spaces by DigitalOcean **Create endpoint** button. The Create a DigitalOcean endpoint page appears.



3. Fill out the **Create a DigitalOcean endpoint** fields as follows:
    - In the **Name** field, type a human-readable name for the endpoint.
    - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
    - In the **Timestamp format** field, optionally type 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, type the name of the DigitalOcean Space in which to store the logs.
    - In the **Access key** field, type the access key associated with the DigitalOcean Space. See the DigitalOcean Spaces Authentication Guide for more information.

- In the **Secret key** field, type the secret key associated with the DigitalOcean Space.
- In the **Period** field, optionally type 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 type the path within the container to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the container's root path. Our guide on changing where log files are written provides more information.
   - In the **Domain** field, type 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 type 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 **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
- In the **Gzip level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).

6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: Elasticsearch

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.

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

NOTE: This logging endpoint is disabled by default. To enable this endpoint for your account, contact support@fastly.com and request it.

NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

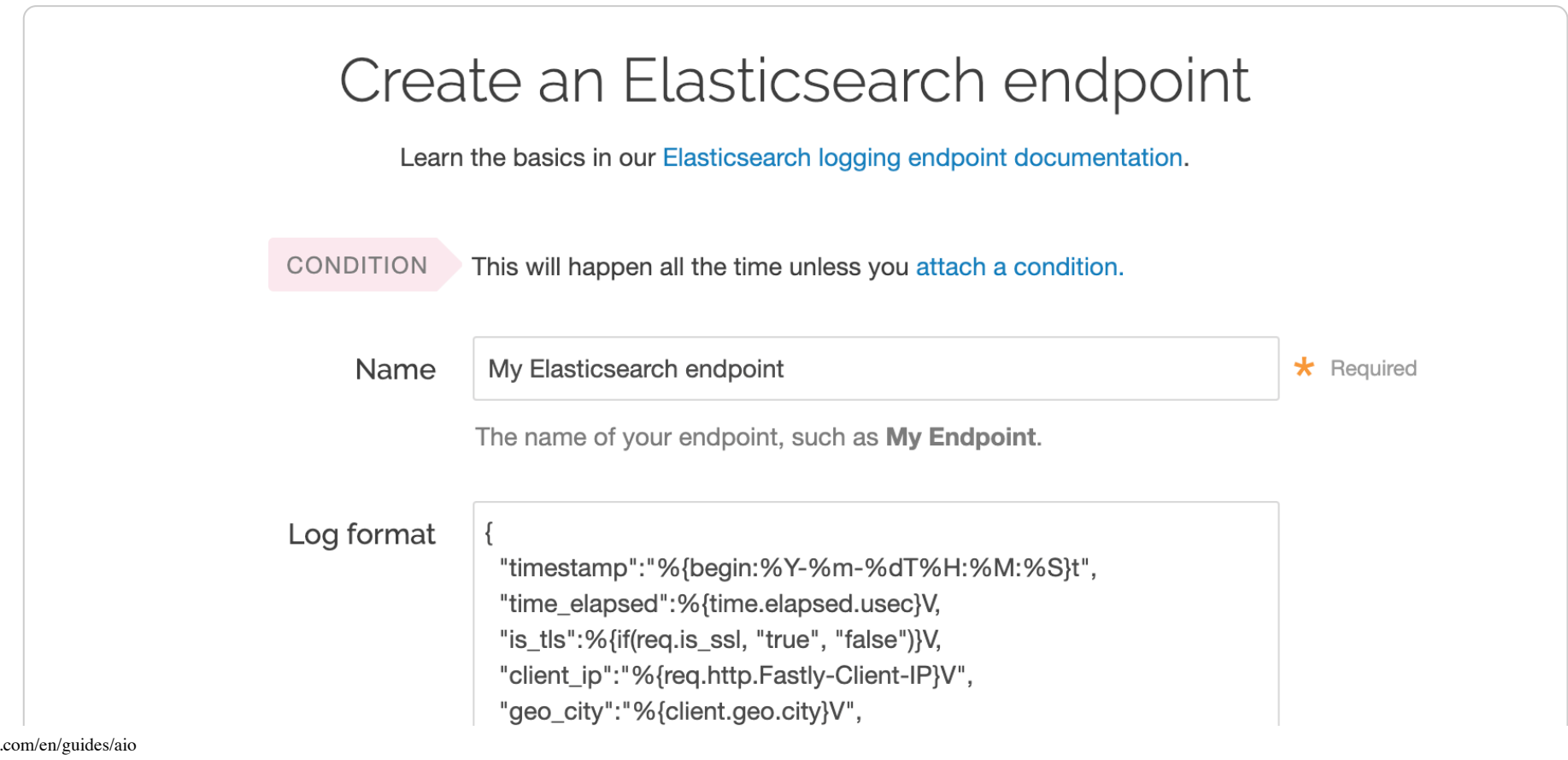
Prerequisites

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.

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.



```
  "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"
}
```

A suggested format string appears by default. See Fastly's log files docs, Varnish's descriptions of VCL variables, and Fastly's available VCL variables for more info.

URL

* Required

The Elasticsearch endpoint URL.

Index

* Required

The Elasticsearch index to send logs to.

Maximum logs

0

Maximum number of logs to append to a batch, if non-zero.

Maximum bytes

0

Maximum size of log batch, if non-zero.

BasicAuth user

BasicAuth user.

BasicAuth password

BasicAuth password.

TLS hostname

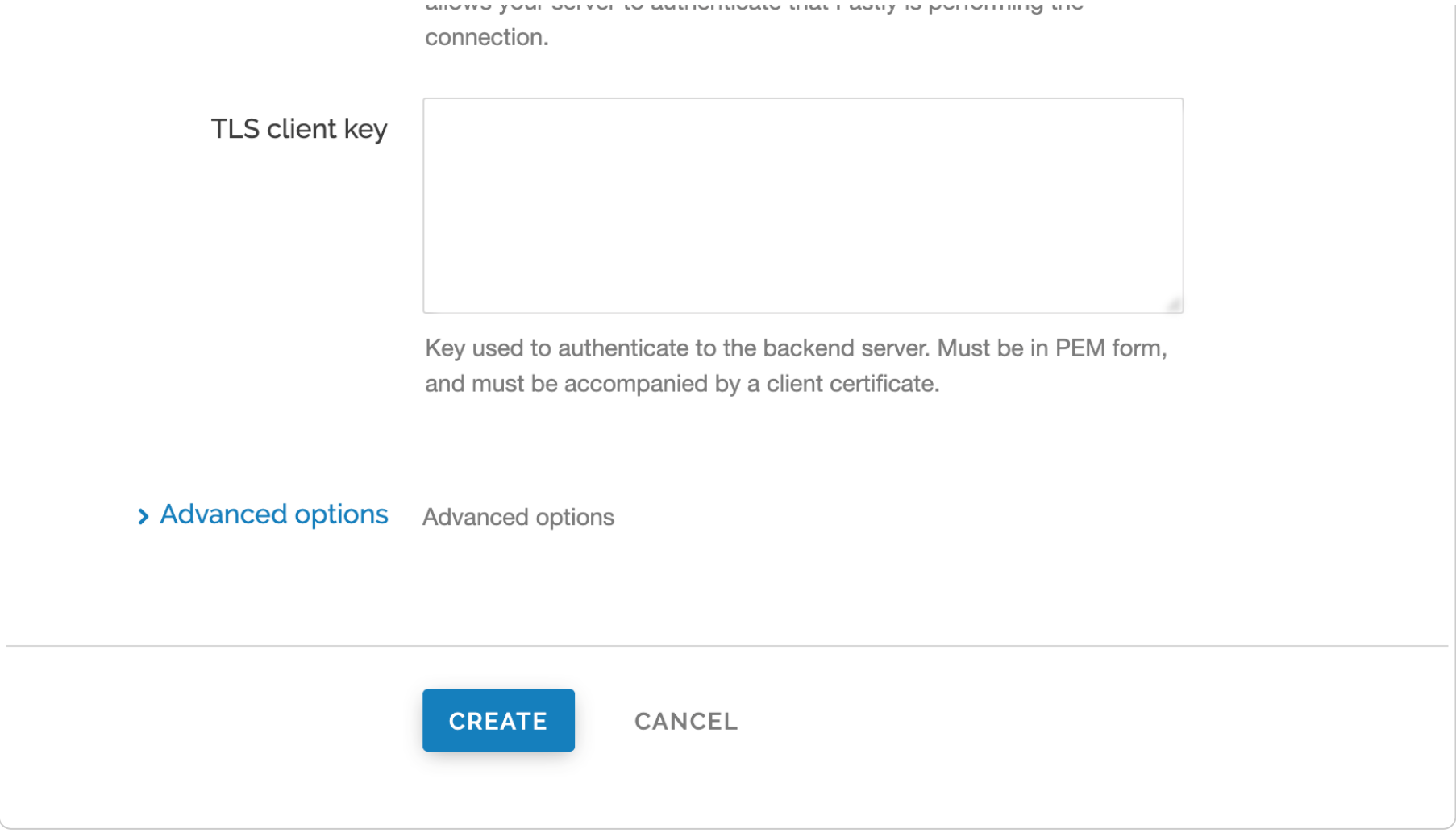
The hostname used to verify the server's certificate. It can either be the Common Name or in subAltNames.

TLS CA certificate

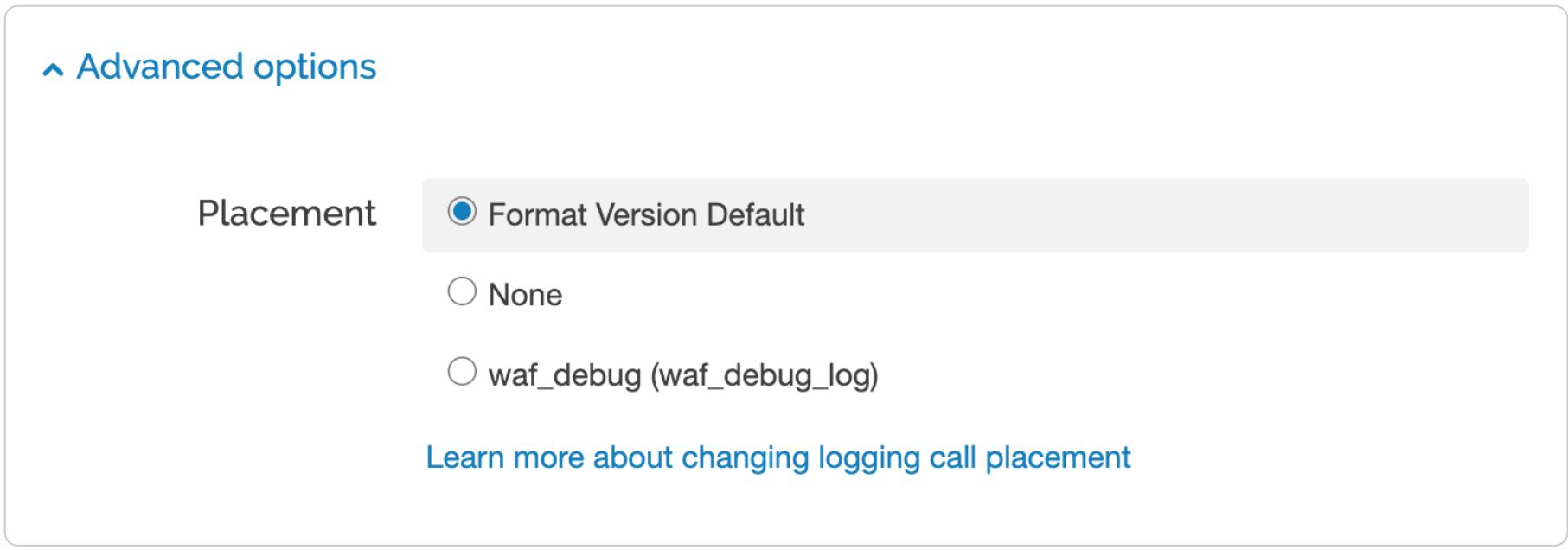
CA certificate used to verify the certificate from origin. Must be in PEM form.

TLS client certificate

Certificate used to authenticate to the origin server. Must be in PEM form, and must be accompanied by a client certificate. It is advised to not used keys from anywhere else. A client certificate and client key allows your server to authenticate that Fastly is performing the



3. Fill out the **Create an Elasticsearch endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, enter the data to send to Elasticsearch. See the example format section for details.
   - In the **URL** field, type the Elasticsearch endpoint URL that includes a port to which logs should be sent. Be sure this port 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 **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 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 type the hostname used to verify the server's certificate. This can be either the Common Name (CN) or Subject Alternate Name (SAN).
   - 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 **Advanced options** link of the **Create an Elasticsearch endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. 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":"%{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.method}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"
}
```

Log streaming: FTP

https://docs.fastly.com/en/guides/log-streaming-ftp

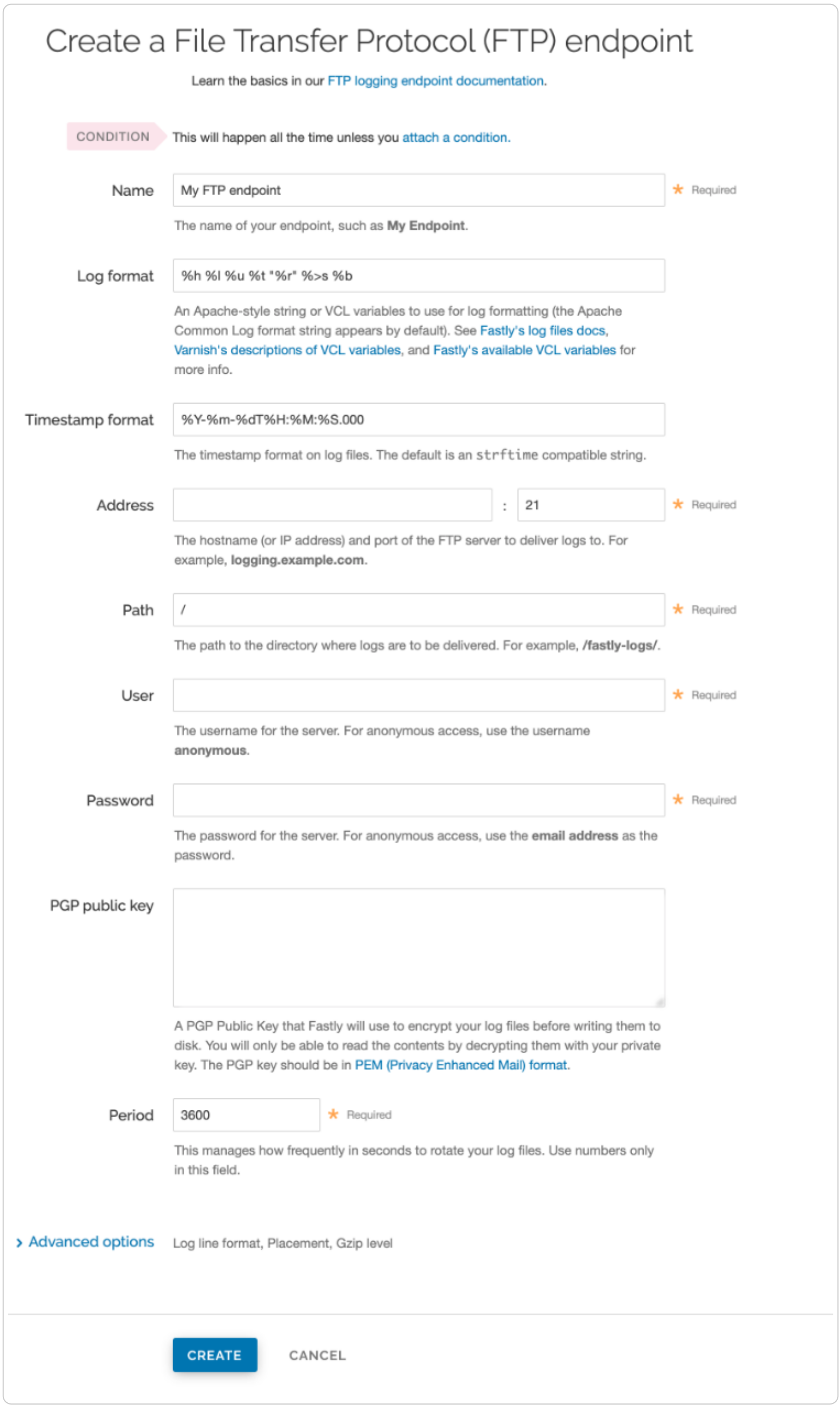
Fastly's Real-Time Log Streaming feature can send log files to password-protected and anonymous FTP servers.

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:

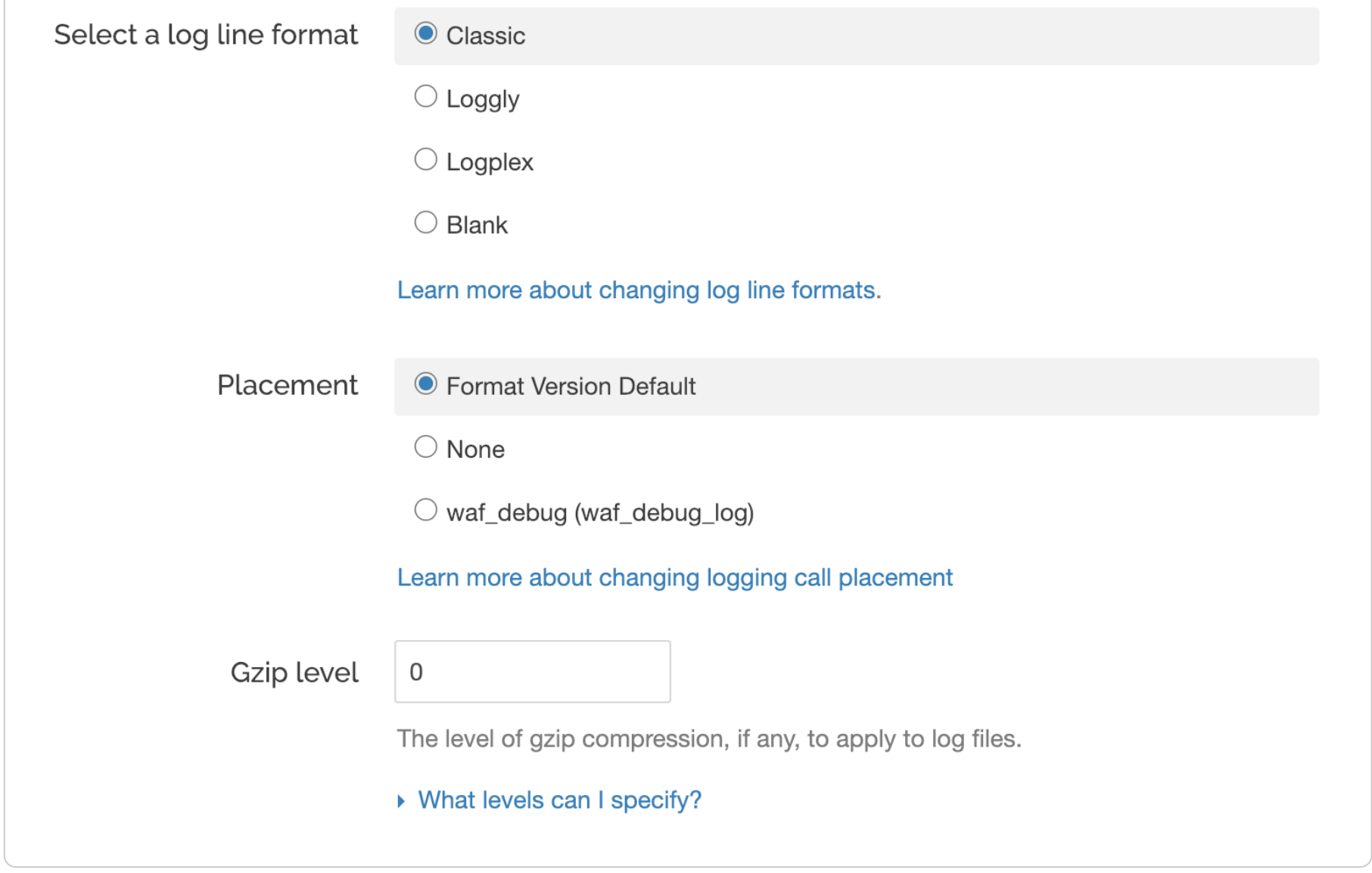
1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the FTP **Create endpoint** button. The Create a File Transfer Protocol (FTP) endpoint page appears.



3. Fill out the **Create a File Transfer Protocol (FTP) endpoint** fields as follows:
    - In the **Name** field, type a human-readable name for the endpoint.
    - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
    - In the **Timestamp format** field, optionally type 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, type the hostname or IP address of the FTP server. In the port field, type the port number you're using for FTP (the default is `21`).
- In the **Path** field, optionally type the path to store the files. The path ends with a trailing slash. If this field is left empty, the files will be saved in the root path. Our guide on changing where log files are written provides more information.
- In the **User** field, type the username used to authenticate to the FTP server. For anonymous access, use the username `anonymous`.
- In the **Password** field, type 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 type 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, type 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 **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
   - In the **Gzip Level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: Google BigQuery

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, see Google's guide on generating service account credentials. When you create the service account, set the key type to JSON.

Obtaining the private key and client email

After you create the service account, download the JSON file to your computer. This file contains the credentials for your BigQuery service account. Open the file and make a note of the `private_key` and `client_email`.

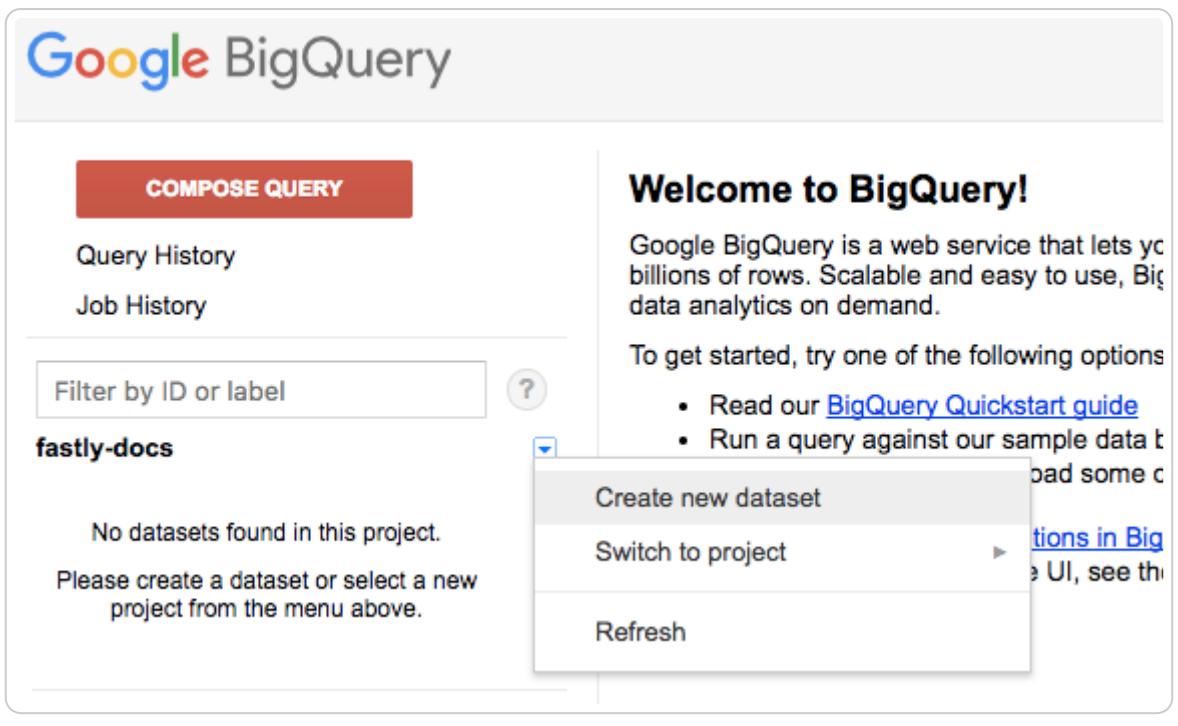
Enabling the BigQuery API

To send your Fastly logs to your BigQuery table, you'll need to enable the BigQuery API in the Google Cloud Platform API Manager.

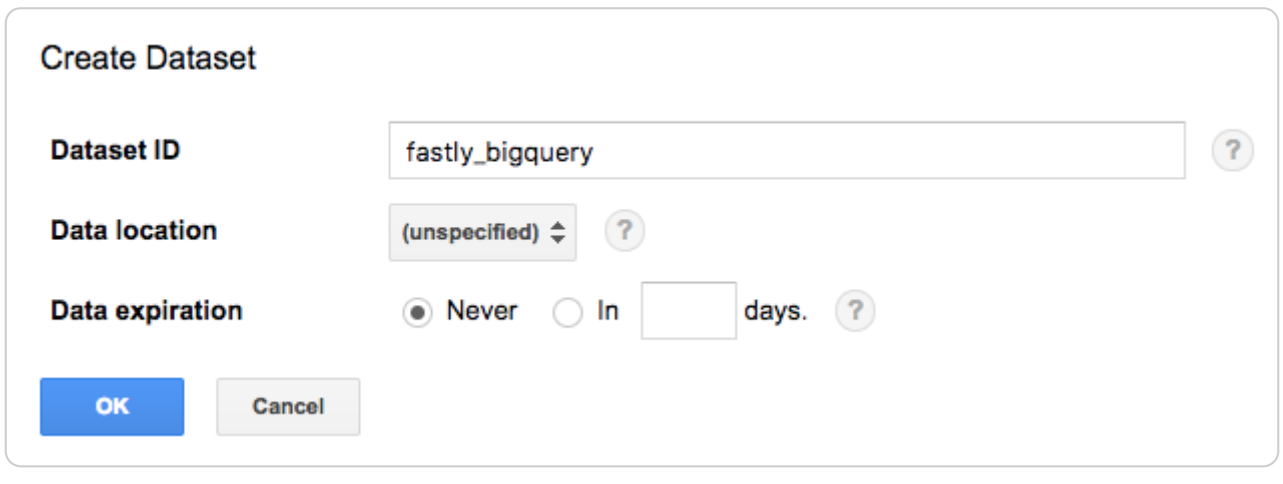
Creating the BigQuery dataset

After you've enabled the BigQuery API, follow these instructions to create a BigQuery dataset:

1. Log in to BigQuery.
2. Click the arrow next to your account name on the sidebar and select **Create new dataset**.



The Create Dataset window appears.



3. In the **Dataset ID** field, enter a name for the dataset (e.g., `fastly_bigquery`).

4. Click the **OK** 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:

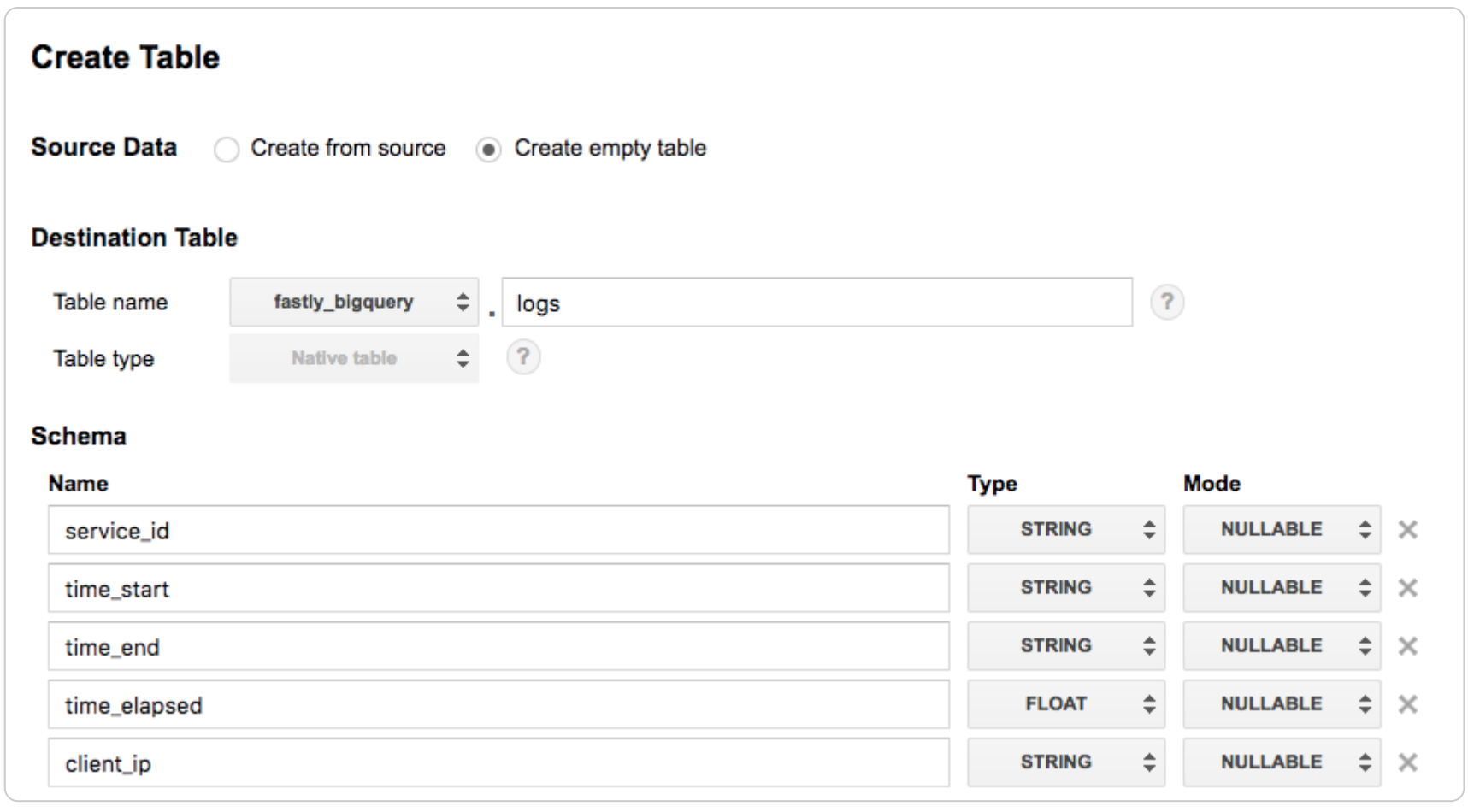
- Edit the schema using the BigQuery web interface.
- Edit the schema using the text field in the BigQuery web interface.
- Use an existing table.
- Set the table to automatically detect the schema.

> **NOTE:** Setting the table to automatically detect the schema may give unpredictable results.

Follow these instructions to add a BigQuery table:

1. On the BigQuery website, click the arrow next to the dataset name on the sidebar and select **Create new table**.

The Create Table page appears.

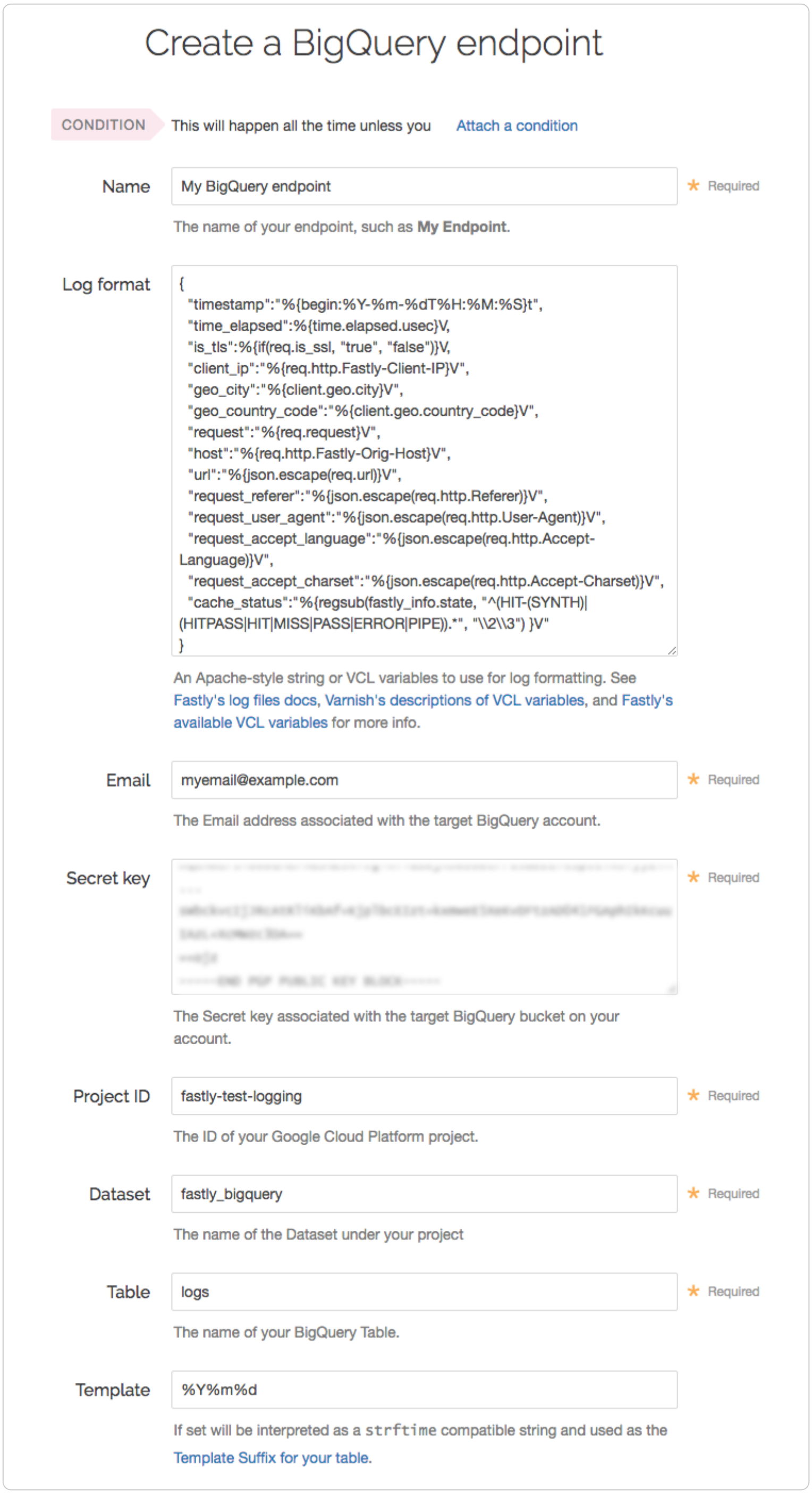


2. In the **Source Data** section, select **Create empty table**.
3. In the **Table name** field, enter a name for the table (e.g., `logs`).
4. In the **Schema** section of the BigQuery website, use the interface to add fields and complete the schema. See the example schema section for details.
5. Create the **Create Table** button.

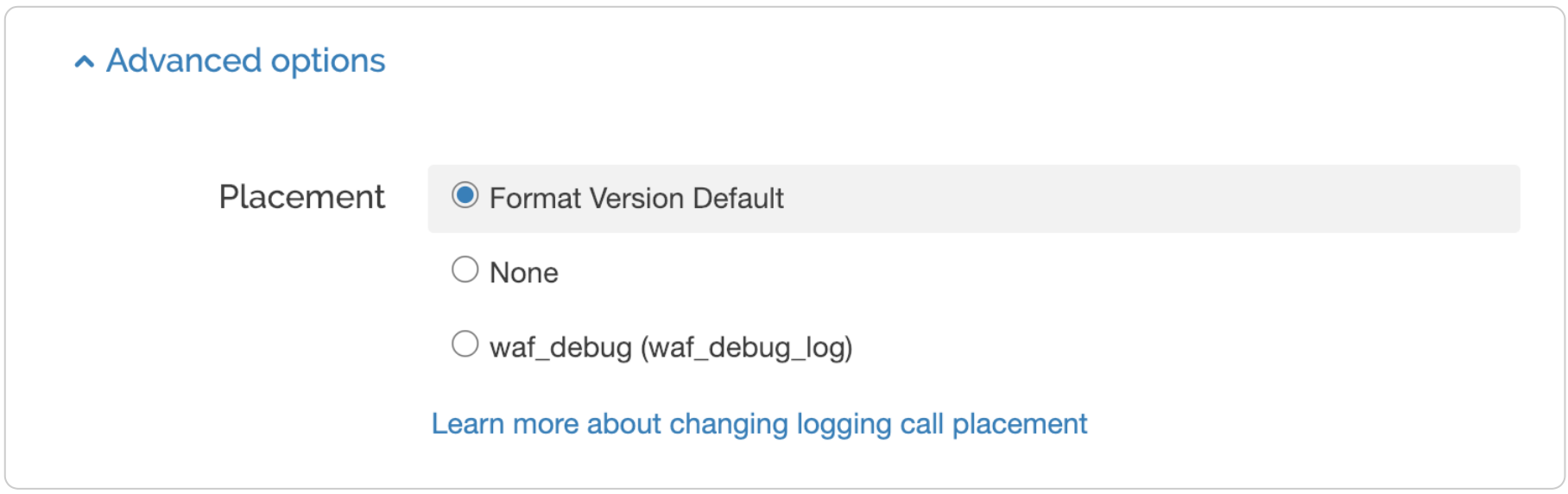
Adding BigQuery as a logging endpoint

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 **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 **Advanced options** link of the **Create a BigQuery endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click **Create** to create the new logging endpoint.
7. 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":"%{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.method}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"
}


```

Example schema

The BigQuery schema for the example format shown above would look something like this:

```
timestamp:TIMESTAMP,time_elapsed:FLOAT,is_tls:BOOLEAN,client_ip:STRING,geo_city:STRING,geo_country_code:STRING,reques
t:STRING,host:STRING,url:STRING,request_referer:STRING,request_user_agent:STRING,request_accept_language:STRING,reque
st_accept_charset:STRING,cache_status:STRING
```

Log streaming: Google Cloud Storage

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`.

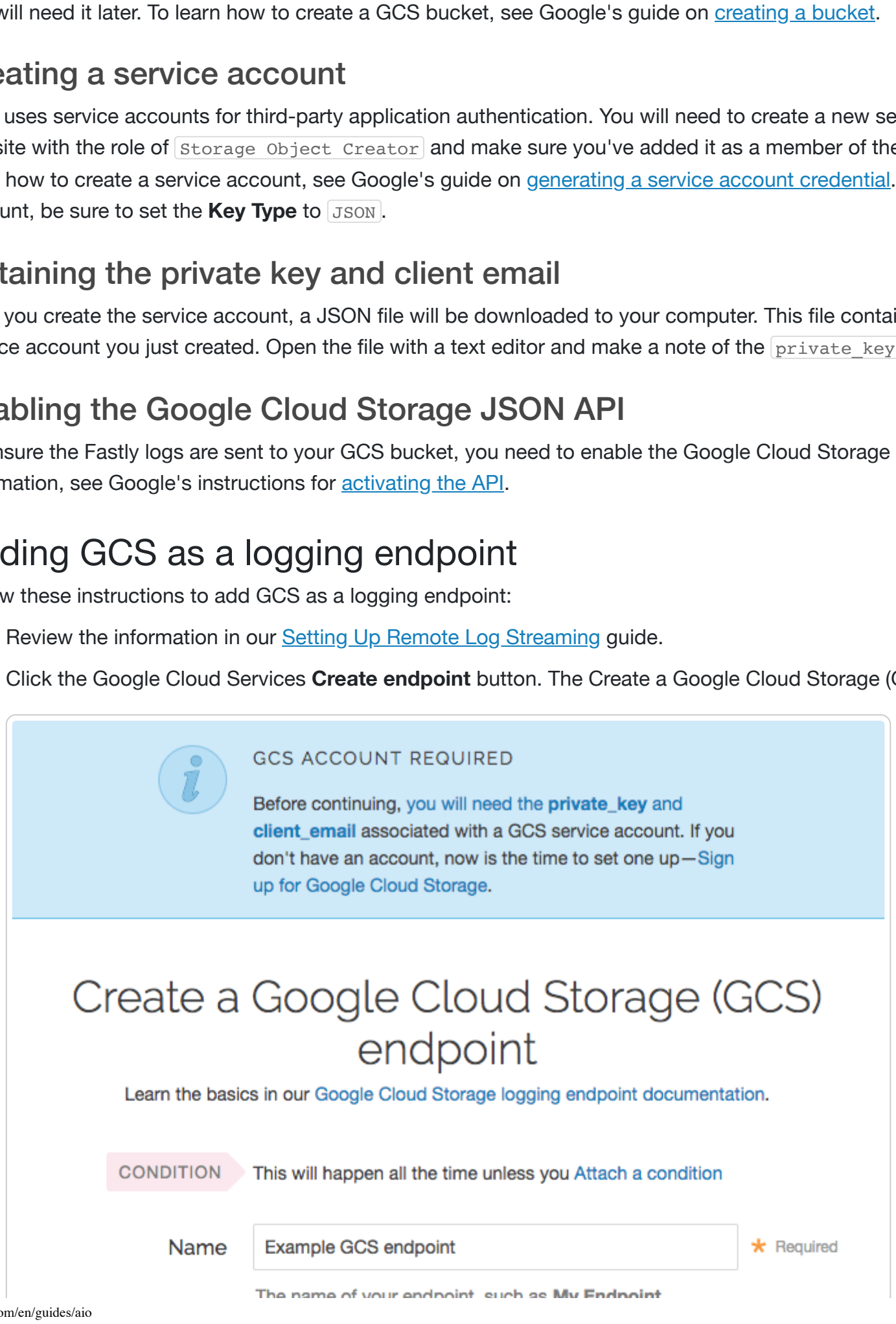
Enabling the Google Cloud Storage JSON API

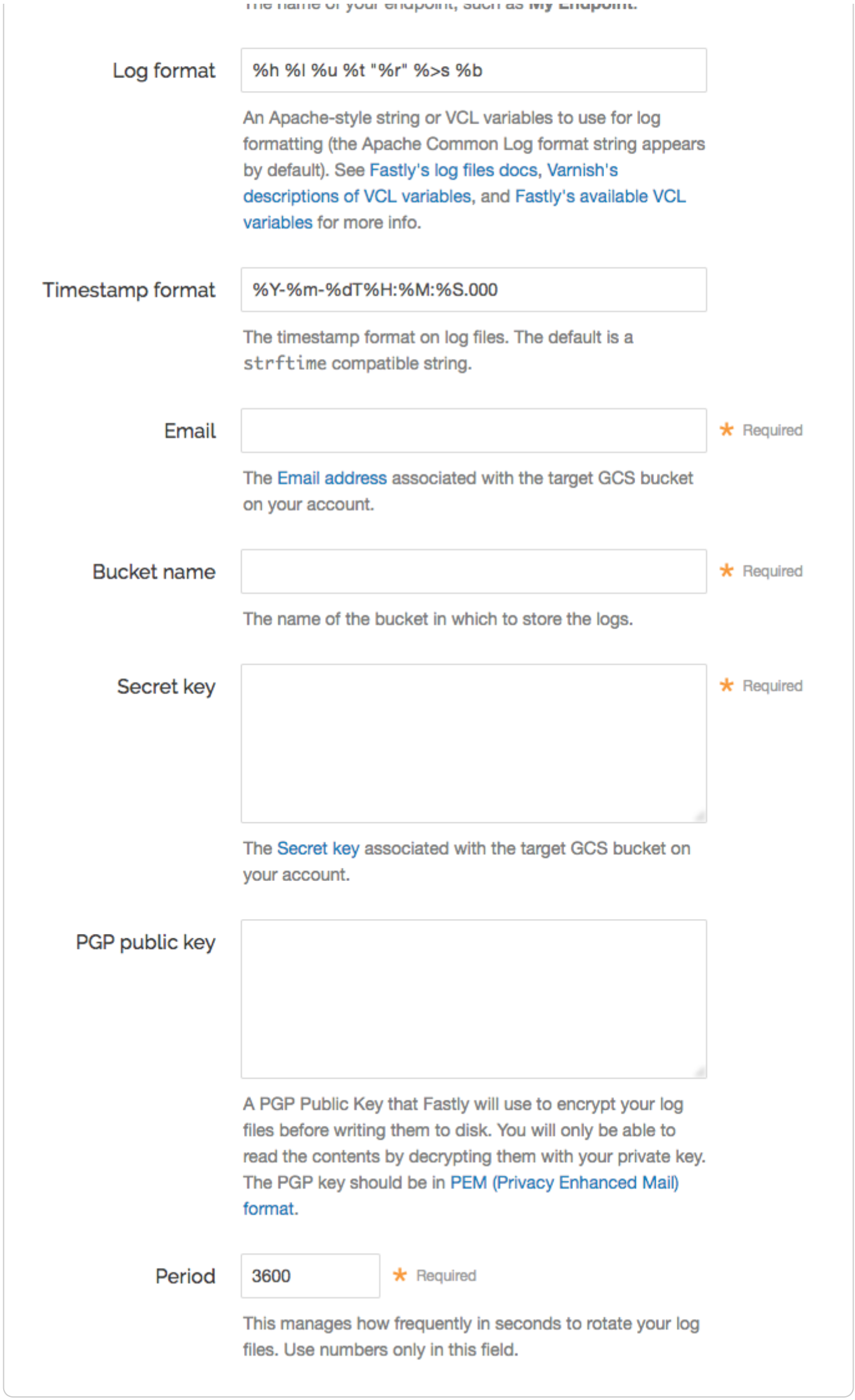
To ensure the Fastly logs are sent to your GCS bucket, you need to enable the Google Cloud Storage JSON API. For more information, see Google's instructions for activating the API.

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.

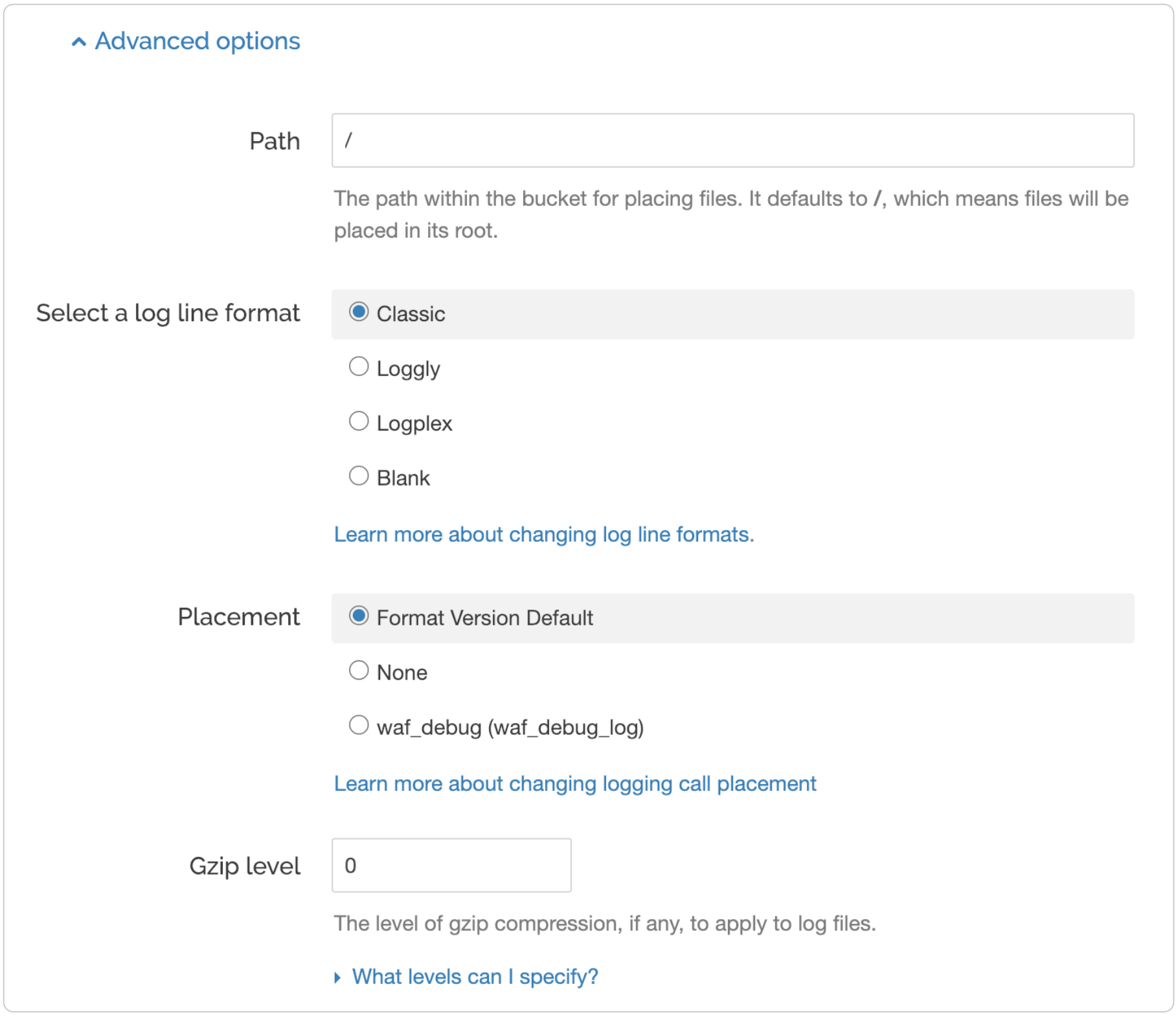




3. Fill out the **Create a Google Cloud Storage (GCS) endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings for more information.
   - In the **Timestamp format** field, optionally type 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, type 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, type the name of the GCS bucket in which to store the logs.
   - In the **Secret key** field, type 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 type 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 type 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 type the path within the bucket to store the files. Specify a directory by ending the path with a trailing slash (`/`). Leaving this field empty saves the files 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 **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
   - In the **Gzip Level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).
6. Click **Create** to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: Honeycomb

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.

