Fastly Guides

Generated: Sat, 13 Dec 2014 16:58:12 +0000

Guides (/guides/)

Getting Started with Fastly

• Getting Started (/guides/getting-started/)

· About Domain Names and Fastly's CDN Service (/guides/getting-started/about-domain-names-and-fastlys-cdn-service)

A domain name is a component of a Uniform Resource Locator (http://en.wikipedia.org/wiki/Uniform_Resource_Locator) (URL). A domain name represents an Internet Protocol (http://en.wikipedia.org/wiki/Internet_Protocol) (IP) resource and, in this context, refers to the IP address of a server computer hosting a website (http://en.wikipedia.org/wiki/Web_site), the website itself, or any other service communicated via the Internet. Customers associate their domain names with their origin (content source) when provisioning a Fastly service.

Domain names are made up of components as follows:

Domain Name Component	Example
URL	http://www.example.com/index.html
Top-Level Domain Name	com
Second-Level Domain Name	example.com
Host Name	www.example.com

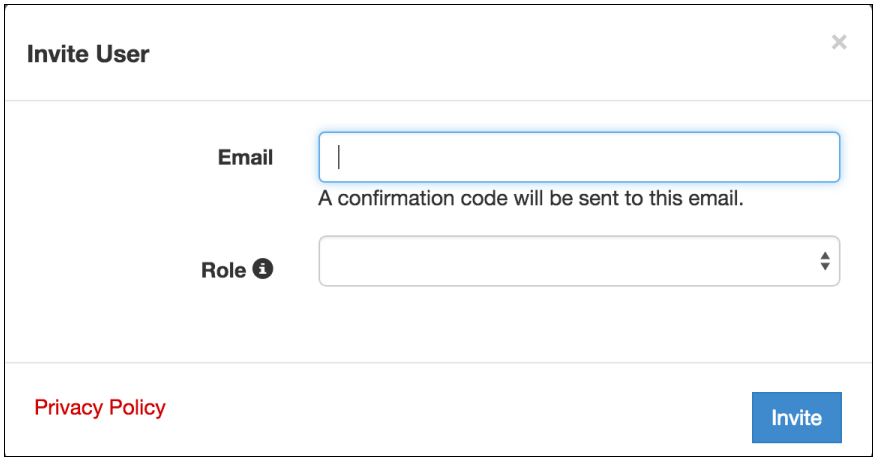
Domain names are registered with a domain name registrar. Fastly is not a domain name registrar.

Fastly supports the use of multiple subdomains for the same origin server, and allows the specification of any number of subdomains for each origin. Some customers use only their second level or apex domain (/guides/domains-and-dns/can-i-use-fastly-with-a-second-level-domain) (e.g., example.com rather than www.example.com) as their canonical domain. Unfortunately, due to limitations in the DNS specification, we don't recommend using Fastly with domains this way.

· Adding New Users to Your Account (/guides/getting-started/adding-new-users-to-your-account)

Adding new users to your account can be done by anyone assigned the superuser role (/guides/customer-support/what-type-of-access-do-different-users-have). If you have been assigned the role of superuser, follow the steps below to invite a new user to be added your account.

1. Log in to the Fastly application (https://app.fastly.com/).
2. Click the **account** tab to access your Account Settings.
3. In the **User Invitations** area, click the **Invite** button. The **Invite 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.
5. From the **Role** drop down menu select the role to assign (/guides/customer-support/what-type-of-access-do-different-users-have) once the user accepts the invitation.
6. Click the **Invite** button to send an invitation to the email you specified.

Remember: If you're adding a new user to make them the billing contact for your account, be sure you complete the additional steps (/guides/billing-and-prices/can-i-get-a-copy-of-my-bill-sent-to-a-person-other-than-the-owner-of-the-account) once they accept the invitation to be added to your account.

· Browser Recommendations When Using the Fastly Application (/guides/getting-started/browser-recommendations-when-using-the-fastly-application)

The Fastly application (https://app.fastly.com/) works well with all modern browsers. We recommend always using the latest version of your favorite browser to prevent malicious websites from compromising your systems. Up-to-date browser versions also (usually) correct bugs and limitations that show up when you visit websites, making those sites harder to use.

You can find the latest version of all major browsers online. The list at Browse Happy (http://browsehappy.com/) may help you.