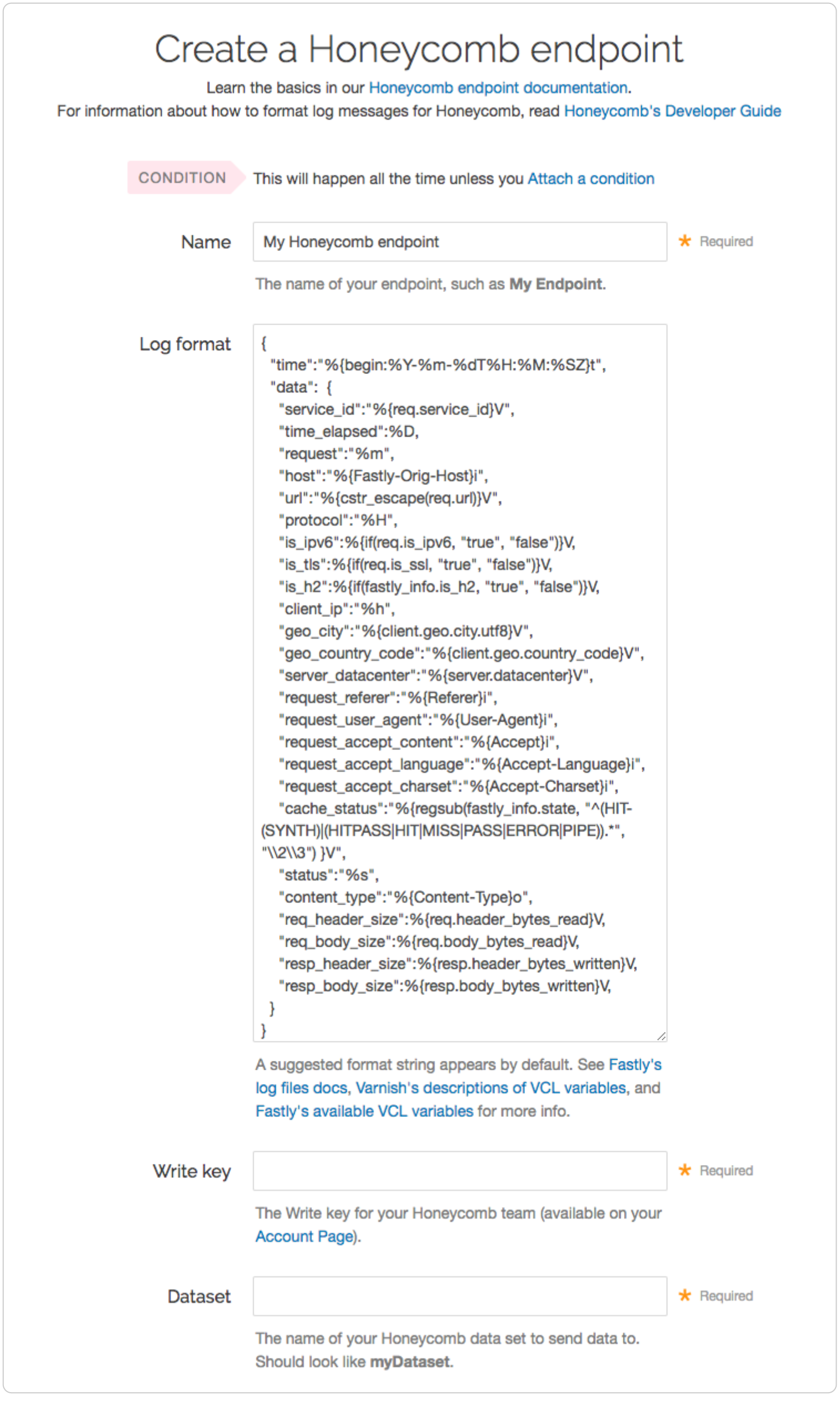
Prerequisites

Before adding Honeycomb as a logging endpoint for Fastly services, you'll need to perform the following steps:

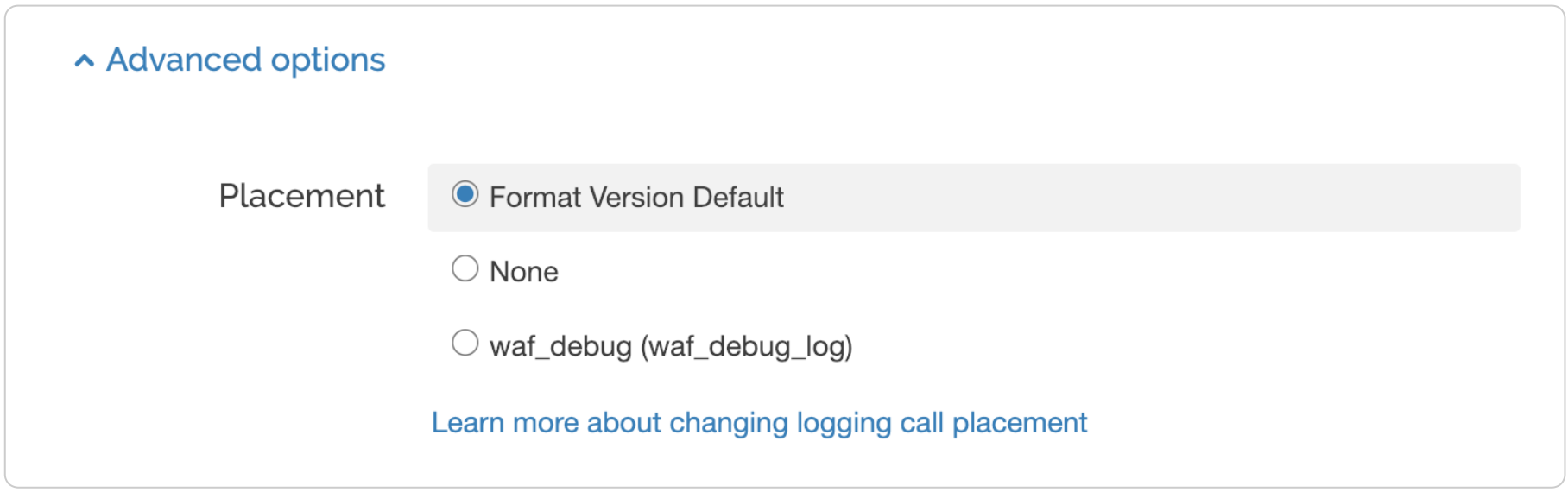
- Sign up for a Honeycomb account if you don't already have one.
- Obtain the Write Key for your team on the Honeycomb Account page.
- Choose a Dataset name. If you plan to collect data from multiple environments (like production, development, staging), Honeycomb recommends creating a Dataset for each 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, type a human-readable name for the endpoint.
   - In the **Log format** field, enter the data to send to Honeycomb. See the example format section for details.
   - In the **Write Key** field, type the write key for your Honeycomb team. This is available on the Honeycomb Account page.
   - In the **Dataset** field, type the name of the Honeycomb Dataset (e.g., `myDataset`).
4. Click the **Advanced options** link of the **Create a Honeycomb endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. 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":"%{Fastly-Orig-Host}i",
    "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: Kafka

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.

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

NOTE: This logging endpoint is disabled by default. To enable this endpoint for your account, contact support@fastly.com and request it.

NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

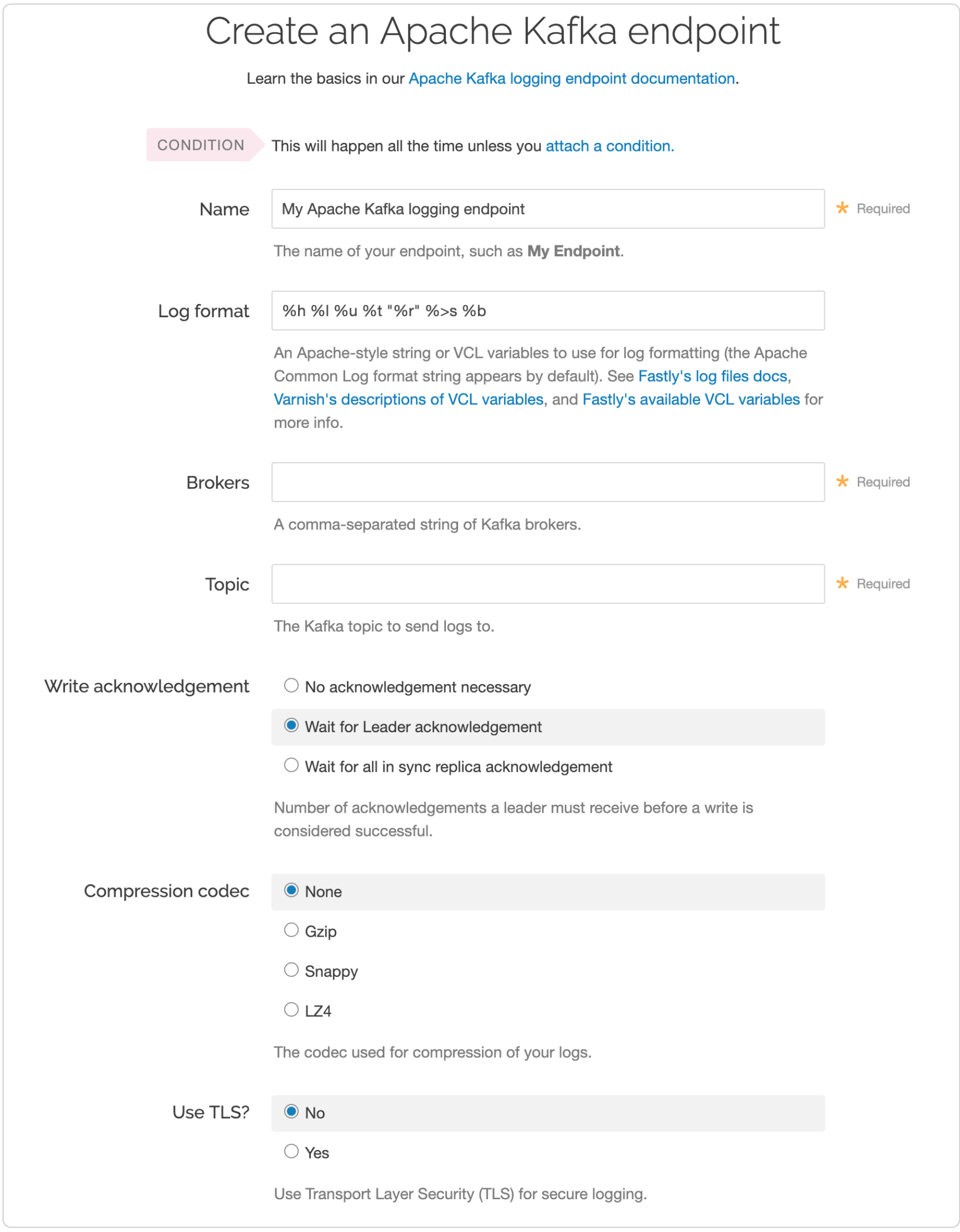
Prerequisites

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:

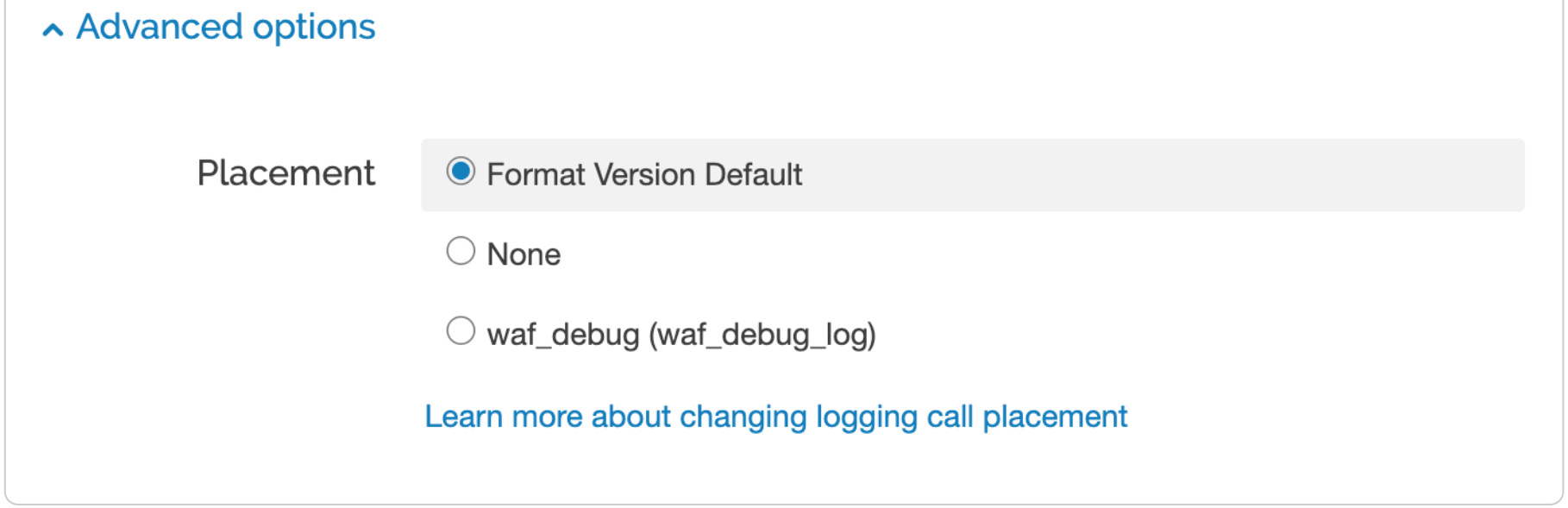
1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Apache Kafka **Create endpoint** button. The Create an Apache Kafka endpoint page appears.



3. Fill out the **Create an Apache Kafka endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings for more information.
   - In the **Brokers** field, type the hostname or IP address of one or more servers (Kafka brokers). Specify multiple servers using a comma-separated string.
   - In the **Topic** field, type the name of the topic to send logs to.
   - In the **Write acknowledgement** area, select the appropriate write acknowledgement a leader must receive before a write is considered successful.
   - In the **Compression codec** area, select the appropriate codec to use for compression of your logs.

- From the **TLS** menu, select **No** to disable encryption for the Kafka endpoint, or **Yes** to enable it. When you select Yes, additional TLS fields appear.
- In the **TLS Hostname** field, optionally type the hostname used to verify the server's certificate. This can be either the Common Name (CN) or Subject Alternate 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.

4. Click the **Advanced options** link of the **Create an Apache Kafka endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: Log Shuttle

https://docs.fastly.com/en/guides/log-streaming-log-shuttle

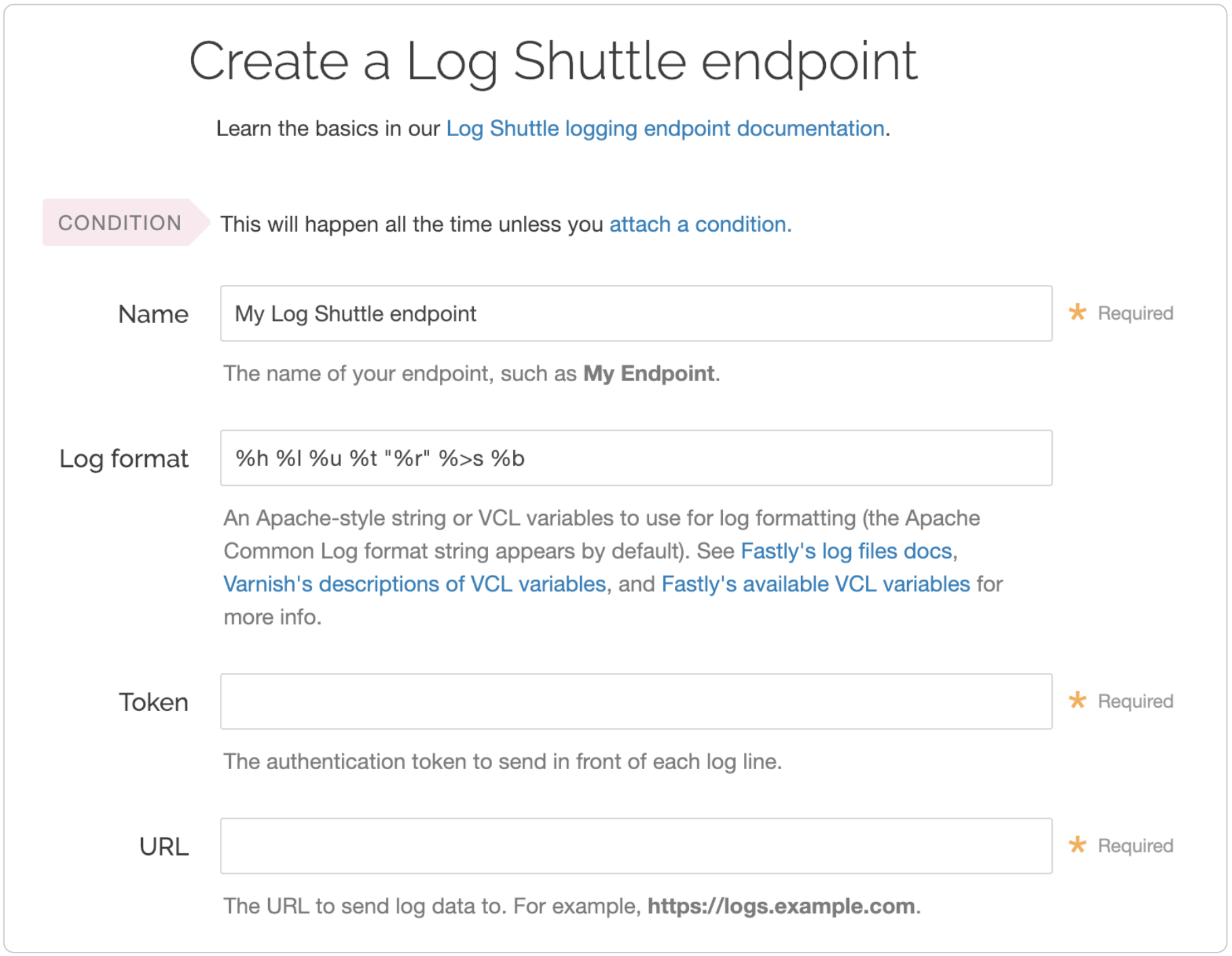
Fastly's Real-Time Log Streaming feature can send log files to Log Shuttle. Log Shuttle is an open source application designed to provide simpler encrypted and authenticated log delivery.

NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

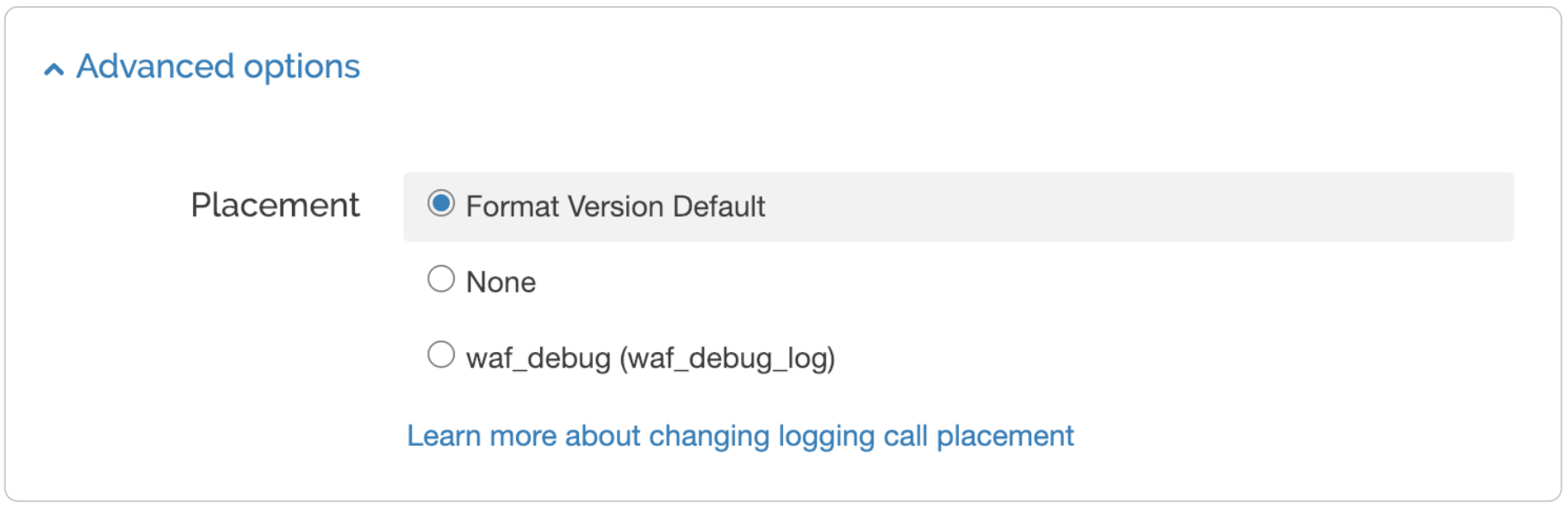
Adding Log Shuttle as a logging endpoint

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, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
   - In the **Token** field, type the data authentication token. This is required for some endpoints like Heroku's Log Integration.
   - In the **URL** field, type the URL to which log data will be sent (e.g., `https://logs.example.com/`).
4. Click the **Advanced options** link of the **Create a Log Shuttle endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: LogDNA

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.

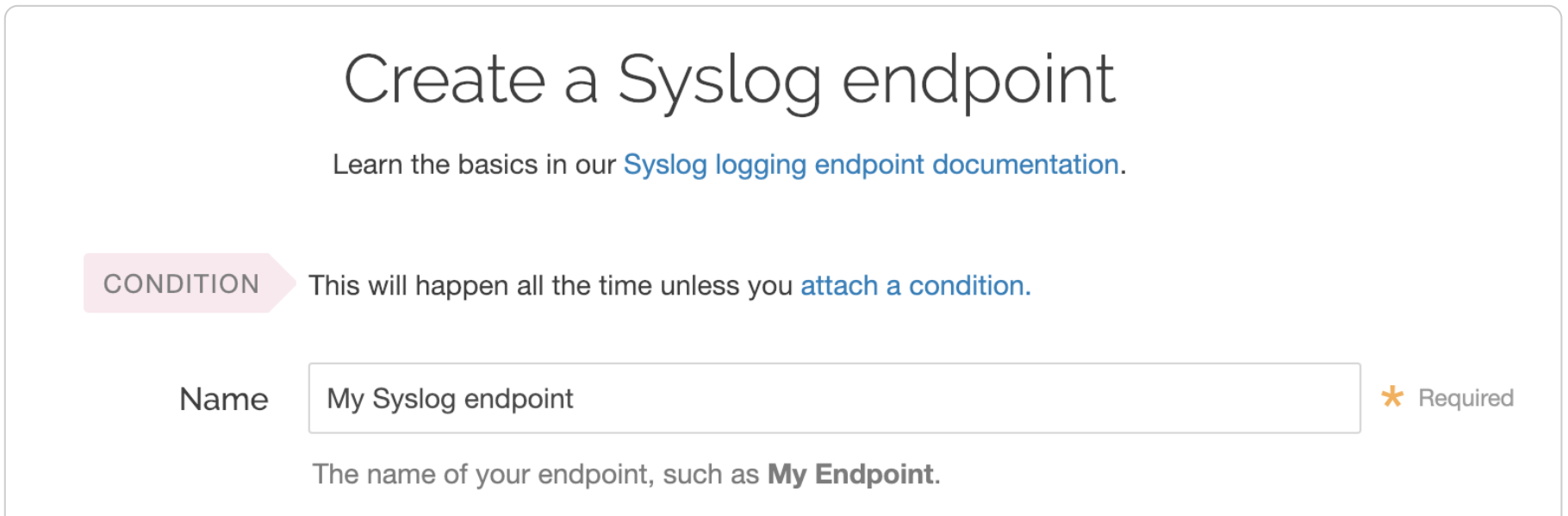
Prerequisites

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.
- Fetch the LogDNA root CA certificate and save it for use during endpoint setup.

Adding LogDNA as a logging endpoint

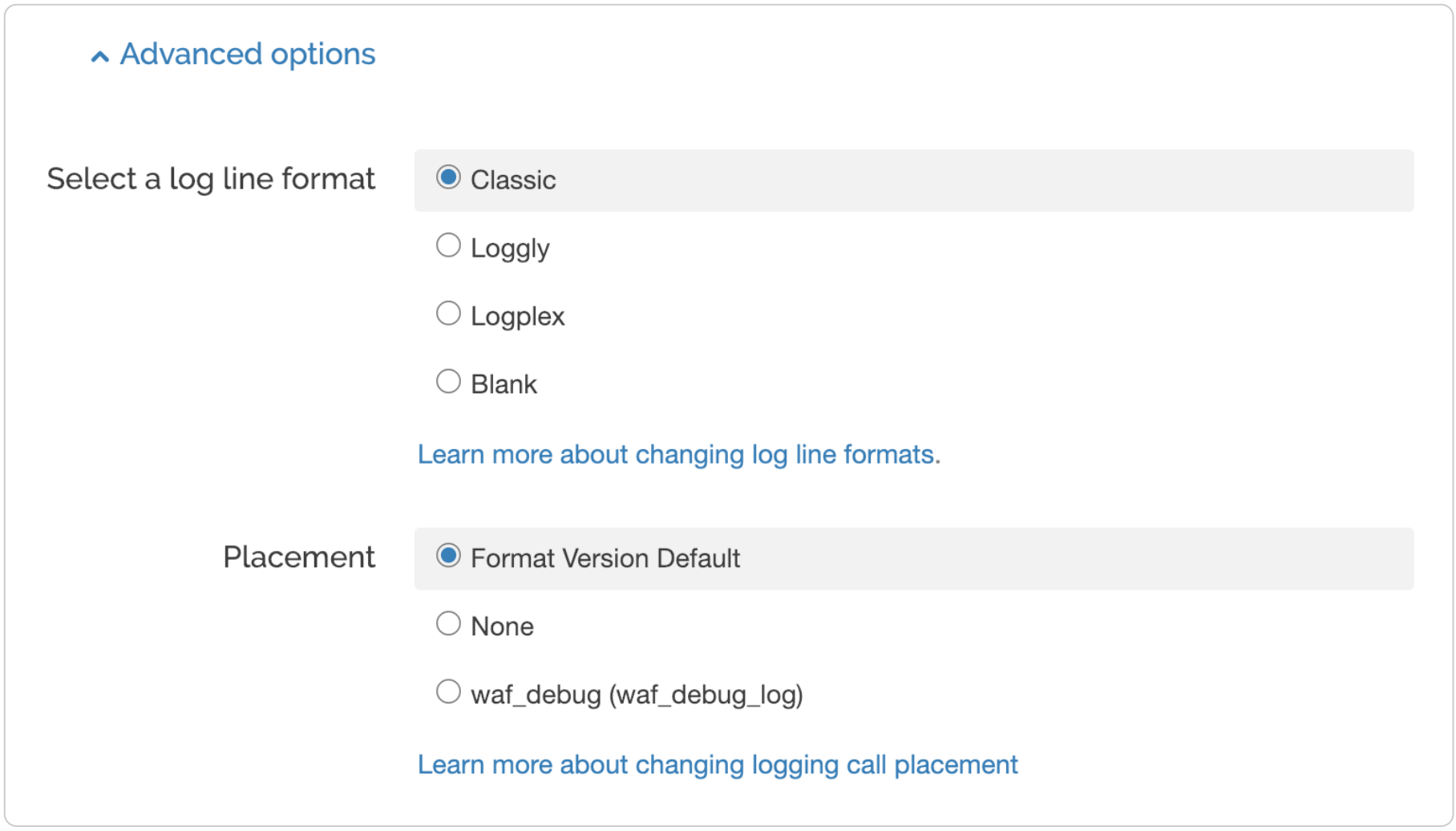
1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the LogDNA (via Syslog) **Create endpoint** button. The Create a Syslog endpoint page appears.



3. Fill out the **Create a Syslog endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings for more information.
   - In the **Syslog address** field, type `syslog-a.logdna.com` in the domain field before the colon, and in the port field after the colon type 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, type `syslog-a.logdna.com`. This is the hostname Fastly will use to verify the syslog server's certificate.
- In the **TLS CA certificate** field, copy and paste the contents of the LogDNA root Certificate file you fetched.

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.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
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.

Log streaming: Logentries

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.

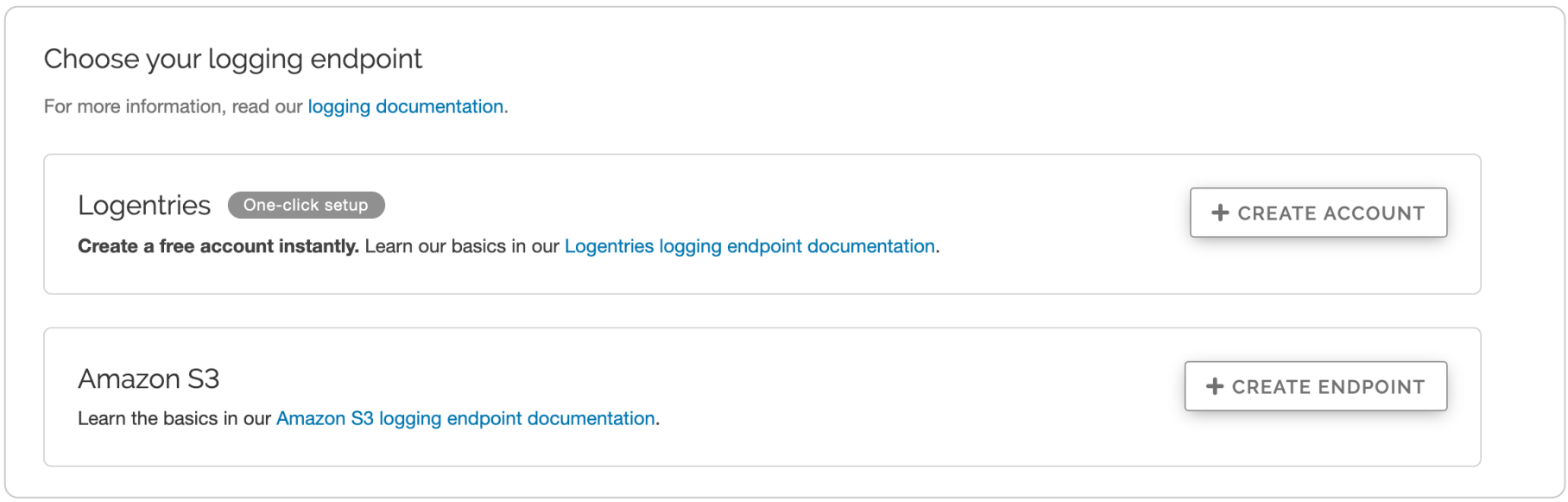
One-click Logentries account setup

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

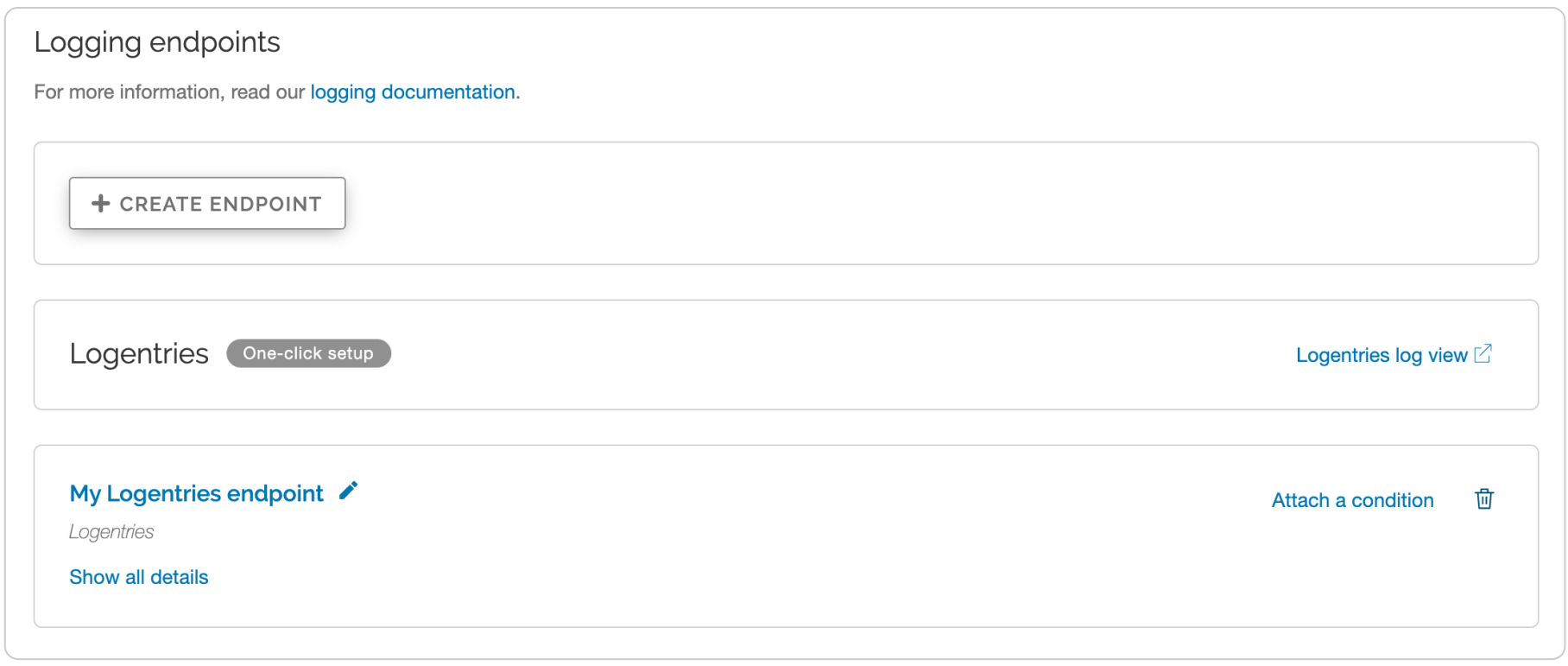


5. In the Logentries One-click setup box, click the **Create Account** button. The Logentries log is automatically created.
6. Click the **Activate** button to deploy your configuration changes.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

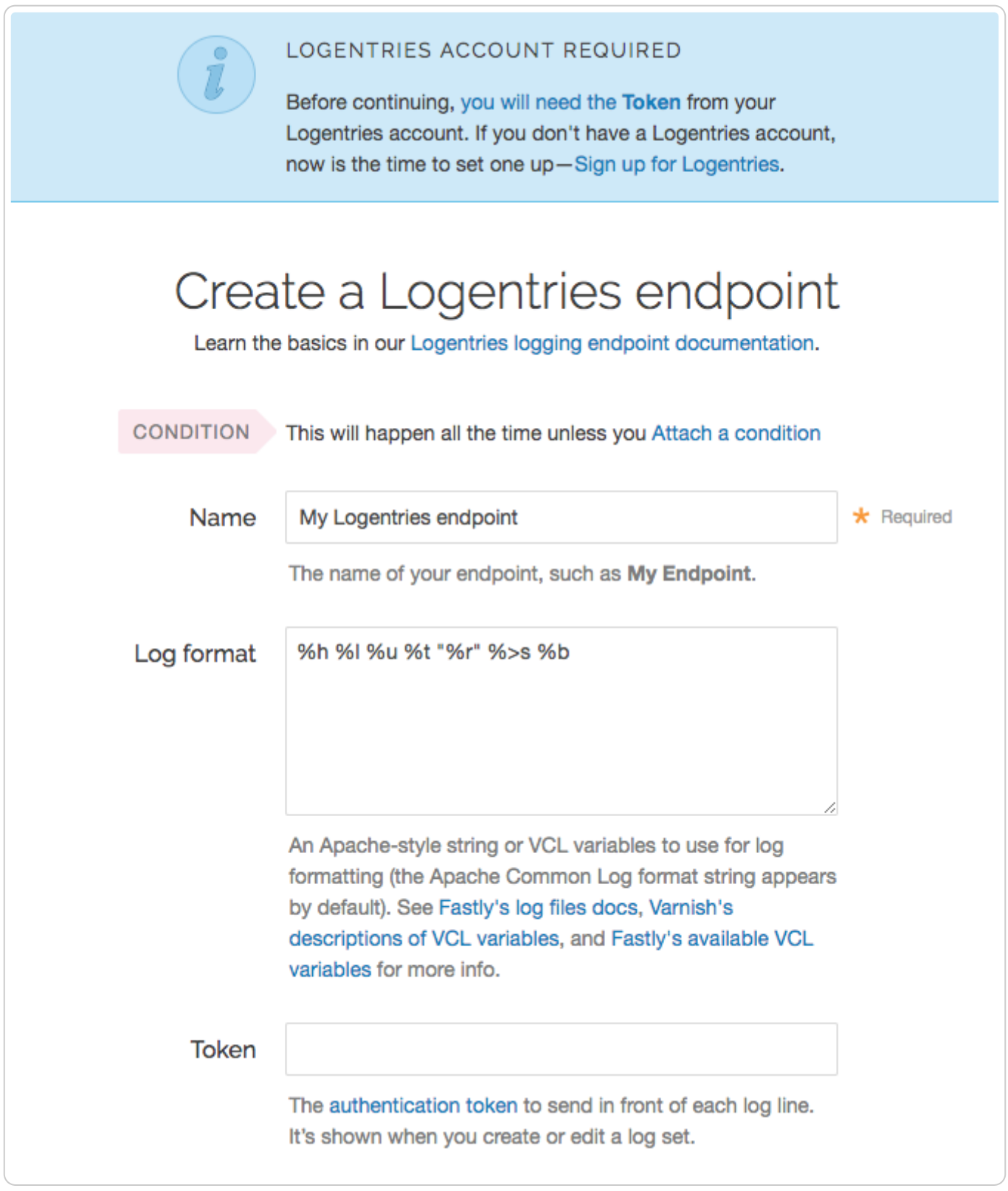
Prerequisites

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.

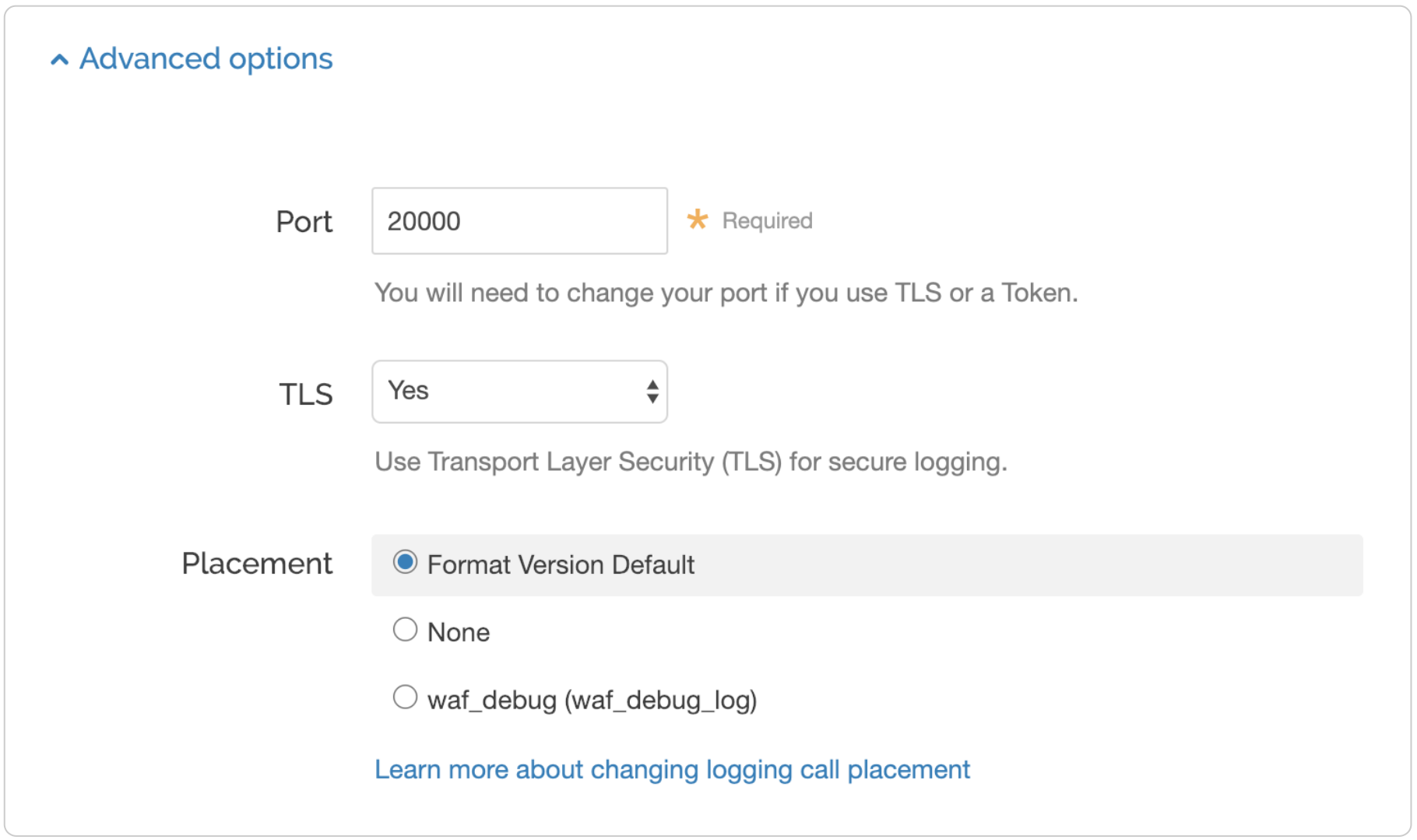
Creating the logging endpoint in the web interface

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, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. Our discussion of format strings also provides more information.
   - In the **Token** field, type the token provided in the Logentries configuration panel. Though you can use the provided secret port number, there are additional options to consider when deciding on token settings.
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:
   - In the **Port** field, type `20000`. Though we recommend this specific setting when adding your endpoint, there are additional options to consider when deciding on the port and TLS settings.
   - From the **TLS** menu, optionally select **Yes**.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Additional selections for tokens, ports, and TLS

Using your token. You can add a Logentries endpoint by using your secret account token. To use your token, set your port to `10000`. However, we strongly recommend sending your logs via TLS. To do this, set TLS to `Yes` and the port number to `20000`. See the Logentries guide Token TCP for more information.

Using your port number. You can add a Logentries endpoint by using your secret Logentries port number (e.g., `56789`). However, we strongly recommend sending your logs via TLS. To do this, set TLS to `Yes` and add `10000` to your secret port number (e.g., `66789`). See the Logentries guide Plain TCP/UDP for more information.

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

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.

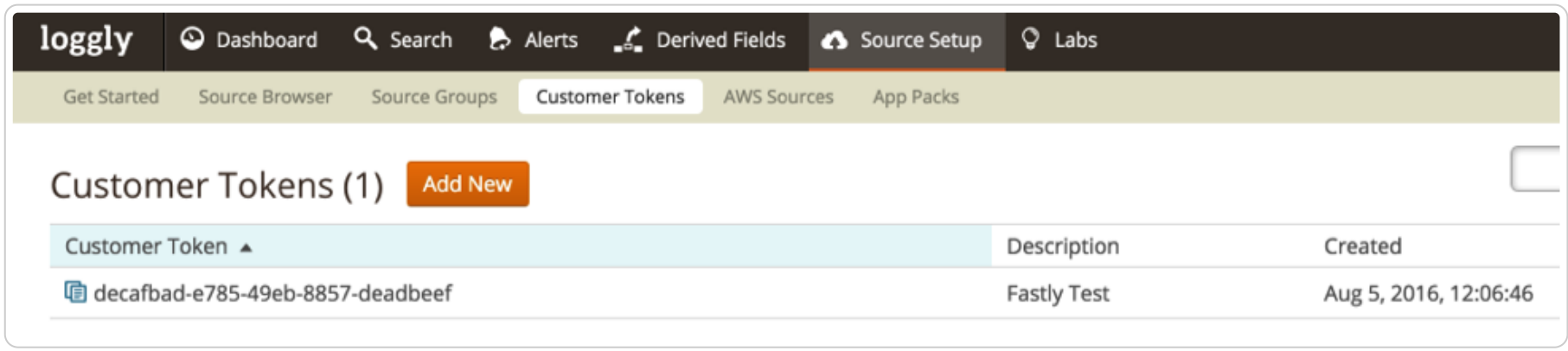
NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

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.

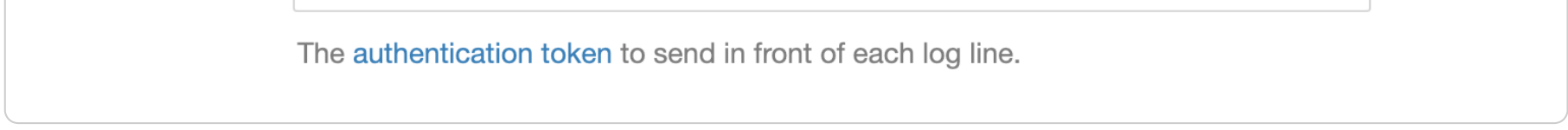


2. Make note of your Loggly customer token. Loggly uses this to associate data you send them with your account.

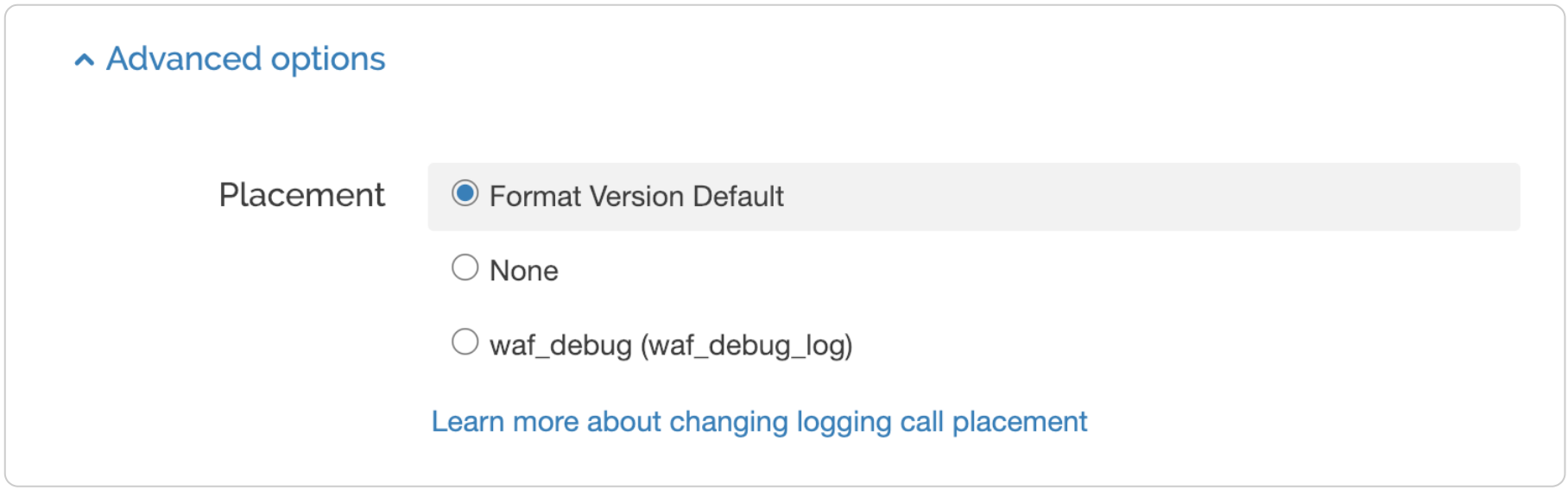
Adding Loggly as a logging endpoint

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, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. Our discussion of format strings provides more information.
   - In the **Token** field, type your Loggly customer token.
4. Click the **Advanced options** link of the **Create a Loggly endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: Heroku's Logplex

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

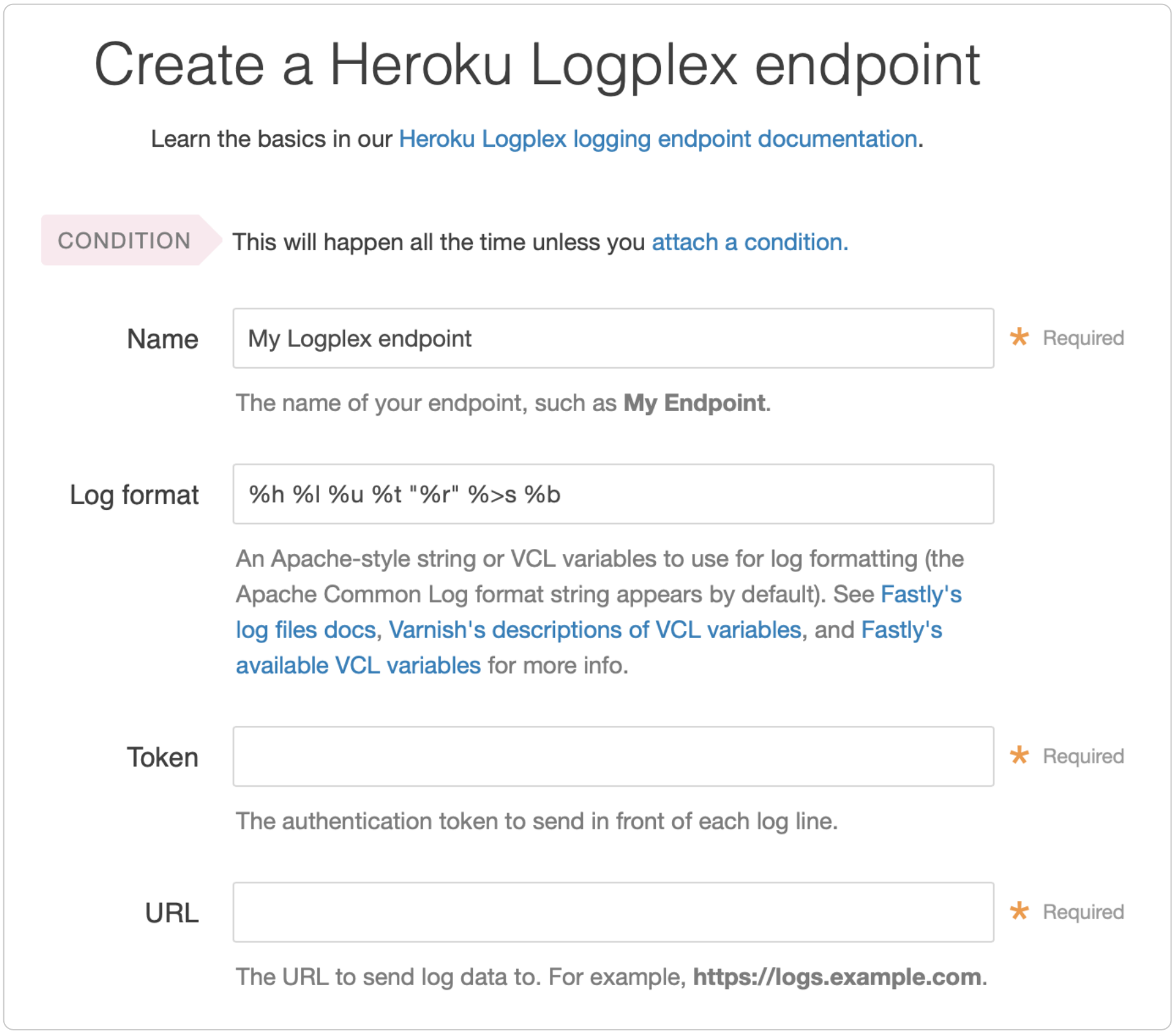
Before continuing, you will need the token from your Heroku Logplex account. If you don't have a Heroku Logplex account, now is the time to set one up by signing up for Heroku.

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.