· CNAME Instructions for most providers (/guides/getting-started/cname-instructions-for-most-providers)

In order to use Fastly with your website you will need to CNAME your domains to point at `global.prod.fastly.net`. This page provides a list of instructions for many of the major Domain Name providers. Select your provider from the list below to get detailed instructions for how to CNAME your domain.

If you cannot find your provider or are still having trouble configuring your CNAME records, contact us at support@fastly.com (mailto:support@fastly.com).

GoDaddy.com

1. Log in to your account at www.godaddy.com (http://godaddy.com/) by clicking the **My Account** tab.
2. Under the **Domains** drop down, find the domain you're using for your service.
3. Select your domain by clicking on the link that appears in the **Domain Name** column of the table of domains. The Settings, DNS Zone File, and Contacts tabs appear with the Settings tab selected by default.
4. Click the **DNS Zone File** tab.
5. Above the **CNAME (Alias)** section, click **Add Record**. A new row appears in the CName (Alias) table.
6. In the new row that appears, enter the following information:
    - **Host:** Enter the subdomain name for the alias assignment. For example, if you chose `www.example.com` as your host address, type `www`.

- **Points to:** Enter `global.prod.fastly.net`.
- **TTL:** Select how long the server should cache the information.

7. Click **Save**.

DynECT Managed DNS

1. Log in to your account at manage.dynect.net (https://manage.dynect.net/).
2. Click on **Manage Domain**.
3. Click **Add New Node**.
4. Enter the address for your Fastly service and click **OK**. For example, if you chose `www.example.com` as the address, you should enter `www.example.com`.
5. Click **Add New Record** and then choose **CNAME** from the drop-down menu.
6. Set your TTL (Time To Live) if desired and then add the full path to Fastly's server. Example: if you chose `www.example.com` you should enter `www.example.com.global.prod.fastly.net`.
7. Click **Publish Now**.

IX Web Hosting

1. Log in to your account at manage.ixwebhosting.com (https://manage.ixwebhosting.com/).
2. Click **Manage** below the **Hosting Account** section.
3. On the left side, click the domain you'd like to use with Fastly.
4. Next to **DNS Configuration**, click **EDIT**.
5. Click **Add DNS CNAME Record**.
6. Under **Name**, enter only the part of your website address that you designated for your Fastly Service. For example, if you picked `www.example.com` as your address, just enter `www` as the entry under Name.
7. Enter `global.prod.fastly.net` under **Data**.
8. Click **Submit**.

1and1

1. Log in to your account at admin.1and1.com (https://admin.1and1.com/).
2. If it's not already selected, click the **Administration** tab.
3. Click **Domains**. The Domain Overview page appears.
4. From the **New** drop-down menu, select **Create Subdomain**. (If you've already created a subdomain for your website's address, skip to step six.)
5. Enter the prefix of the address for your Fastly service, and click **OK**. Example: if you chose `www.example.com` you should enter `www`.
6. Check the box next to the subdomain.
7. From the **DNS** menu, select **Edit DNS Settings**.
8. Click the radio button next to **CNAME**.
9. Next to **Alias**, enter `global.prod.fastly.net`
10. Click **OK**.

EveryDNS.net

1. Log in to your account at www.everydns.net (http://www.everydns.net/).
2. On the left side, click the domain you'd like to use with Fastly.
3. Since EveryDNS.net is your hosting service, and not your domain registrar, be sure that your domain points to EveryDNS.net's nameservers. This will allow your CNAME record configuration to take effect.
4. Below **Add a Record:**, you can create your CNAME record.
5. Next to Fully **Qualified Domain Name**, enter only the part of your website address that you designated for your Fastly service. For example, if you picked `www.example.com` as your website's address, just enter `www` as the entry next to Fully

Qualified Domain Name.
6. Select **CNAME** as the **Record Type**.
7. Enter `global.prod.fastly.net` as the **Record Value**.
8. Click **Add Record**.

Yahoo! Small Business

1. Log in to your account at smallbusiness.yahoo.com (http://smallbusiness.yahoo.com/services).
2. Click **Domain Control Panel** below the domain you'd like to use with Fastly.
3. Click **Manage Advanced DNS Settings**.
4. Click **Add Record**.
5. In the **Source** field, enter the part of the address you chose for your website using Fastly. For example, if you chose `www.example.com`, enter `www` in the **Source** field.
6. Enter `global.prod.fastly.net` in the **Destination** field.
7. Click **Submit**.