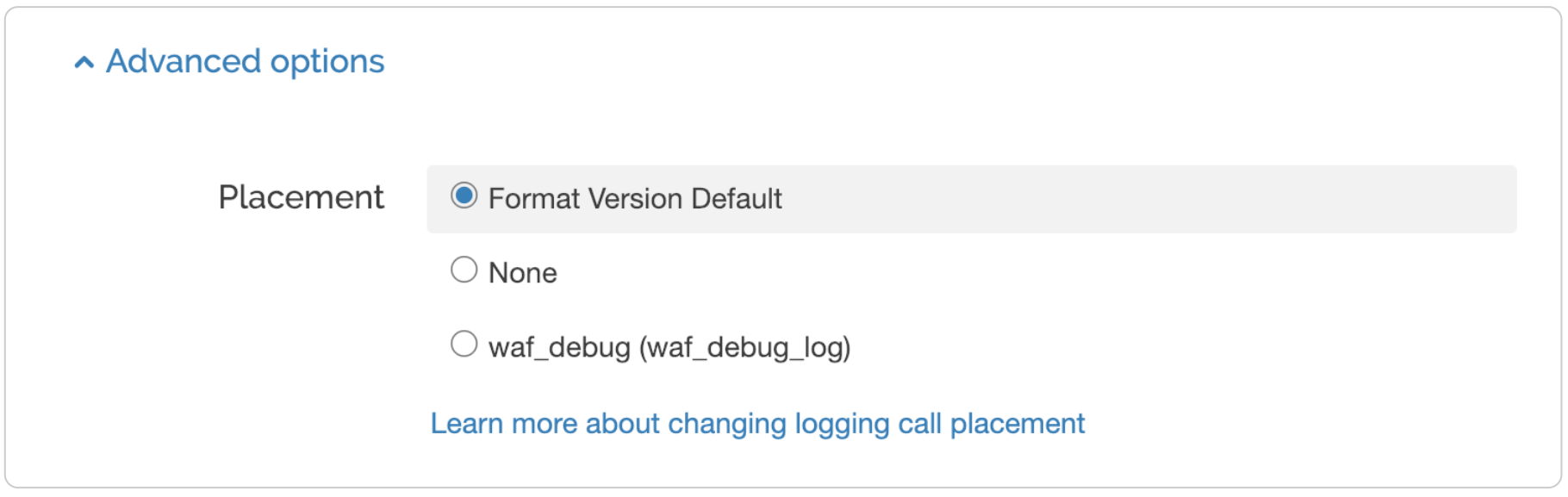
Adding Heroku Logplex as a logging endpoint

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, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. Our discussion of format strings also provides more information.
   - In the **Token** field, type your Heroku Logplex token.
   - In the **URL** field, type `https://1.us.logplex.io/logs` unless otherwise instructed by our support staff.
4. Click the **Advanced options** link of the **Create a Heroku Logplex endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: OpenStack

https://docs.fastly.com/en/guides/log-streaming-openstack

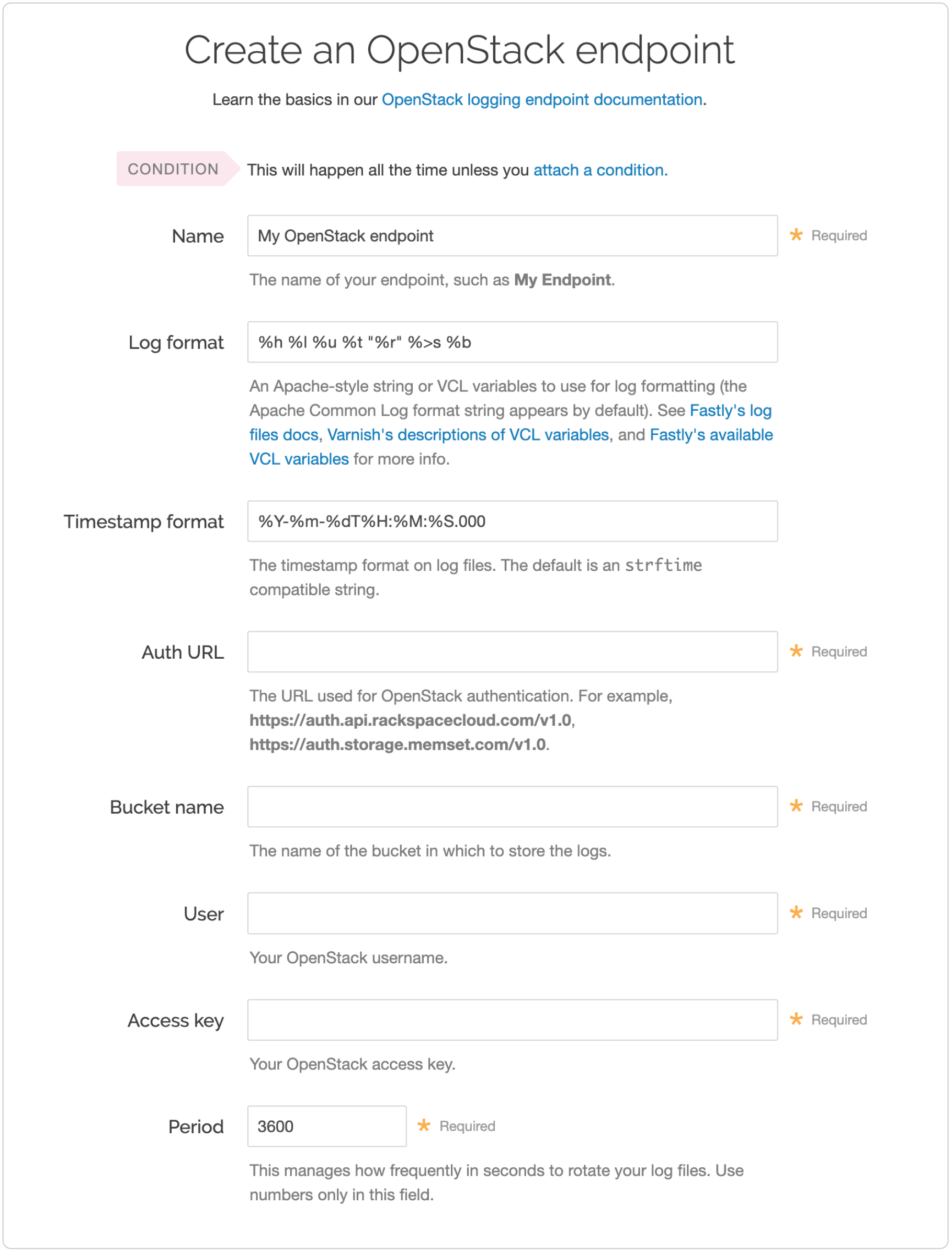
Fastly's Real-Time Log Streaming feature can send log files to OpenStack. OpenStack is an open-source platform for cloud-computing that many companies deploy as an infrastructure-as-a-service.

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:

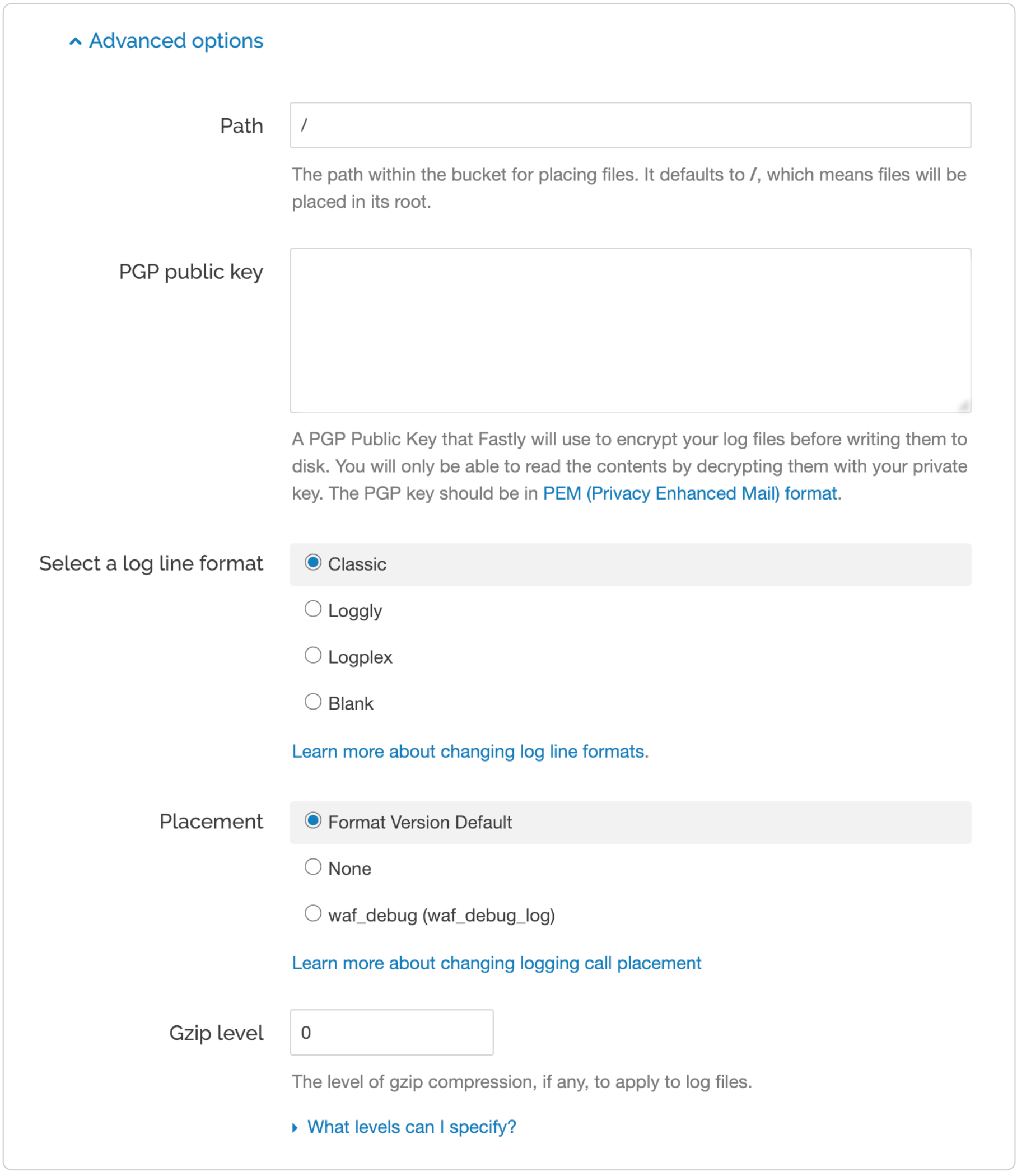
1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the OpenStack **Create endpoint** button. The Create an OpenStack endpoint page appears.



3. Fill out the **Create an OpenStack endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings for more information.
   - In the **Timestamp format** field, optionally type 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, type the URL used for OpenStack authentication (e.g., `https://auth.api.rackspacecloud.com/v1.0`).
   - In the **Bucket name** field, type the name of the OpenStack bucket in which to store the logs.
   - In the **User** field, type your OpenStack username.

- In the **Access Key** field, type your OpenStack access key.
- In the **Period** field, type an interval (in seconds) to control how frequently to rotate your log files. 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 type 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 type 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 **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used

in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.

- In the **Gzip Level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).

6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: Papertrail

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

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.

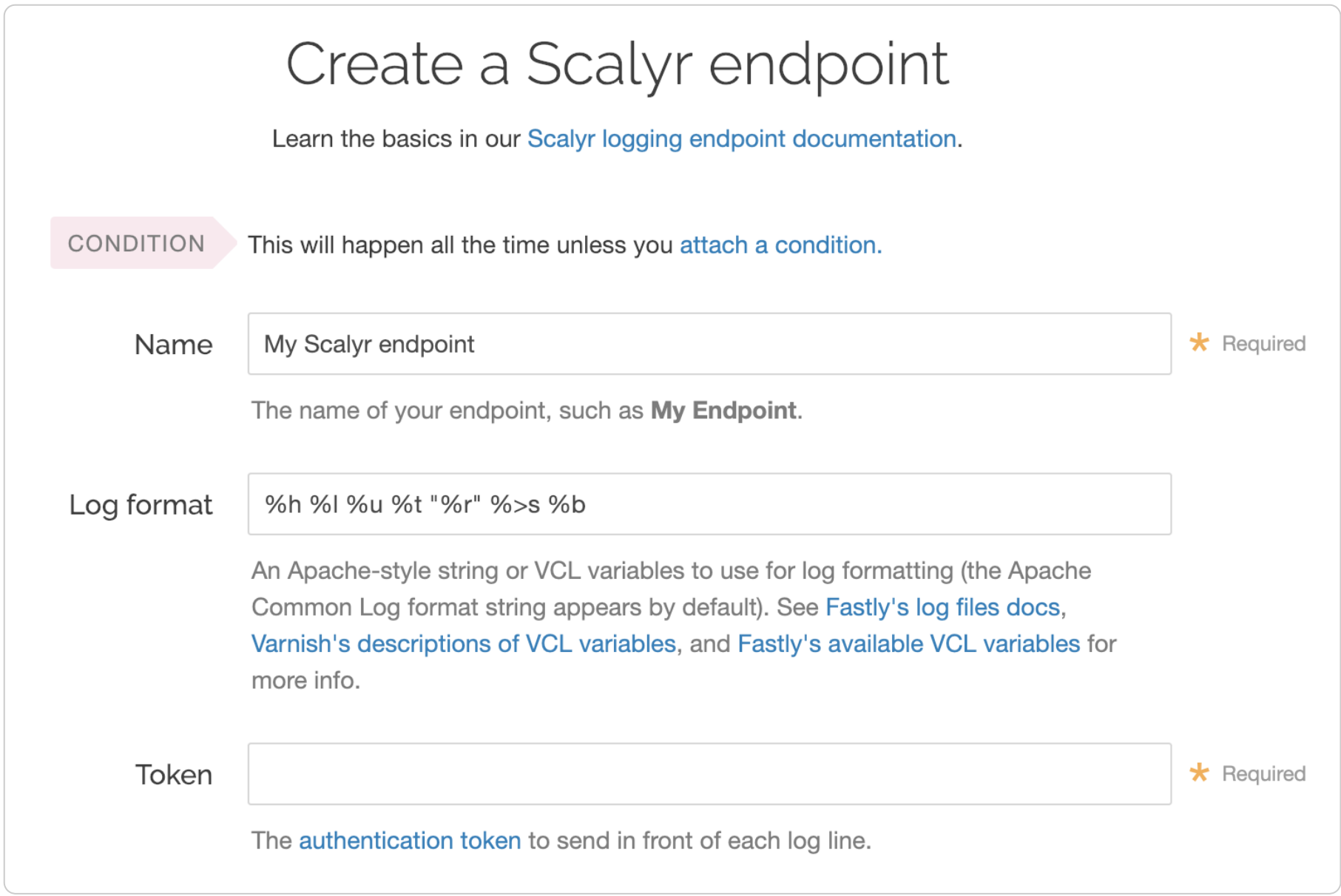
Once you've signed up, navigate to the **API Keys** area in the **Settings** on your Scalyr dashboard and make note of your Scalyr Write Token. Scaylr uses this to associate data you send them with your account. You'll need this token when you set up your endpoint with Fastly.

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.

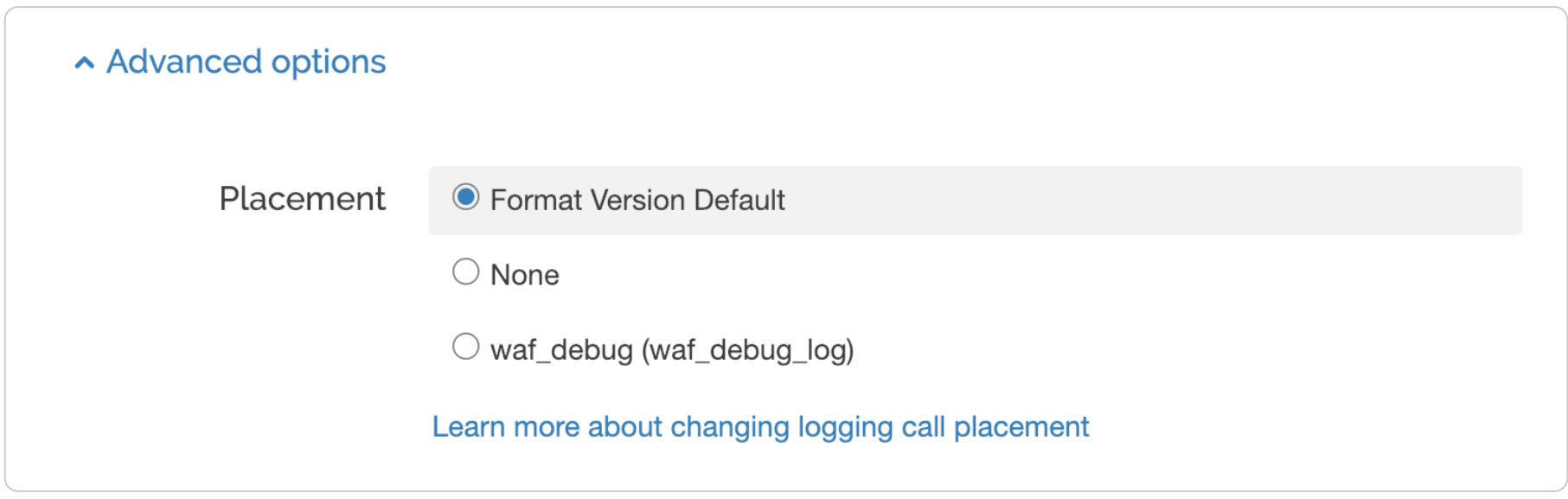
Adding Scalyr as a logging endpoint

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, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings for more information.
   - In the **Token** field, type the Scalyr Write Token provided in the Scalyr dashboard.
4. Click the **Advanced options** link of the **Create a Scalyr endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: SFTP

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, type a human-readable name for the endpoint.
- In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default.
- In the **Timestamp format** field, optionally type a timestamp format for log files. The default is an `strftime` compatible string. Our guide on changing where log files are written](/en/guides/changing-where-log-files-are-written) provides more information.
- In the **Address** field, type the hostname or IP address of the SFTP server. In the port field after the colon, type the port number you're using for SFTP (the default is `22`).
- In the **Path** field, type 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.

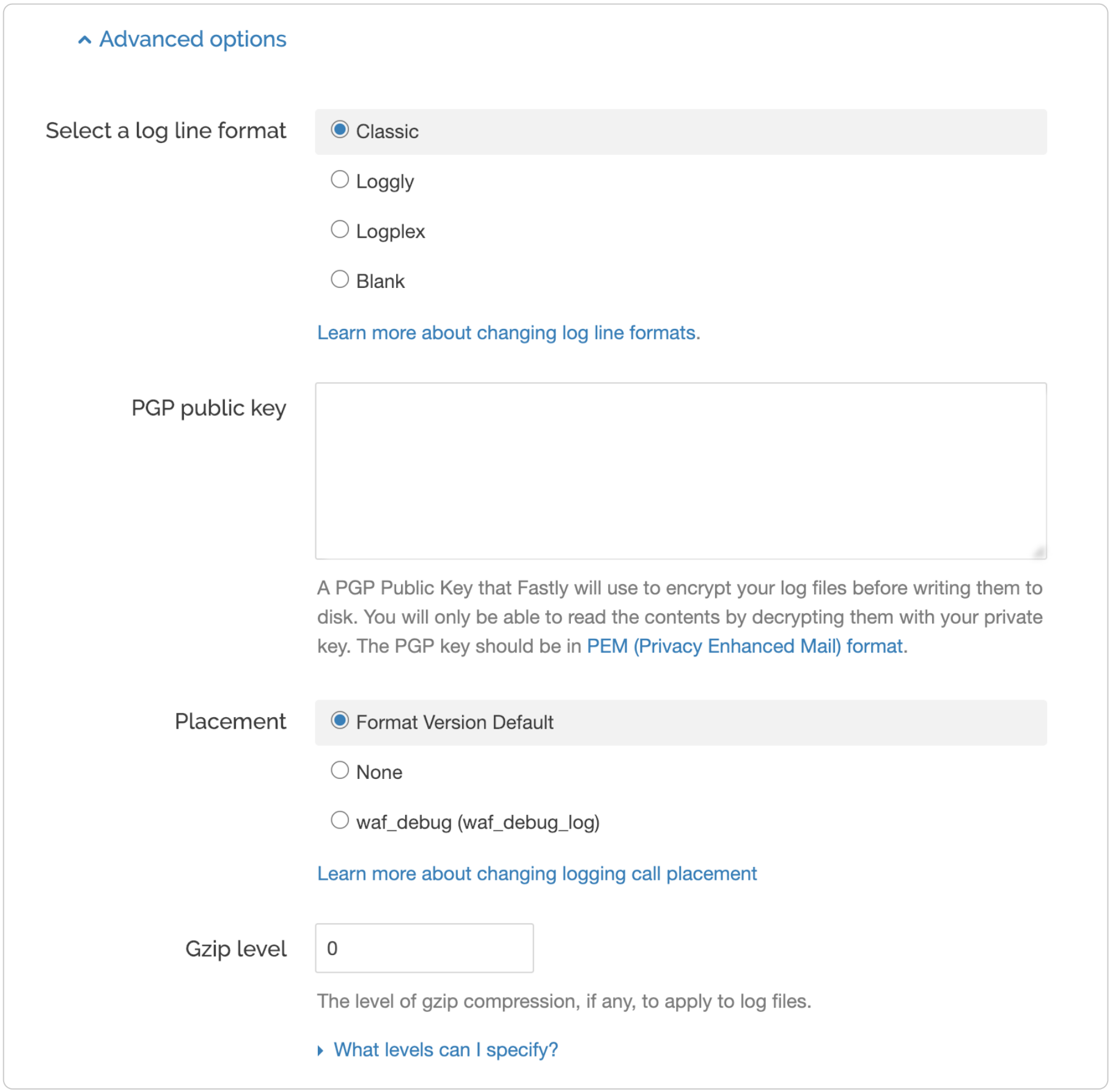
- In the **User** field, type the username used to authenticate to the SFTP server.
- In the **Known hosts** field, type a host key for each host you can connect to over SFTP. Each host key you enter must be on its own line. Known hosts entries should match what's stored in your known_hosts file located 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, type 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, type 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, type 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 type a PGP public key that Fastly will use to encrypt your log files before writing them to disk. You only can read the contents by decrypting them with your private key. The PGP key should be in PEM (Privacy-Enhanced Mail) format. Our guide on log encryption provides more information.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
   - In the **Gzip level** field, optionally type the level of gzip compression you want applied to the log files. You can specify any whole number from `1` (fastest and least compressed) to `9` (slowest and most compressed). This value defaults to `0` (no compression).
6. Click the **Create** button to create the new logging endpoint.
7. Click the **Activate** button to deploy your configuration changes.

Log streaming: Splunk

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.

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.

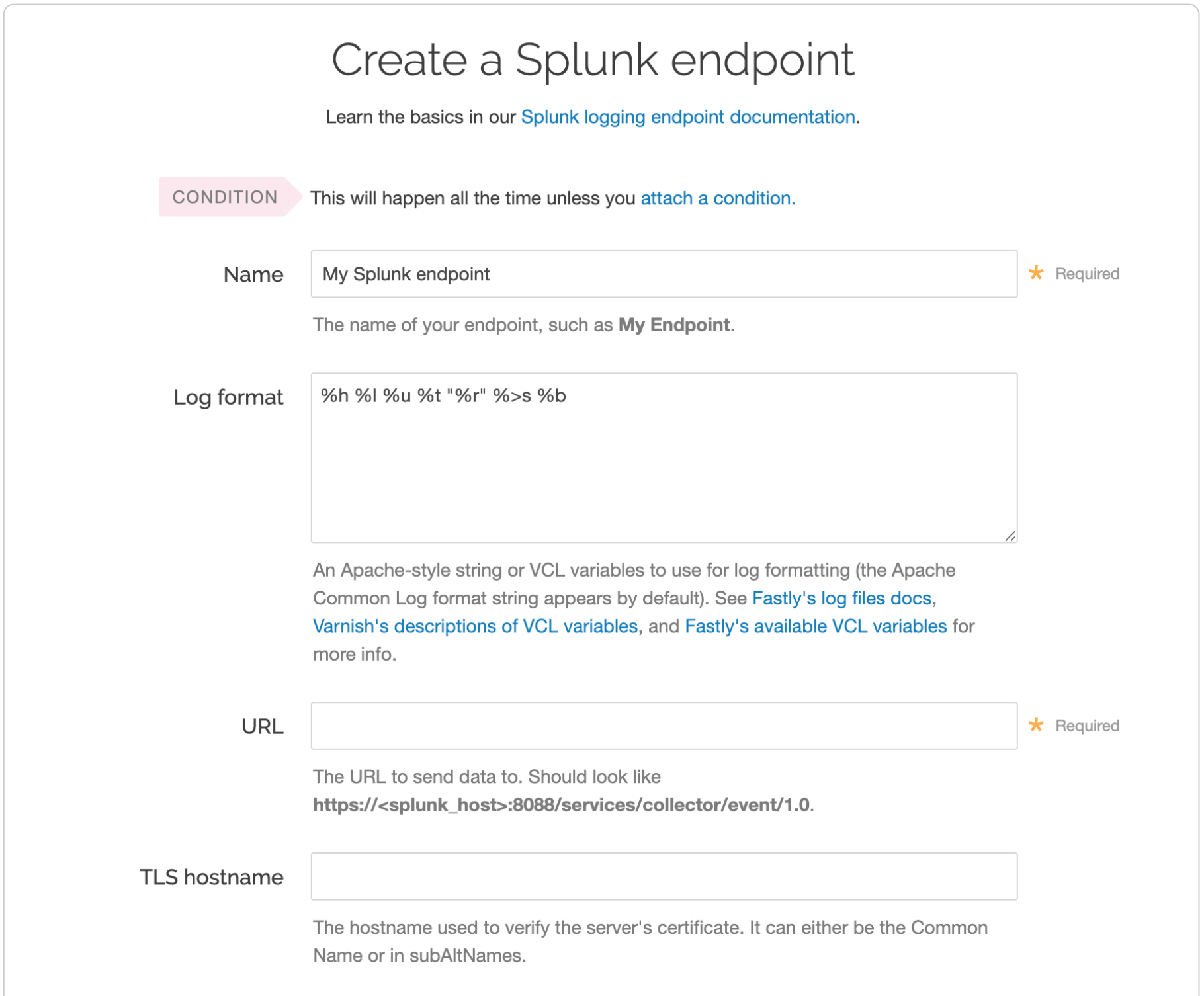
Type	URL
Self hosted	`https://<hostname>:8088/services/collector/event`
Self-service Splunk Cloud plans	`https://input-<hostname>:8088/services/collector/event`
All other Splunk Cloud plans	`https://http-inputs-<hostname>:8088/services/collector/event`

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:

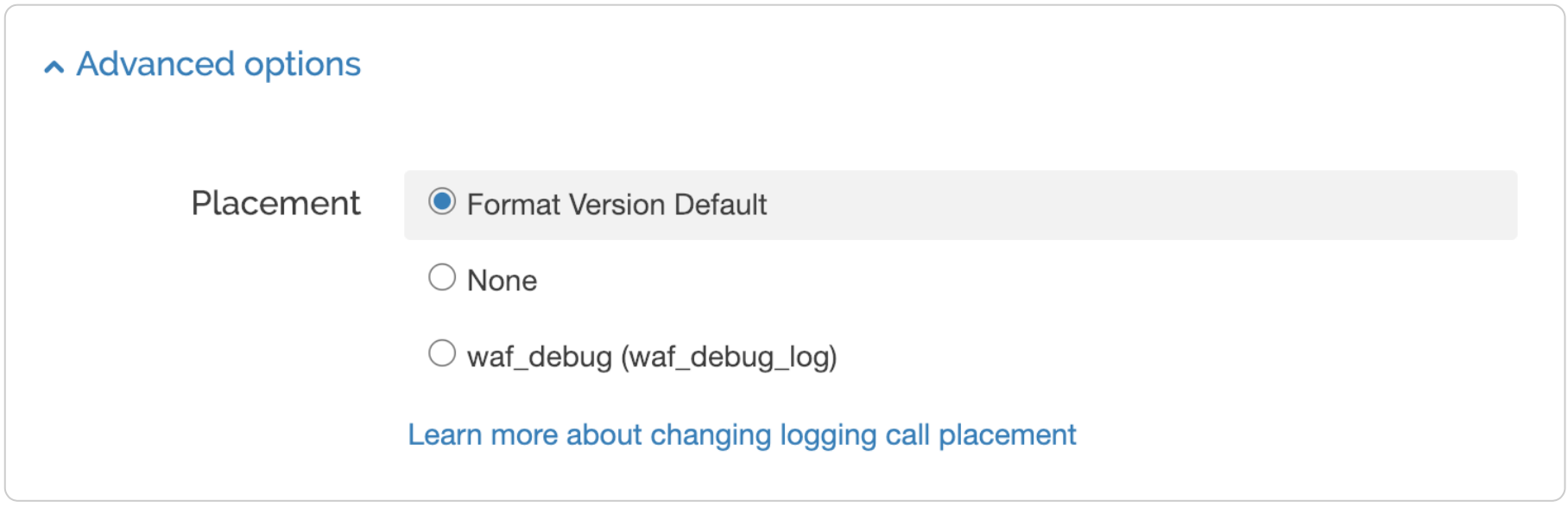
1. Review the information in our Setting Up Remote Log Streaming guide.
2. Click the Splunk **Create endpoint** button. The Create a Splunk endpoint page appears.



3. Fill out the **Create a Splunk endpoint** fields as follows:
   - In the **Name** field, type a human-readable name for the endpoint.
   - In the **Log format** field, type an Apache-style string or VCL variables to use for log formatting. You can use our recommended log format.
   - In the **URL** field, type the URL to send data to (e.g., `https://<splunk_host>:8088/services/collector/event/1.0`).
   - In the **TLS hostname** field, type the hostname used to verify the server's certificate. If you're using Splunk Enterprise, type `SplunkServerDefaultCert`.

- In the **TLS CA certificate** field, type 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 **Token** field, type the token for the HEC.

4. Click the **Advanced options** link of the **Create a Splunk endpoint** page. The Advanced options appear.



5. In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
6. Click the **Create** button to create the new logging endpoint.
7. 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
  }
}
```

```
4
7
4
8
4
9
5
0
5
1
5
2
5
3
5
4
5
5
5
6
5
7
5
8
5
9
6
0
6
1
6
2
6
3
6
4
6
5
6
6
6
7
6
8
6
9
7
0
7
1
7
2
7
3
```

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

In the Fastly web interface, type `SplunkServerDefaultCert` in the **TLS hostname** field.

For Splunk Enterprise, the default set up has the following CA certificate.

> **NOTE:** These settings work for a default Splunk Enterprise installation, but we encourage you to install your own TLS Certificate.

```
-----BEGIN CERTIFICATE-----
MIIDejCCAmICCQCNHBN8tj/FwzANBgkqhkiG9w0BAQsFADB/MQswCQYDVQQGEwJV
UzELMAkGA1UECAwCQ0ExFjAUBgNVBAcMDVNhbiBGcmFuY2lzY28xDzANBgNVBAoM
BlNwbHVuazEXMBUGA1UEAwwOU3BsdW5rQ29tbW9uQ0ExITAfBgkqhkiG9w0BCQEW
EnN1cHBvcnRAc3BsdW5rLmNvbTAeFw0xNzAxMzAyMDI2NTRaFw0yNzAxMjgyMDI2
NTRaMH8xCzAJBgNVBAYTAlVTMQswCQYDVQQIDAJDQTEWMBQGA1UEBwwNU2FuIEZy
YW5jaXNjbzEPMA0GA1UECgwGU3BsdW5rMRcwFQYDVQQDDA5TcGx1bmtDb21tb25D
QTEhMB8GCSqGSIb3DQEJARYSc3VwcG9ydEBzcGx1bmsuY29tMIIBIjANBgkqhkiG
9w0BAQEFAAOCAQ8AMIIBCgKCAQEAzB9ltVEGk73QvPlxXtA0qMW/SLDQlQMFJ/C/
tXRVJdQsmcW4WsaETteeWZh8AgozO1LqOa3I6UmrWLcv4LmUAh/T3iZWXzHLIqFN
WLSVU+2g0Xkn43xSgQEPSvEK1NqZRZv1SWvx3+oGHgu03AZrqTj0HyLujqUDARFX
sRvBPW/VfDkomHj9b8IuK3qOUwQtIOUr+oKx1tM1J7VNN5NflLw9NdHtlfblw0Ys
5xI5Qxu3rcCxkKQuwz9KRe4iijOIRMAKX28pbakxU9Nk38Ac3PNadgIk0s7R829k
980sqGWkd06+C17OxgjpQbvLOR20FtmQybttUsXGR7Bp07YStwIDAQABMA0GCSqG
SIb3DQEBCwUAA4IBAQCxhQd6KXP2VzK2cwAqdK74bGwl5WnvsyqdPWkdANiKksr4
ZybJZNfdfRso3fA2oK1R8i5Ca8LK3V/UuAsXvG6/ikJtWsJ9jf+eYLou8lS6NVJO
xDN/gxPcHrhToGqi1wfPwDQrNVofZcuQNklcdgZ1+XVuotfTCOXHrRoNmZX+HgkY
gEtPG+r1VwSFowfYqyFXQ5CUeRa3JB7/ObF15WfGUYplbd3wQz/M3PLNKLvz5a1z
LMNXDwN5Pvyb2epyO8LPJu4dGTB4jOGpYLUjG1UUqJo9Oa6D99rv6sId+8qjERtl
ZZc1oaC0PKSzBmq+TpbR27B8Zra3gpoA+gavdRZj
-----END CERTIFICATE-----
```

Log streaming: Sumo Logic

https://docs.fastly.com/en/guides/log-streaming-sumologic

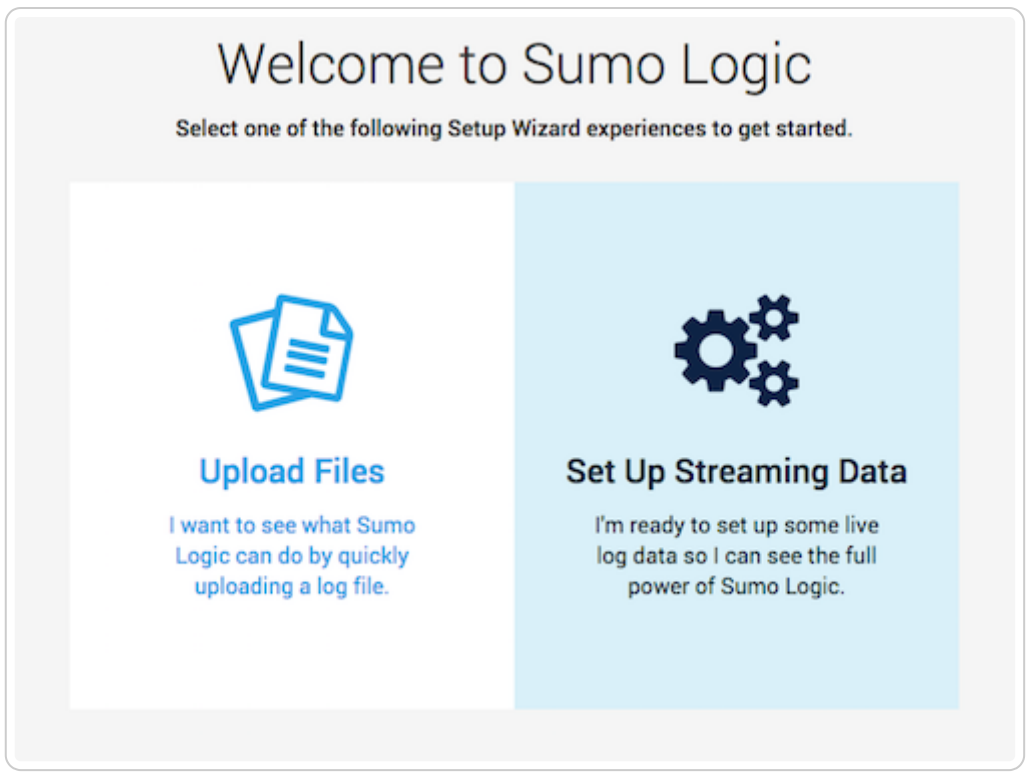
Fastly's Real-Time Log Streaming feature can send log files to Sumo Logic. Sumo Logic is a web-based log analytics platform used by developers and IT teams.

> **NOTE:** Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

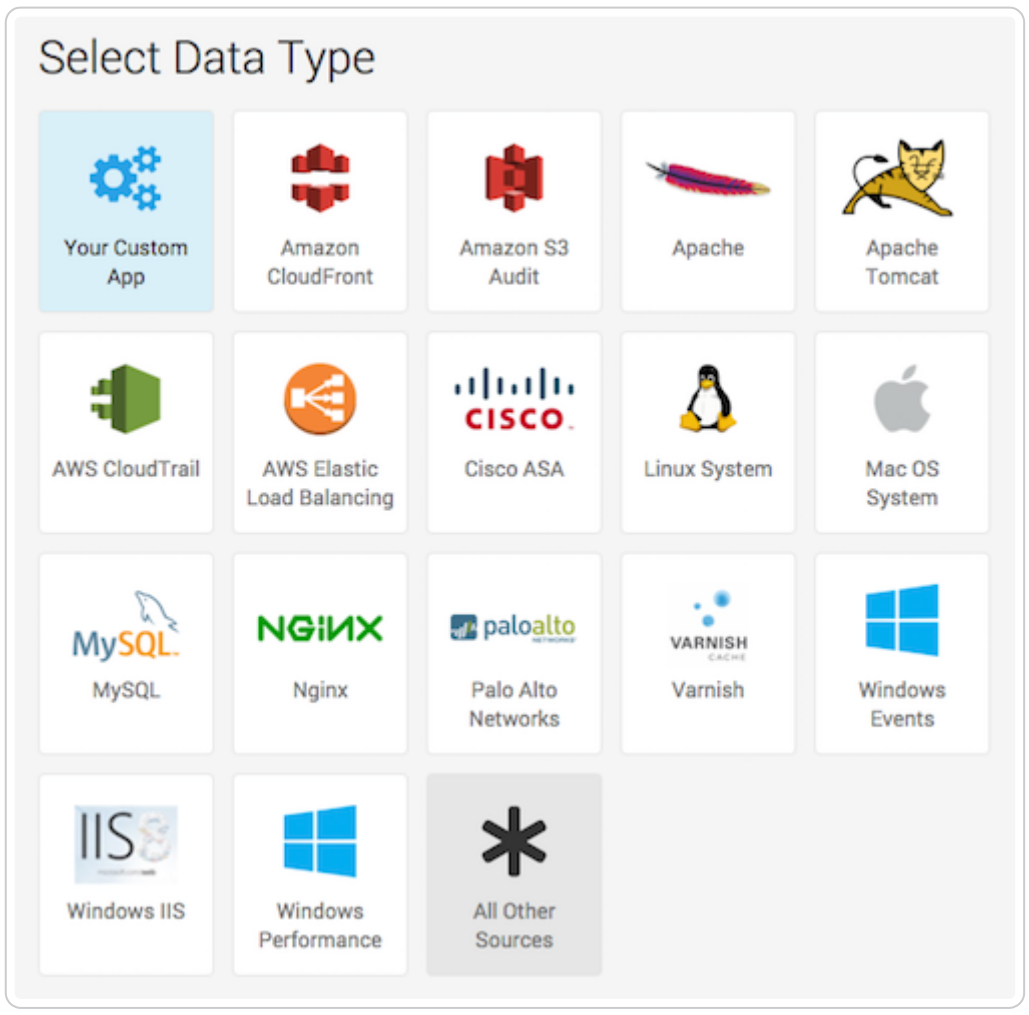
Setting up Sumo Logic

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:

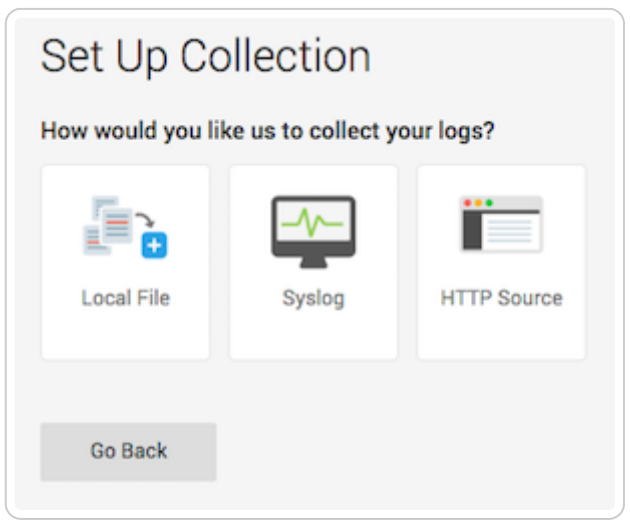
1. The process starts with the Sumo Logic Setup Wizard, which appears immediately after you create your Sumo Logic account. If you already have an account, you can access the wizard by selecting **Setup Wizard** from the **Manage** menu at the top of the Sumo Logic application.



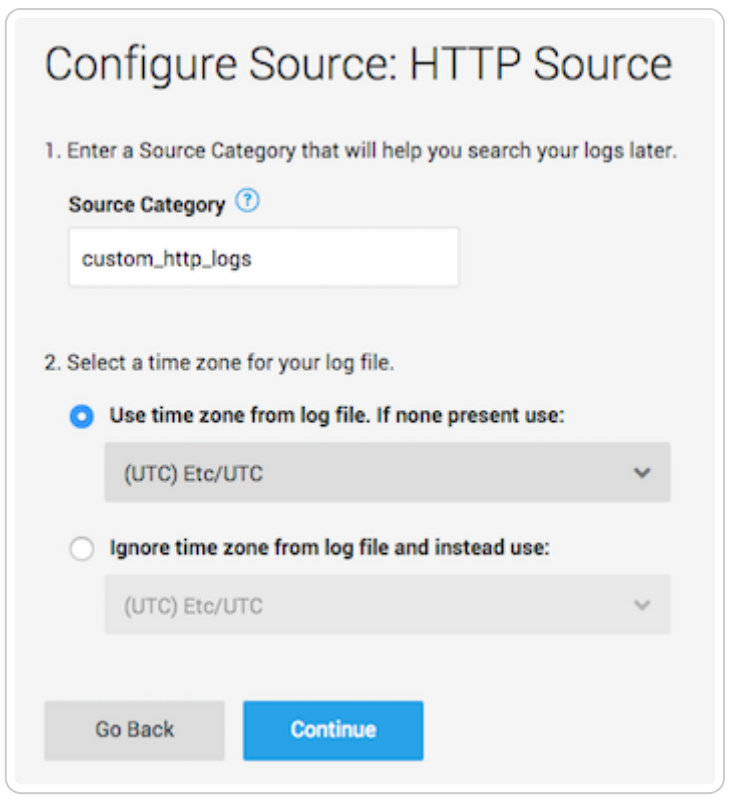
2. Click **Set Up Streaming Data**. The Select Data Type window appears.



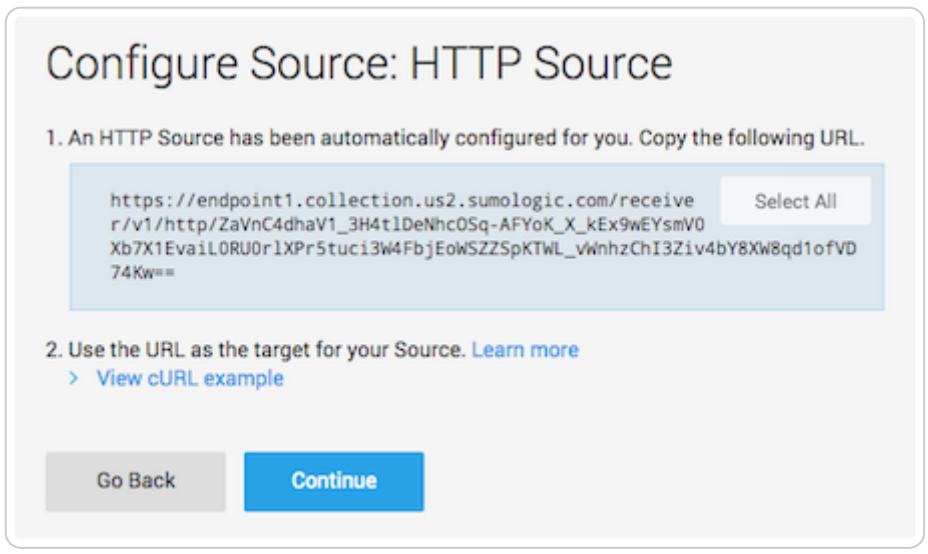
3. Click **All Other Sources**. The Set Up Collection window appears.



4. Click **HTTP Source**. The Configure Source: HTTP Source window appears.



5. In the **Source Category** field, type 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.

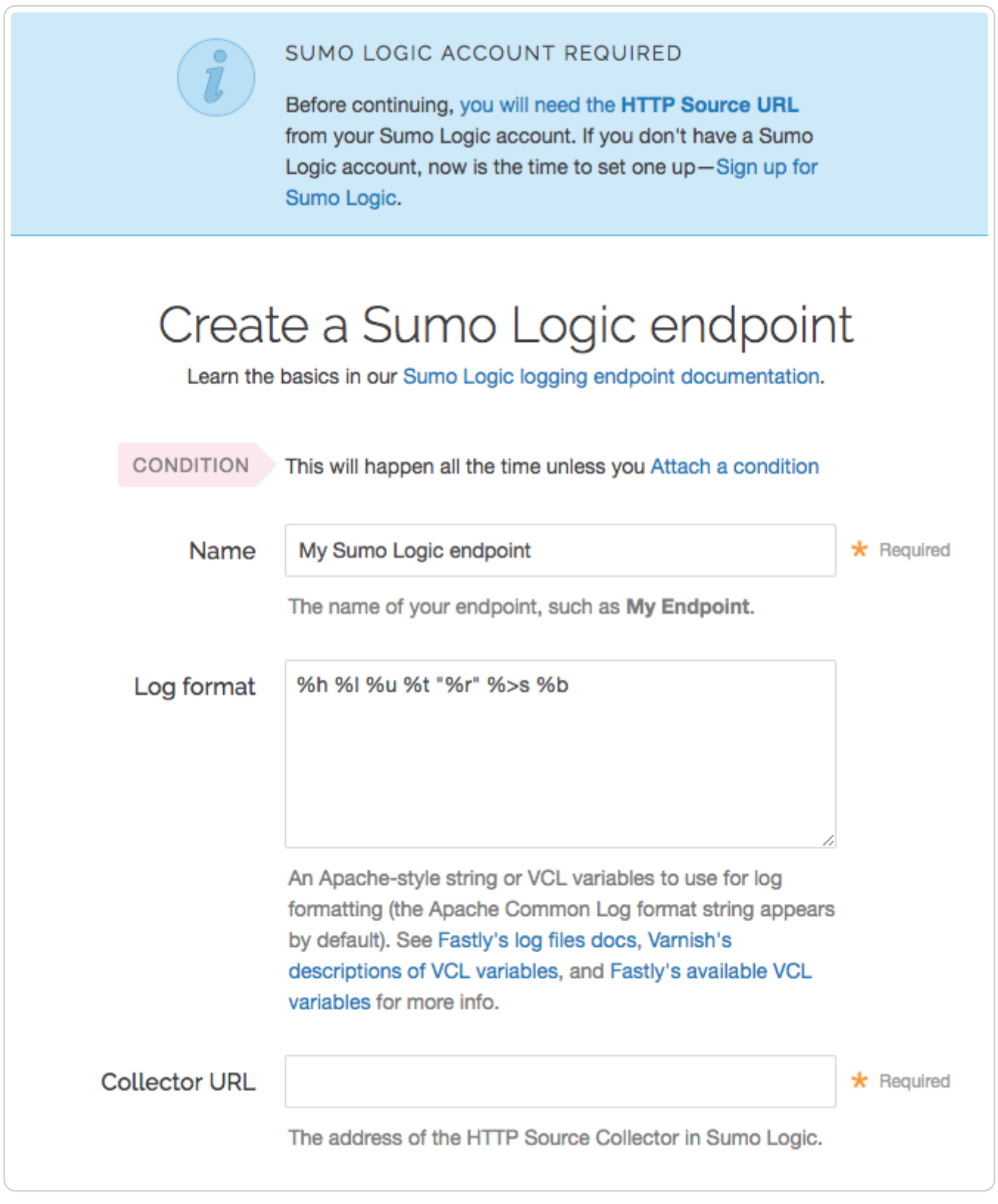


7. Copy the HTTP Source URL. You will enter this value in the Fastly web interface.
8. Click **Continue**. Sumo Logic will add the new source.

Adding Sumo Logic as a logging endpoint

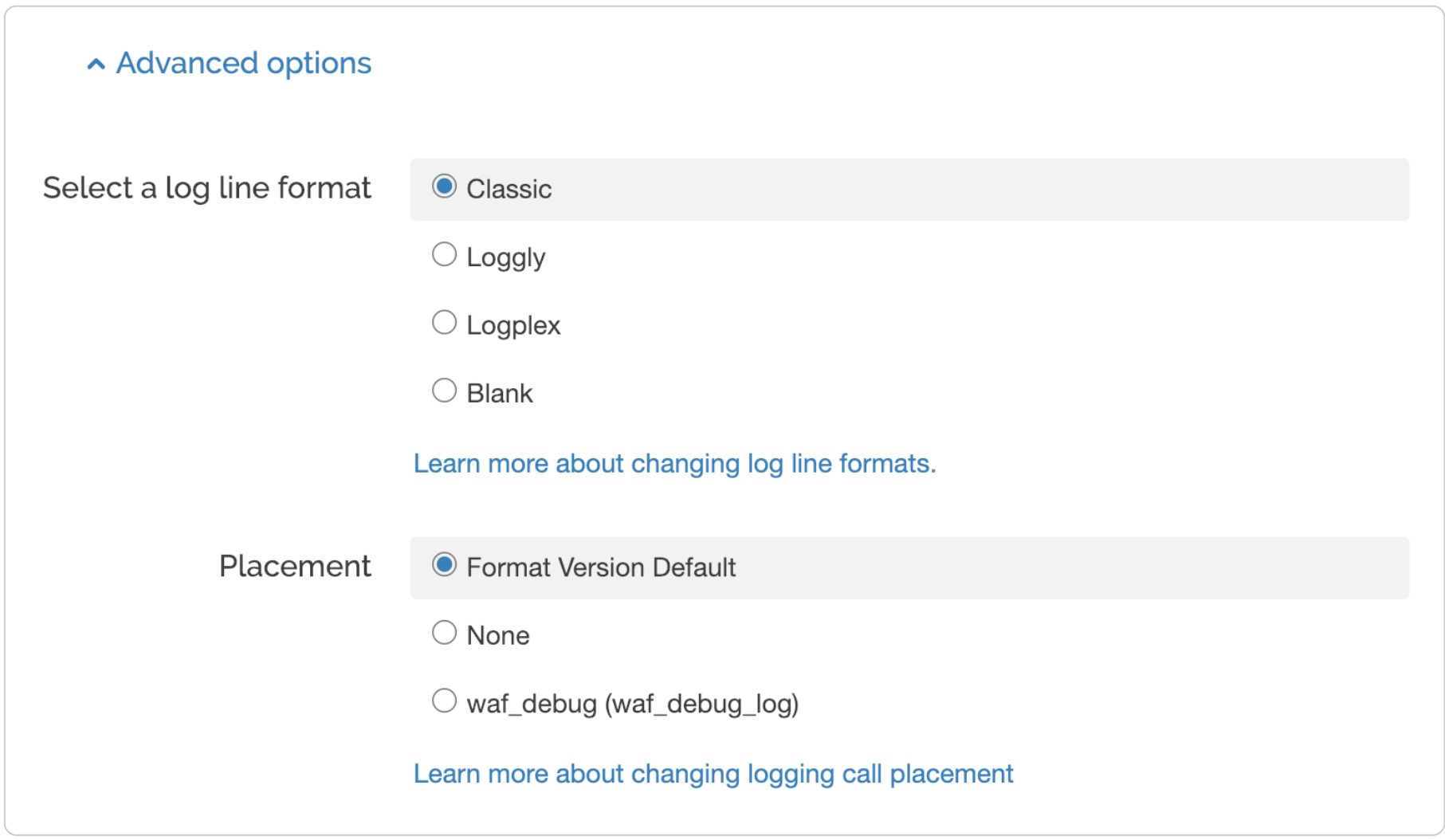
After you've created a Sumo Logic account and obtained the HTTP Source URL, follow these instructions to add Sumo Logic as a logging endpoint for Fastly services:

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, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. Our discussion of format strings provides more information.
   - In the **Collector URL** field, type 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.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
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.

Log streaming: Syslog

https://docs.fastly.com/en/guides/log-streaming-syslog

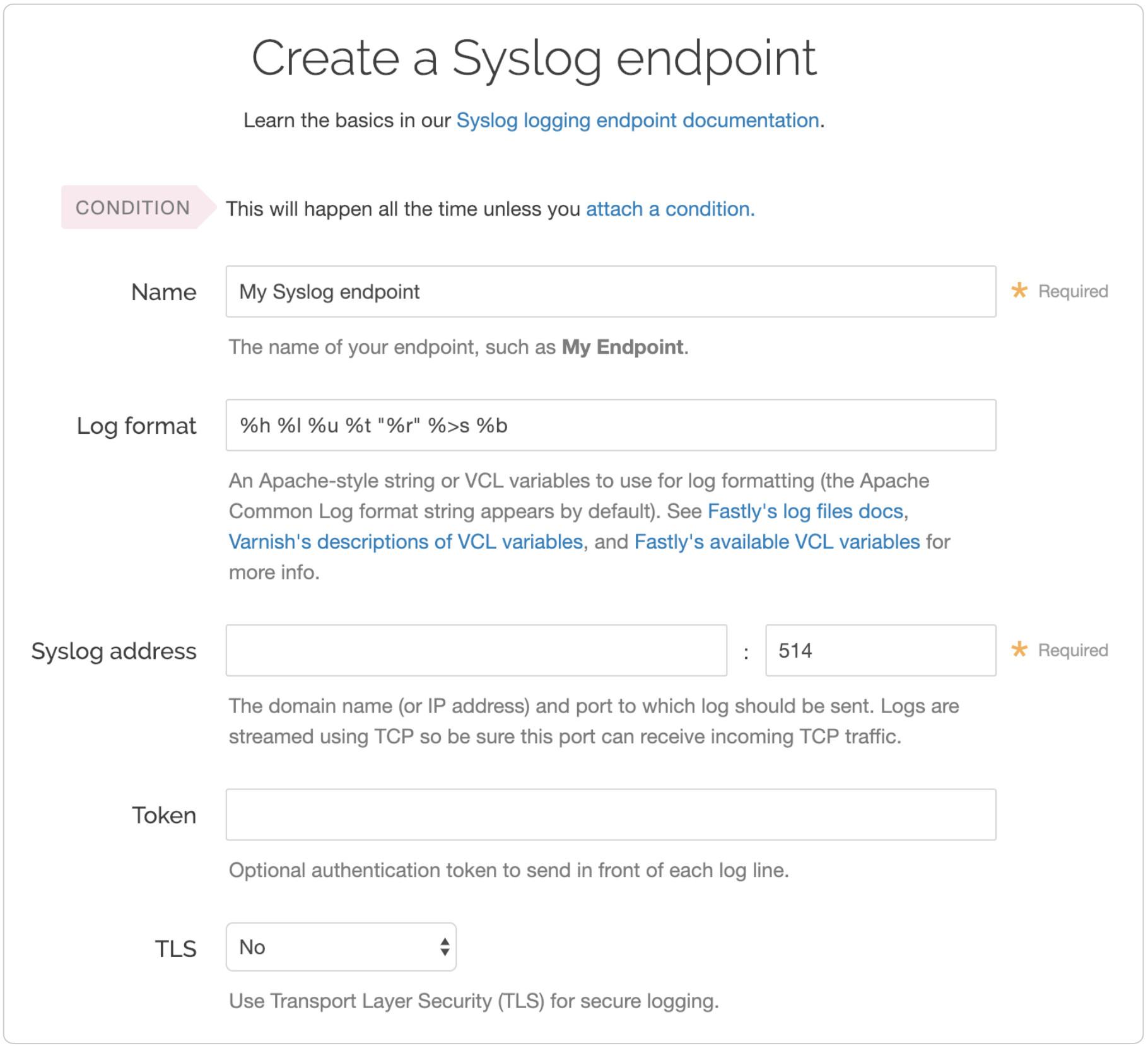
Fastly's Real-Time Log Streaming feature can send log files to syslog-based logging software. Syslog is a widely used standard for message logging.

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

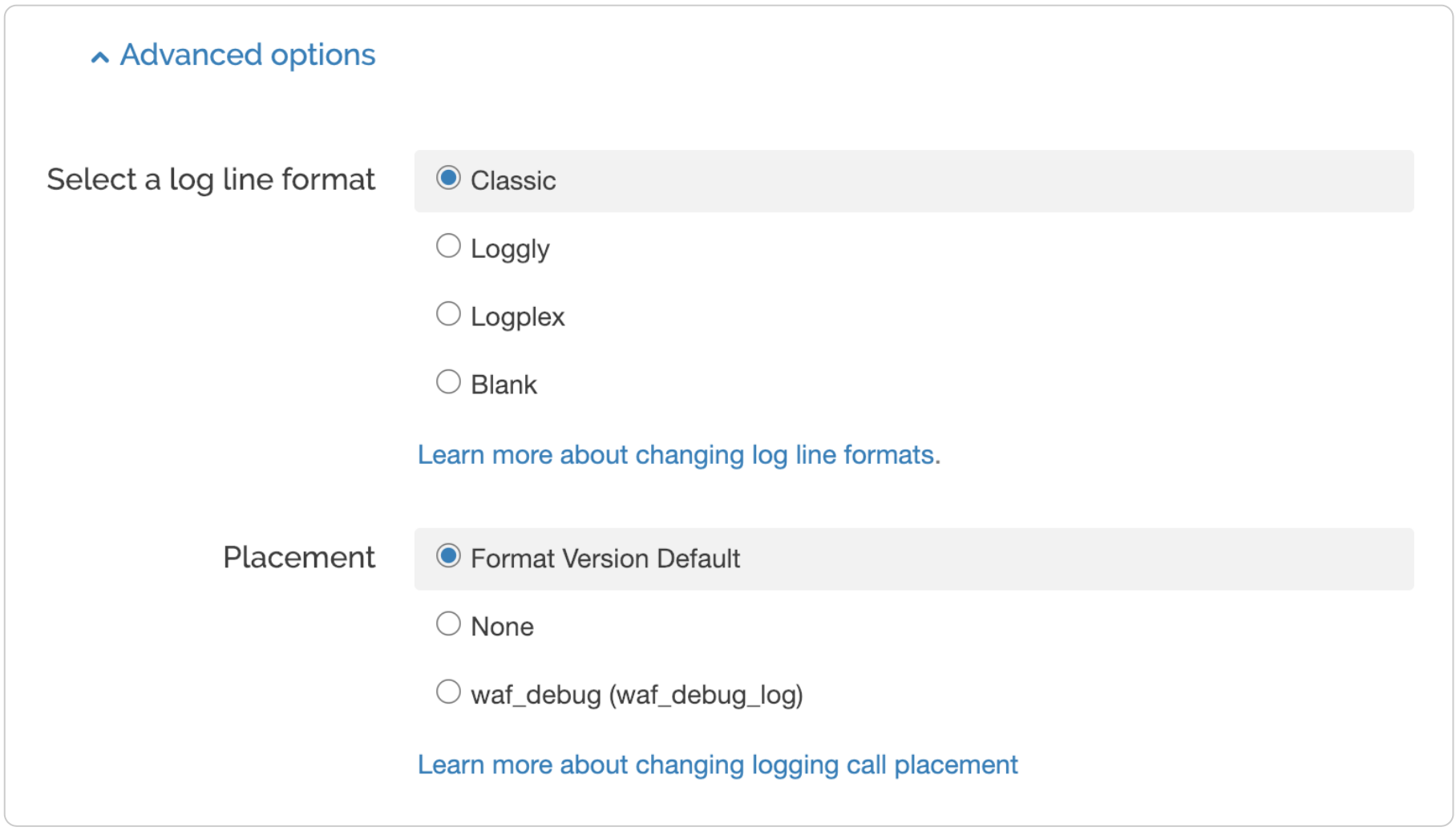
Follow these instructions to add syslog as a logging endpoint:

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, type a human-readable name for the endpoint.
   - In the **Log format** field, optionally type an Apache-style string or VCL variables to use for log formatting. The Apache Common Log format string appears in this field by default. See our guidance on format strings for more information.
   - In the **Syslog address** field, type 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 type 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 type the hostname used to verify the syslog server's certificate. This can be either the Common Name (CN) or Subject Alternate Name (SAN). 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.
   - In the **Placement** area, select where the logging call should be placed in the generated VCL. Valid values are **Format Version Default**, **None**, and **waf_debug (waf_debug_log)**. Selecting **None** creates a logging object that can only be used in custom VCL. See our guide on WAF logging for more information about `waf_debug_log`.
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, type any suitable header name (for example, `http.X-Separator`).
   - In the **Source** field, type 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 address blocks is dynamic, so we recommend programmatically obtaining the list from our JSON feed whenever possible.

Non-Fastly services

These articles describe how non-Fastly services interoperate with Fastly.

https://docs.fastly.com/en/guides/integrations#_non-fastly-services

Amazon S3

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>.<REGION>.amazonaws.com`. Use the table in the S3 section of the Amazon S3 regions and endpoints documentation as a guide. For example, if your bucket name is `fastlytestbucket` and your region is `s3`, your hostname would be `fastlytestbucket.s3.amazonaws.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 region (e.g., `fastlytestbucket.s3.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.amazonaws.com`.
4. In the **Override host** field in the **Advanced options**, enter an appropriate address for your host (e.g., `fastlytestbucket.s3.amazonaws.com`). You entered this information during host creation.

Enabling cross-origin resource sharing (CORS)

We recommend enabling CORS (Cross-Origin Resource Sharing) when using Amazon S3 as your origin. To enable this, follow the instructions in our guide on enabling cross-origin resource sharing (CORS).

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.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 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 Amazon S3 configuration in our Cache Control tutorial.

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.

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:

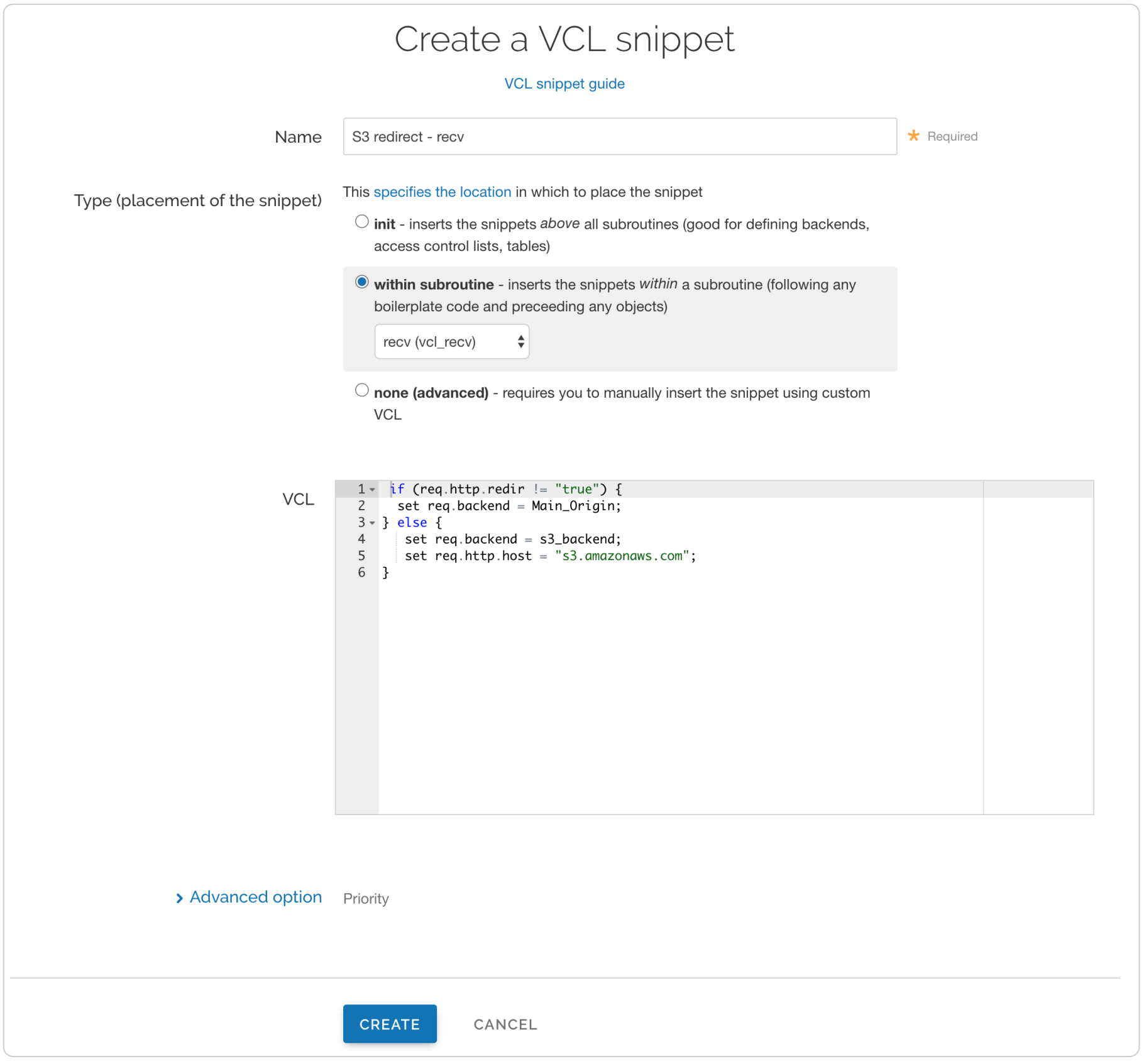
```
unset beresp.http.x-amz-id-2;
unset beresp.http.x-amz-request-id;
unset beresp.http.x-amz-delete-marker;
unset beresp.http.x-amz-version-id;
```

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:

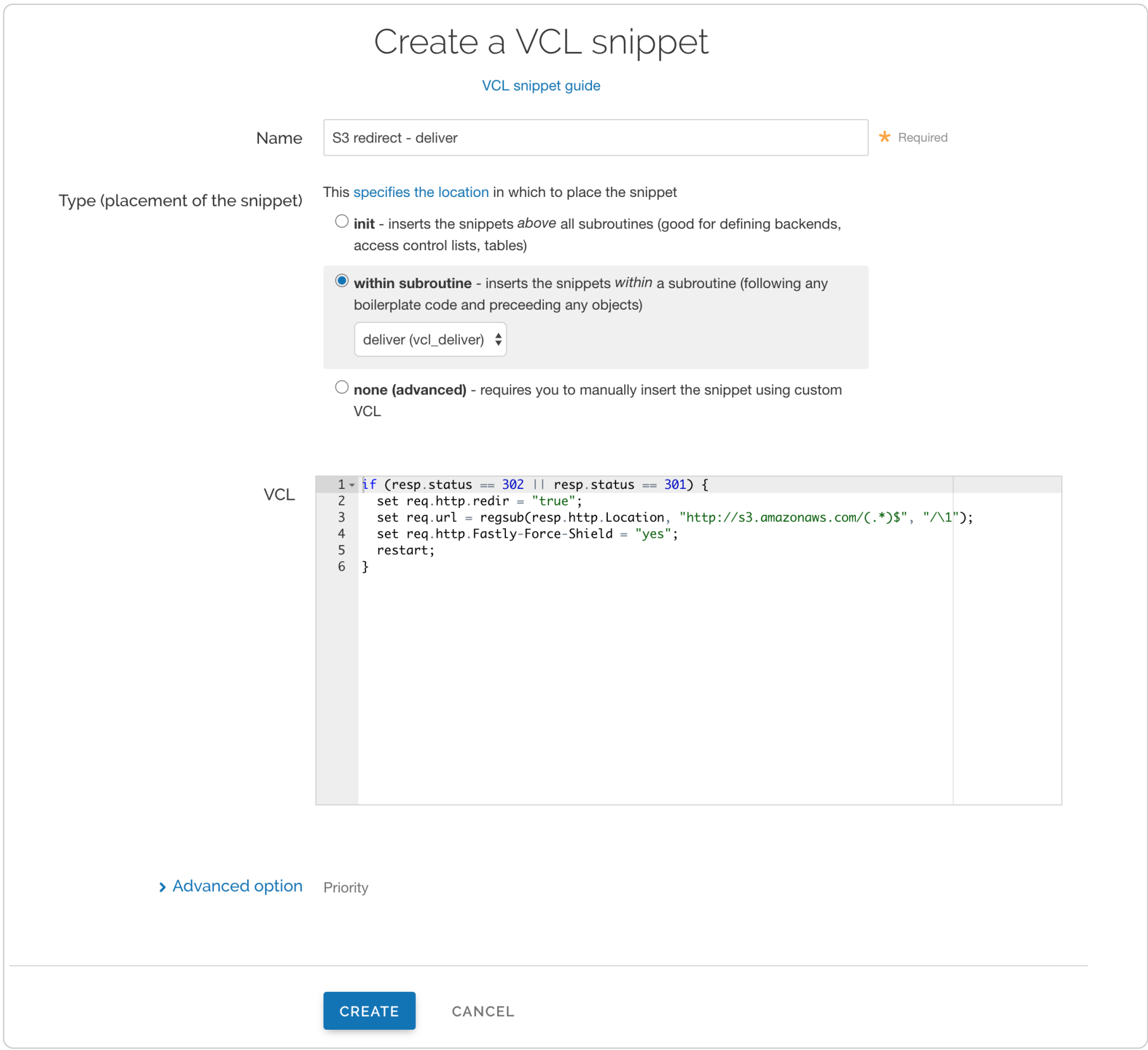
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **VCL Snippets** link. The VCL Snippets page appears.
5. Click the **Create snippet** button. The Create a VCL snippet page appears.



6. In the **Name** field, type 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";
}
```