No-IP

1. Log in to your account at www.no-ip.com (http://www.no-ip.com/).
2. On the left side, click **Host/Redirects**.
3. Click **Manage** underneath **Host/Redirects**.
4. Click **Add** for a new entry, or click **Modify** and skip to step six for an existing entry.
5. Enter the part of the address that you picked for Fastly as the host name, and select your domain name. For example, if you entered `www.example.com` as your address, enter `www` as the host name.
6. Select **DNS alias CNAME** at the host type.
7. Enter `global.prod.fastly.net` as the **Target Host** and click **Modify**.

DNS Park

1. Log in to your account at www.dnspark.net (https://www.dnspark.net/).
2. On the left side, click **DNS Hosting**.
3. Click the domain you'd like to use with Fastly.
4. Since DNS Park is your hosting service, and not your domain registrar, be sure that your domain points to DNS Park's nameservers. This will allow your CNAME record configuration to take effect.
5. Click **Alias Records**.
6. Under **Host Name**, enter only the part of your website address that you designated for Fastly. If you picked `www.example.com` as your website's address, enter `www`.
7. Enter `global.prod.fastly.net` under **Destination Name**.
8. Click **Add Alias**.

eNom

1. Log in to your account at www.enom.com (http://www.enom.com/).
2. From the drop-down menu at the **Domains** tab, select **My Domain**. You'll be directed to the **Manage Domains** page.
3. Click the domain that you'd like to use with Fastly.
4. Click **Host Settings**.
5. To add a CNAME record, click **NEW ROW**. If you've already created a CNAME record for the address, simply edit the existing CNAME record.
6. Enter the part of the address that you want to use with Fastly. For example, if you entered `www.example.com` as your address, enter `www`.
7. Enter `global.prod.fastly.net` for the **host name**.
8. Click **Save**.

Network Solutions

1. Log in to your account at www.networksolutions.com (http://www.networksolutions.com/).
2. In the left navigation bar, open the **nsWebAddress (Domains)** menu by clicking the **+ icon**.
3. Click **Manage Domain Names**.
4. On the Domain Details page for the domain you're using with Fastly, select the **Designated DNS** radio button to the right of **Change domain to point to**. (If you've already configured advanced DNS settings, click **Edit** next to **Advanced DNS Settings**.)
5. Click the **Apply Changes** button.
6. Under the **Advanced DNS Manager** heading, click **Manage Advanced DNS Records**.
7. Under the **Host Aliases (CNAME Records)** heading, click **Add/Edit**.
8. In the **Alias** box, enter your custom URL prefix (such as `www`).
9. Select the radio button to the left of the **Other Host** box.
10. Enter `global.prod.fastly.net` in the **Other Host** box.
11. Click **Continue**.
12. Review your changes and click **Save Changes** to create the CNAME record.

MyDomain.com

1. Log in to your account at www.mydomain.com/login (http://www.mydomain.com/login/).
2. Click on the **My Services** tab.
3. Under **Manage My Services**, click on **Manage Services**.
4. Go to **Domain Overview**.
5. Click on **Domain Administration**.
6. Select the domain you're using with Fastly.
7. Click the **DNS** tab, and select **DNS Records**.
8. Click **Add New DNS record**.
9. From the drop-down menu, select **CNAME**, and click **Next**.
10. Enter your custom URL prefix (such as `www`), and then enter `global.prod.fastly.net`.
11. Click **Finish**.

Domain Direct

1. Go to www.domaindirect.com (http://www.domaindirect.com/).
2. Click **my account**, and log in to your domain.
3. On the **Domain** tab, click **Advanced Settings**.
4. Click **Edit Zone File**. Note: Do not delete the A record for `*.domain.com` and `domain.com`, or you will have problems later.
5. Under **CNAME**, create an entry for the part of the address that you want to use with Fastly. For example, if you are using `www.example.com` as your address, enter `www`.
6. Point this CNAME record to `global.prod.fastly.net`.

Gandi.net

1. Log in to your account at www.gandi.net (http://www.gandi.net/).
2. Click the **Administration** tab, and then click the **control panel** link.
3. Click the domain you want to manage, and go to **Technical Settings**.
4. Click **Manage DNS zone file**.
5. Click **Add an entry**, and for each field, do the following:
    - **Name:** Enter your custom URL prefix (such as www).
    - **Type:** Select CNAME.
    - **Value:** Enter `global.prod.fastly.net.`, making sure to include a trailing dot (.) at the end of the value.
    - Click **Submit**.