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, type 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.

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.

> This article describes an integration with a service provided by a third party. Please see our note on integrations.

DigitalOcean Spaces

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 in the **Advanced options**, 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), use the Amazon S3 configuration in our Cache Control tutorial as an example.

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:

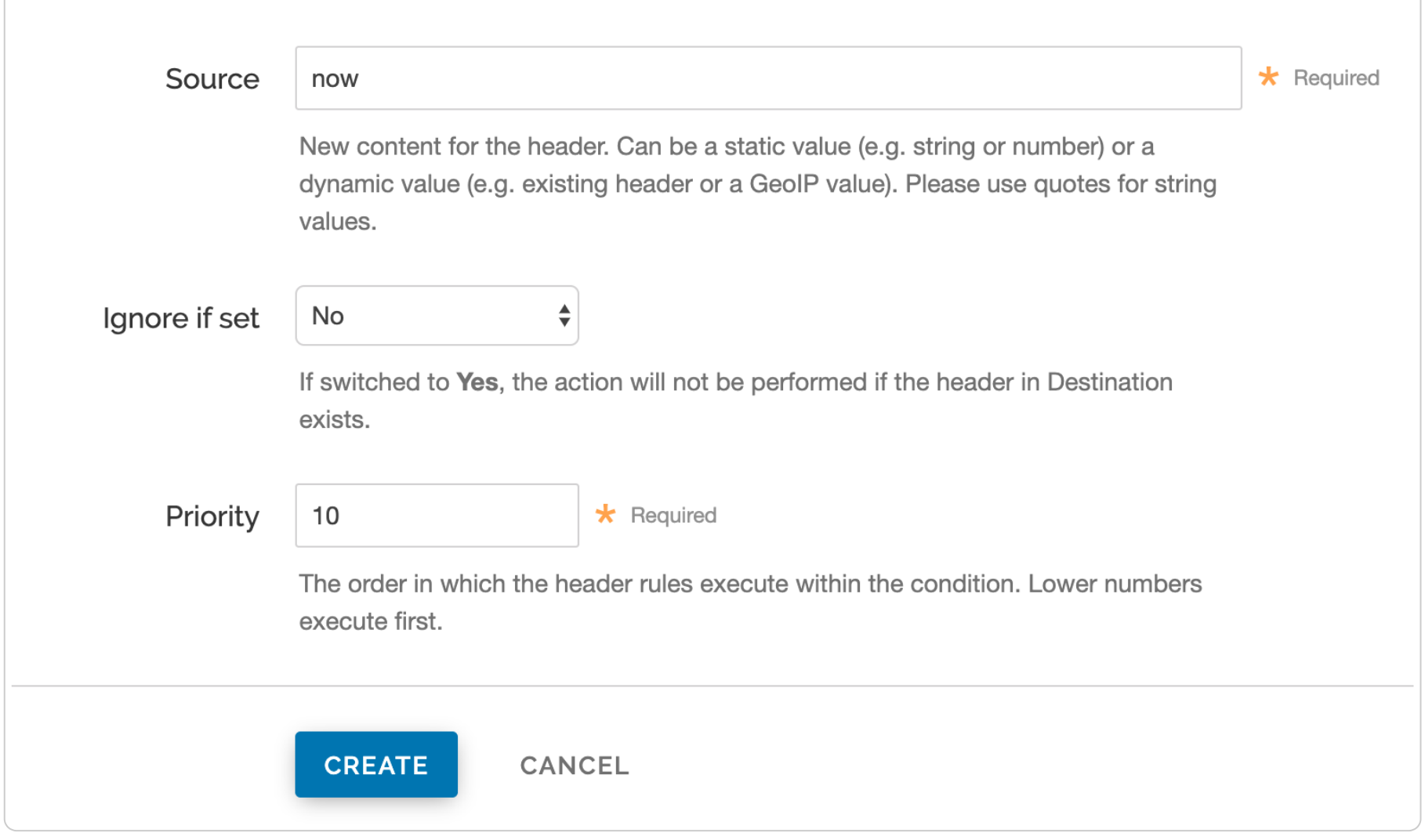
1. the **access key** and **secret key**
2. your **Space** name

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Date`.

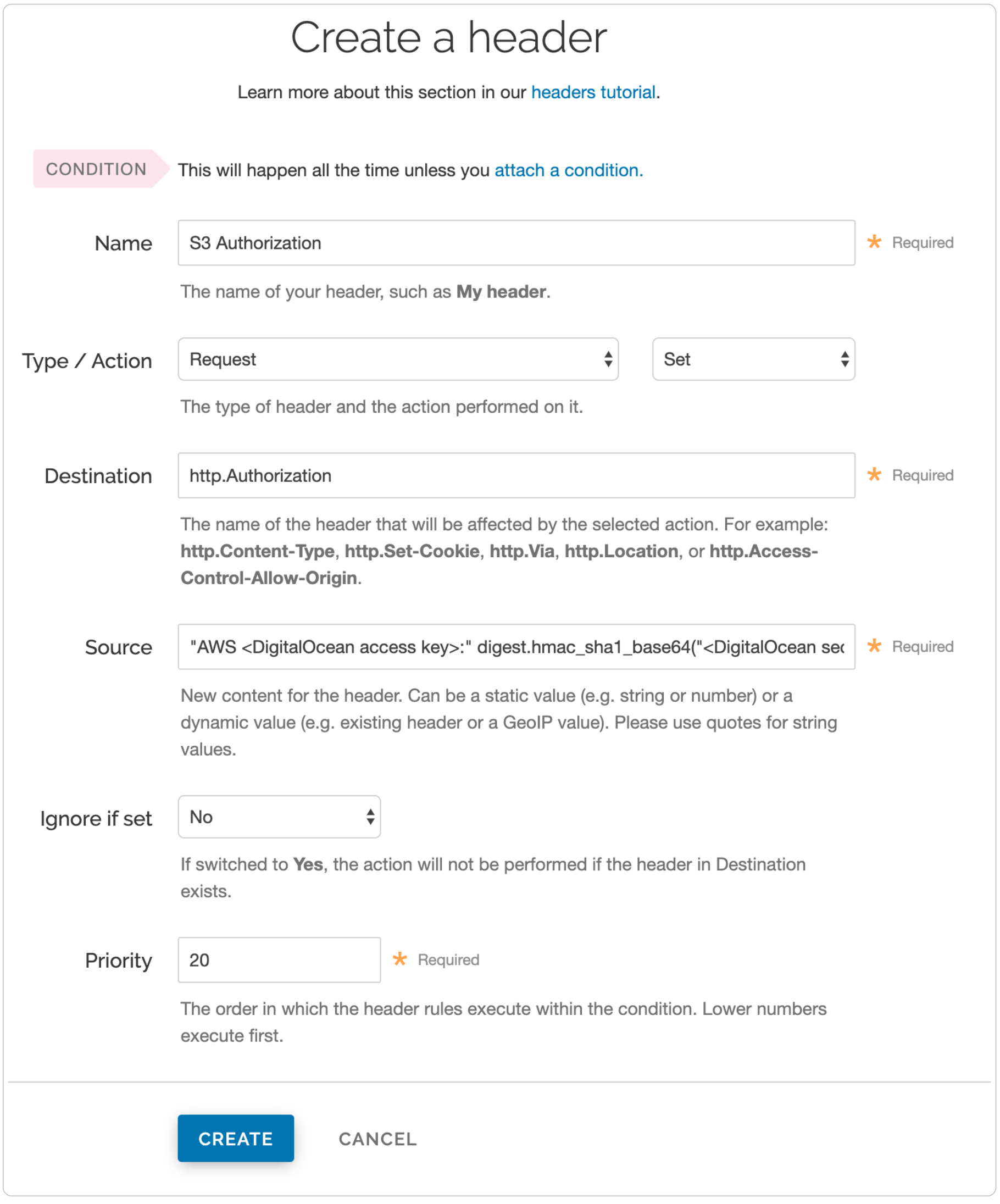
- In the **Source** field, type `now`.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `10`.

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, type `Spaces Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Authorization`.

- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `20`.

3. In the **Source** field, type the header authorization information using the following format:

```
"AWS <DigitalOcean access key>:" digest.hmac_sha1_base64("<DigitalOcean secret key>", if(req.method == "HEAD", "GET", req.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. Please see our note on integrations.

Discounted egress from Google

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 Origin Shields are routed over Fastly's network, leveraging optimized TCP connection handling, quick-start, and opened connections to ensure 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.

This article describes an integration with a service provided by a third party. Please see our note on integrations.

Google Cloud Storage

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.

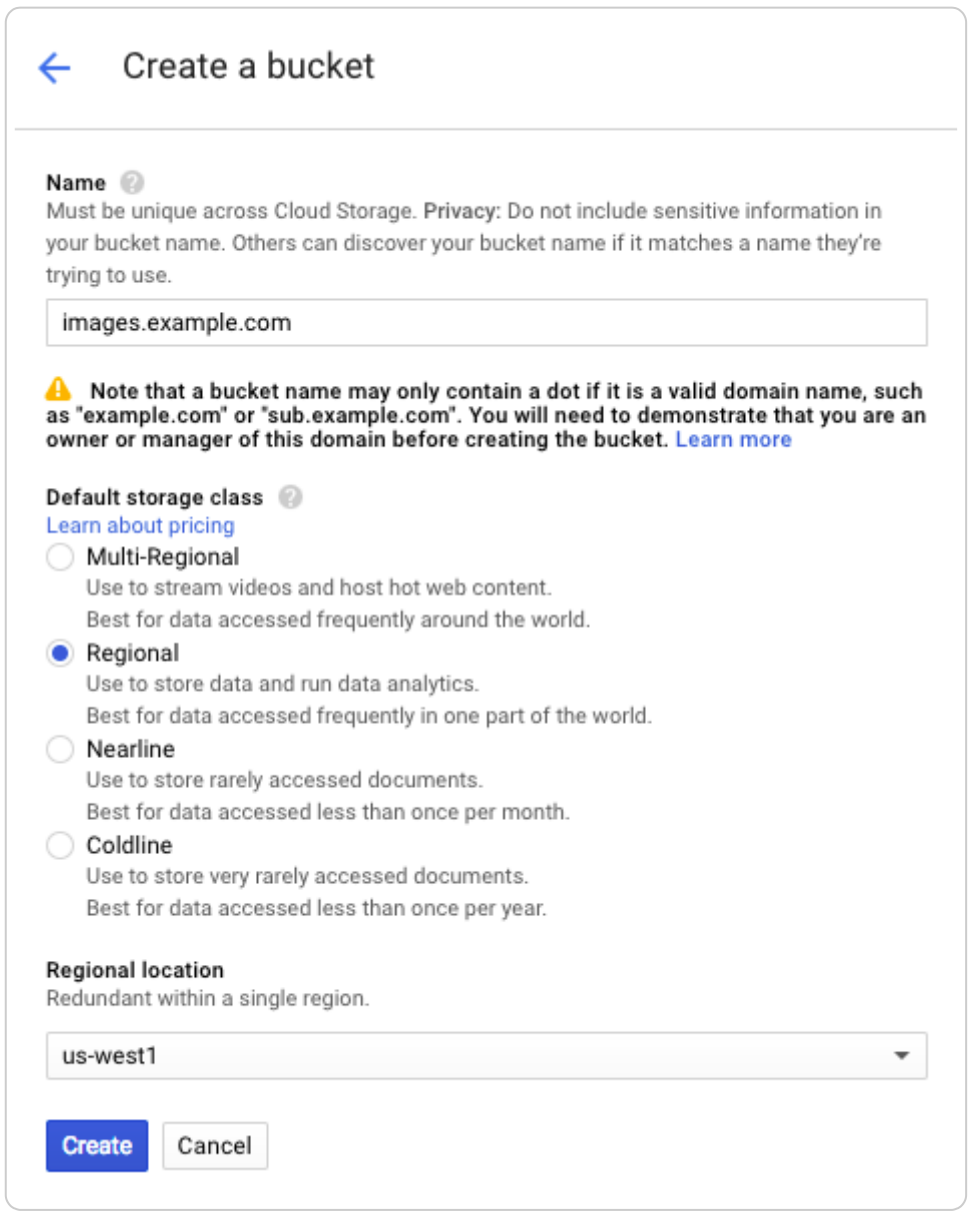
★ **TIP:** Google offers an integration discount that applies to any Google Cloud Platform product. If you're a Fastly customer and would like to take advantage of this discount, email salesgcp@fastly.com.

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. Use Google's Search Console to verify ownership of your domain name, if you have not already done so. See the instructions on Google's website.
4. Fill out the **Create a bucket** fields as follows:
   - In the **Name** field, type your domain name (e.g., `example.com` or `images.example.com`) to create a domain-named bucket. 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.
5. Click the **Create** button.

You should now add files to your bucket and make them externally accessible by selecting the **Public link** checkbox next to each of the files.

Adding your GCS bucket as an origin server

To add your GCS bucket as an origin server, follow the instructions for connecting to origins. 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 override 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 `test123`, your hostname would be `test123.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 the same address that appears in the Address field (e.g., `test123.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 in the **Advanced options** area, 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 below, 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 GCS bucket's region.

Interconnects exist in the following locations within North America:

- Ashburn (BWI)
- Ashburn (DCA)
- Ashburn (IAD)
- Atlanta (FTY)
- Chicago (MDW)
- Dallas (DFW)
- Los Angeles (LAX)
- New York (JFK)
- Palo Alto (PAO)
- Seattle (SEA)
- San Jose (SJC)
- Toronto (YYZ)

Interconnects outside of North America exist in:

- Amsterdam (AMS)
- Frankfurt (FRA)
- Frankfurt (HHN)
- Hong Kong (HKG)
- London (LCY)
- London (LHR)
- Madrid (MAD)
- Paris (CDG)
- Singapore (SIN)
- Stockholm (BMA)
- Sydney (SYD)
- Tokyo (TYO)
- Tokyo (HND)

Review our caveats of shielding and select an interconnect accordingly.

Setting the Cache-Control header for your GCS bucket

GCS performs its own caching, 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 files in your GCS bucket:

```
gsutil setmeta -h "Cache-Control: max-age=0, s-maxage=86400" gs://<bucket>/*.html
```

Replace `<bucket>` in the example above with your GCS bucket's name. Note that `max-age` should instruct GCS to cache your content for zero seconds, and Fastly to cache your content for one day. See Google's setmeta docs 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.

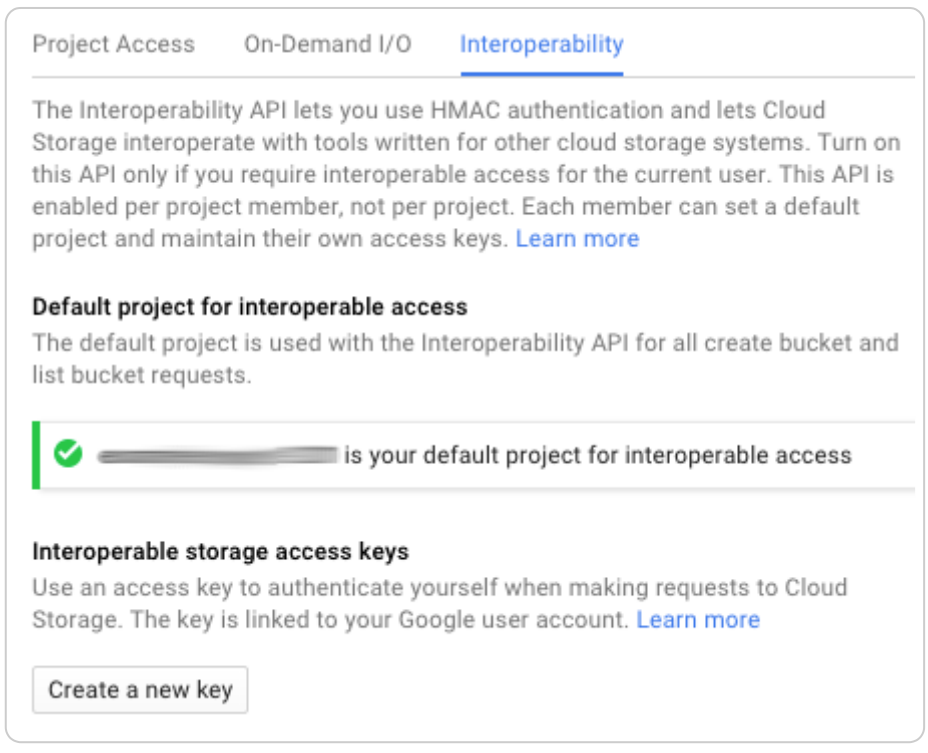
Using GCS with private objects

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.

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.

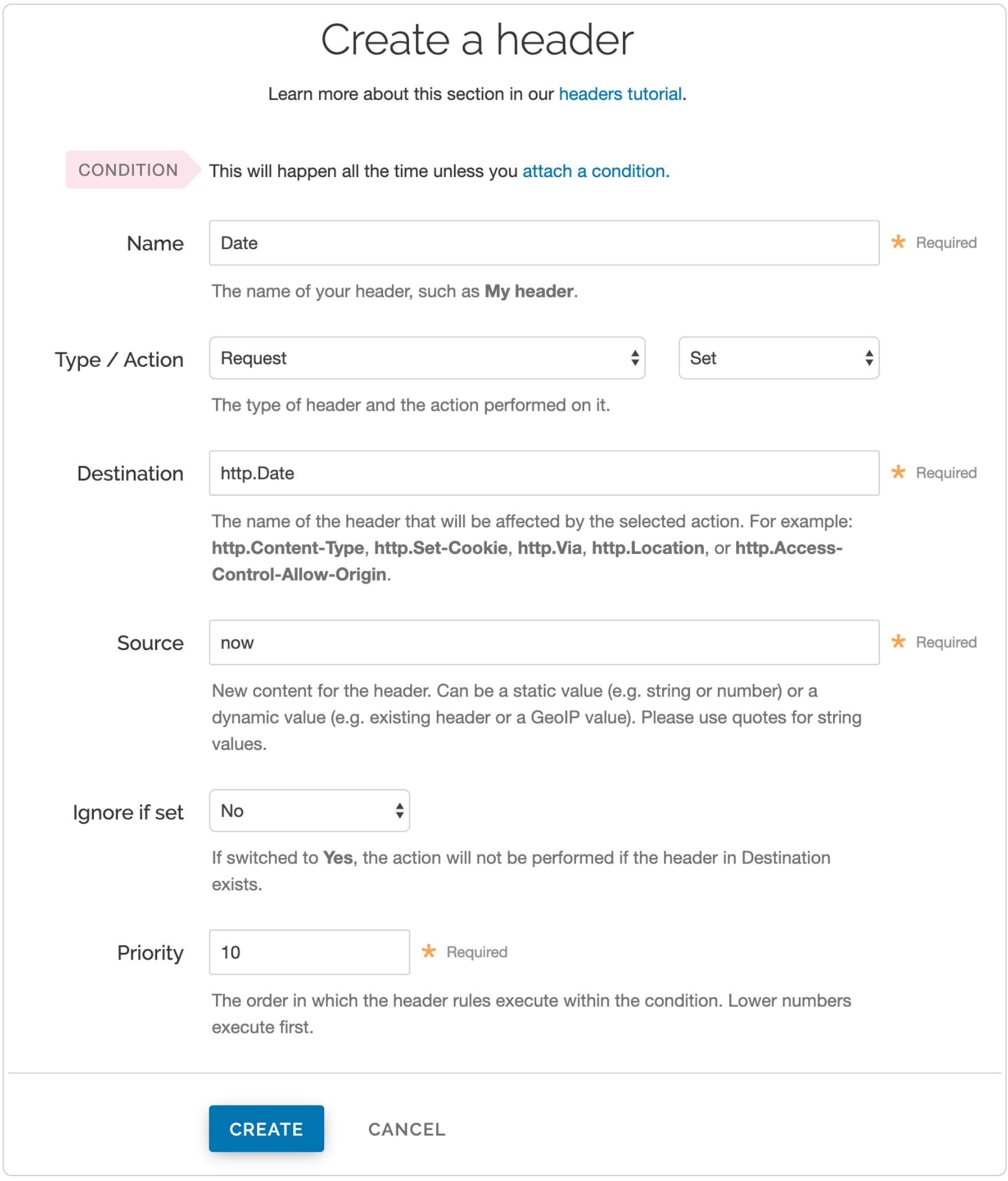
Setting up Fastly to use GCS private content

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.

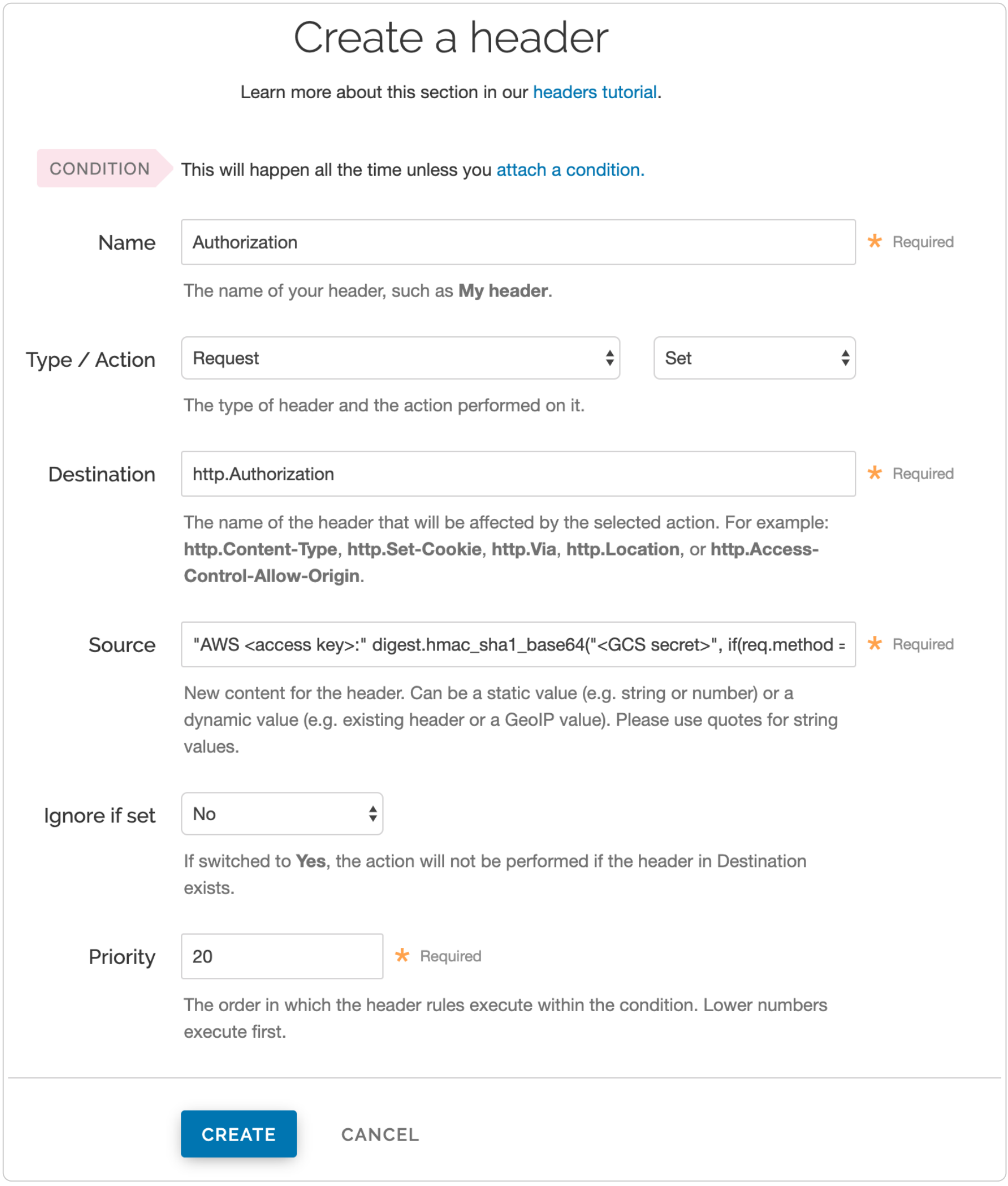
4. Click the **Content** link. The Content page appears.
5. Click the **Create header** button. The Create a new header page appears.



6. Fill out the **Create a new header** fields as follows:
   - In the **Name** field, type `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Date`.
   - In the **Source** field, type `now`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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, type `Authorization`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Authorization`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `20`.
3. In the **Source** field, type 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
== "HEAD", "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. Please see our note on integrations.

Google Compute Engine

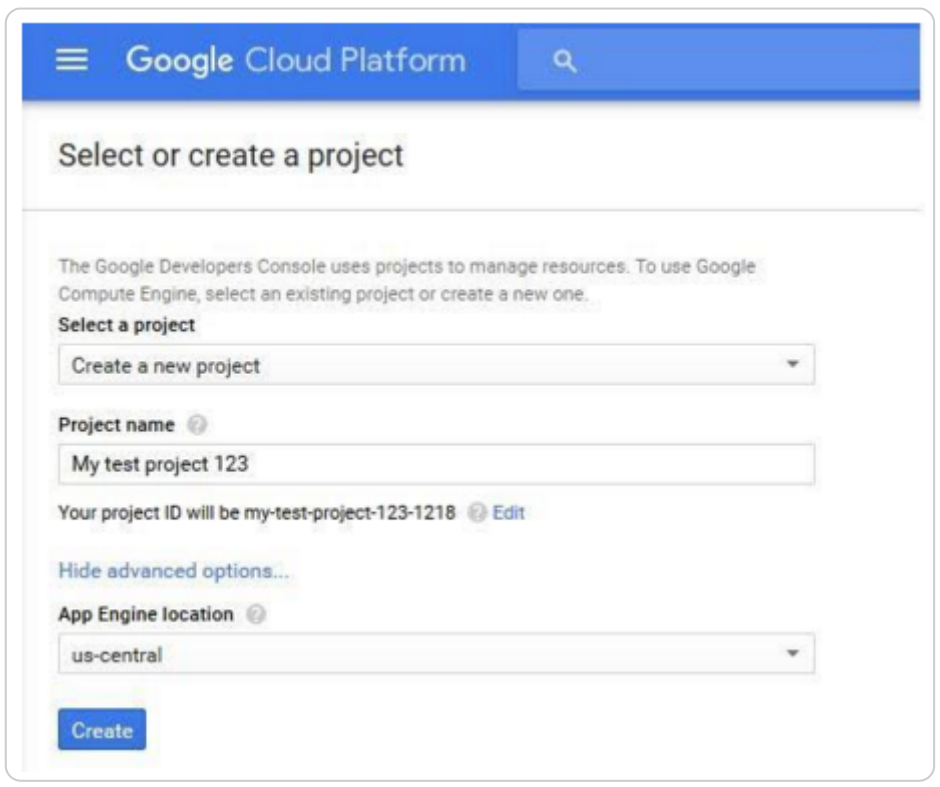
https://docs.fastly.com/en/guides/google-compute-engine

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.

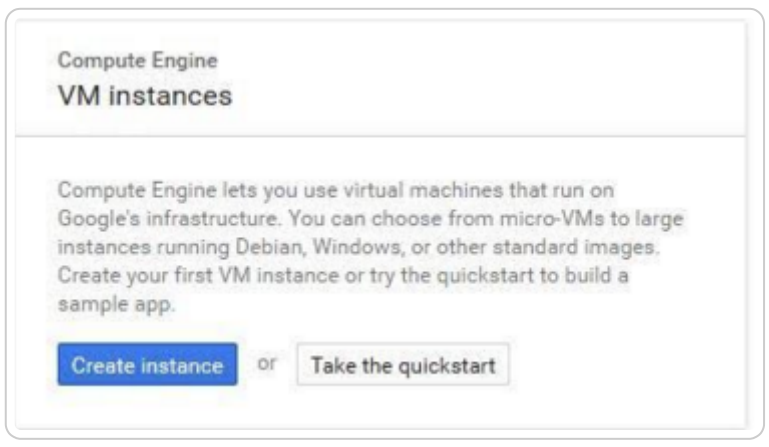
TIP: Google offers an integration discount that applies to any Google Cloud Platform product. If you're a Fastly customer and would like to take advantage of this discount, email salesgcp@fastly.com.

Creating and setting up your GCE instance

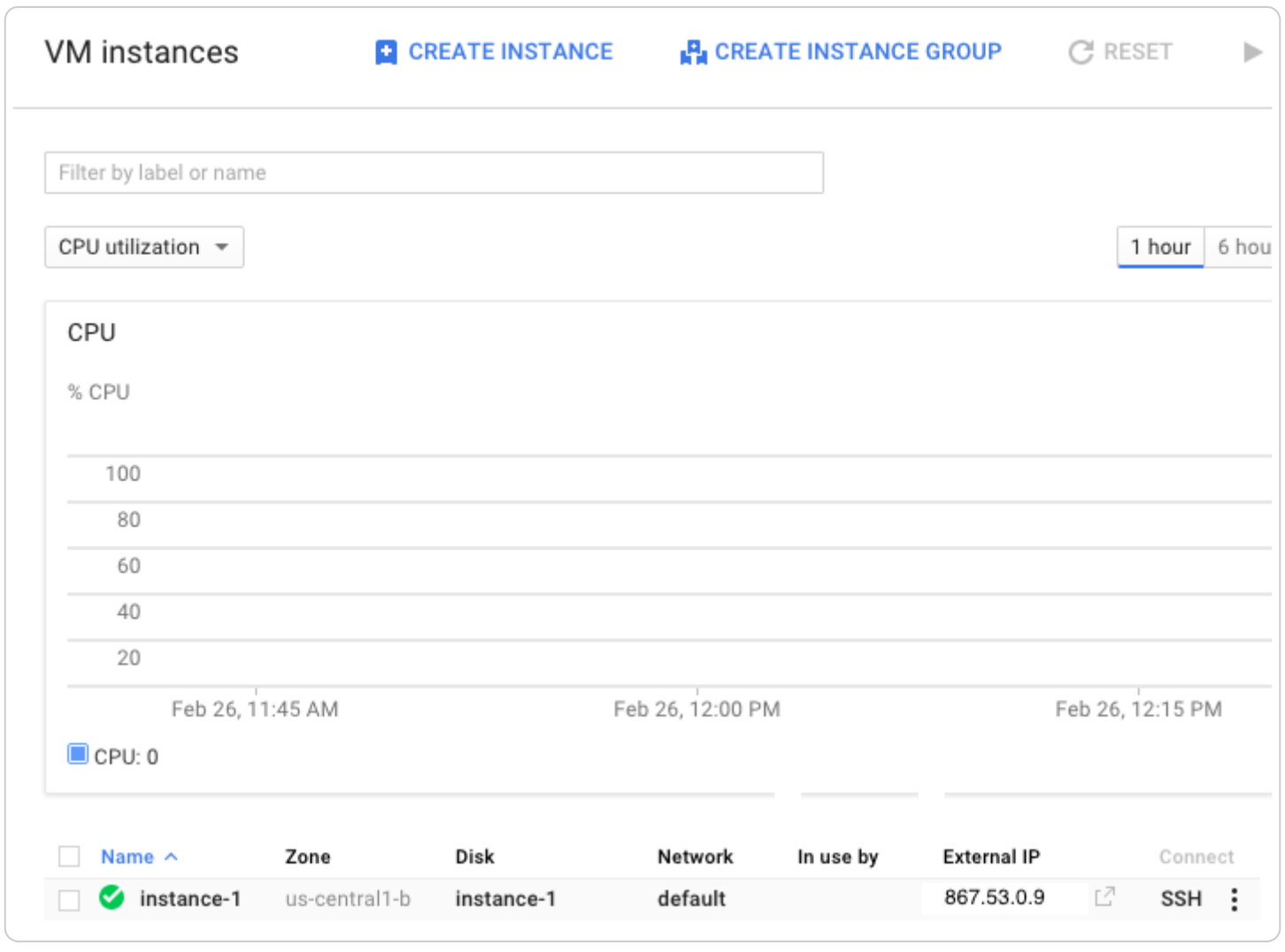
1. Sign up for Google Compute Engine and start the basic set up. If you are already signed up and at your dashboard, click the **Get started** link in the Try Compute Engine area.
2. Create or select a project to hold your origin's data.



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 and click the **Configure** link.
2. Create a new service if you don't already have one set up.
3. From the service menu, select the appropriate service.
4. Follow the instructions for connecting to origins. You'll add specific details about your origin server when you fill out the **Create a host** fields:
   - In the **Name** field, type the name of your server (for example, `Google Compute Engine`).
   - In the **Address** field, type 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 below, 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.

Interconnects exist in the following locations within North America:

- Ashburn (DCA)
- Ashburn (IAD)
- Atlanta (ATL)
- Chicago (MDW)
- Dallas (DFW)
- Los Angeles (LAX)
- New York (JFK)
- Seattle (SEA)
- San Jose (SJC)
- Toronto (YYZ)

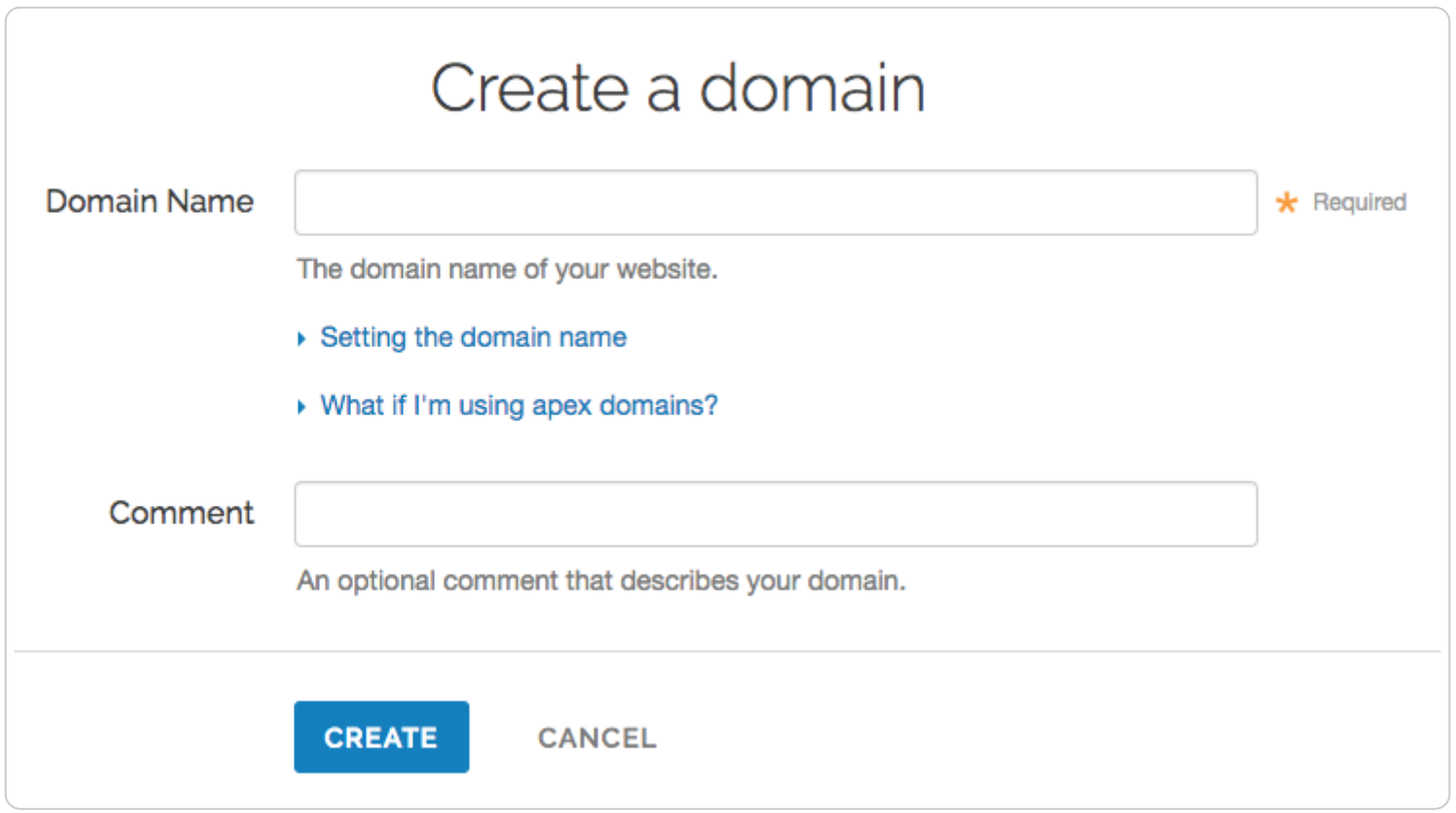
Interconnects outside of North America exist in:

- Amsterdam (AMS)
- Frankfurt (FRA)
- Frankfurt (HHN)
- Hong Kong (HKG)
- London (LCY)
- London (LHR)
- Madrid (MAD)
- Paris (CDG)
- Singapore (SIN)
- Stockholm (BMA)
- Tokyo (TYO)
- Tokyo (HND)

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Create domain** button. The Create a domain page appears.



5. In the **Domain Name** field, type the name that users will type in their browsers to access your site.
6. In the **Comment** field, optionally type 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. Please see our note on integrations.

Microsoft Azure Blob Storage

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

To make your Azure Blob Storage stores available through Fastly, follow the steps below.

Creating a new service

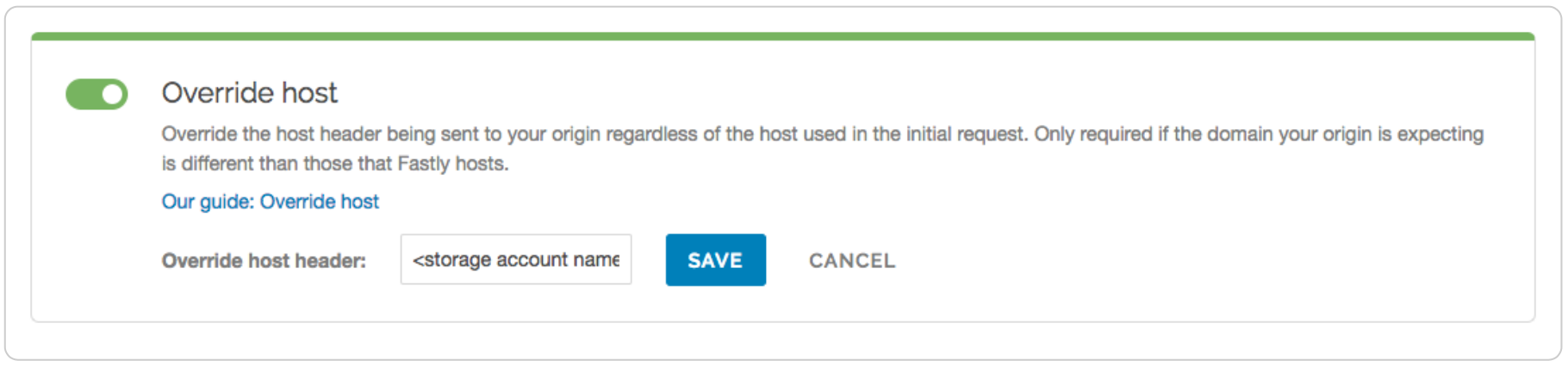
Follow the instructions for creating a new service. You'll add specific details about your origin when you fill out the **Create a new service** fields:

- In the **Name** field, type any descriptive name for your service.
- In the **Domain** field, type the hostname you want to use as the URL (e.g., `cdn.example.com`).
- In the **Address** field, type `<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, type `<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 service menu, select the appropriate service.
2. Click the **Edit configuration** button and then select **Clone active**. 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. Type 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, type `Modify URL`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `url`.
   - In the **Source** field, type `"/<your container name>" req.url`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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

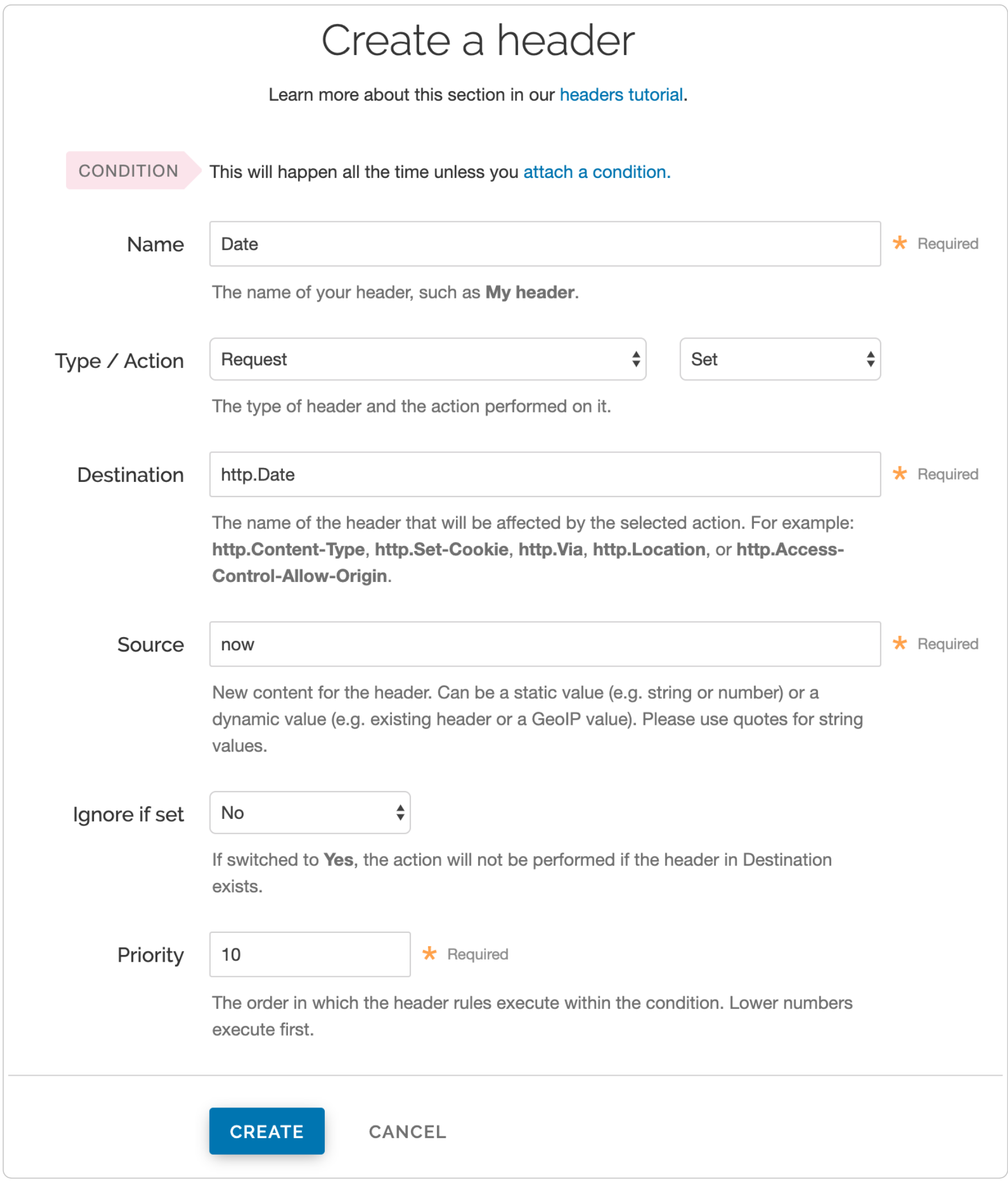
⚠ **WARNING:** Your account's Shared Key does not have detailed access control. Anyone with access to your Shared Key can read and write to your container. Consider using a Shared Access Signature (SAS) instead.

To access an Azure Blob Storage private container with Fastly using a Shared Key, read Microsoft's "Authorize with Shared Key" page. 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

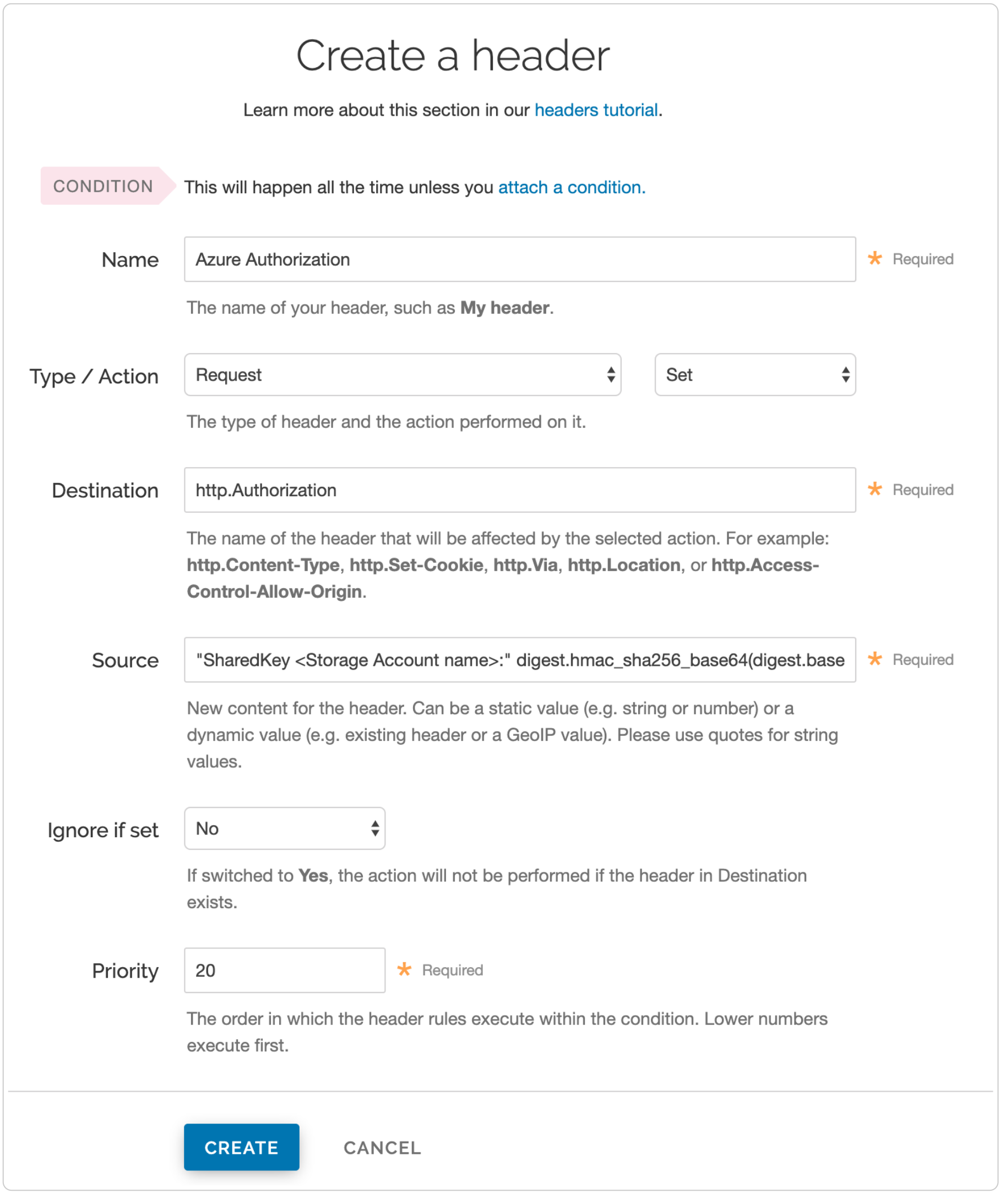


6. Fill out the **Create a header** fields as follows:
   - In the **Name** field, type `Date`.
   - From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
   - In the **Destination** field, type `http.Date`.
   - In the **Source** field, type `now`.
   - From the **Ignore if set** menu, select **No**.
   - In the **Priority** field, type `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, type `Azure Authorization`.
- From the **Type** menu, select **Request**, and from the **Action** menu, select **Set**.
- In the **Destination** field, type `http.Authorization`.
- From the **Ignore if set** menu, select **No**.
- In the **Priority** field, type `20`.

3. In the **Source** field, type the header authorization information using the following format:

```
"SharedKey <Storage Account name>:" digest.hmac_sha256_base64(digest.base64_decode("<Azure Storage Account share
d key>"), if(req.method == "HEAD", "GET", req.method) LF LF LF req.http.Date LF "/<Storage Account name>" req.ur
l.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("UDJXUFN1NjhCZmw4OWo3MnZUK2JYWVpCN1NqbE93aFQ
0d2hxdDI3"), 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.

Element	Description
`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 "Delegating Access with a Shared Access Signature" page. Then, obtain the SAS and sign the access URL.

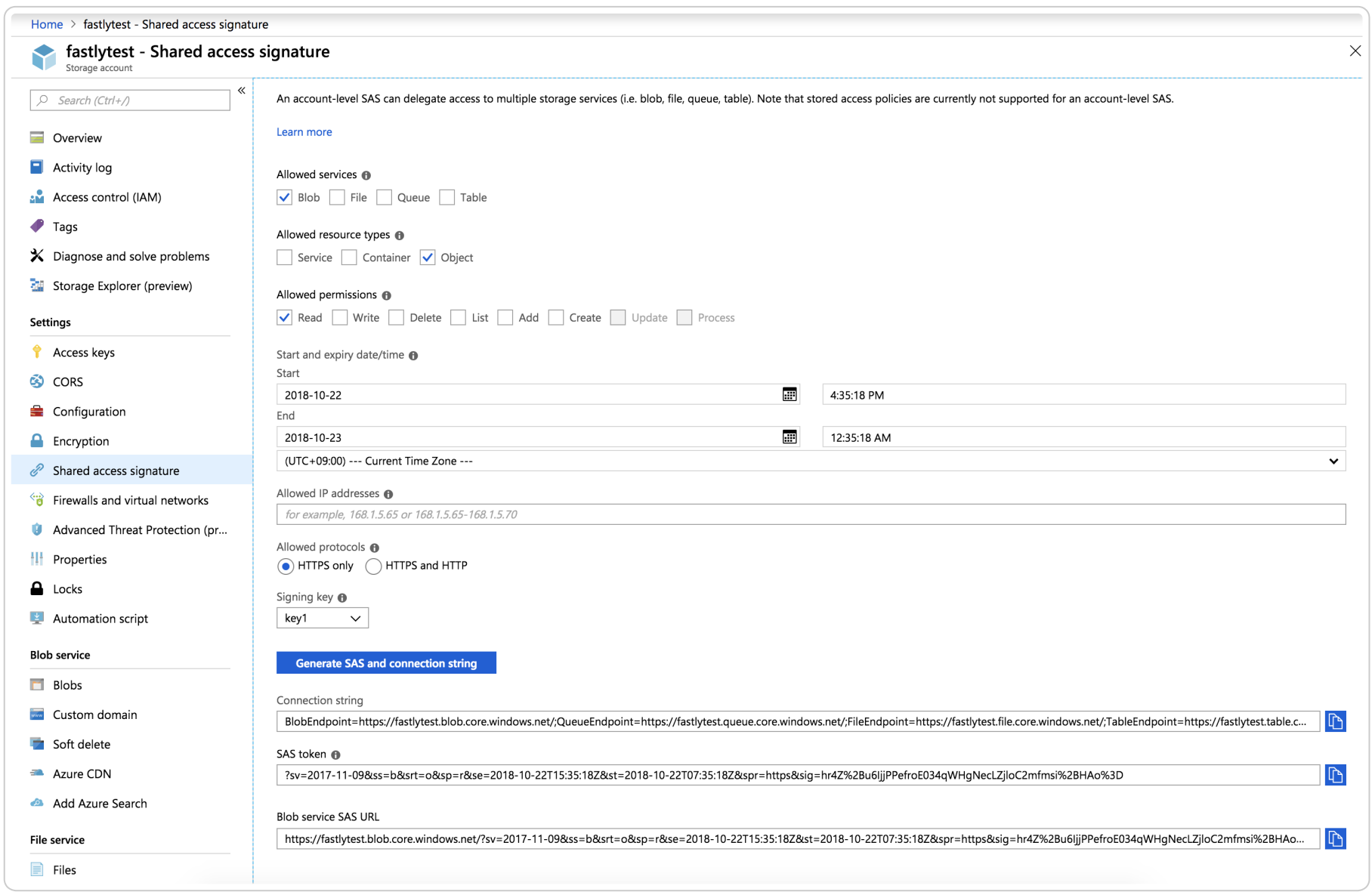
TIP: Using a Service Shared Access Signature gives you more detailed control over:

- 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:

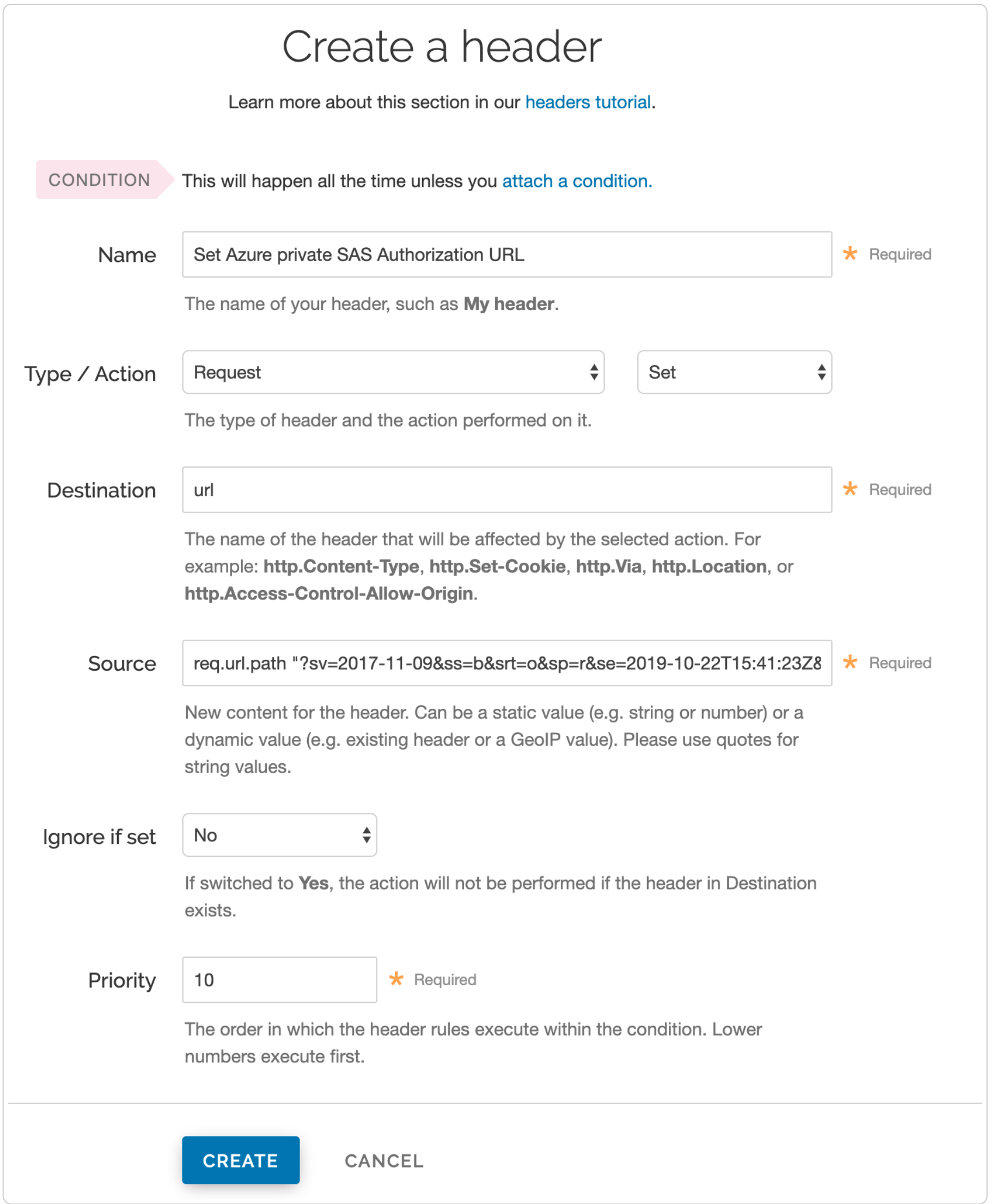
```
?sv=2017-11-09&ss=b&srt=o&sp=r&se=2019-10-22T15:41:23Z&st=2018-10-22T07:41:23Z&spr=https&sig=decafbaddeadbeef
```

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type `url`.
- In the **Source** field, type `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, type `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 "Constructing 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. Please see our note on integrations.

Outbound data transfer from Azure

https://docs.fastly.com/en/guides/outbound-data-transfer-from-azure

Fastly has integrated a local circuit with Microsoft Azure ExpressRoute Direct. Once you've signed up for Fastly services and configured them correctly, outbound data transfers from the appropriate Azure regions to Fastly will use this local circuit when configured with Fastly shielding.

To configure your Azure origin to use this direct connectivity, choose `East US` as the Azure service region and then be sure to choose `Ashburn (BWI)` as the shielding location when configuring your Fastly service.

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. Please see our note on integrations.

PerimeterX Bot Defender

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 more information, see 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. Please see our note on integrations.

Wasabi

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

Wasabi public and private buckets can be used as origins with Fastly.

Using Wasabi as an origin

To make your Wasabi 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 bucket's region. For the us-east-1 region, type `<BUCKET>.s3.wasabisys.com`. For all other regions, type `<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 in the **Advanced options**, 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 bucket as follows:

Wasabi bucket region	Shielding location
eu-central-1	Amsterdam, NL
us-east-1	Ashburn, VA
us-west-1	Seattle, WA

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), use the Amazon S3 configuration in our Cache Control tutorial as an example.

Using private Wasabi buckets

To use a Wasabi 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 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 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, type `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, type `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, type `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. Please see our note on integrations.

Diagnostics

These articles describe how to log data, troubleshoot problems, and tune performance.

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

Streaming logs

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

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.
- **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 line formats

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. By default, 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`).

Available log line message formats

Although the default message prefix works for most logging services and processors, we allow you to choose one of several formats:

- `classic` is the default prefix format. A standard syslog prefix as defined by RFC 3164.
- `loggly` is a structured syslog prefix format based on RFC 5424.
- `logplex` is a Heroku-style length prefixed syslog format.
- `blank` means no prefix. Just your log message. Useful when writing to JSON and CSV files.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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/<yo
ur Fastly service ID>/version/<version number>/logging/<logging endpoint>/<log name>' --data-binary '{"message_typ
e":"<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/SU1
Z0isxPaozGVKXdv0eY/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`).

Changing where log files are written

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

Custom log formats

https://docs.fastly.com/en/guides/custom-log-formats

Fastly provides two versions of custom log formats. 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.
`%A`	The local IP address.
`%B`	The size of response in bytes, excluding HTTP headers.

Format String	Description
`%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.
`%{FOOBAR}e`	Not supported. Always returns `"-"`.
`%f`	The filename.
`%h`	The remote IP address.
`%H`	The request protocol.
`%{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.
`%k`	The number of keepalive requests handled on this connection. Always returns `0`.
`%l`	Not supported. Always returns `"-"`.
`%m`	The request method.
`%{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.
`%p`	The canonical port of the server serving the request. Always returns `80`.
`%{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).
`%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.
`%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).
`%T`	The time taken to serve the request, in seconds.
`%u`	Not supported. Always returns `"-"`.
`%U`	The URL path requested, not including any query string.
`%v`	The domain name of the request. Equal to `%{req.http.host}V`.

Format String	Description
`%V`	The same as `%v`.
`%{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.

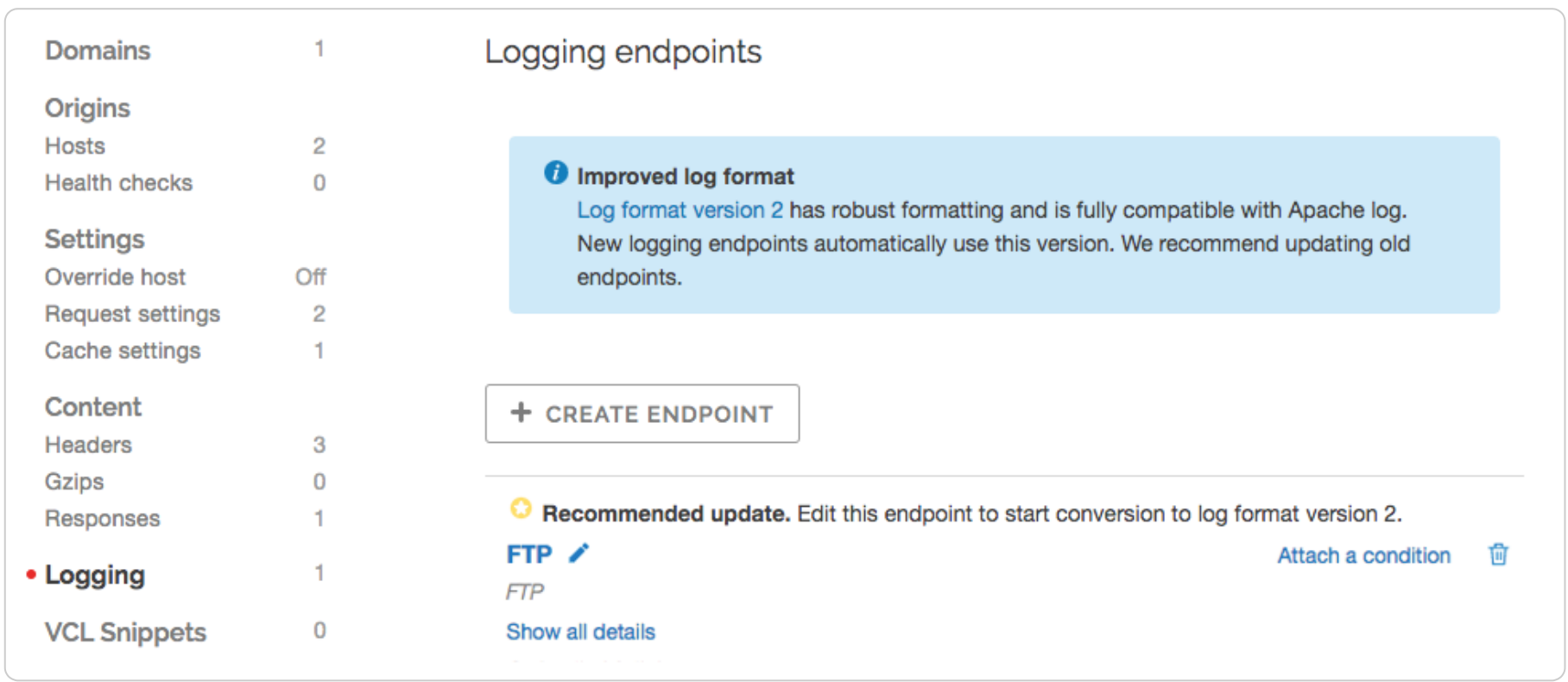
Format String	Description
`%b`	The content size of the response, calculated using the `Content-Length` header rather than actually checking the length of the response (and may therefore be wrong).
`%h`	The remote IP address.
`%l`	The remote log name. Always returns the hardcoded value `"-"`.
`%r`	The 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

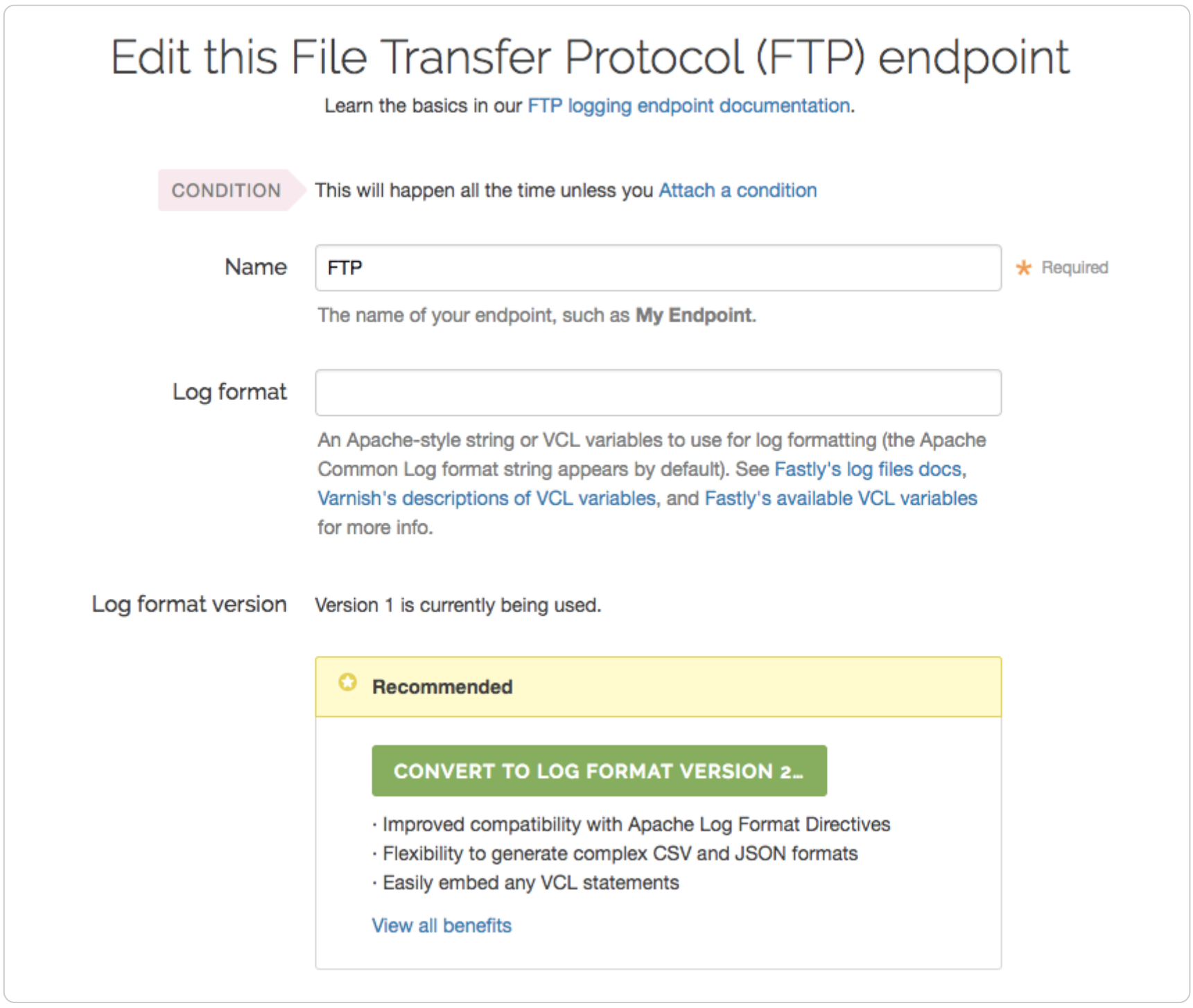
> ⚠ **WARNING:** Upgrading is a permanent change. Logging objects using version 2 formatting cannot be downgraded to version 1.

Follow these instructions to upgrade a logging endpoint to the version 2 custom log format:

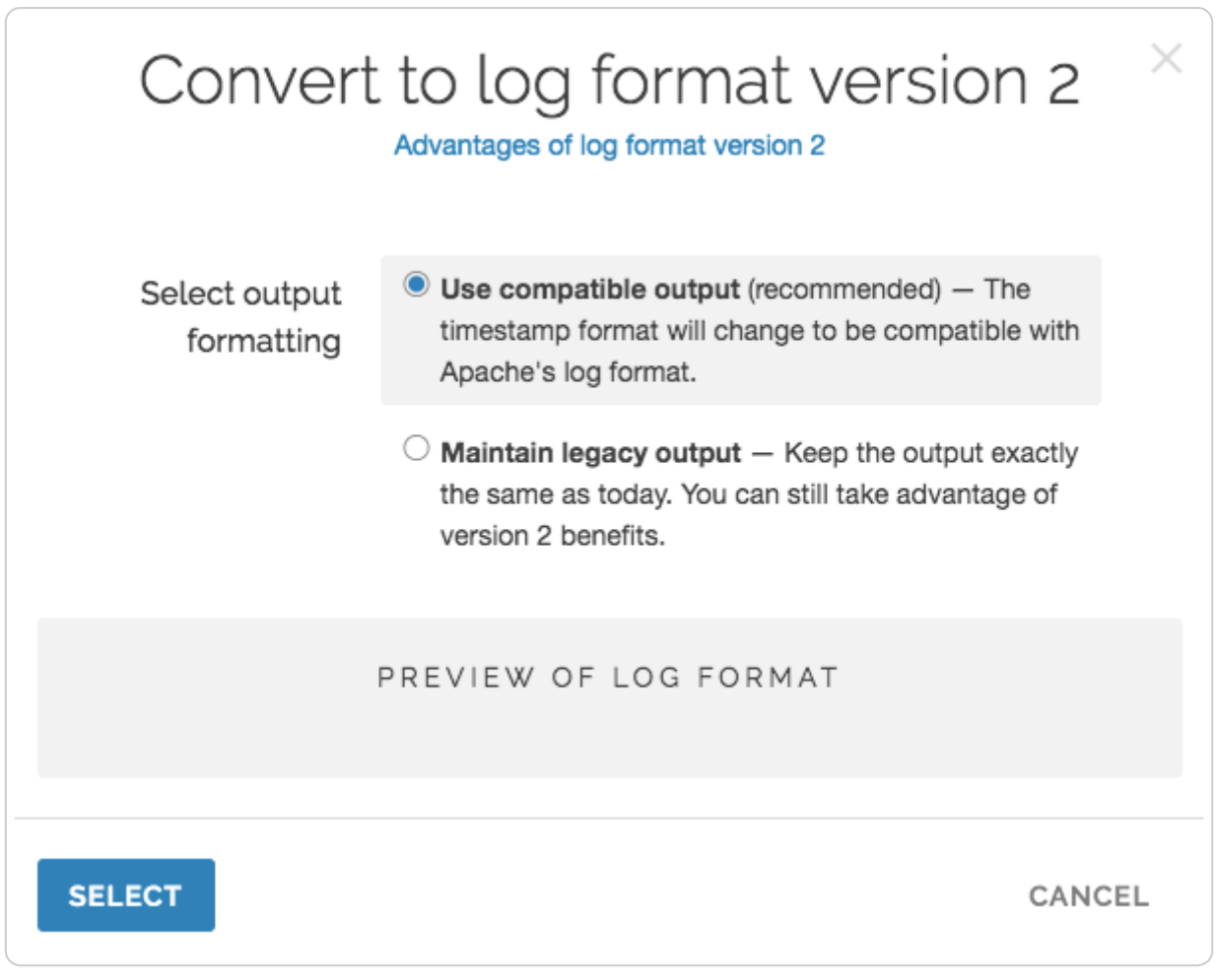
1. Log in to the Fastly web interface and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. The Domains page appears.
4. Click the **Logging** link. The Logging endpoints page appears. If 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/<yo
ur Fastly service ID>/version/<version number>/logging/<logging endpoint>/<log name>' --data-binary '{"format_versio
n":"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/SU1
Z0isxPaozGVKXdv0eY/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/<vers
ion number>/logging/<logging endpoint>/<log name>'
```

The cURL command will produce JSON output detailing the configuration of your service version. For example:

```
1
2
3  {
4      "address": "logs.papertrailapp.com",
5      "created_at": "2016-04-01T15:37:30+00:00",
6      "deleted_at": null,
7      "format": "time.start.msec time.to_first_byte time.elapsed req.body_bytes_read req.bytes_read resp.http.conten
8  t-length server.region client.ip %>s \"req.method req.url req.proto\" \"req.http.referer\" \"req.http.user-agent\"
9  ",
1      "format_version": "2",
0      "hostname": "logs.papertrailapp.com",
1      "name": "fastly",
1      "port": "11111",
1      "public_key": null,
2      "response_condition": "LOG /",
1      "service_id": "1a2b3c4d5e6f7g8h9j0k",
3      "updated_at": "2016-04-01T19:47:47+00:00",
1      "version": "123"
4  }
1
5
```

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

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:

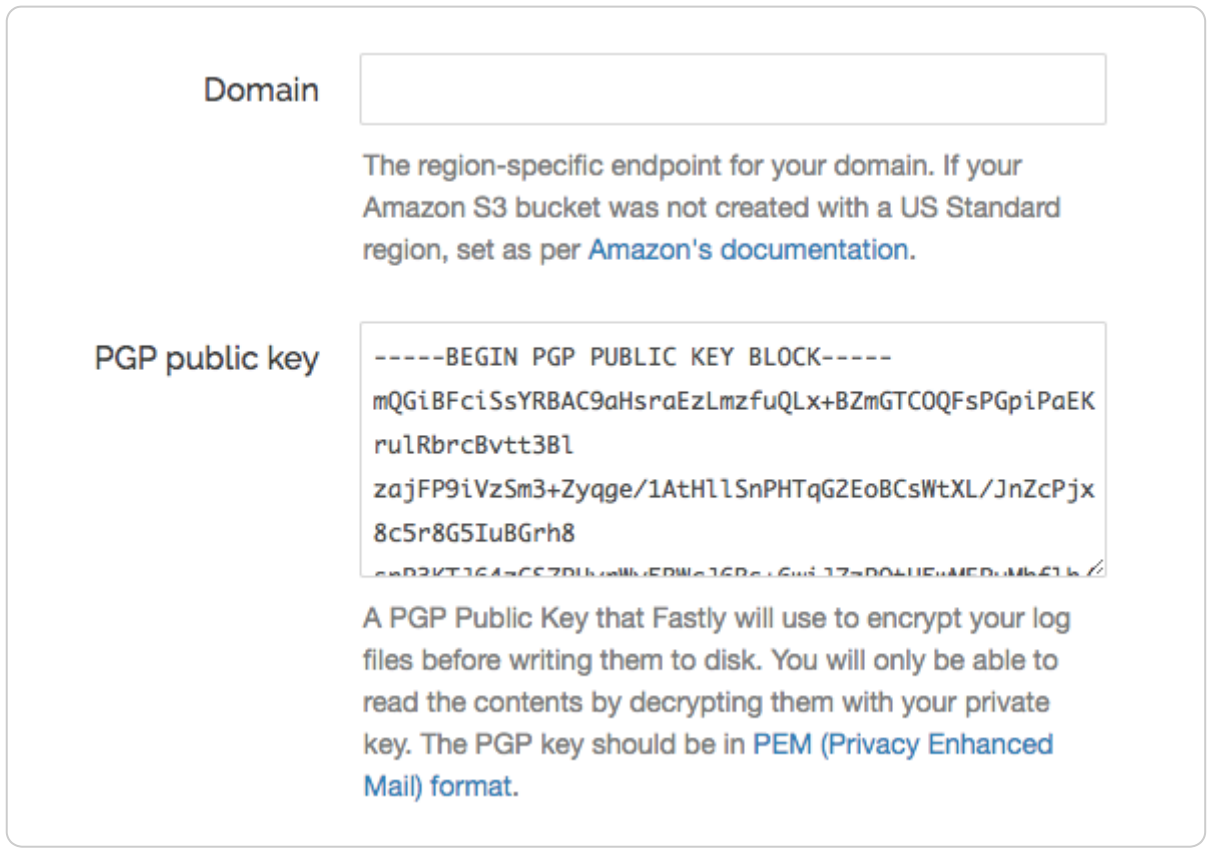
```
gpg --armor --export <your email>
```

The output will be in PEM (Privacy-Enhanced Mail) format and will look similar to the following:

```
1  -----BEGIN PGP PUBLIC KEY BLOCK-----
2  mQGiBFciSsYRBAC9aHsraEzLmzfuQLx+BZmGTCOQFsPGpiPaEKrulRbrcBvtt3Bl
3  zajFP9iVzSm3+Zyqge/1AtHllSnPHTqG2EoBCsWtXL/JnZcPjx8c5r8G5IuBGrh8
4  snP3KTJ64zCS7PUvrWy5RWcJ6Rs+6wiJ7zPOtU5wMEPuMbflh/soy50zrwCg74XN
5  [...REDACTED...]
6  -----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

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.

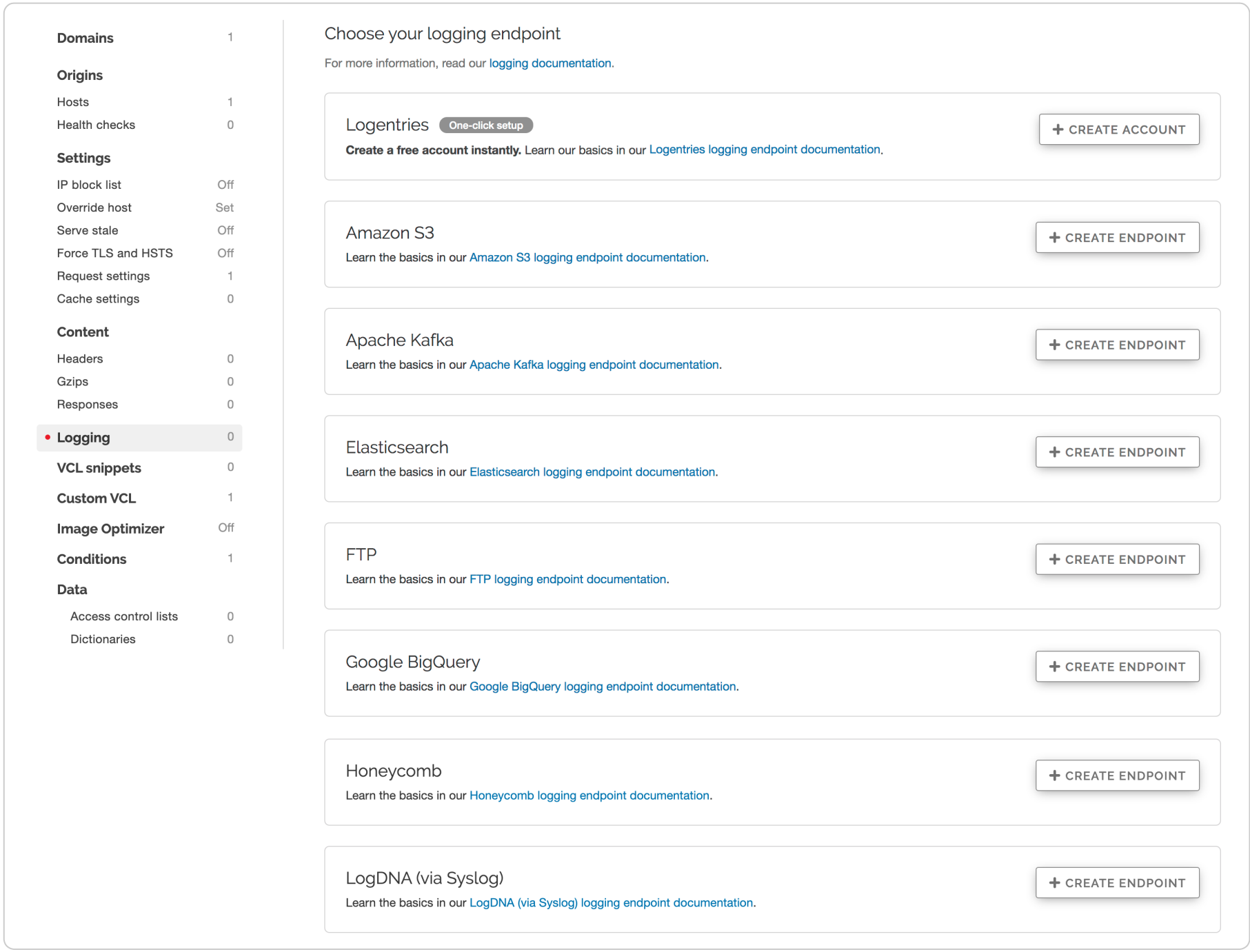
NOTE: Fastly does not provide direct support for third-party services. See Fastly's Terms of Service for more information.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.



NOTE: Some third-party services 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.

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-<server id>`. You can change both the frequency and path of these files. Our guide on changing where log files are written provides more information.

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.

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 adding a condition that prevents Fastly from generating log entries. Follow these instructions to add the condition:

1. On the Logging endpoints page, click the **Attach a condition** link next to the appropriate logging service. The Add a condition window appears.

2. Click **Create a new response condition**. The Create a new response condition window appears.
3. Fill out the **Create a new response condition** window as follows:
   - In the **Name** field, type a human-readable name for the condition.
   - In the **Apply if** field, type `false`.
   - Leave the default value set in the **Priority** field.
4. Click **Save and apply to**.
5. 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

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

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

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 RFC5424 to another provider

You'll have to create a regular Syslog logging object pointing at your endpoint. For example, if Fastly didn't have a Loggly logging object then this would mean setting `hostname` to `logs-01.loggly.com`, `port` to `6514`, `use_tls` to `true`, and the `message format` field to `blank`. Then, in the `format` field, you would put the following:

```
<134>1 %{"%Y-%m-%dT%T%z"}t %{server.datacenter}V <log name> - - [<token>@<PEN> tag="fastly" tag="other tag" <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. In the example we have two: fastly and other tag 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.

Name	VCL Value	Type	Description
Start Time	`begin:%Y-%m-%dT%H:%M:%S%z`	time	The time at the start of the request in [ISO 8601] (https://en.wikipedia.org/wiki/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 `DATETIME` field requires the timestamp format in the JSON above.

CSV

```
%H, %{time.start.sec}V, %{begin:%Y-%m-%dT%H:%M:%S%z}t, %{regsub(req.http.User-Agent, {"""}, {"""""""})}V, %{if(req.is_ipv6, "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_ipv6:%{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-Agent)}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

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

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.
`%{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.

Variable	Description
`%{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
`%{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 datacenter 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"`)

Debugging techniques

Because Fastly doesn't store customer logs, we provide information about debugging techniques that help you gain insights into your service configurations.

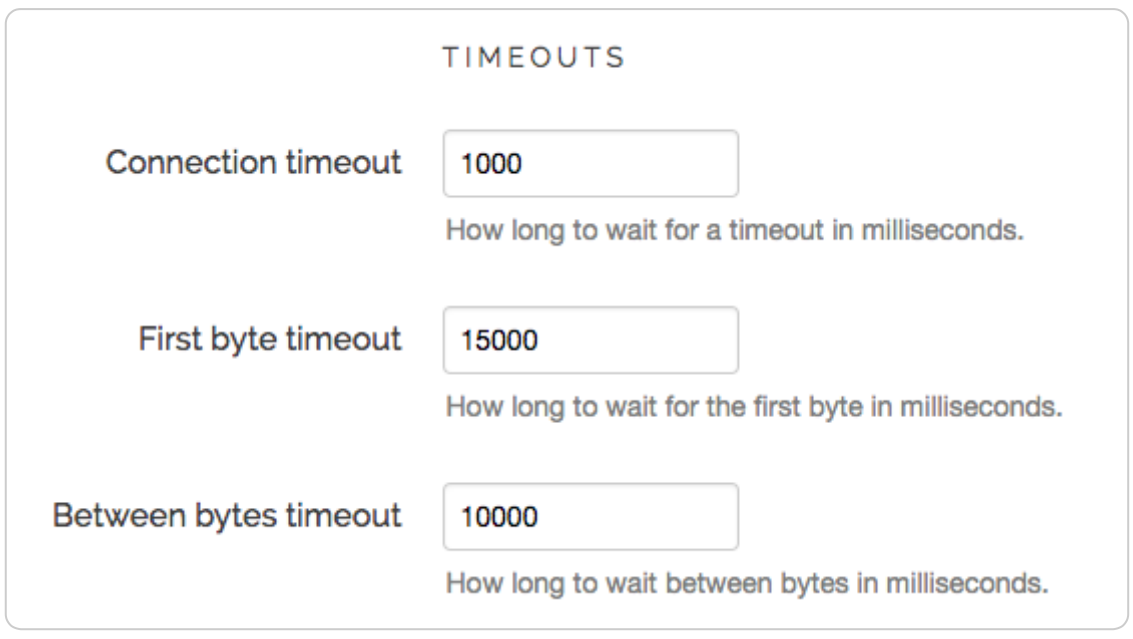
https://docs.fastly.com/en/guides/diagnostics#_debugging-techniques

Changing connection timeouts to your origin

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

> **NOTE:** Fastly enforces a 60 second timeout between nodes unless you're passing requests in `vcl_recv`.