6. You should be back to the list of entries. Click **Submit Changes**.

EuroDNS

1. Log in to your account at www.eurodns.com (https://www.eurodns.com/).
2. Click **My Domains** in the **My Domains** menu on the left side of the page. A table listing your domains should appear.
3. Click the **DNS** column on the row of the domain you wish to change. The DNS records for your domain should now be displayed.
4. From the **Add...** drop-down menu of the **HOST NAME** table, choose **Host Alias**, and click the **Add** button next to it.
5. In the **Alias:** field, add the name you chose for your service Fastly. For example, if you chose `www.example.com` as your address, enter `www`.
6. Enter `global.prod.fastly.net` in the **Host Name** field.
7. Click the **Save Settings** button.

Register.com

1. Log in to your account at www.register.com (http://www.register.com/).
2. Under the **My Accounts** tab, click the domain that you're using with Fastly.
3. Scroll down to the **Advanced Technical Settings** heading, and click **Edit Domain Aliases Records**.
4. In the box to the left of your domain name, enter your URL prefix (such as `www`).
5. Enter `global.prod.fastly.net` in the **points to** box.
6. Click **Continue**.
7. Review your changes, and click **Continue** to create the CNAME record.

MyHosting

1. Go to myhosting.com (http://myhosting.com/).
2. Click on the **Account** link at the top of the page.
3. Log in with your **Domain Name** and **Password**.
4. Click on the **Domain Name** tab.
5. Click **DNS Management** in the left-hand pane.
6. Click the **Manage DNS button**.
7. Add an **ALIAS (CNAME) RECORD** by doing the following in the **ZONE SERVER** area:
   - Enter your URL prefix (such as `www`).
   - Enter `global.prod.fastly.net`.

Namecheap

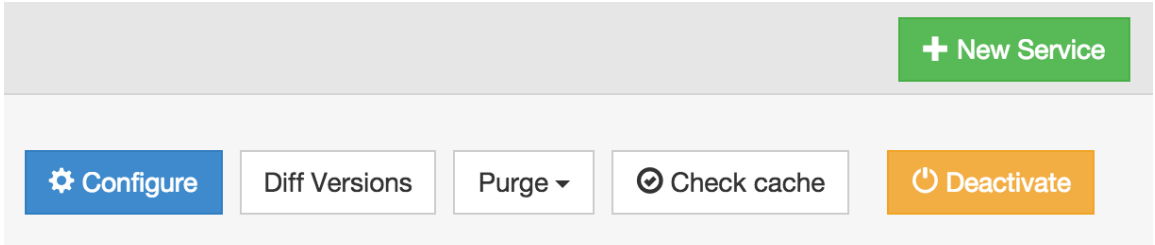
1. Log in to your account at www.namecheap.com/myaccount/login-only.aspx (http://www.namecheap.com/myaccount/login-only.aspx).
2. Click on **Your Domain/Products** in the right hand sidebar.
3. Click on the domain you are using under the main **Your Domains** heading.
4. Click on **URL Forwarding** under the **Host Management** tab on the left.
5. Enter `global.prod.fastly.net` into the IP Address/URL column for both @ and `www` as a CNAME.
6. Click **Save Changes.**

· Creating a New Service (/guides/getting-started/creating-a-new-service)

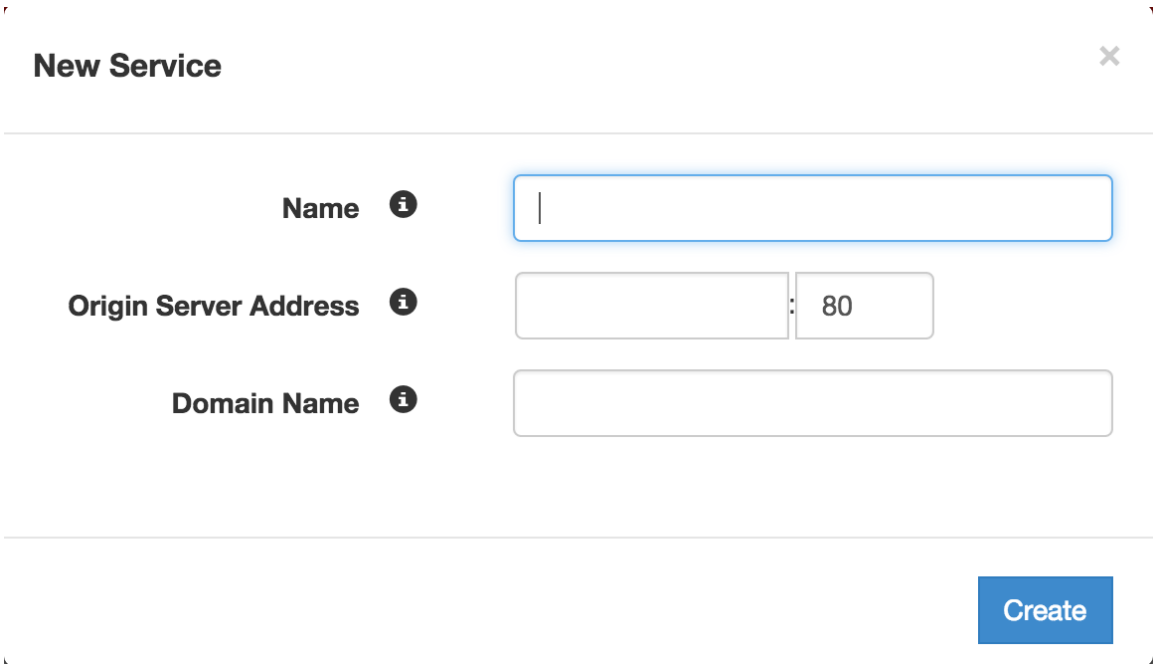
To create a new service, follow the steps below.

1. Log in to the Fastly application.
2. Click the **configure** button (the wrench icon at the top of the window).

3. At the top right of the window, click the green **New Service** button.



The New Service window appears.



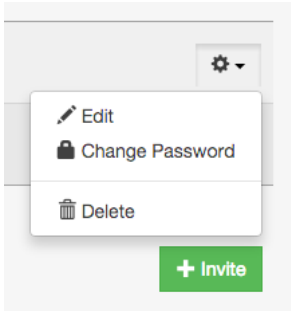
4. Fill out the **New Service** window as follows:
   - In the **Name** field, type the name of the first service you'd like to create. You can rename this service (/guides/faqs/how-do-i-rename-my-service) at any time.
   - In the **Origin Server Address** field, type the IP address (or hostname) and port number (usually `80`) for your website's server.
   - In the **Domain Name** field, type the domain name of your website. We need this information to properly route requests to your website.
5. Click the **Create** button. A new service appears in the list of services available.

· Deleting Account Users (/guides/getting-started/deleting-account-users)

To delete a user from your account, follow the steps below.

REMEMBER: You must be assigned the role of super user (/guides/customer-support/what-type-of-access-do-different-users-have) in order to delete other user accounts.

1. Log in to the Fastly application (https://app.fastly.com/).
2. Click on the **account** tab. The Account Settings controls appear.
3. Scroll down to the **Users** area and search for the name of the user you wish to delete.
4. Click the gear icon to the right of the user's name and select **Delete** from the menu that appears.



A confirmation window appears.

5. Click **Confirm** on the confirmation window.

· Enabling an IP Whitelist for Account Logins (/guides/getting-started/enabling-an-ip-whitelist-for-account-logins)

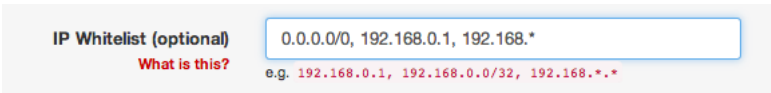
Fastly allows you to define the range of IP addresses authorized to access your Fastly account. This optional IP whitelisting functionality is not enabled by default.

IMPORTANT: ***If you decide to use optional IP whitelisting, 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 whitelist 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 (/guides/customer-support/help!-im-locked-out-of-my-account!) based on an email request alone***.

To restrict access to your Fastly account based on a specific list or range of IP addresses, follow these steps.

1. Log in to the Fastly application.
2. Click the **account** tab to access the **Account Settings** information.

3. In the **IP Whitelist** field of the **Company Settings** area, type the IP addresses to which your account access should be restricted.