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

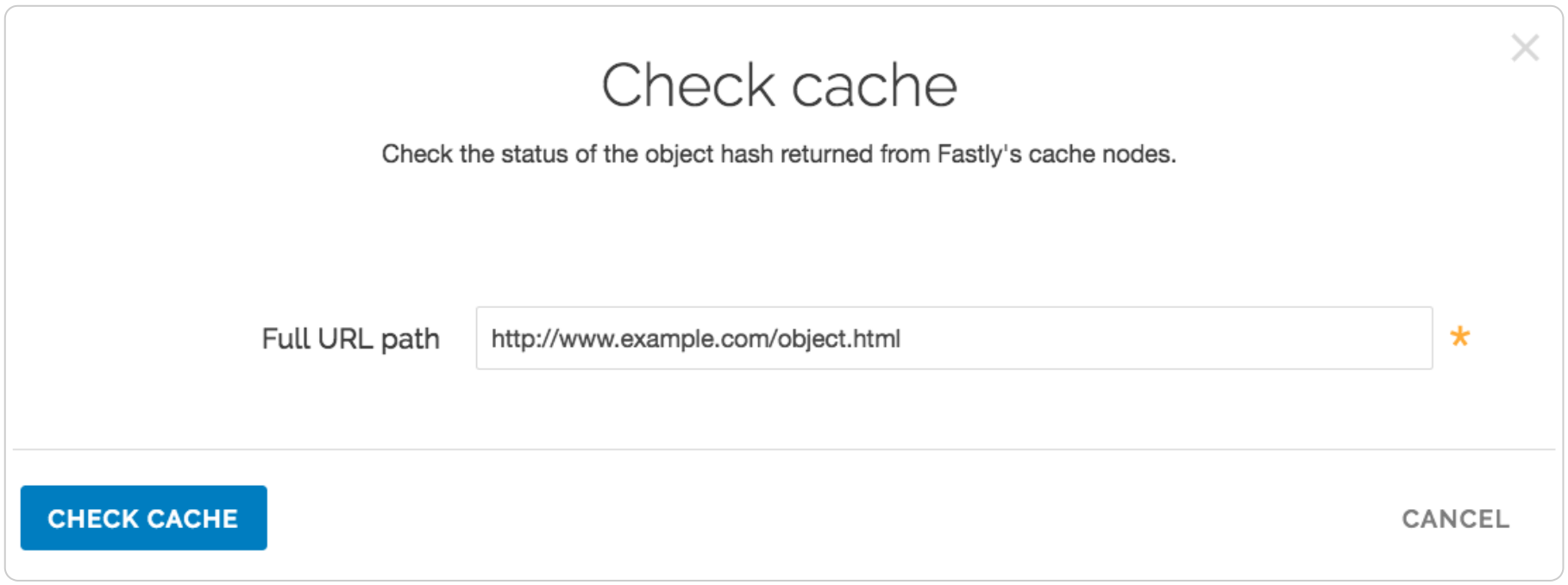
https://docs.fastly.com/en/guides/checking-cache

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 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.

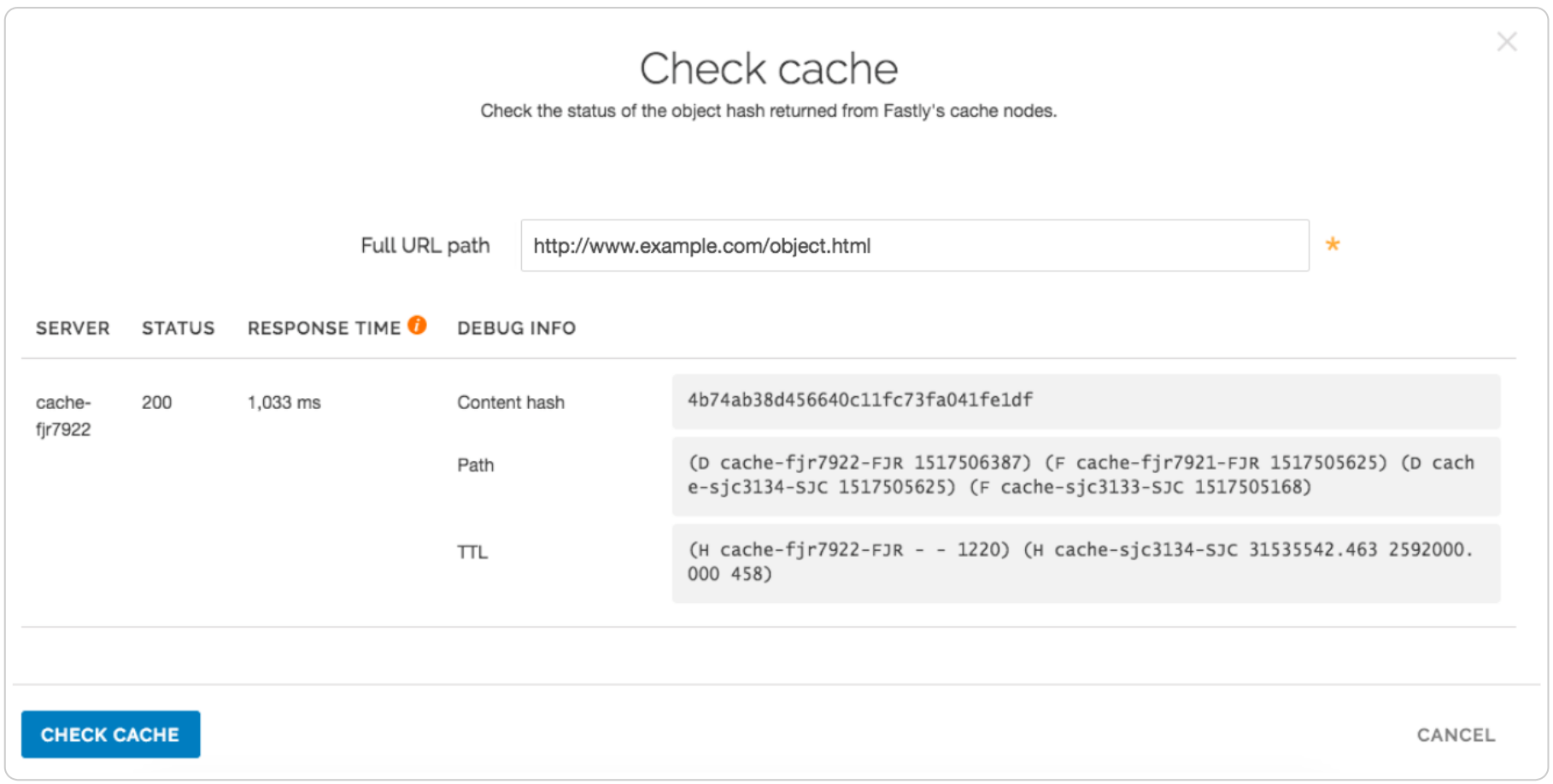
Using the web interface

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 **Configure** link.
2. Click the **Check Cache** button. The Check Cache window appears.



3. In the **Full URL path** field, type 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 cURL. We recommend using 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 147067
2629) (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 datacenter.

> **NOTE:** Our guide to understanding cache HIT and MISS headers provides in depth details key to understanding the `X-Served-By`, `X-Cache`, and `X-Cache-Hits` headers with shielded services.

Debugging with mtr

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 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:

```
 1  ~  mtr -c 20 -w -r www.fastly.com
 2  Start: Mon Feb  2 15:27:20 2015
 3  HOST: test-local-machine.local                Loss%   Snt   Last   Avg  Best  Wrst StDev
 4    1.|-- 10.100.20.2                            0.0%    20    2.1   2.2   1.6   4.3   0.5
 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
 6    3.|-- ae5.cr2.sjc2.us.zip.zayo.com           0.0%    20    4.6   6.5   2.9  35.3   7.7
 7    4.|-- ae10.mpr4.sjc7.us.zip.zayo.com         0.0%    20    4.7   4.8   3.6  14.5   2.3
 8    5.|-- be6461.ccr21.sjc03.atlas.cogentco.com  5.0%    20    5.1   5.9   4.2  15.3   2.6
 9    6.|-- fastly-inc.edge2.sanjose3.level3.net   0.0%    20    5.0   4.7   4.2   8.2   0.8
10    7.|-- ???                                   100.0    20    0.0   0.0   0.0   0.0   0.0
11    8.|-- 23.235.47.184                          0.0%    20    4.7  14.3   3.8  74.6  20.3
```

Debugging with WebPageTest

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.

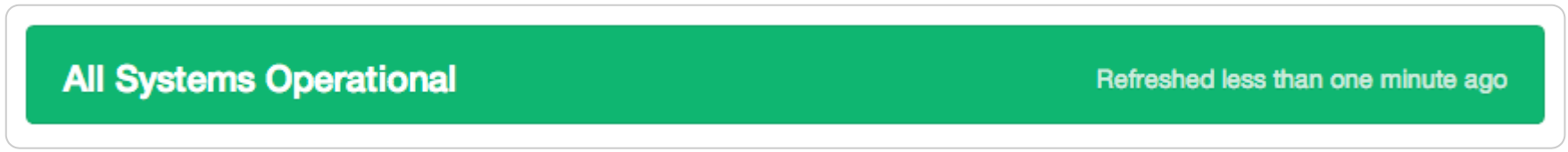
Fastly's network status

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.

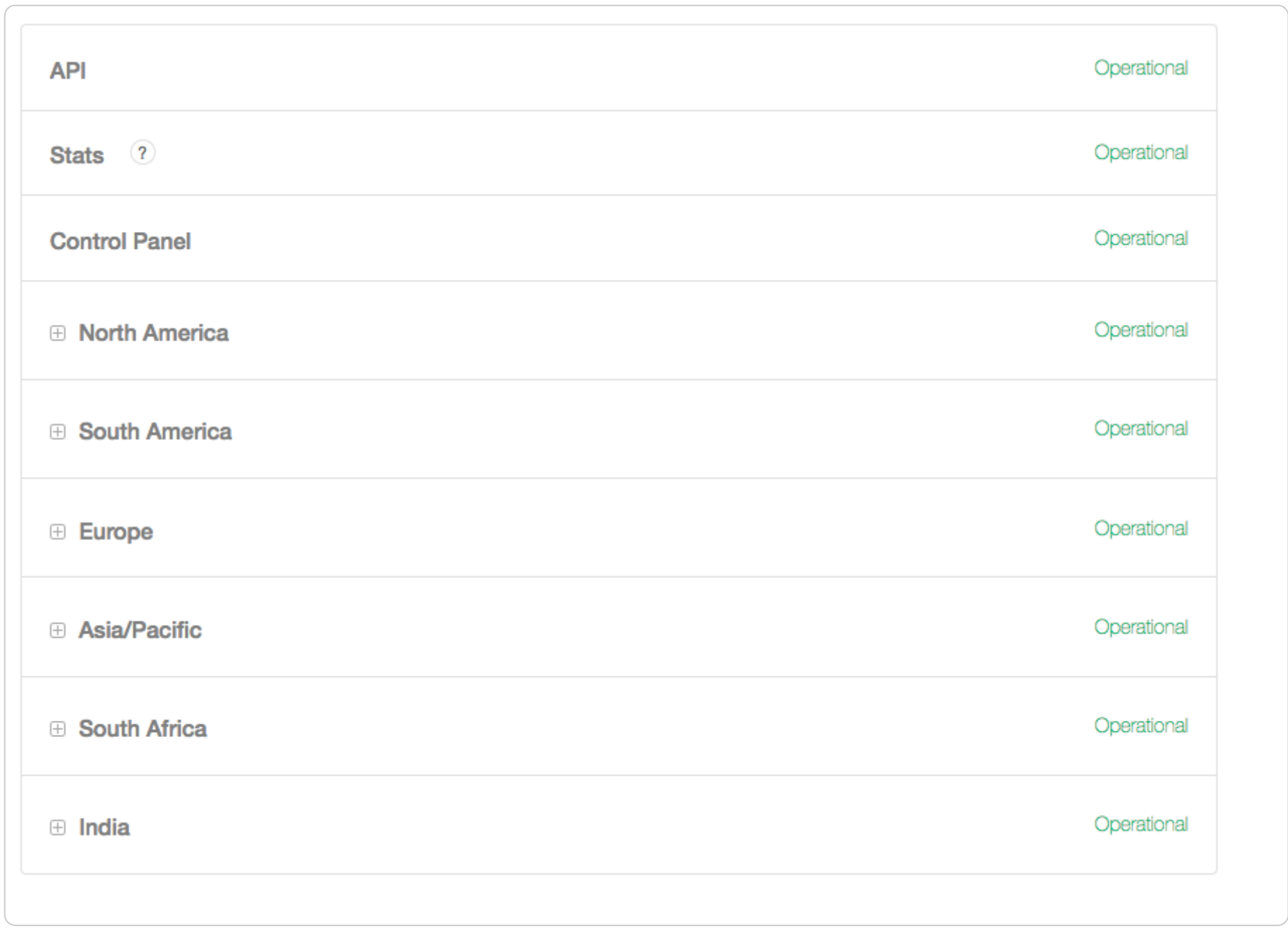
Overall system status

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.



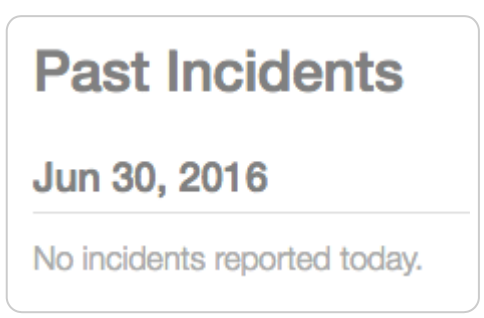
Individual component statuses

The status of the Fastly API, the Fastly web interface, statistics collection and delivery, and each Fastly point of presence (POP) appears immediately below the overall status. POPs are grouped by region. You can see the status of all POPs in a region by clicking the + icon next to the region's name.



Past incident statuses

Fastly keeps track of past incidents. Past incidents, if any, for approximately the past two weeks appear immediately below the individual component statuses.



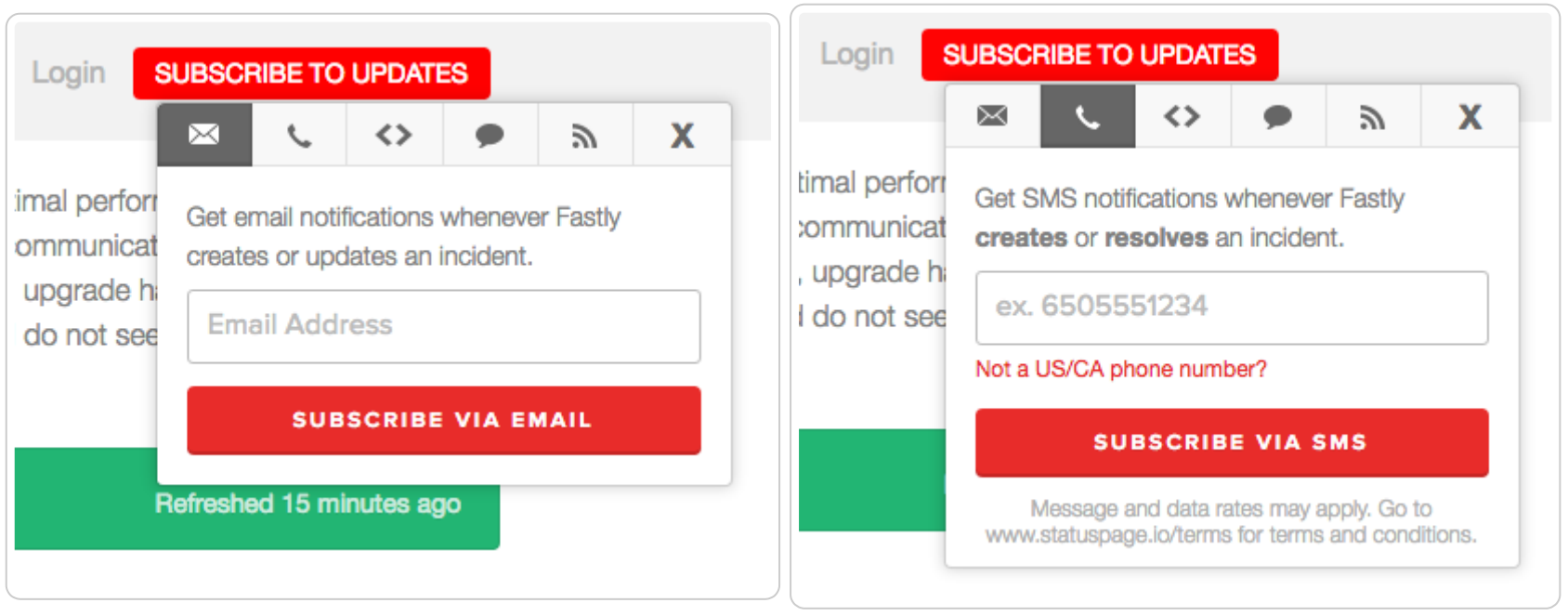
In addition to the textual description, each incident status appears in a color that indicates the level of service impact. The color indicators are as follows:

- Black - no component marked specifically as out of service or degraded
- Blue - scheduled maintenance
- Yellow - minor degradation or disruption
- Orange - significant degradation or traffic rerouting
- Red - component offline

We also keep track of all past incidents in an incident history page.

Subscribing to notifications

Fastly allows you to subscribe to status notifications via email or SMS text messaging. Simply click the **Subscribe to Updates** button in the upper right corner of the status page screen. Once subscribed, we'll email you any time we create or update an incident.



To subscribe to email notifications, click the letter icon, type your email address in the displayed field, and click **Subscribe Via Email**. You can unsubscribe at any time by clicking the unsubscribe link that appears at the bottom of every status email.

To subscribe via SMS text messaging, click the telephone icon, type your telephone number in the displayed field, and click **Subscribe Via SMS**. Unsubscribe from SMS text messaging at any time by replying STOP to any status message you receive.

Google Pagespeed module errors

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 Cache Control Tutorial

Googlebot crawl stats

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 "Fetch as Google" tool article 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

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
```

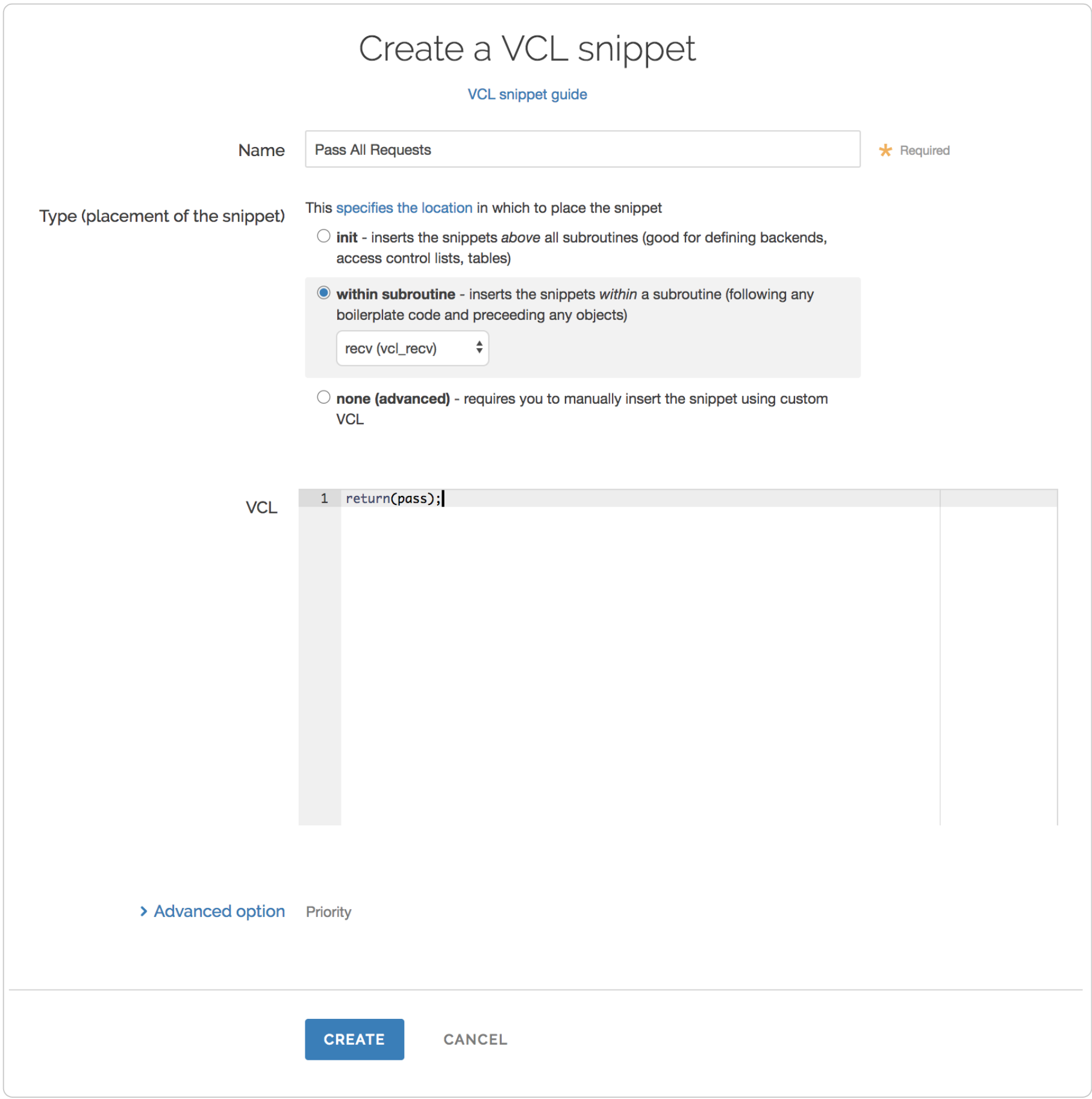
In addition, IE8 has some odd behavior to do with the back button. Adding `Vary: *` to the headers seems to fix the problem.

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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

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.

Common errors

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

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.

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, you'll want to extend your first byte timeout.

By default, the first byte timeout is set to 15 seconds. For cacheable objects, extend the timeout to a maximum of 60 seconds. For non-cacheable objects, disable clustering by calling `return(pass);` in `vcl_recv` to extend the maximum timeout to 600s. If the object is cacheable, you can disable clustering to increase the maximum timeout allowed by adding the `fastly-no-shield` header to the request 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 it will affect your cache hit ratio.

Origin configuration errors

The following describes typical origin configuration errors you may encounter.

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. This error may mistakenly appear instead of the backend.max_conn reached error the first time Fastly encounters the maximum number of connections to your backend. 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 service and domain errors

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:

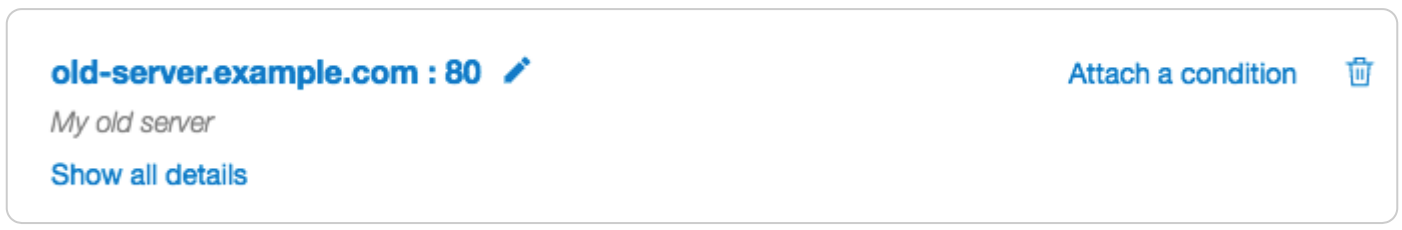
```
{
  "msg": "An error occurred while connecting to the fastly API, please try your request again.",
  "detail": "Exceeding max number of domains: 10"
}
```

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.

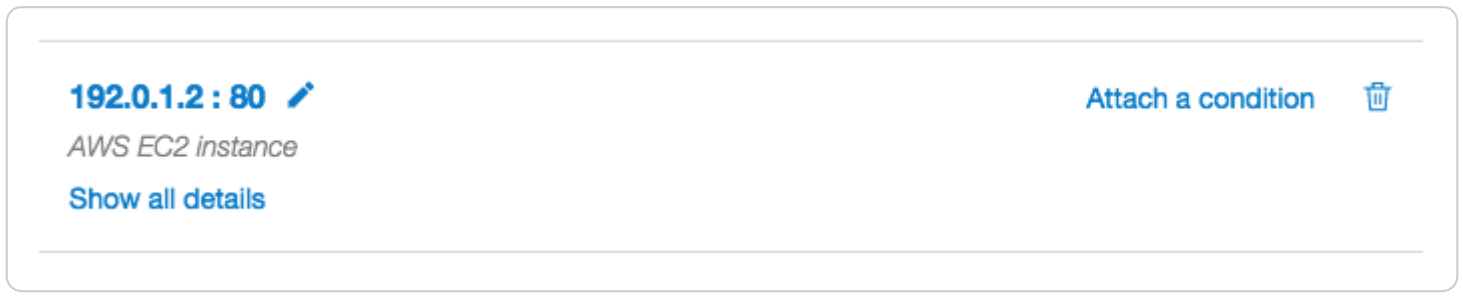
Error 1000 with CloudFlare DNS

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";
}
```

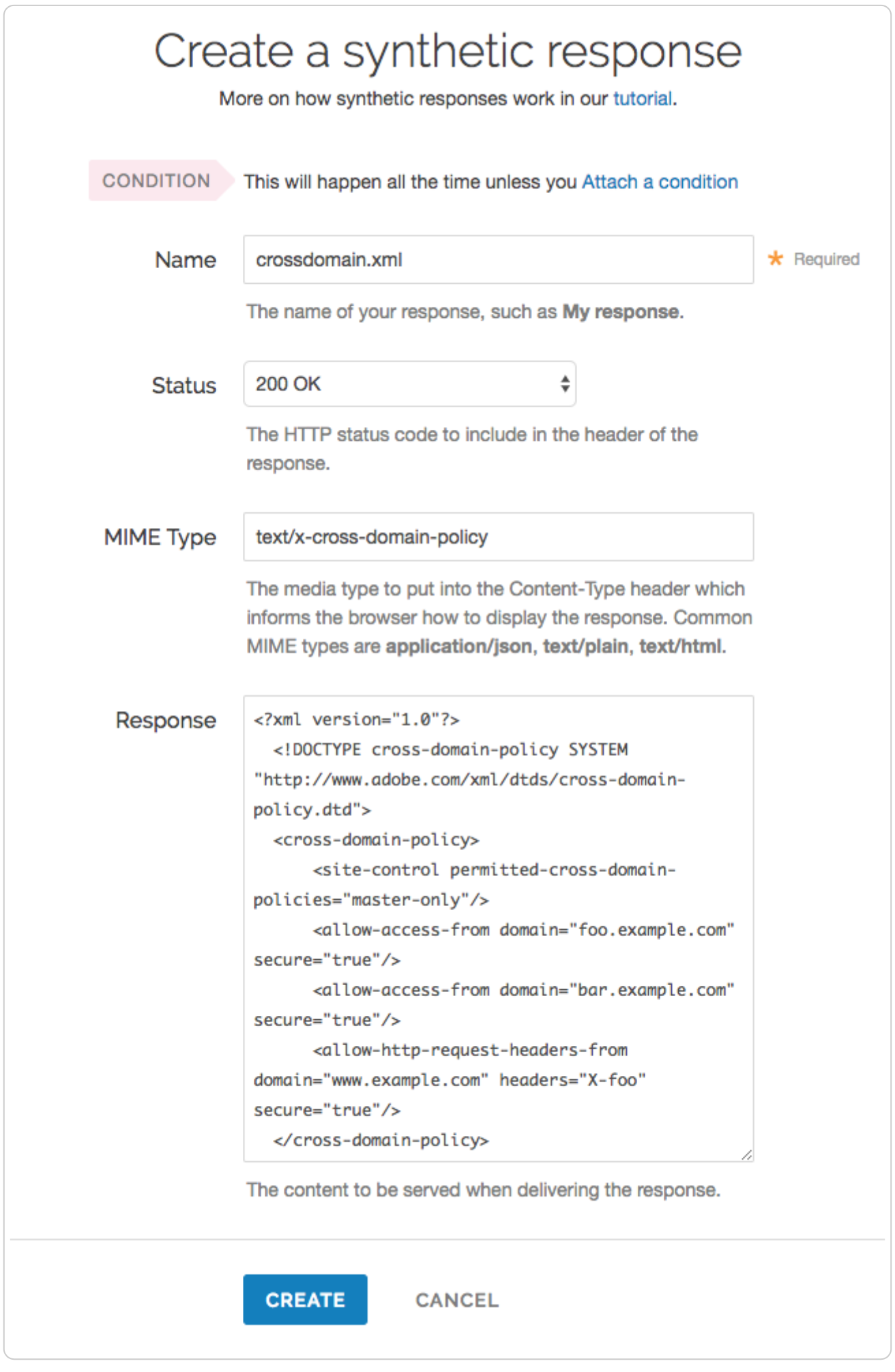
Fixing cross-domain errors

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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, type `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, type a human-readable name for the response condition. For example `crossdomain.xml`.
   - In the **Apply if** field, type `req.url == "/crossdomain.xml"`.
10. Click **Save and apply to** 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.

Various permissive and restrictive examples of `crossdomain.xml` files appear in Adobe's information on Cross-domain XML for streaming.

Loop detection

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.

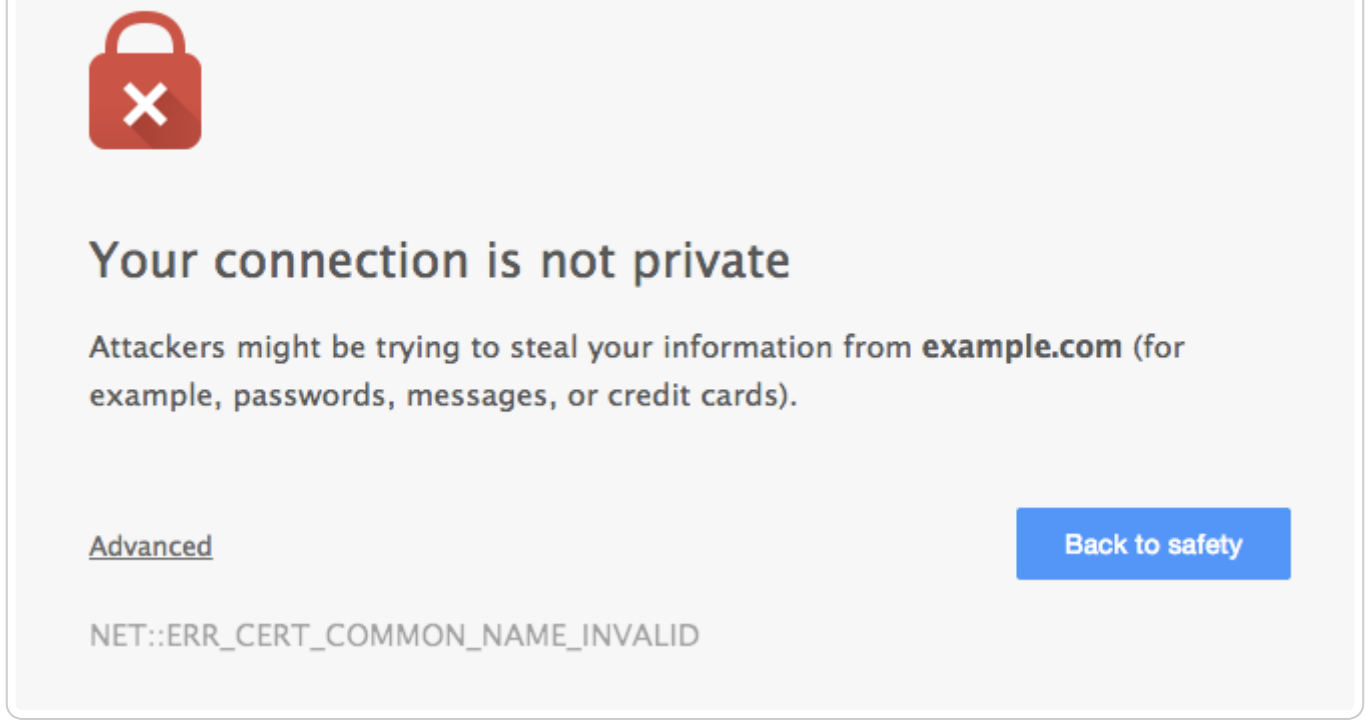
```
HTTP/1.1 503 Loop detected
Error-Reason: loop detected
Connection: close
Content-Type: text/plain
Fastly-Host: <hostname>
Fastly-FF: <hostname>
Server: Varnish
```

TLS certificate errors

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

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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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, type 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`, type 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 Certificate 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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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 1.2, TLS 1.1, and TLS 1.0.
- Confirm that you can connect to your origin. For example, if you're using TLS 1.2, enter a command like `echo Q | openssl s_client -connect ${IP}:443 -tls1_2`. To test other versions of TLS, you can replace `-tls1_2` with `-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 and click the **Configure** link.
2. From the service menu, select the appropriate service.
3. Click the **Edit configuration** button and then select **Clone active**. 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.

Account info

These articles describe how to manage account access, billing, and security.

https://docs.fastly.com/en/guides/account-info

Account management

These articles describe how to manage account access.

https://docs.fastly.com/en/guides/account-info#_account-management

Account lockouts

https://docs.fastly.com/en/guides/account-lockouts

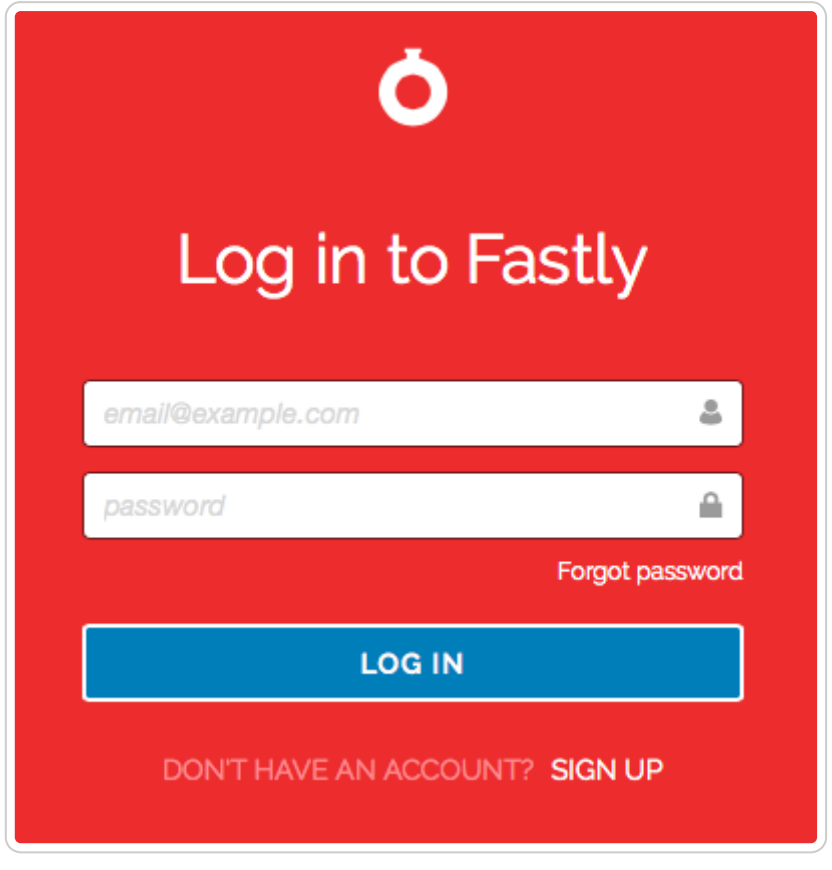
Why is my account locked?

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.

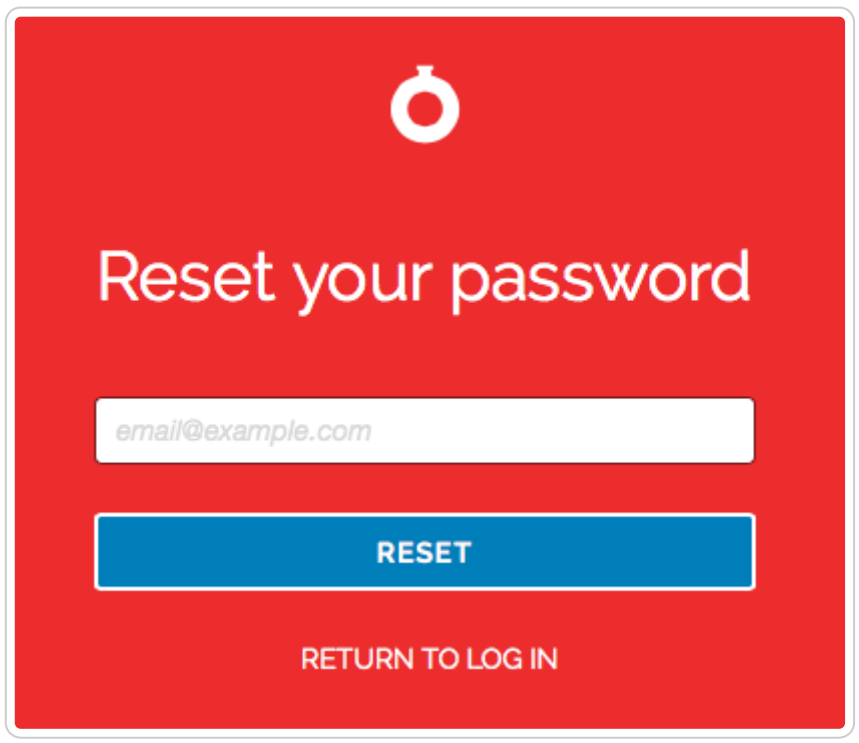
I am not using two-factor authentication. How can I access my account?

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:

1. Point any standard web browser to the Fastly login page. The login controls appear.



2. Click the **Forgot Password** button underneath the password field. The Reset your password page appears.



3. In the **Email (login)** field, type the email address you normally use to log in to your Fastly account, and then click the **Reset** button. Password reset instructions will be emailed to you.
4. Click on the password reset link in the emailed instructions that the system sends you. The Reset Your Password page appears.
5. Click the **Reset Password** button. The system sends you a temporary password to the email address you supplied.
6. Using the temporary password you receive, log in to your account. The controls to create a new password appear.
7. Fill out the **Reset Password** controls as follows:
   - In the **Current Password** field, type the temporary password that the system emailed to you when you requested a password reset.
   - In the **New Password** field, type a new password to replace the temporary password you were sent.
   - In the **Confirm Password** field, type the new password a second time to confirm it.
8. 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 your account's company name

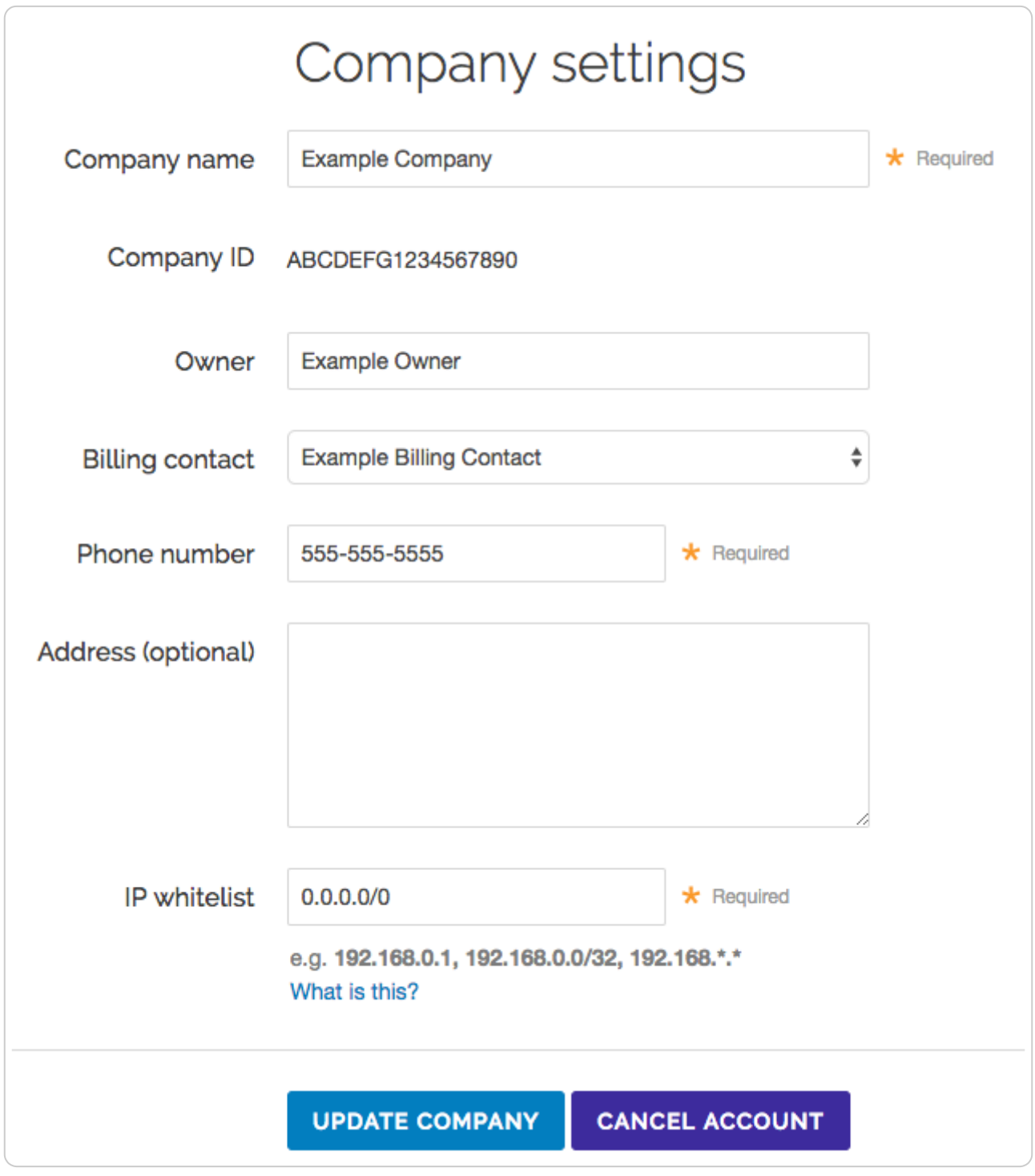
https://docs.fastly.com/en/guides/changing-your-accounts-company-name

Fastly allows you to change your account's company name at any time after it's been created.

IMPORTANT: You must be the account owner or be assigned the role of superuser to change your account name.

To change the company name, 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 **Company name** field of the **Company settings** area, replace the current company name.



3. Click the **Update Company** button.

Enabling an IP allowlist for account logins through the web interface

https://docs.fastly.com/en/guides/enabling-an-ip-allowlist-for-account-logins-through-the-web-interface

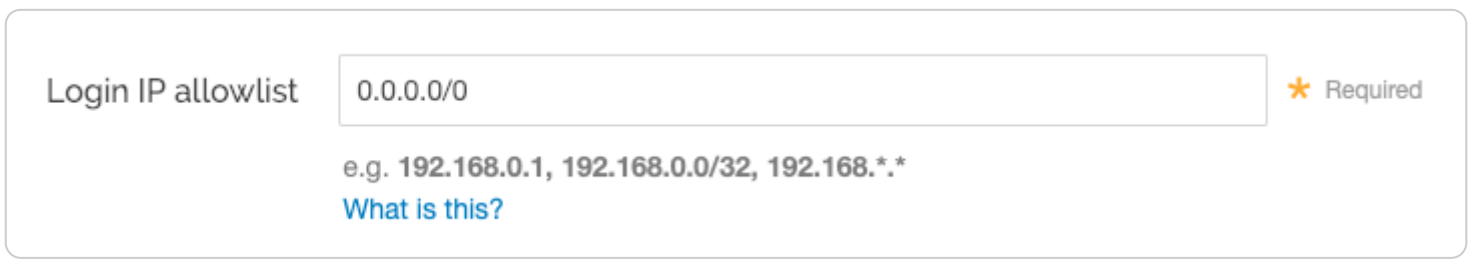
Fastly allows you to define the range of IP addresses authorized on your Fastly account from which users are able to login to the Fastly web interface. This optional IP allowlisting functionality is not enabled by default. It can restrict access to Fastly's API endpoints.

⚠ **WARNING:** ***If you decide to use optional IP allowlisting, your account owner must have a valid telephone number on file.*** 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. 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***.

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, type `0.0.0.0/0` (the default IP range indicating no allowlisting).
3. Click the **Update Company** button.

Enabling and disabling two-factor authentication

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

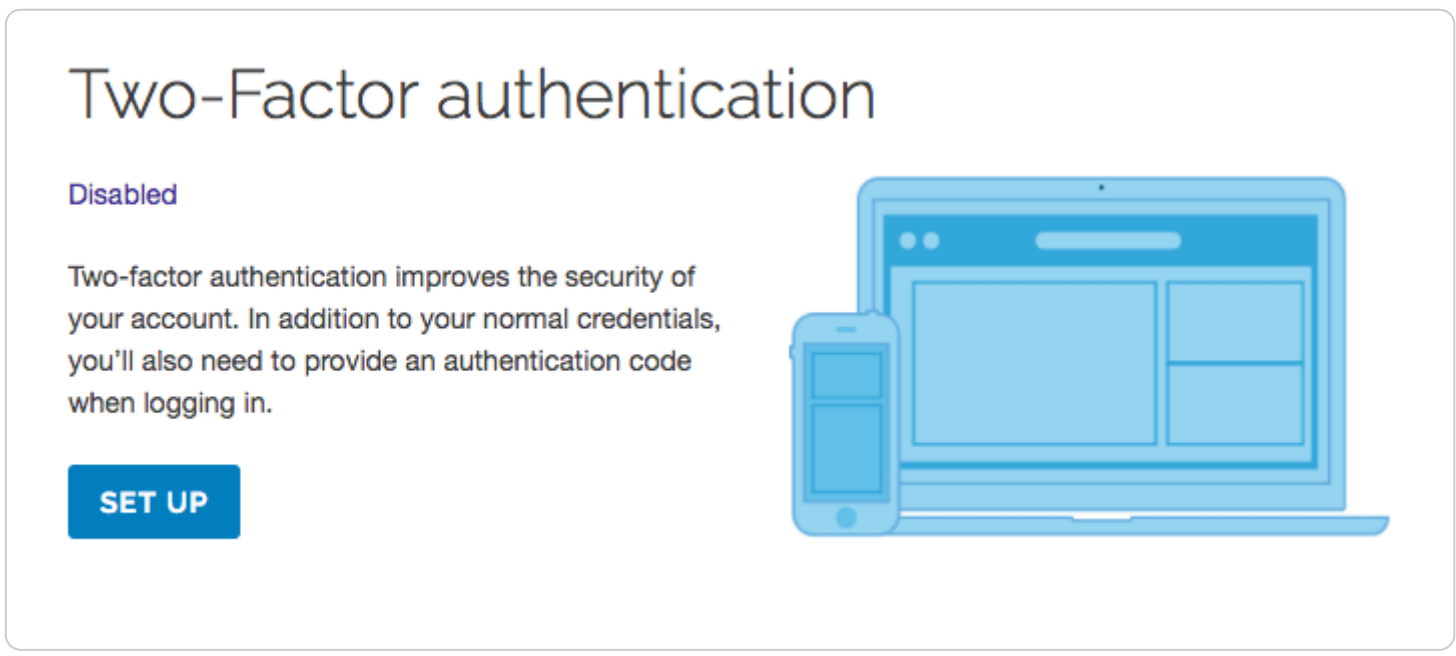
Depending on whether or not your organization has enabled company-wide two-factor authentication, you may be able to enable and disable two-factor authentication for your personal account. We also have instructions for recovering access to your account if you lose your mobile device.

Enabling two-factor authentication

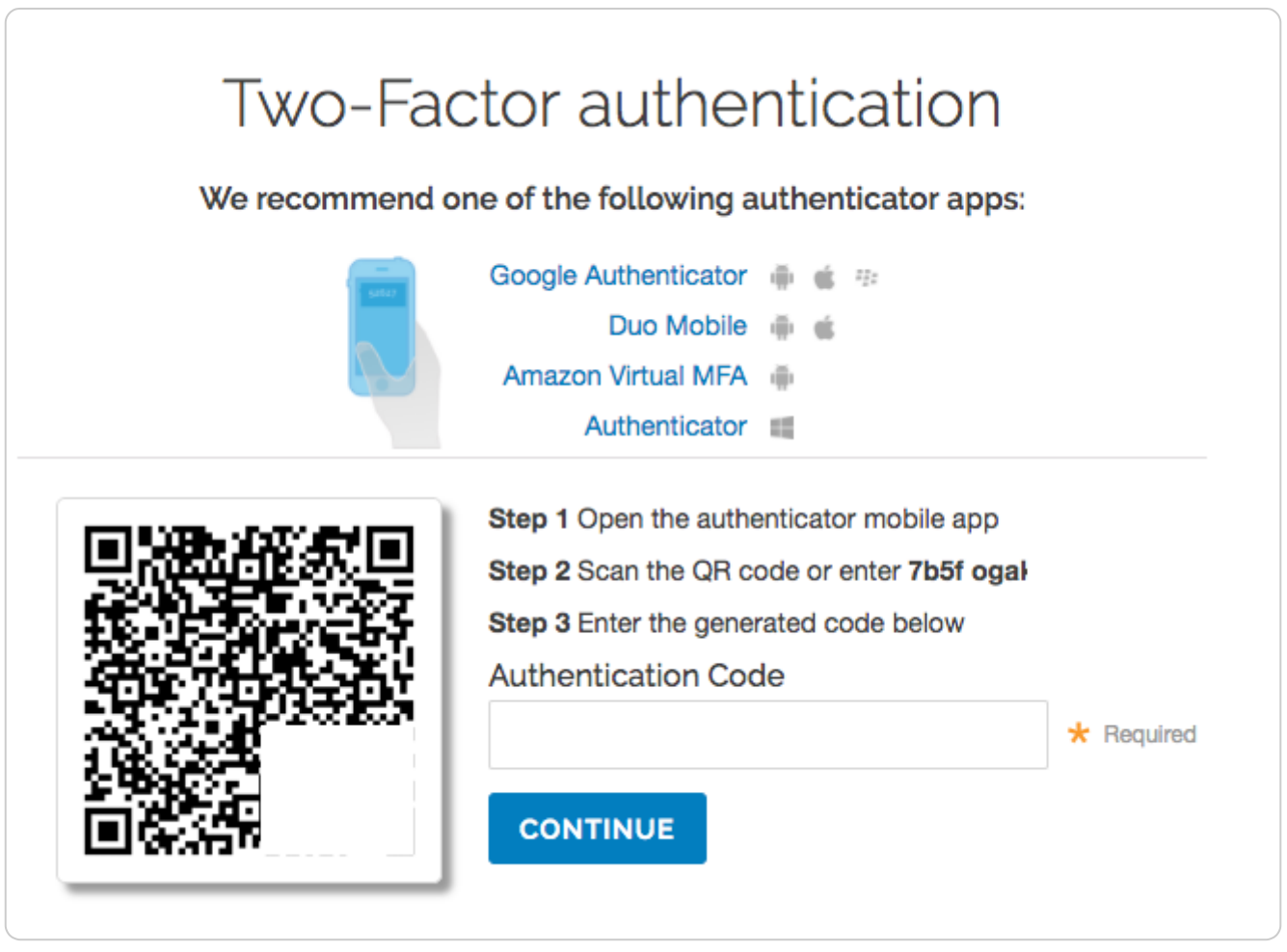
To enable two-factor authentication for your user account, follow the steps below.

> **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.

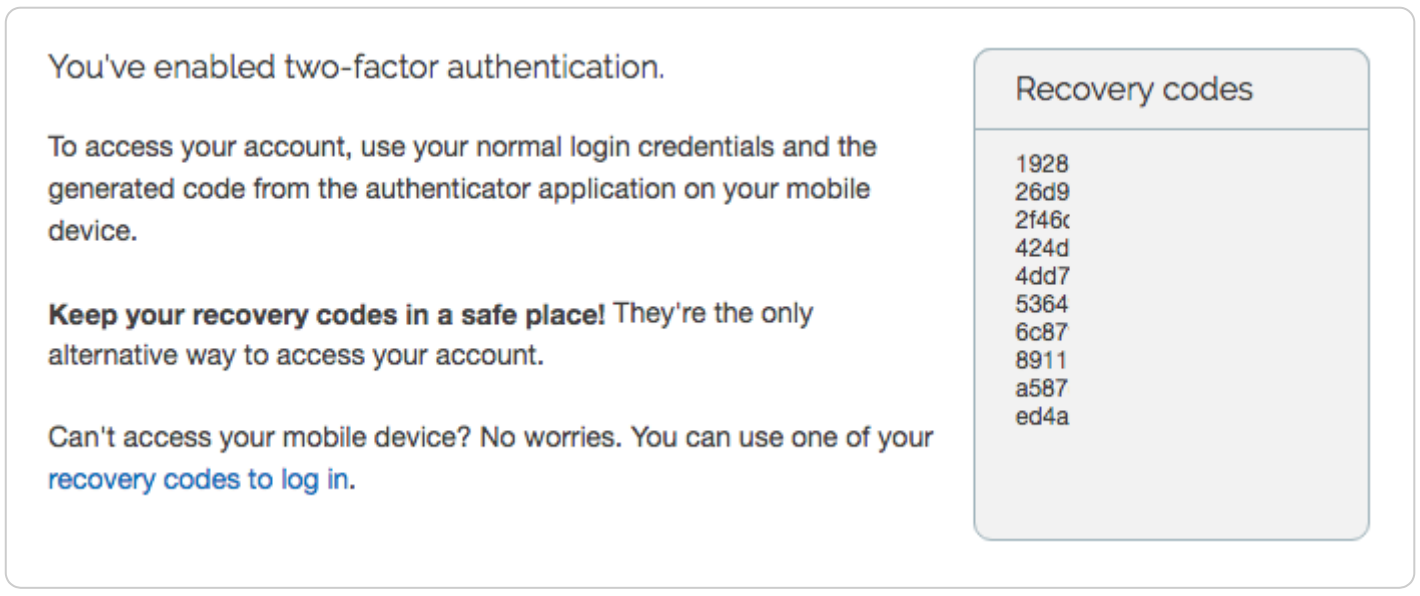


> **IMPORTANT:** The QR code above is an example. Scan the one that appears in the Fastly application, not in this guide.

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, type 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.

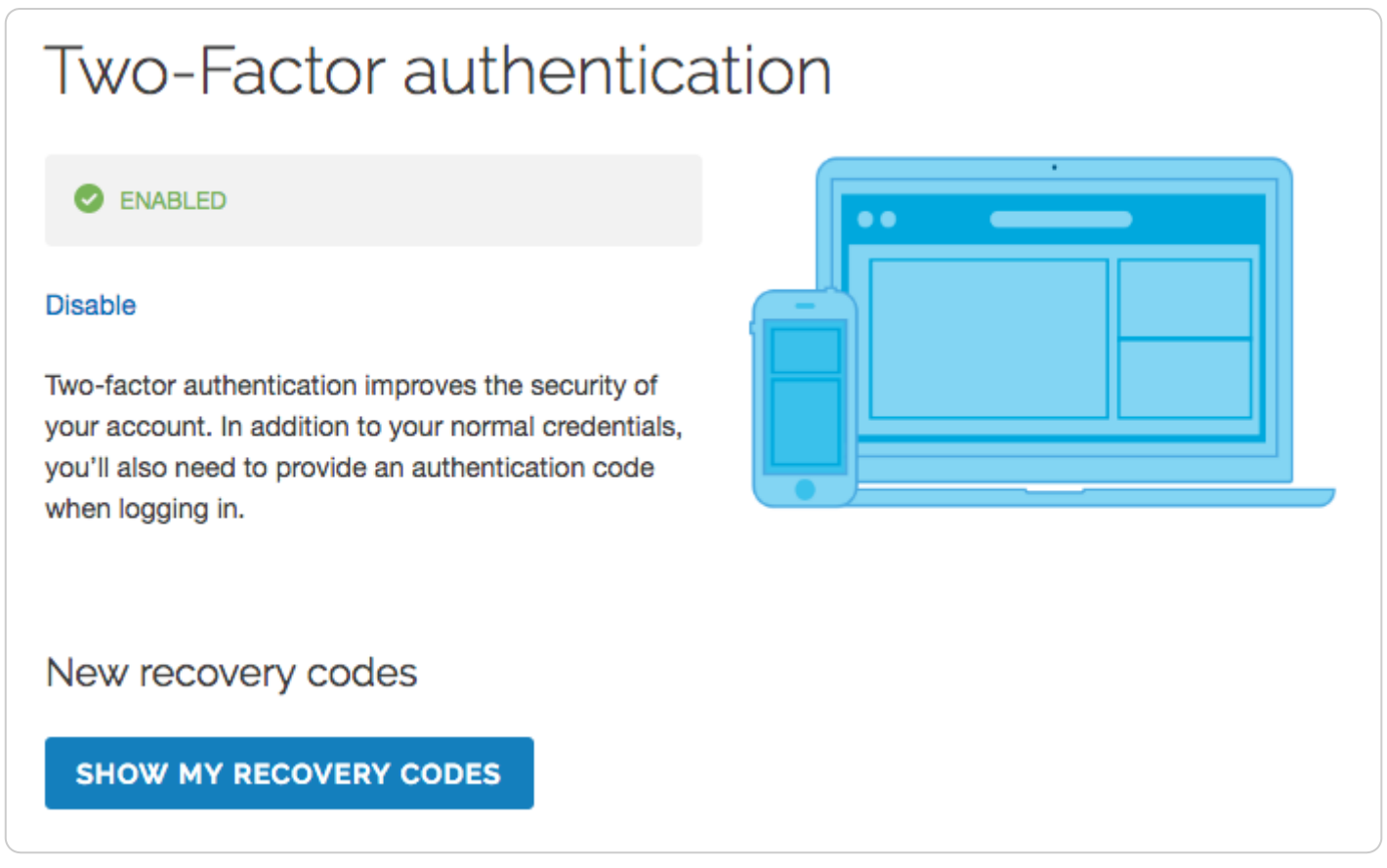
Once you enable two-factor authentication for your account, any other open sessions will require reauthentication. For example, if you enable two-factor authentication in one browser window and you're viewing various aspects of your service through multiple additional browser windows, you will be required 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, type 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.

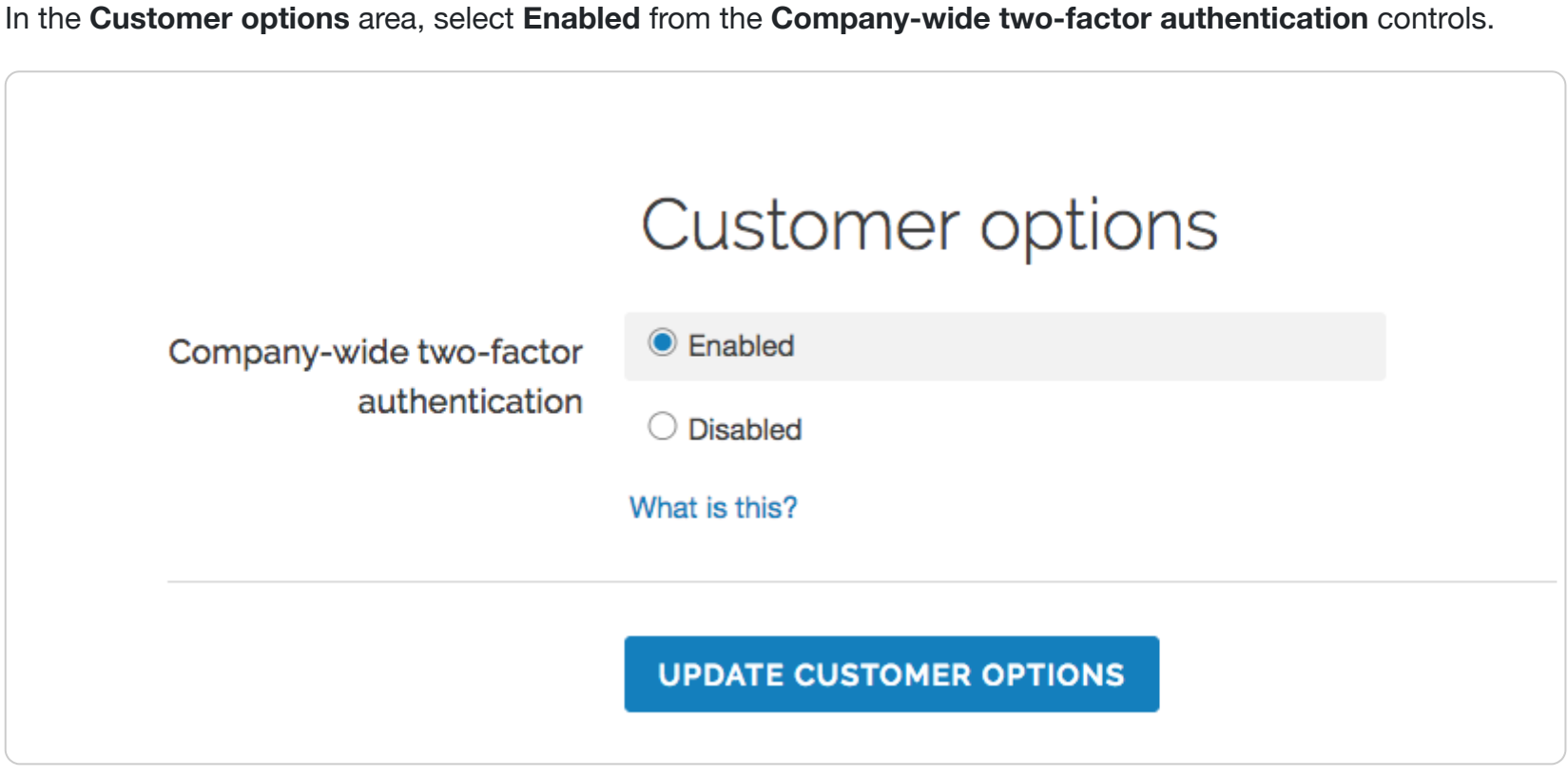
Managing two-factor authentication as a company

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

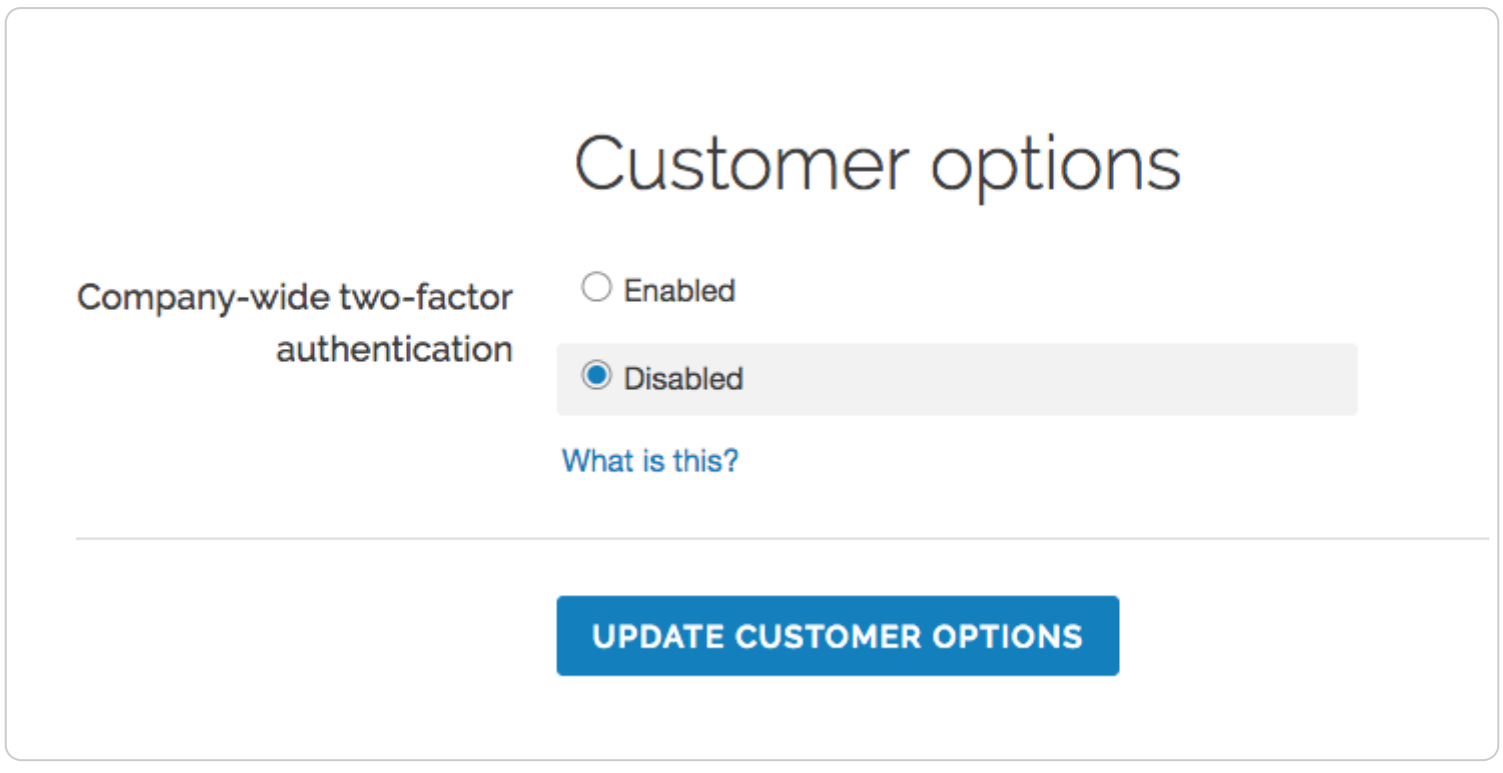
If company-wide two-factor authentication is enabled, a superuser can disable two-factor authentication for a single user's account. This is typically done for user accounts being used for scripts and session authentication. To disable two-factor authentication for a single user's account, follow the 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.

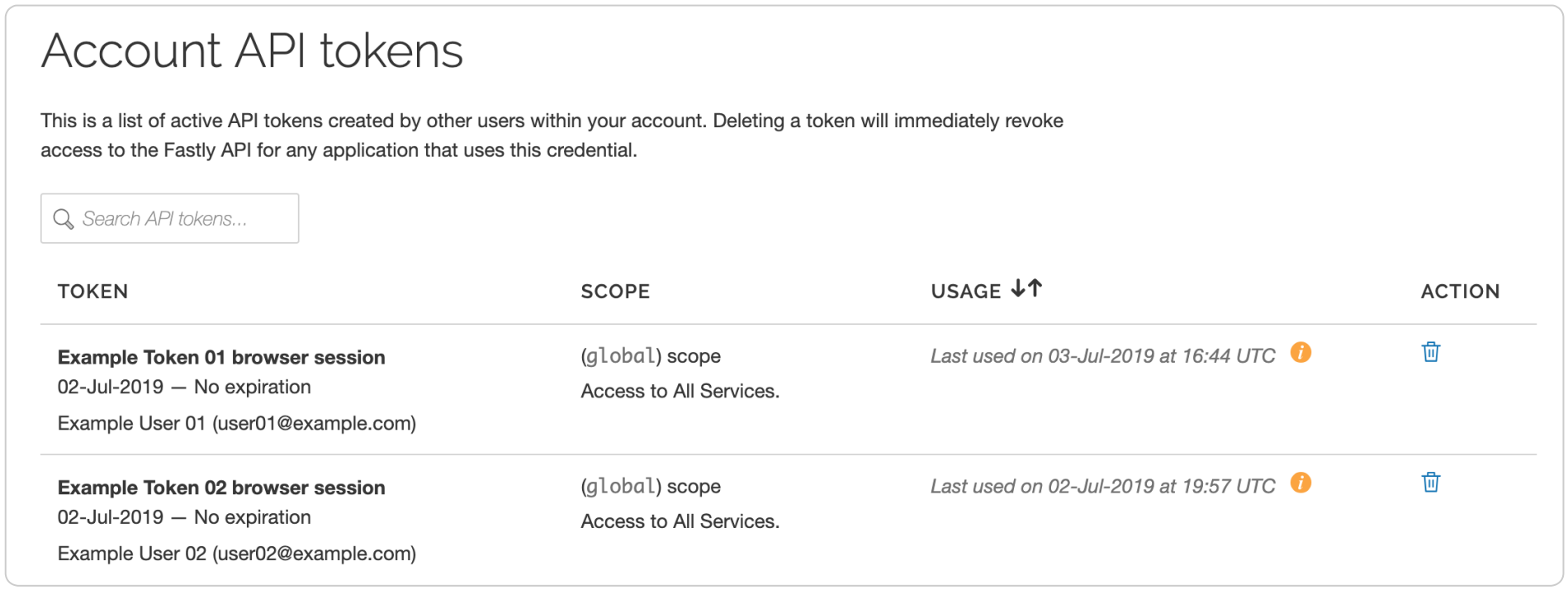
Finding and managing your account info

https://docs.fastly.com/en/guides/finding-and-managing-your-account-info

Account information, including your service ID and your customer ID (also called your company ID) can be accessed directly from the Fastly web interface.

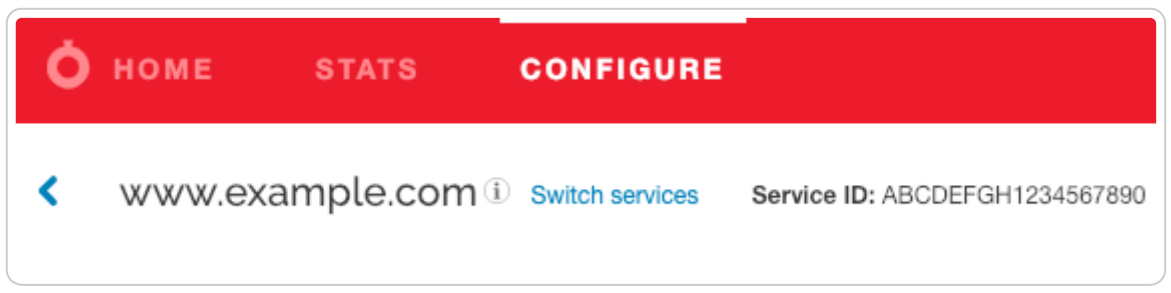
Finding your API tokens

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.



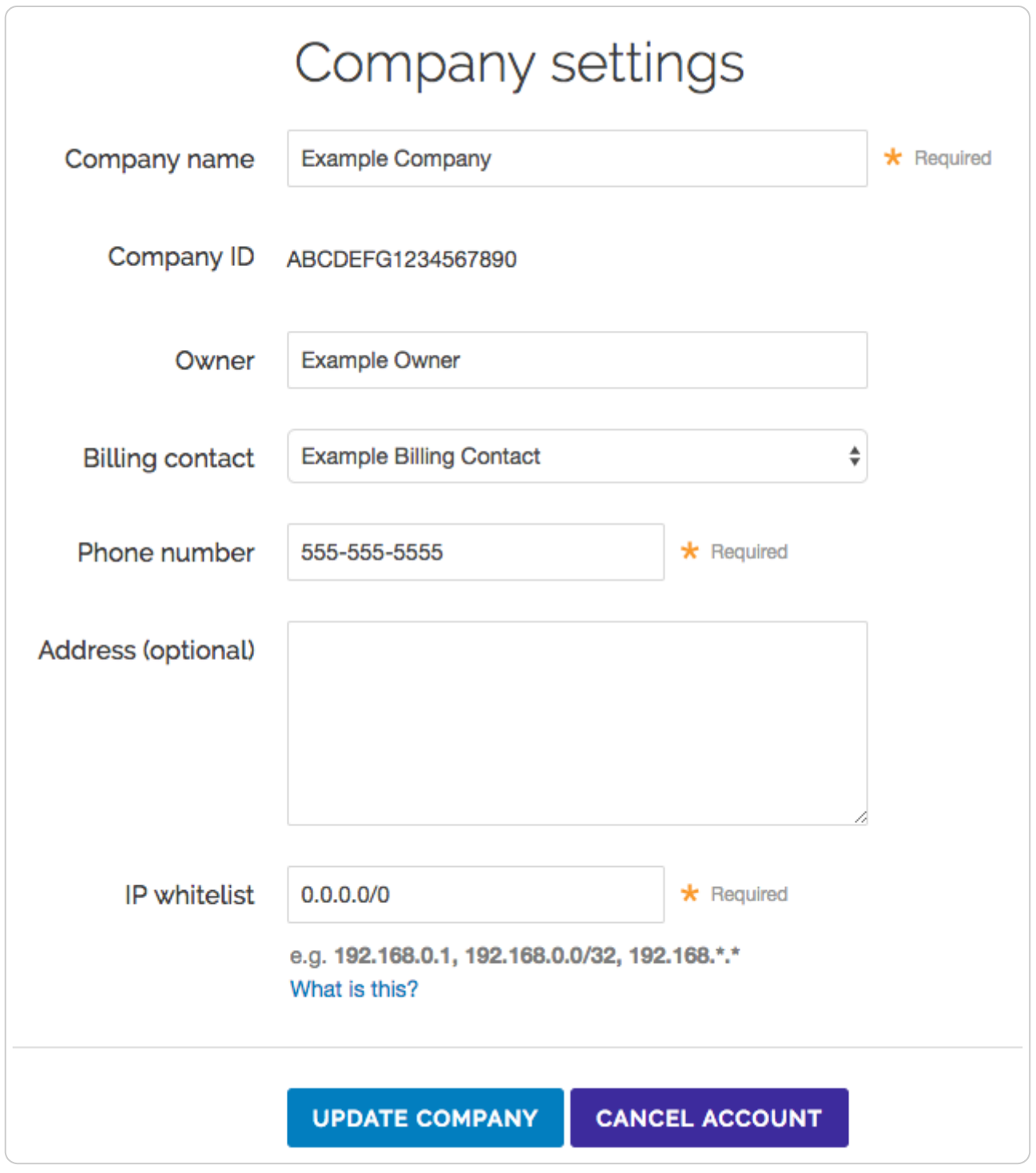
Finding your service ID

Your **Service ID** appears next to the name of your service on any page.



Finding your customer ID

Your **Company ID**, also called your **Customer ID**, appears in the **Company settings** of your **Account** page, which you access by clicking the **Account** link in the user menu:



> **Using API tokens**
>
> 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.

The API Token Management page allows you to create, view, and delete API tokens associated with your personal account. Superusers can view and delete any of the API tokens associated with the organization's Fastly account.

> **TIP:** You can also use the Fastly API to create and manage API tokens.

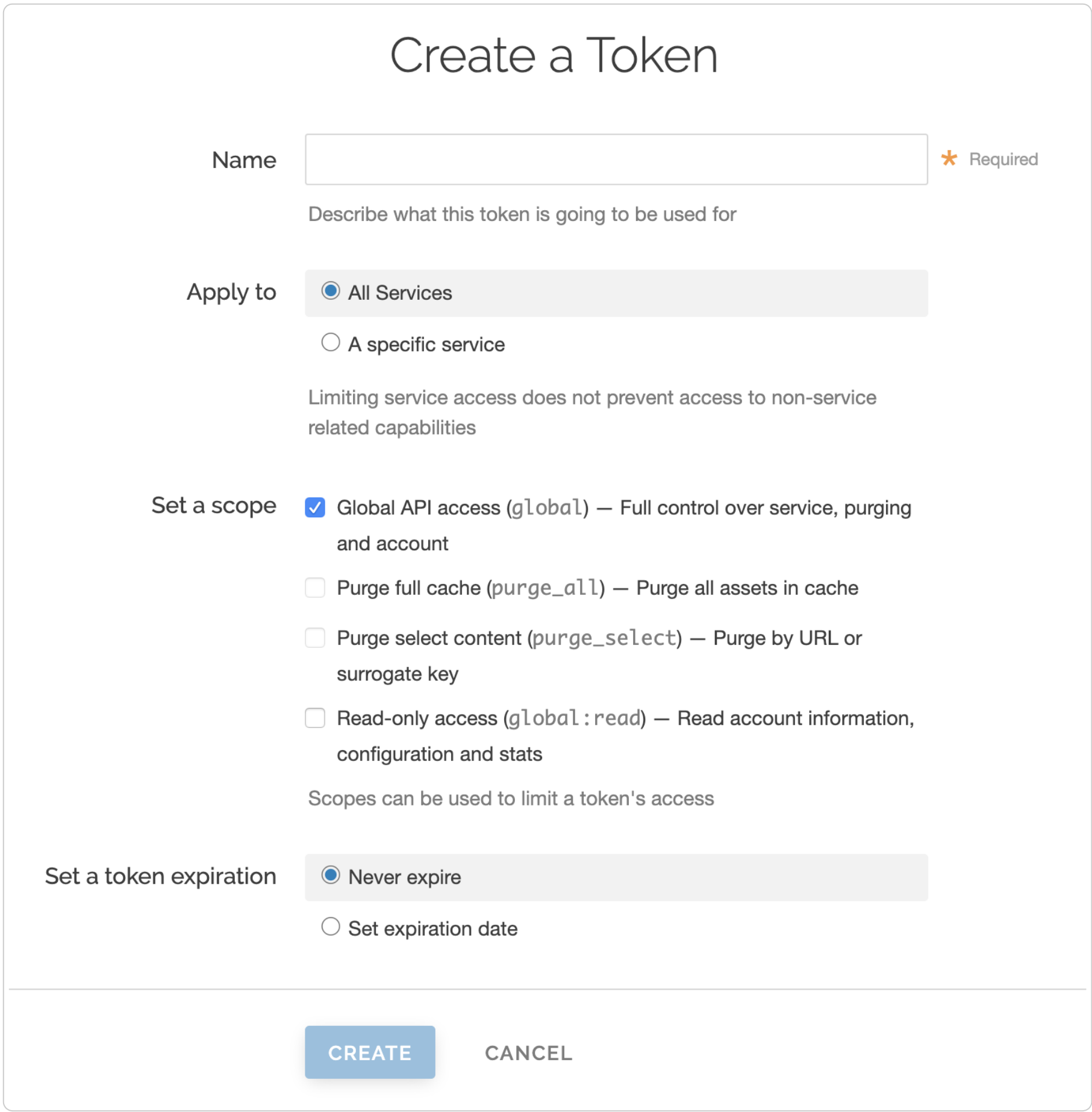
Best practices

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, type a descriptive name for the API token that indicates how or where you will to use the token.
   - In the **Apply to** area, optionally select a service to restrict the service-level access of the token to one service.
   - 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 expire on a specified date. 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.

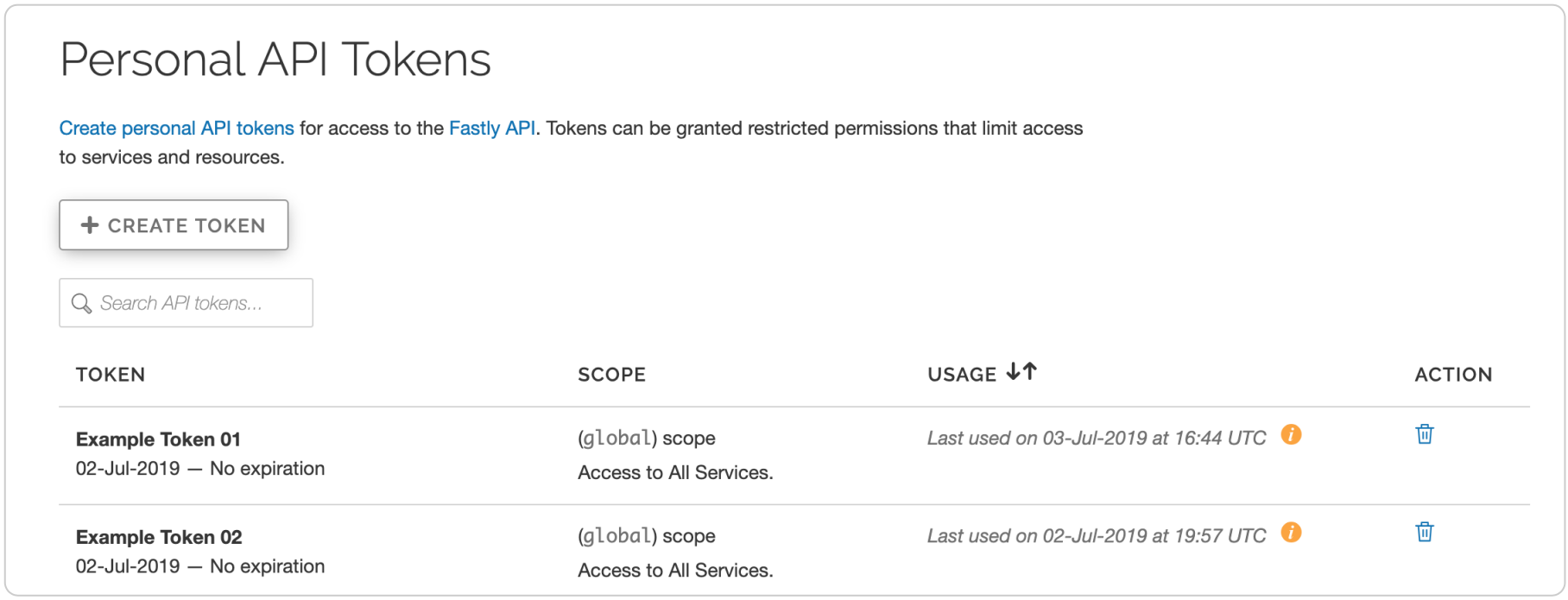
Viewing API tokens

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:

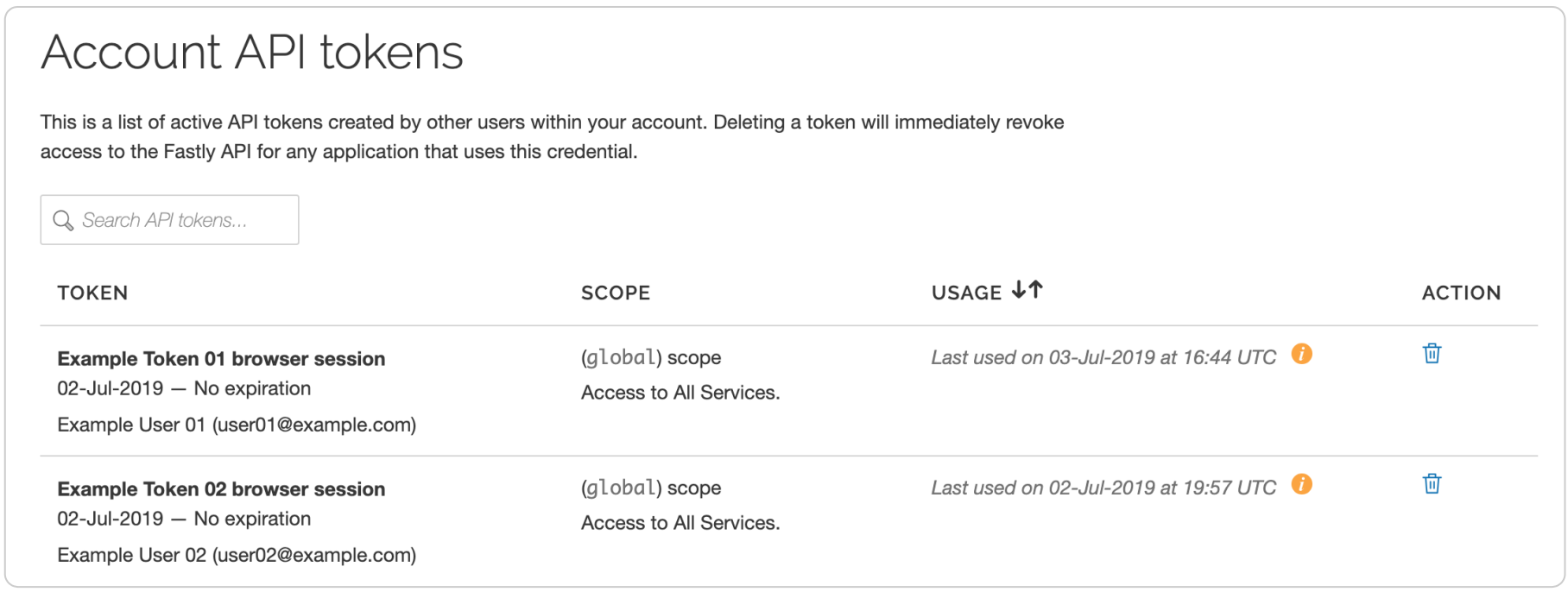
1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **Personal API tokens** link. The Personal API tokens page appears with a list of your personal tokens.



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 **Account API tokens**. 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**. 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`.

Billing

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

https://docs.fastly.com/en/guides/accounts-and-pricing-plans

Types of accounts and plans

Fastly offers a variety of accounts and pricing plans, which we detail below. To estimate your monthly charges using our pricing estimator, see our pricing page.

Free developer trials

We offer a development trial that allows you to test our services free of charge. Simply sign up for a trial and begin testing. We allow you to test up to $50 of traffic for free to ensure everything fits your requirements.

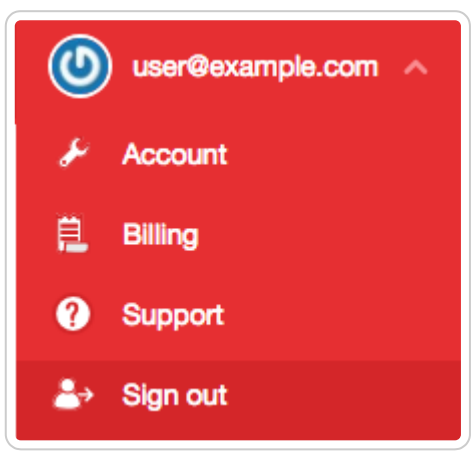
Once your testing is complete and you're ready to start pushing production traffic our way, you can switch your account to a paid account by adding your credit card information. Some add-on options (our TLS certificate options, for example) require you to switch your account to a paid account before that functionality becomes available to you.

Paid accounts without contracts

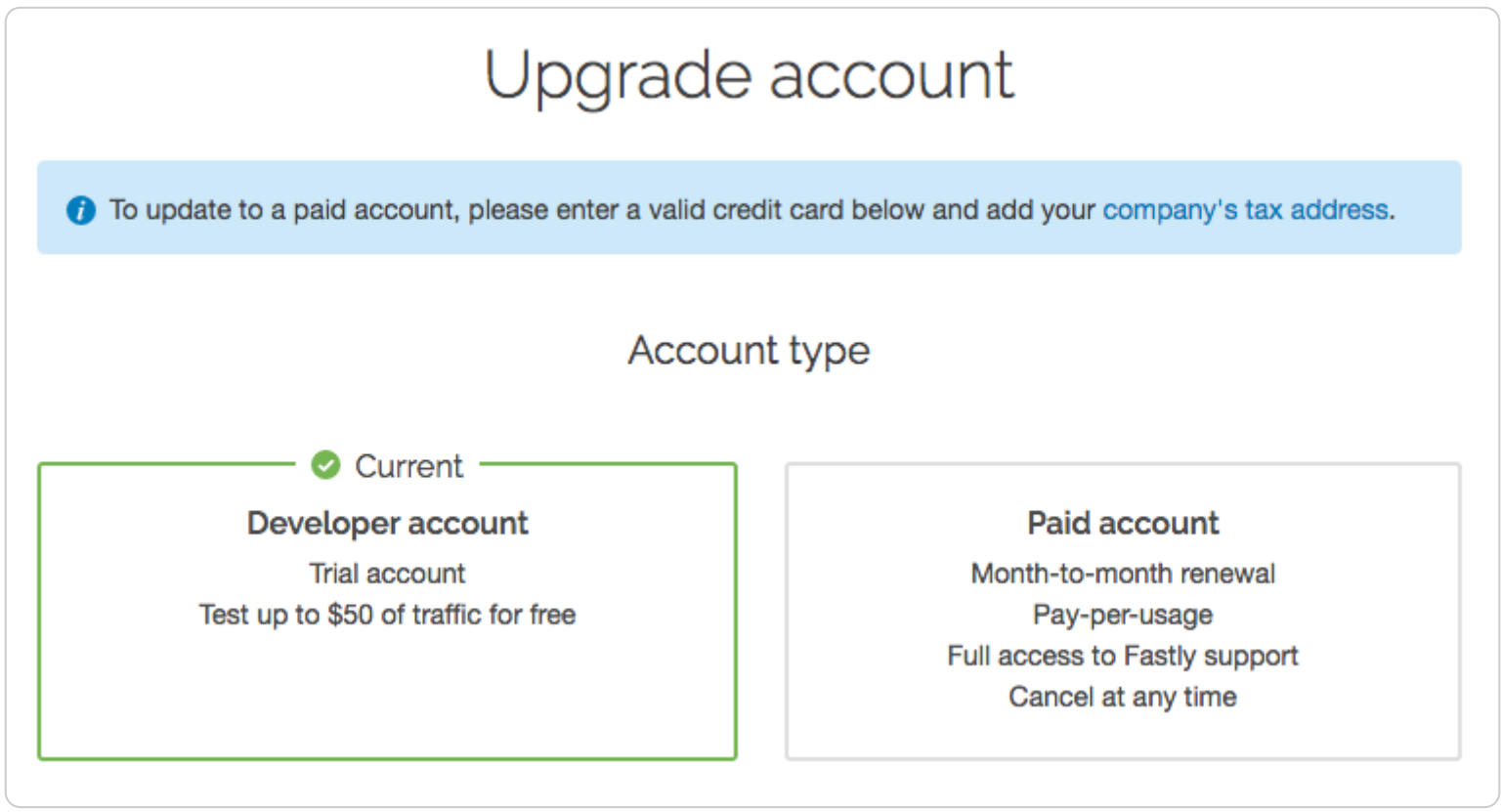
After your trial period ends, you can use Fastly's services on a month-to-month basis without having to sign a contract. Be sure you've provided us with your current billing address as well as your credit card information.

To switch from a developer trial to a paid account without a contract, follow the steps below:

1. Log in to the Fastly web interface.
2. From the user 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.

Once you switch to a paid account, the developer account plan option disappears and we'll begin billing you automatically at the end of every month using your credit card information. You can cancel your paid account at any time.

Paid accounts with contractual commitments

If you plan to push at least 2TB of data per month and require one of our TLS service options, or if you plan to push a minimum of 4TB of data per month, it might be worthwhile to consider a contract with Fastly. Contact us at sales@fastly.com for more information. We also offer solutions targeted to the needs of specific industries.

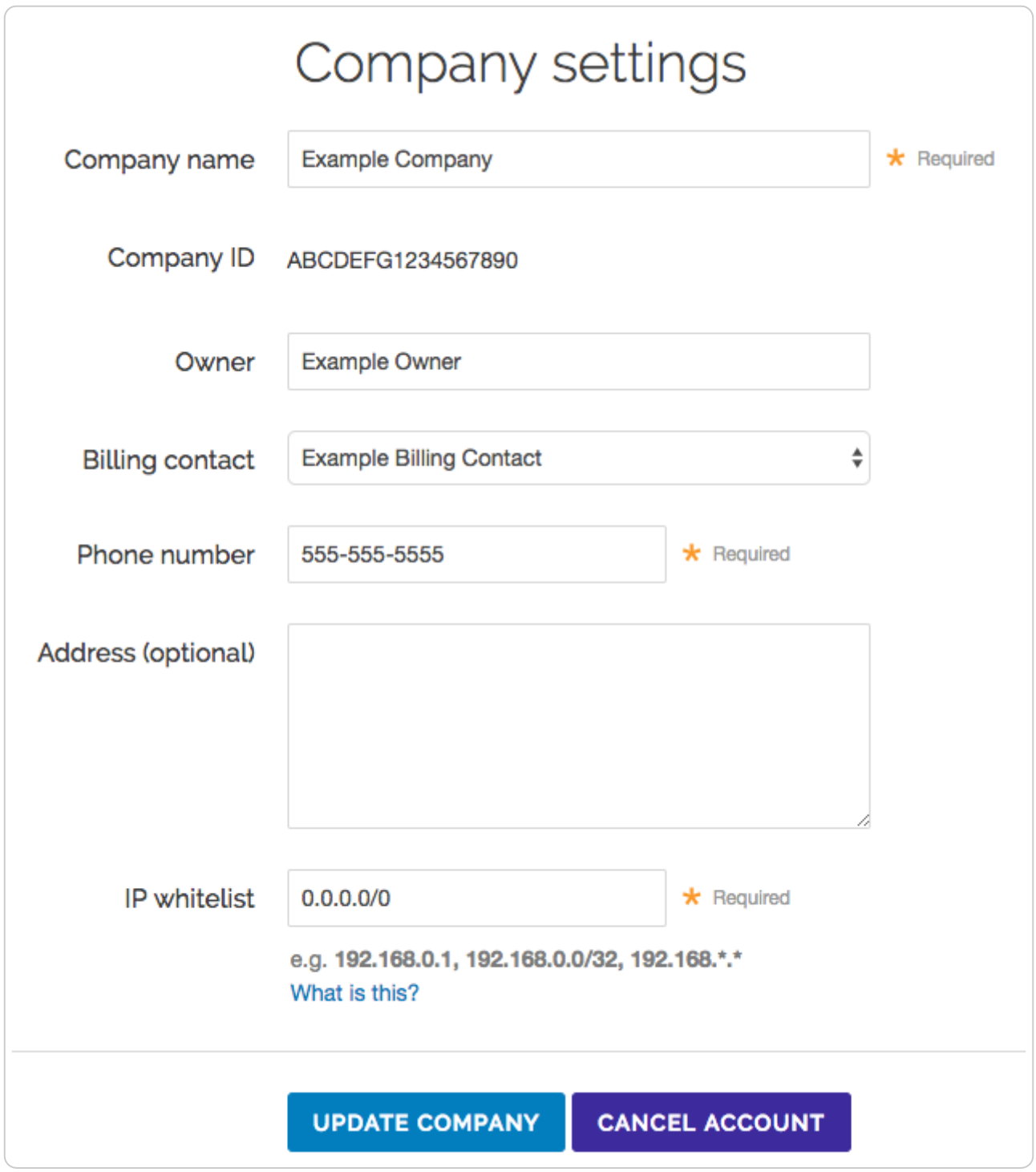
Free open source developer accounts

We're active open source contributors and support the community whenever possible. If you're an open source developer, your bill is on us. Contact us at community@fastly.com to get started.

Canceling your account

To cancel your account, have your account owner follow the steps below:

1. Deactivate and then delete all services on your account.
2. If you're using a TLS certificate, do the following:
   - If you've purchased one of Fastly's shared certificate options, delete your TLS domains.
   - 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, type 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 Customer Support and request reactivation.

How we calculate your bill

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. 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.

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.

Two specific settings related to responses may affect the total charges on your bill. Enabling gzipping can reduce the size of responses which reduces the bandwidth you use and thus can reduce your total bill. Enabling shielding may initially result in greater bandwidth use because requests may need to travel between POPs. The reduced load on your origin servers, however, frequently

offsets this increased cost and the potential increase in your bill's total.

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.

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

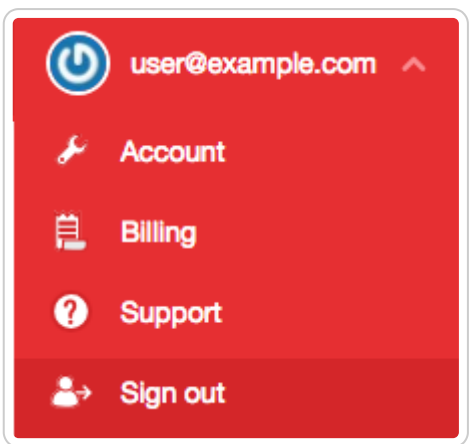
Fastly bills in arrears, not in advance, meaning that we bill you for services after you've used them, not before. For example, if you sign up for and start using Fastly services in January, the bill you receive in February reflects January's charges and services, your March bill reflects February's charges services, and so forth.

How account cancellation affects your bill

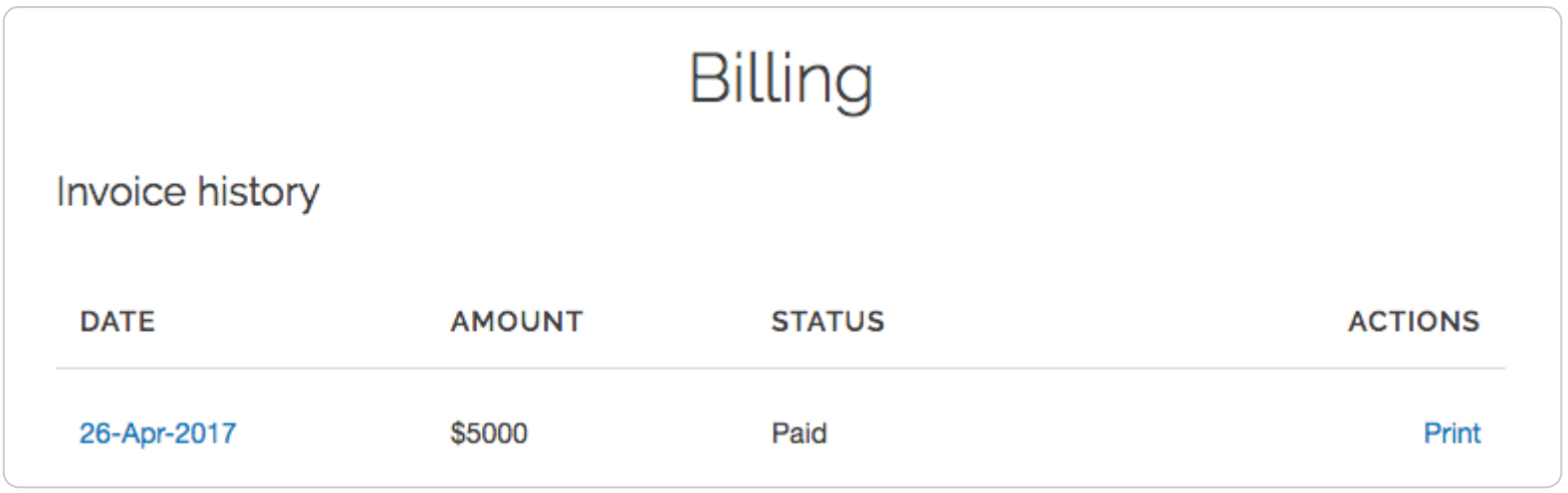
If you ever cancel your account, you'll be billed for any outstanding charges accrued through the day you canceled, or at least the monthly minimum, whichever amount is greater.

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

https://docs.fastly.com/en/guides/paying-your-bill

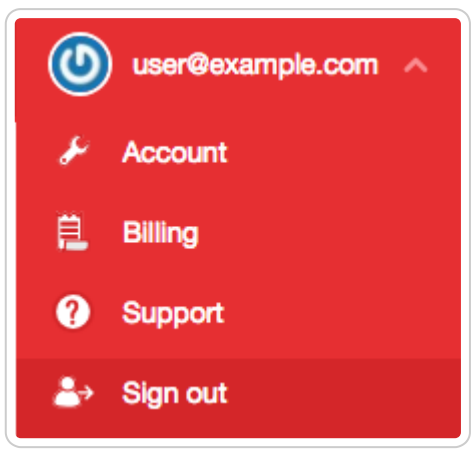
At the end of each month, your account's billing contact will be sent an email summarizing your current usage levels and the charges your account incurred for the month. The email contains a link to an online copy of the related invoice.

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.

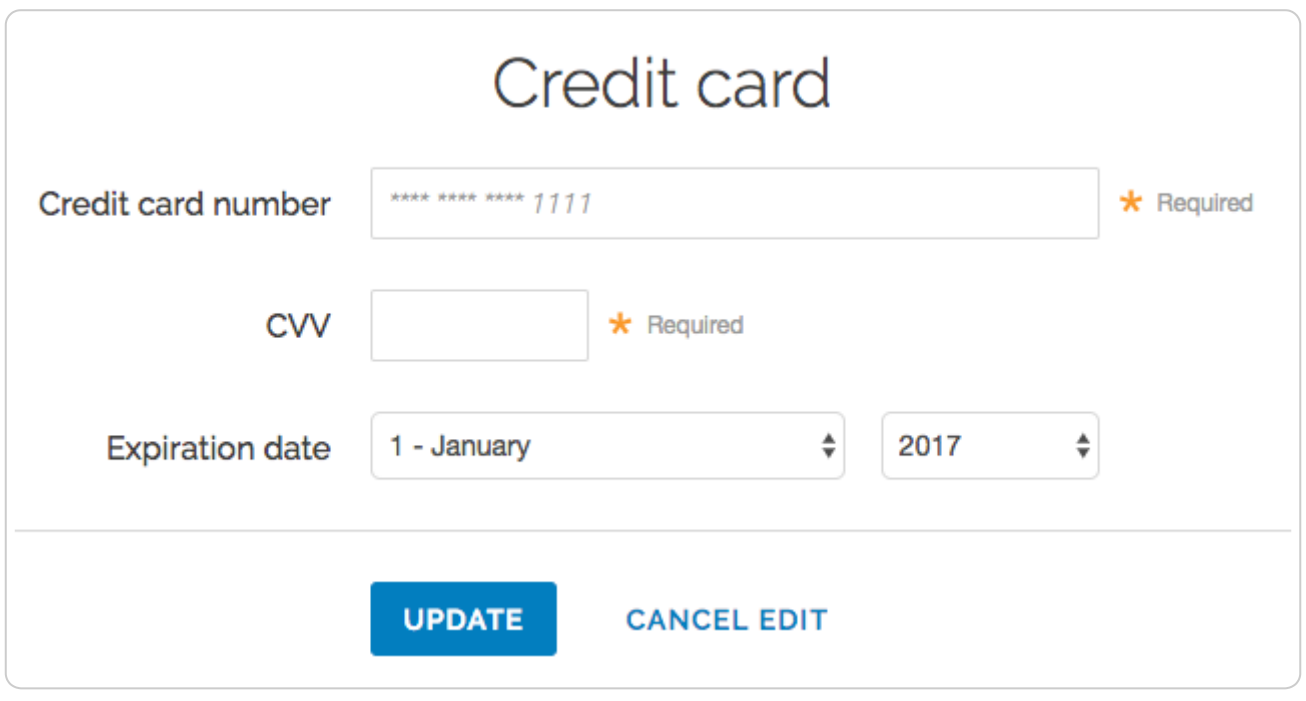
Changing your credit card information

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.



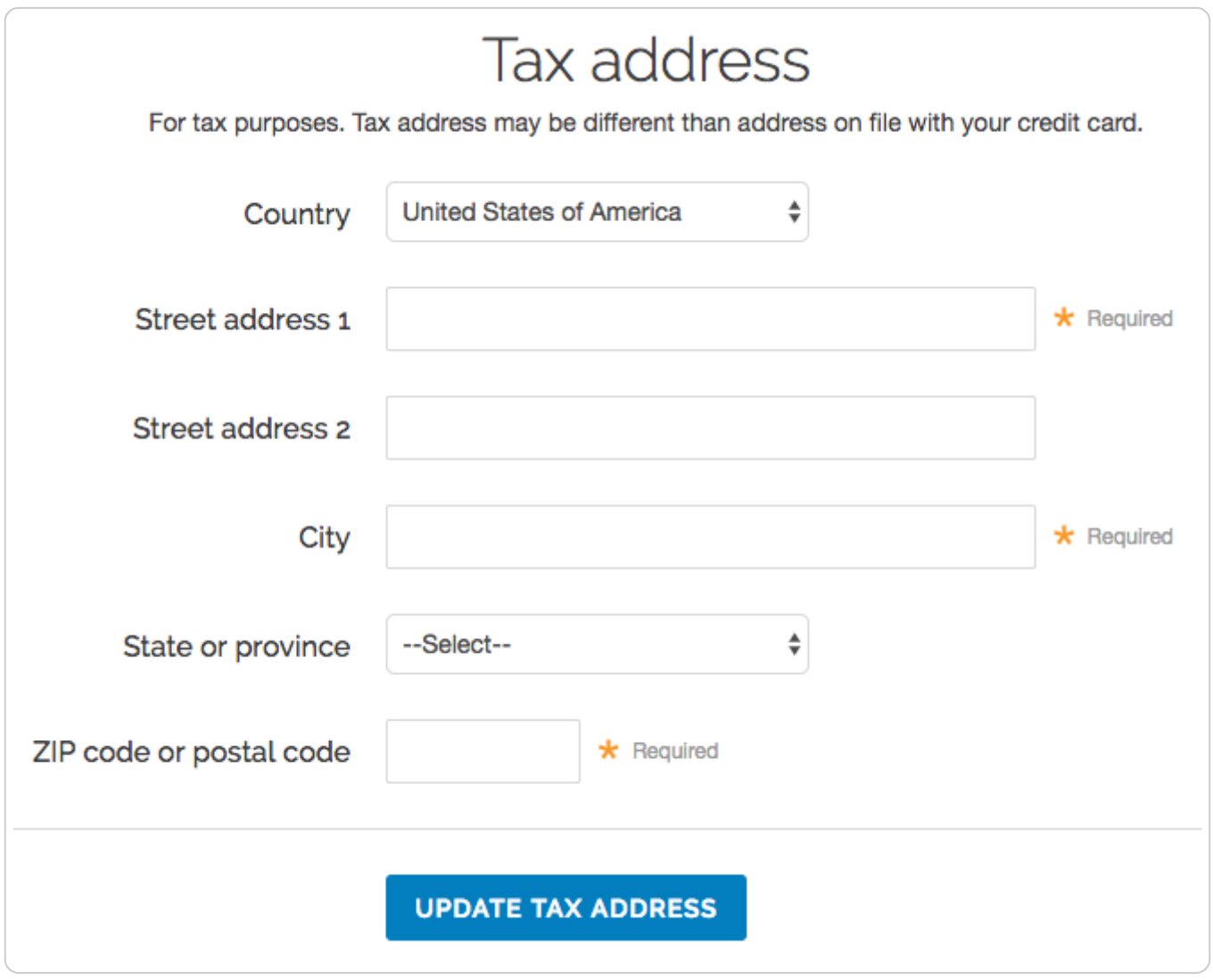
6. Click **Update** to save your credit card information.

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.worldpay.com/en-gb/privacy-policy.

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 type the tax address information you use in the fields provided.



4. Click the **Update Tax Address** button to save the tax address information. 6.

Who receives your bill

https://docs.fastly.com/en/guides/who-receives-your-bill

By default, your account owner is considered your billing contact and will receive your bill for Fastly services. You can change your billing contact at any time if you've been assigned the superuser role for an account. If you ever delete your billing contact, billing will automatically revert to the account owner.

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.

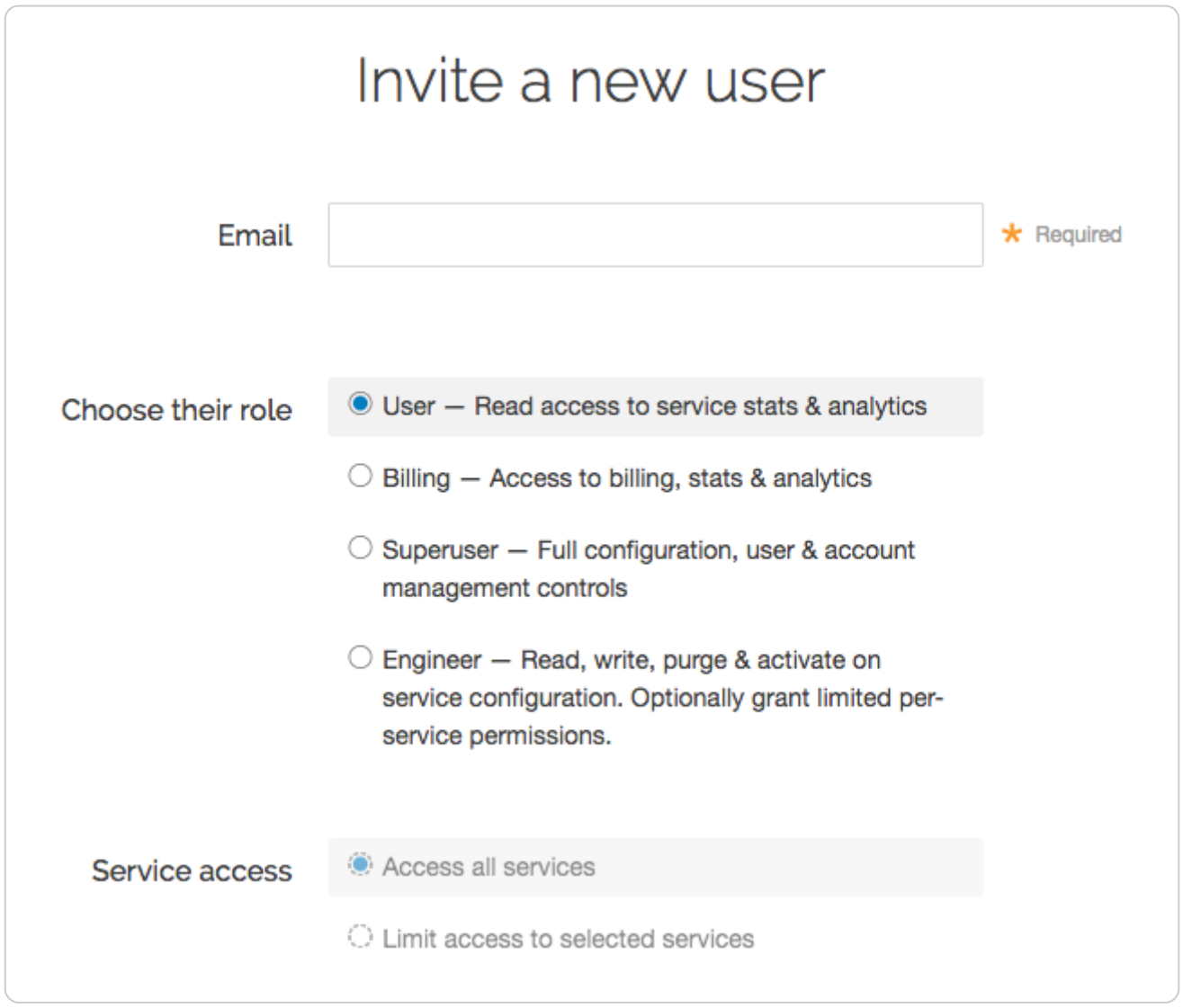
Changing who receives your bill

Follow the steps below to have your billing invoice sent to a person other than the owner of your account.

For new users

To send the billing invoice to a user who has not yet created an account, follow these steps.

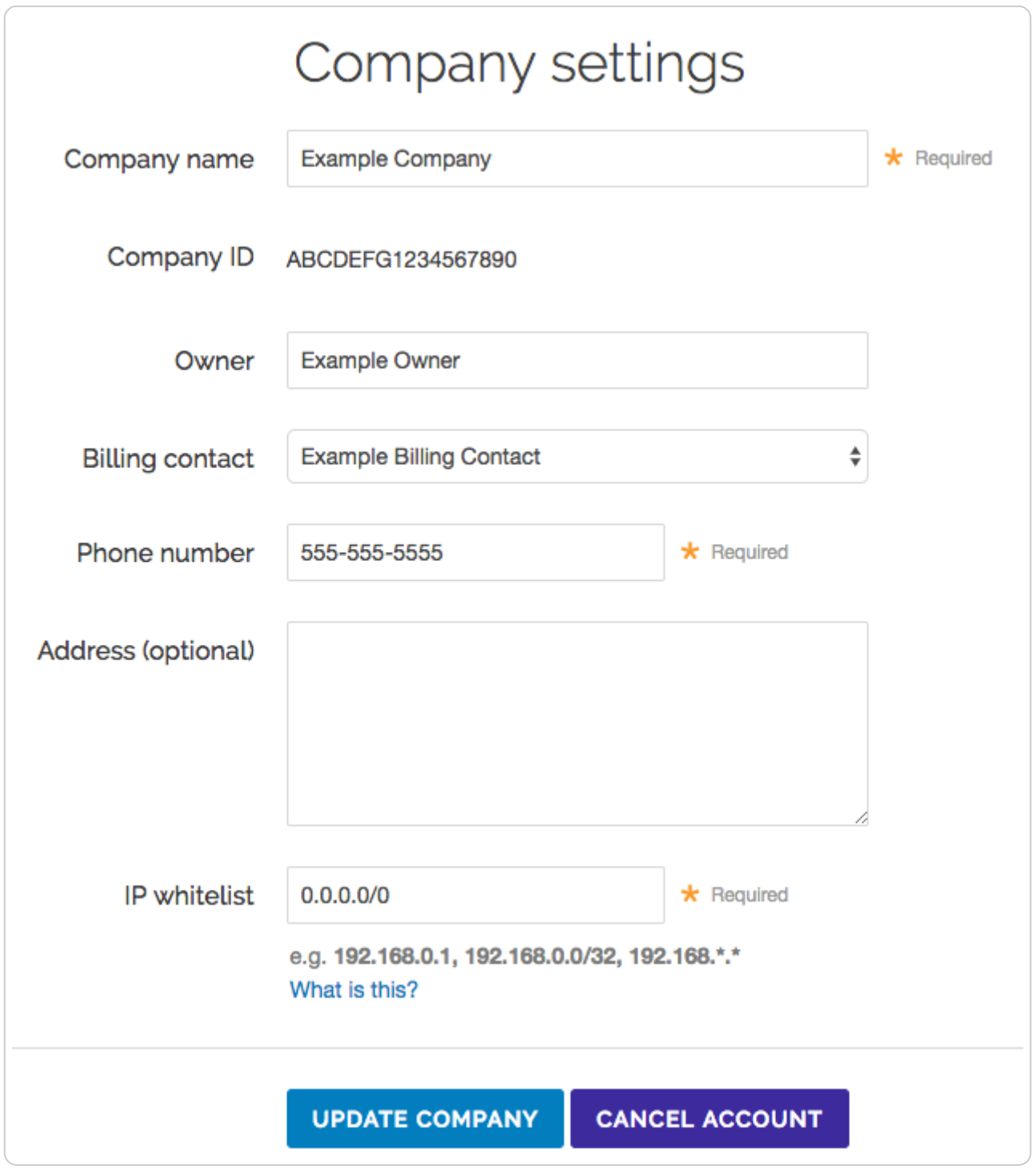
1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link. The User management page appears.
3. In the **User Invitations** area, click the **Invite** button. The Invite a new user window appears.



4. In the **Email** field, type the email address of the user you'd like to invite to become a billing contact.

> ★ **TIP:** To send invoices to multiple people, we recommend setting up a group email address and setting that email address as your Billing Contact user.

5. From the **Choose their role** options, select **Billing**.
6. Click the **Invite** button to send an invitation to the email you specified.
7. Once the user has accepted the invitation, return to the account information in the web interface.
8. Click the **Company settings** link.
9. In the **Company settings** area, select the user's name from the **Billing contact** menu.



10. Click the **Update Company** button to set the billing contact.

For existing users

To send the billing invoice to a user who already has an account, 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 **Company Settings** area, select the user's name from the **Billing Contact** menu. Make sure the user name you select has the correct role assigned to view and manage billing information.
3. Click **Update Company** to save the billing contact.

User access and control

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

https://docs.fastly.com/en/guides/adding-and-deleting-user-accounts

Fastly allows you to add users to your account, assigning them different roles and permissions as appropriate. You can delete user accounts when you no longer want them to have access.

IMPORTANT: You must be assigned the role of superuser to add users to or delete users from an account.

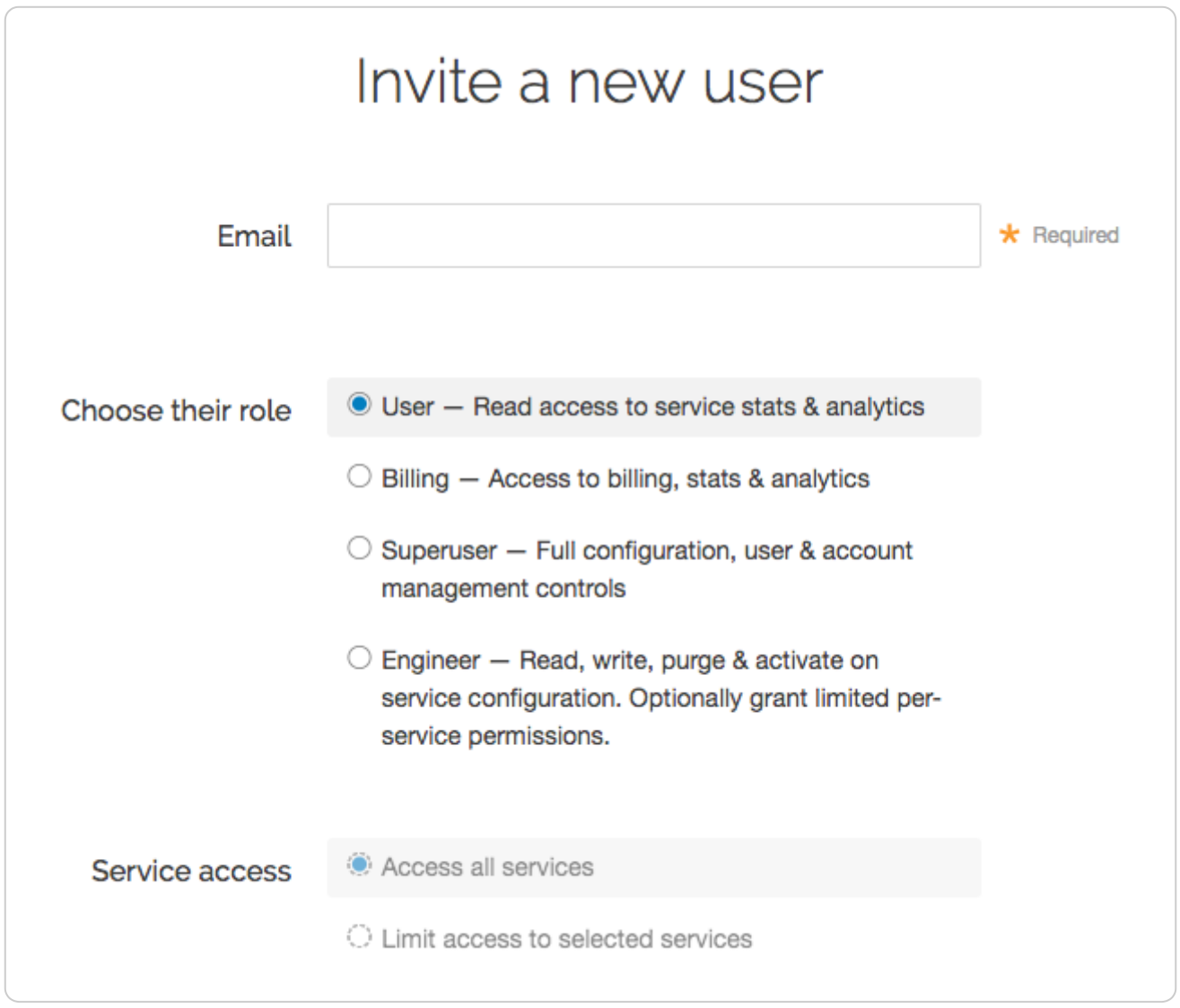
Adding account users

TIP: Adding a new user to make them the billing contact for your account? Follow our billing contact instructions instead.

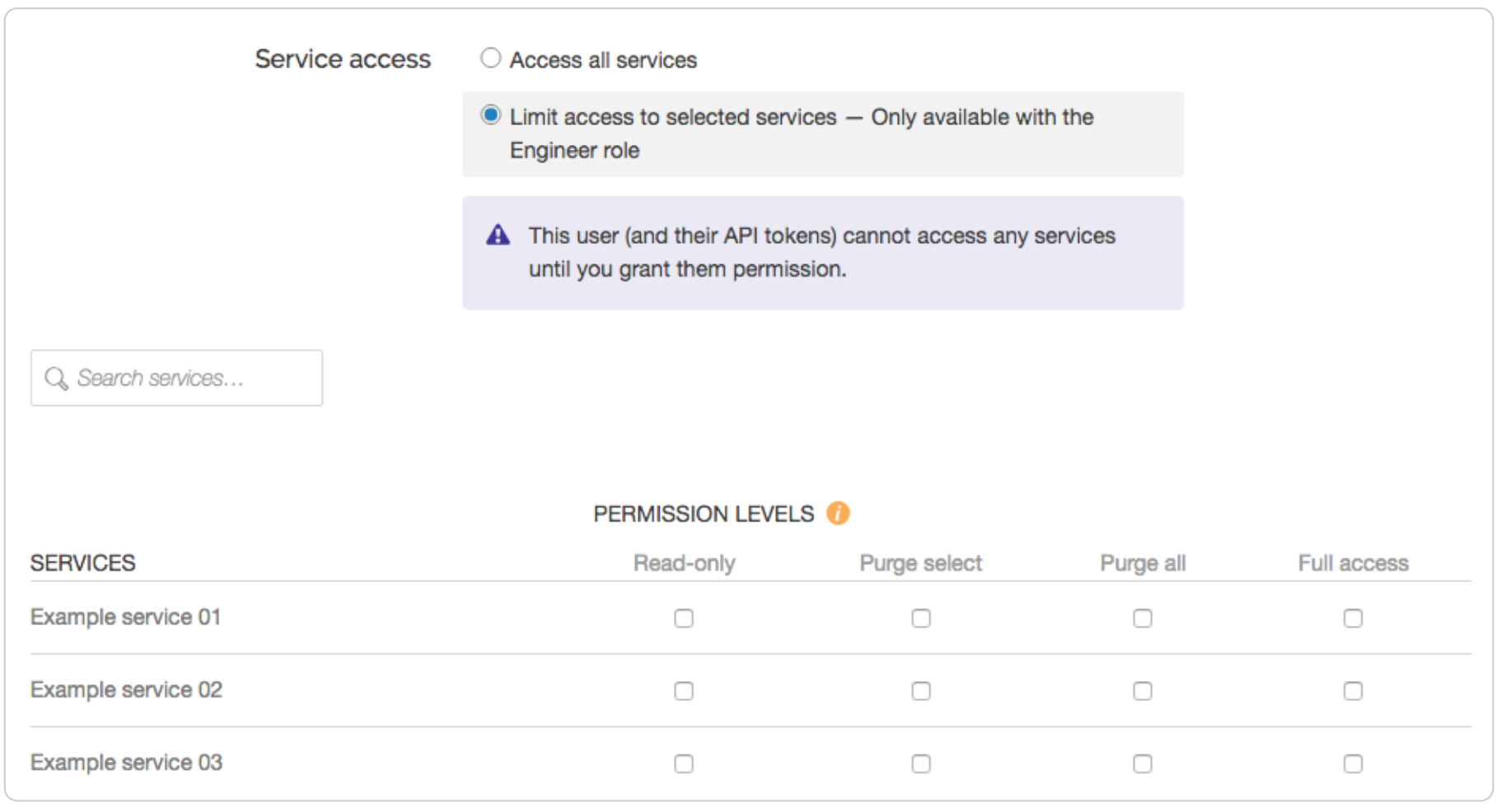
Adding a new user to your account

To add a new user to your account, send them an invitation to join following the steps below:

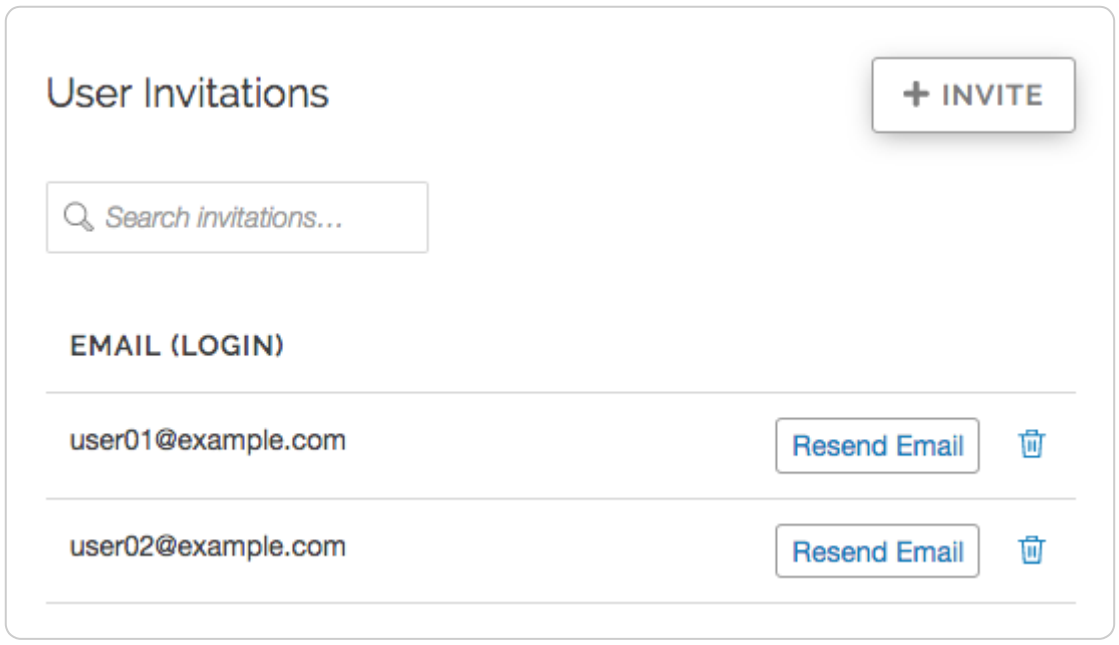
1. Log in to the Fastly web interface and click the **Account** link from the user menu. Your account information appears.
2. Click the **User management** link.
3. In the **User Invitations** area, click the **Invite** button. The Invite a new user page appears.



4. In the **Email** field, type 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 User Invitations area and remains there until the invitation is accepted.



TIP: If you need to send an invitation to a user again, click the **Resend Email** button.

Adding an existing user to your account

To add an existing user to your Fastly account, have them cancel their existing account and then re-invite them by following the steps to add a new account user to your account. We associate a user's email address with an account. Canceling that account allows the email address to be reused.

NOTE: Account cancellation might not be an option in some situations. Contact support@fastly.com to discuss how accounts can be combined.

Deleting account users

TIP: Deleting the owner of the account? Be sure to transfer ownership first.

To delete a user from your 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, find the name of the user to delete.
4. Click the gear icon to the right of the user to be deleted, then select **Delete** 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

https://docs.fastly.com/en/guides/configuring-user-roles-and-permissions

Your Fastly account can be managed by multiple users. You can control each user's role, as well as control the scope of their service access and their specific permission levels for that service access.

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, analytics, and service configuration information 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 full (✔) or potentially restricted (—) access to the following functionality:

	User	Billing	Engineer	Superuser
Stats dashboards				
View historical stats	X	X	X	X
View real-time service stats	X	X	X	X
Configure				
View service configurations	?	X	X	X
Create services			X	X
Delete services			?	X
Configure services			?	X
Compare service versions			?	X
Deactivate services			?	X
Purge			?	X
View and download generated VCL			?	X
Customize VCL			?	X
TLS management	?	?	?	X
Account & Organization				
Update personal profile settings	X	X	X	X
Update company settings				X
Invite all new user roles				X
Invite new engineer and user roles (API only)			X	
Assign and change roles and permissions				X
Issue password resets				X
Delete account users				X
Enable and disable personal 2FA	X	X	X	X
Enable and disable company-wide 2FA				X
Manage personal API tokens	X	X	X	X
Revoke account API tokens				X
Billing				
View invoices		X		X
View billing history		X		X
Pay bills		X		X
Update credit card info		X		X

Change account type	X	X

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.

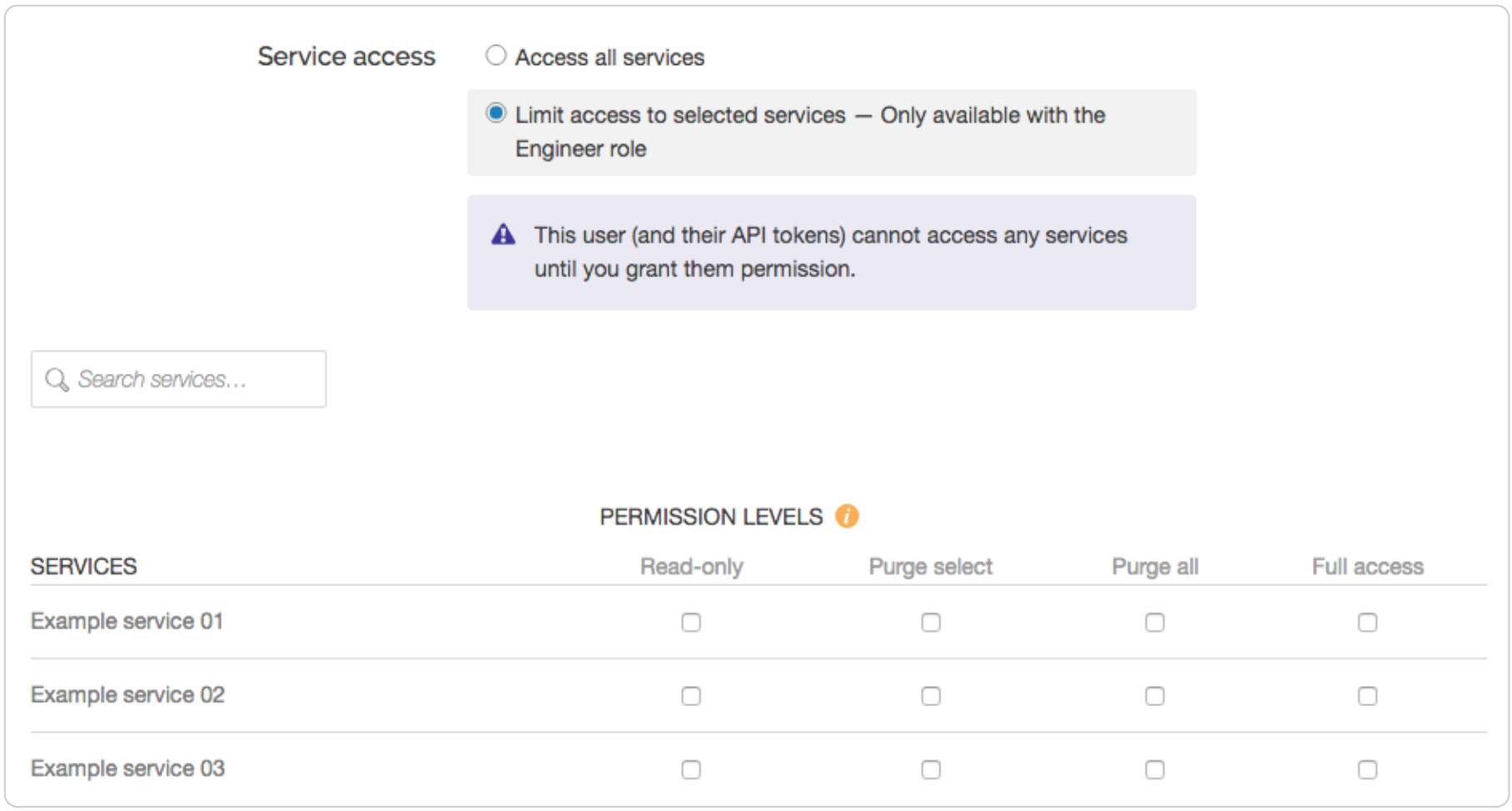
Users assigned the role of `engineer` can create new services (this is especially useful for learning about configuration options without affecting production services). By default, an engineer will automatically have full access to any service they create until their permission levels on that service 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 **Users** area, click the gear icon 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 and how to transfer it

We assign account "ownership" to the first user who signs up for an account for your organization. We automatically assign owners the superuser role, though that role can be changed by another superuser once additional users are added.

Accounts can only be canceled by owners. In addition, account owners 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.

To transfer account ownership to another user, contact support@fastly.com for assistance.

Email and password changes

https://docs.fastly.com/en/guides/email-and-password-changes

The Fastly web interface allows you to change the name, email address, and password currently associated with your account.

Changing your name or email address

Follow these instructions to change the name or email address 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. Fill out the page as follows:
   - In the **Name** field, type your name.
   - In the **Email (login)** field, type your email address.
4. Click **Update Profile** to save the changes.
5. If you've changed your email address, confirm your password in the window that appears.

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, type your existing password.
   - In the **New password** field, type the new password.
   - In the **Confirm password** field, type the new password a second time.

4. Click **Change Password** to save the changes.

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 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

Setting up single sign-on (SSO)

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. This feature allows your organization's users to sign in to the Fastly web interface using the IdP instead of an email address and password.

Prerequisites

Review the following prerequisites before enabling SSO for your organization:

- Your IdP must support Security Assertion Markup Language 2.0 (SAML 2.0).
- Your IdP must support IdP-initiated SSO.
- You must have access to your IdP's administration console.
- You must be a superuser.

You should also review this feature's limitations before enabling SSO.

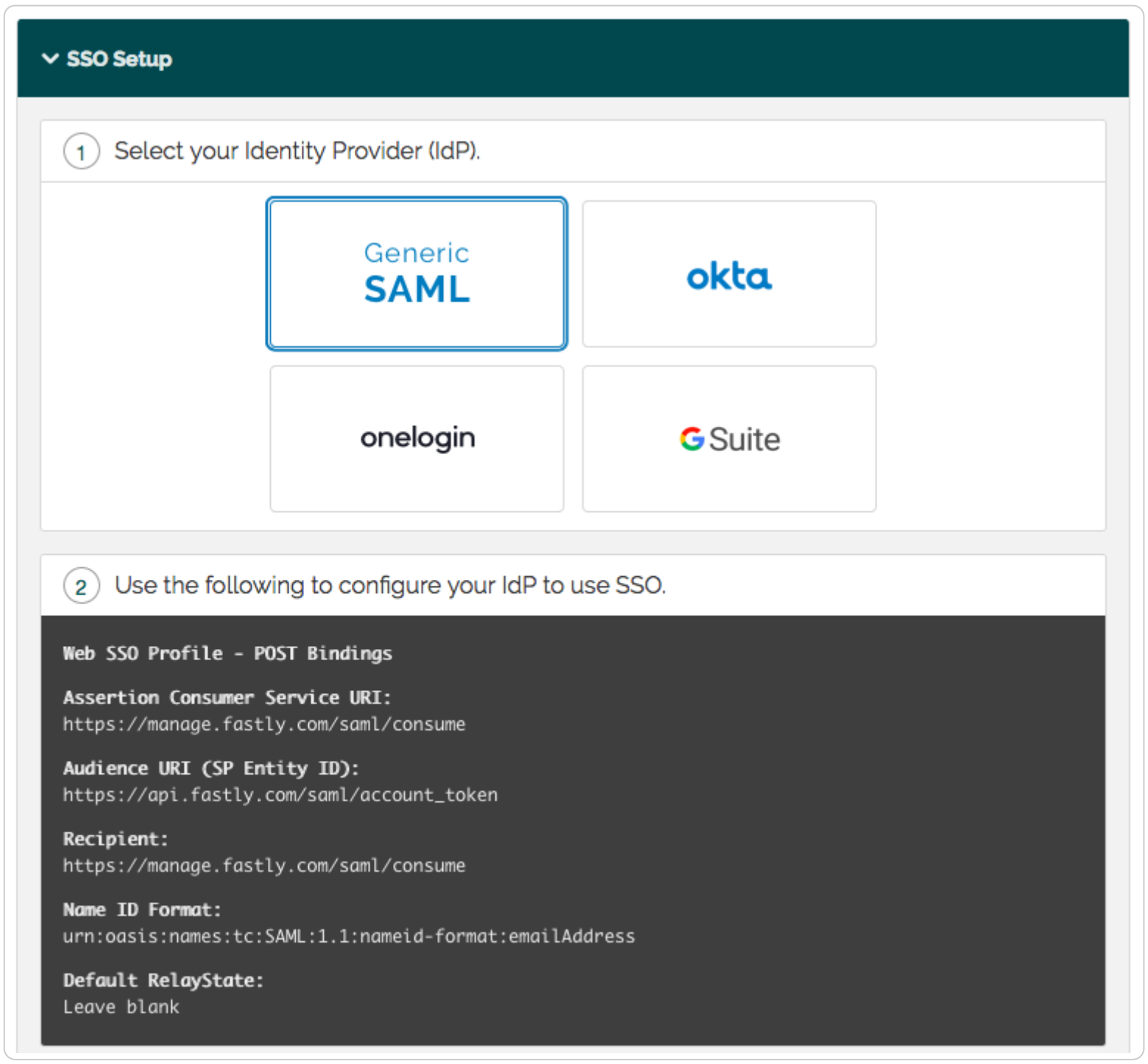
Enabling SSO

Enabling SSO for your organization requires you to set up specific SAML configurations via your IdP and then use those configurations (via a metadata file) in the Fastly web interface.

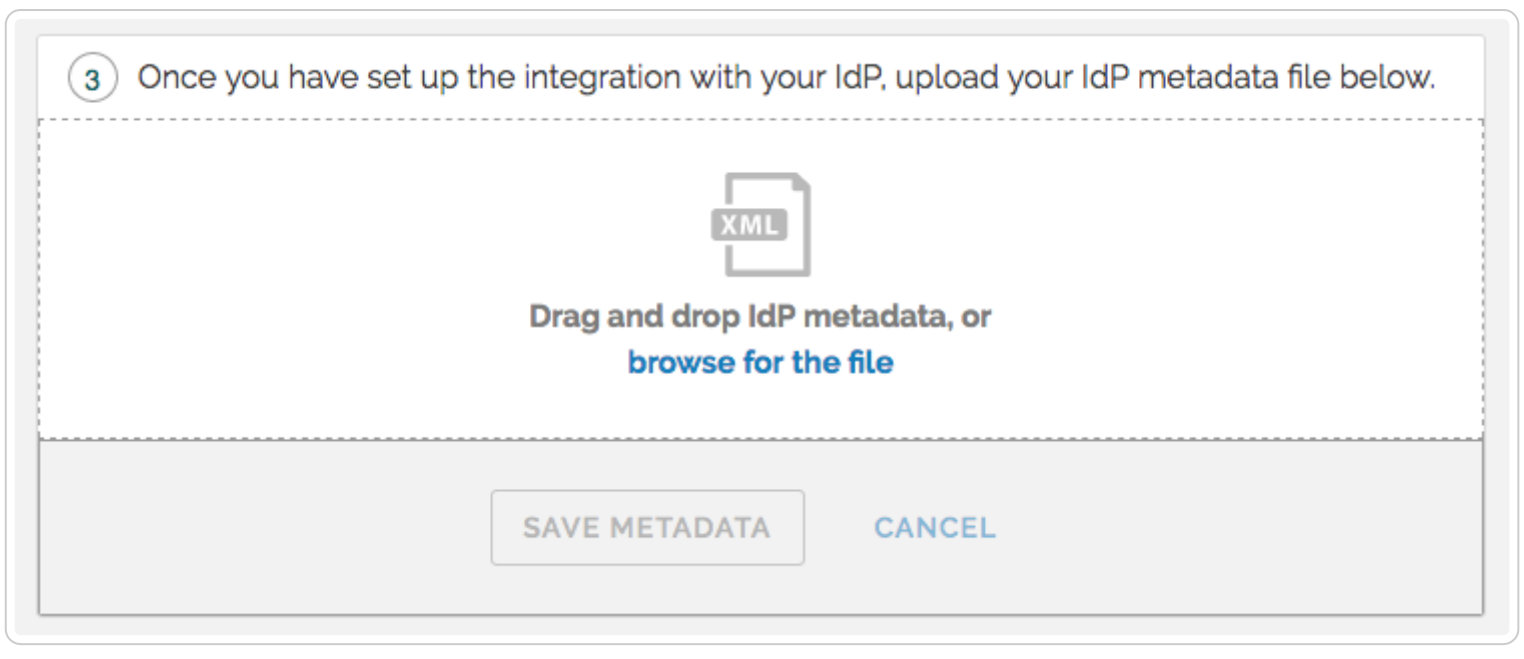
Retrieving your IdP's SAML configurations

Follow these instructions 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 appropriate button to select your organization's IdP.



4. 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.
5. 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.
6. 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.

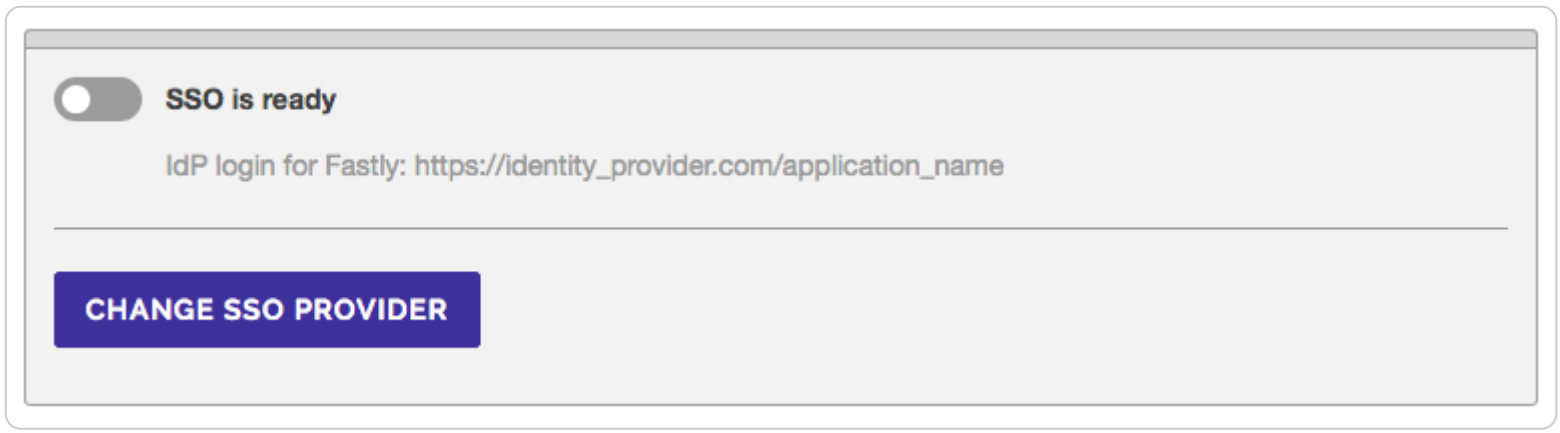


7. Click the **Save Metadata** button.

Enabling SSO for your organization

Follow these instructions to enable SSO for your organization:

1. Click the **SSO is ready** switch to enable SSO for your organization.



2. Click the **Proceed** button in the confirmation window that appears.

SSO is now enabled for your organization. Existing users on your Fastly account can now use SSO to log in to Fastly provided that the user's email address with Fastly matches an email address with your IdP and the user has been explicitly assigned your SAML application.

Performing user tasks 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.

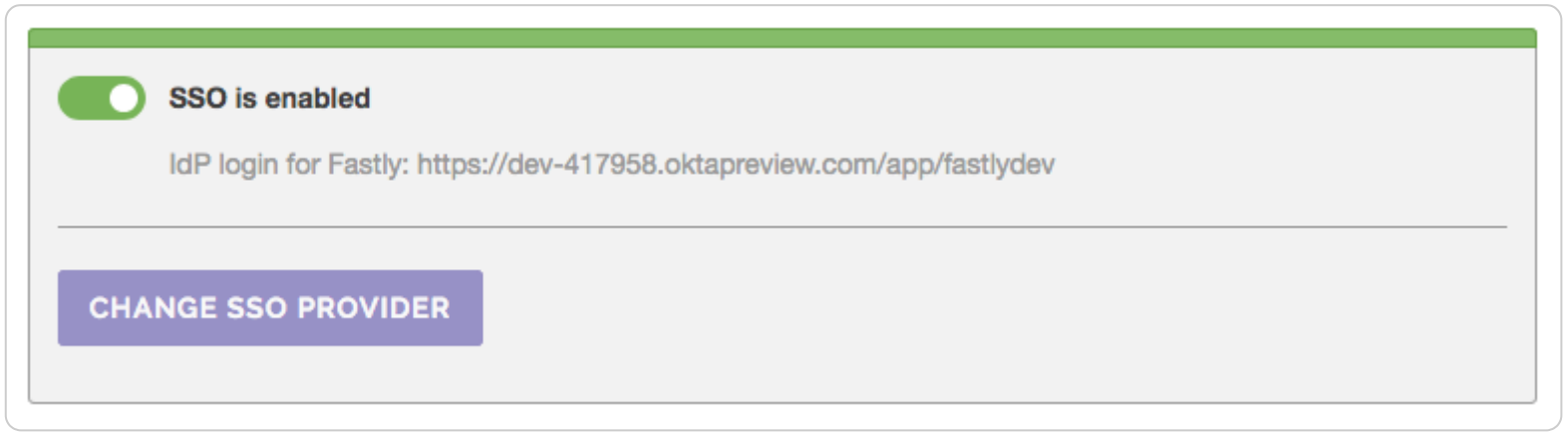
Disabling SSO

> **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.

Follow these instructions to disable SSO for your organization:

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 **SSO is enabled** switch to disable SSO for your organization.

> **NOTE:** Disabling SSO won't delete your SSO settings. You can reenable SSO at any point using the IdP configuration metadata you uploaded when you first enabled SSO.



4. Click the **Disable SSO** button in the confirmation window that appears.

SSO is now disabled. If you need to set up a new IdP, click **Change SSO provider** and follow the instructions in the enabling SSO section.

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 the session's authentication.
- Fastly does not currently support forcing SSO. Existing users can log in to the Fastly web interface by using SSO via the IdP, or by entering the email address and password associated with their Fastly account. New users invited into SSO-enabled accounts won't be prompted to set a password for the Fastly web interface, but they can generate a password for their Fastly account by clicking **Forgot password** on the sign in page.
- Fastly does not currently support SP-initiated (Service Provider initiated) SSO.

Transferring services to other accounts

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.

Moving users from one account to another

To move a user from one account to another, follow these instructions:

1. Deactivate and delete all services on the account.
2. Cancel the account.
3. Accept the invitation from the new account.

Reference

These articles provide reference information about common Fastly terms and configuration settings.

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

Glossary of terms

https://docs.fastly.com/en/guides/glossary-of-terms

These are common Fastly, HTTP, and networking terms you may encounter within our service guides.

ACL

Access control list. A list of permissions that can be attached to an object allowing customers to quickly check a client's IP against a list of known net blocks and then make decisions based on the result.

Activation

The act of deploying a service's configuration changes. Once a service version is activated, it is locked to make rollbacks safer and provide version control.

Altitude

Fastly's customer summit.

Backend

See *origin server*.

Cache-Control

The specific HTTP header that controls who can cache a response, under which conditions, and for how long. Fastly respects `Cache-Control` headers returned from origin servers as one approach to cache management. See also *surrogate-control*, *max-age*, and *s-maxage*.

Cloning

The act of creating a new version of an existing service and placing it in an editable state.

Cookie

HTTP headers used to perform certain functions like authenticating login in secure website areas, information tracking, remembering user preferences, and customizing how information is presented.

cURL

An open-source command line tool for transferring data with URL syntax from or to a server using one of many supported protocols. Fastly users can issue cURL commands to verify requests are caching in the Fastly network.

DNS

Domain Name System. A system for naming computers and network services that translates a domain's numbered IP address into an easy-to-remember alphabetic name.

Edge Dictionary

A type of container Fastly users can create to store data as key-value pairs and turn frequently repeated statements into a single function that acts as constant.

Egress traffic

Bandwidth used when traffic travels from Fastly *points of presence* (POPs) to the end user.

ESI

Edge Side Includes. An XML-based markup language that allows content assembly by HTTP surrogates. Allows Fastly users to cache pages that contain both cacheable and uncacheable content (such as user-specific information).

Gzip

A way of compressing information to make it faster to transmit. Fastly allows users to dynamically gzip content based on file extension or content type.

Header

An HTTP field that precedes the main content of information being sent in a request or response and describes the length of the content, type of content, or other characteristics of the information.

Host (header)

Information used in addition to the IP address and port number to uniquely identify a domain.

Ingress traffic

Bandwidth used when end users make requests that send traffic to Fastly *points of presence* (POPs).

Instant Purge

A feature of Fastly's purging functionality that allows users to actively invalidate content in Fastly caches within milliseconds. See also *Soft Purge*.

manage.fastly.com

The web interface through which customers access Fastly's CDN services.

max-age

An HTTP *Cache-Control* directive that specifies how long (in seconds) an object will remain in the cache before Fastly removes the object from storage. See also *surrogate-control*, *cache-control*, and *s-maxage*.

MTR

A tool that combines traceroute and ping programs in a single network diagnostic tool. Frequently used in debugging network connections.

Origin server

The location or address from which Fastly's network requests the content it will serve.

Origin Shield (Shielding)

A specific Fastly *point of presence* (POP) designated by users as the primary source of content through which all content requests from other POPs will be directed in lieu of contacting a customer's origins directly.

OTFP

On-the-fly packaging. A feature of Fastly's video on demand media and streaming offering that allows customers to dynamically package video for delivery in multiple HTTP streaming formats. Also known as "just in time" video content packaging.

POP

Point of Presence. Datacenter within which Fastly's globally distributed cache servers reside.

Priority

A setting that allows users to specify the order request and cache settings execute within their subroutines. The Priority can be any whole number and always default to 10. The smaller the assigned priority number, the sooner that condition executes (e.g., 1 executes sooner than 10).

private

An HTTP *Cache-Control* directive that allows users to select which objects are not cached. Fastly will not cache responses with a Cache-Control value of `private`.

Purging

The process of picking out one or more objects from the Fastly cache and discarding it along with its variants. See also *Instant Purge* and *Soft Purge*.

Redirect

A function that directs requests for information from their originally intended locations to a more desirable destination.

Role-based access control

A method of regulating access to resources based on the roles of the individuals within an organization.

s-maxage

An HTTP cache control directive similar to *max-age*, but applied only to shared caches. See also *surrogate-control* and *cache-control*.

Service

A user-defined set of caching rules and behavior for a website or application. You can use the Fastly web interface to create, edit, and delete your services.

Service authorization

A function that grants per service access to an Engineer role. In addition, the API calls that limit user access to specified services. See also *service permissions*.

Service permissions

The functions that manage the level of control granted to an Engineer role once they've been authorized to access a service. See also *service authorization*.

Set-Cookie

The header sent by a server in response to an HTTP request and then used to create a cookie on a user's origin. Fastly supports a method for extracting a named value out of `Set-Cookie` headers no matter how many there are. By default, Fastly will not cache responses that contain a `Set-Cookie` header.

Soft Purge

A type of purging that allows users to easily mark content as outdated (expired) instead of immediately deleting it from Fastly's caches. See also *Instant Purge*.

status.fastly.com

Fastly's network status monitoring site. Allows customers to quickly check whether anomalies they see may be due to a known problem currently being worked on by Fastly or if their issues more likely stem from problems within their own infrastructure.

support@fastly.com

The main email address of Fastly's Customer Support team through which customers can ask questions and receive assistance.

Surrogate-Control

An HTTP response header that allows origin servers to use control directives to dictate how intermediate caches, including Fastly, should handle response entities. `Surrogate-Control` will not affect browsers. See also *cache-control*, *max-age*, and *s-maxage*.

Surrogate Key

A unique identifier that allows customers to group content together for faster processing. Fastly uses surrogate keys as part of its purging strategy.

Synthetic response

Custom responses generated within the CDN that users can set if a specific URL is requested or a specific condition, such as a status code, is met. These responses require no origin server interaction.

TLS (SSL)

Transport Layer Security. A cryptographic protocol Fastly follows that ensures privacy between communicating applications and their users on the internet.

URL

Uniform Resource Locator. An address used to find a site or application's objects on the internet.

Varnish

Caching software that helps content-heavy dynamic websites as well as heavily consumed APIs load faster. Fastly's core caching infrastructure is based on a heavily modified version of Varnish.

VCL

Varnish Configuration Language. A scripting language used to configure and add logic to Varnish caches. Fastly users can create custom VCL files with specialized configurations.

WAF

Web Application Firewall. A network security system that monitors, filters, or blocks data packets as they travel to and from a web application.

www.fastly-debug.com

A network debugging tool designed to provide key info to help a Fastly user troubleshoot issues with Fastly's Customer Support team.

Resource limits

https://docs.fastly.com/en/guides/resource-limits

This guide details Fastly resource limits and summarizes the implications of exceeding those limits.

Cache limits

Item	Limit	Implications
Cache file size (with streaming miss)	5GB	Exceeding this limit when trying to cache an object results in a `Service unavailable` error unless Segmented Caching is enabled.
Cache file size (without streaming miss)	2GB	Exceeding this limit when trying to cache an object results in a `Service unavailable` error unless Segmented Caching is enabled.

Item	Limit	Implications
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.

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 header limits

Item	Limit	Implications
URL size	8KB	Exceeding the limit results in a `Too long request string` error.
Cookie size	32KB	Exceeding the limit results in Fastly stripping the cookie and setting `req.http.Fastly-Cookie-Overflow = "1"`.
Header size	69KB	Exceeding the limit results in a `503 backend read error`. See Common 503 errors for more info.
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.
Varnish restart limit	3 restarts	Exceeding the limit results in a `Service Unavailable` error. This limit exists to prevent infinite loops.
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.
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.