In the IP Whitelist field you can include single or multiple (separated by commas) IP addresses or IP ranges as follows:

- a single IPv4 address (e.g., `192.168.0.1`)
- an IPv4 CIDR range (e.g., `192.168.0.0/32`)
- an IPv4 Wildcard range (e.g., `192.168.0.*`, `192.168.*.1`, `192.168.*.*`)

4. Click the **Update Company** button.

· Enabling and disabling two-factor authentication (/guides/getting-started/enabling-and-disabling-twofactor-authentication)

Fastly supports two-factor authentication (/guides/faqs/does-the-fastly-application-support-multifactor-authentication), a two-step verification system, for logging in to the application. To use this security feature with your Fastly account, you must obtain a mobile device such as a smartphone or tablet capable of scanning a barcode or QR code. Then you must download and install an authenticator application. We recommend the following:

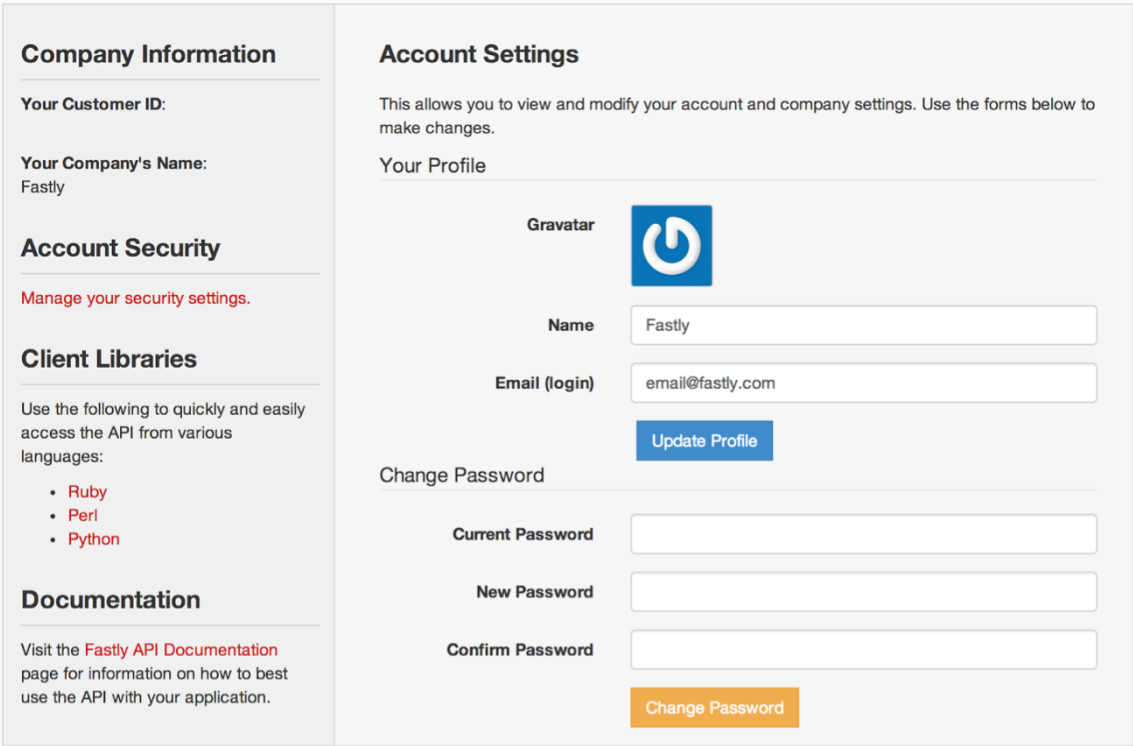
- For Android, iOS, and Blackberry: Google Authenticator (https://support.google.com/accounts/answer/1066447?hl=en)
- For Android and iOS: Duo Mobile (http://guide.duosecurity.com/third-party-accounts)
- For Android: AWS Virtual MFA (http://www.amazon.com/gp/product/B0061MU68M#)
- For Windows Phone: Authenticator (http://www.windowsphone.com/en-us/store/app/authenticator/021dd79f-0598-e011-986b-78e7d1fa76f8)

IMPORTANT: If you enable two-factor authentication via the user interface, you will no longer be able to use a simple username+password when using the API, and must use the API key for authentication.

Enabling two-factor authentication

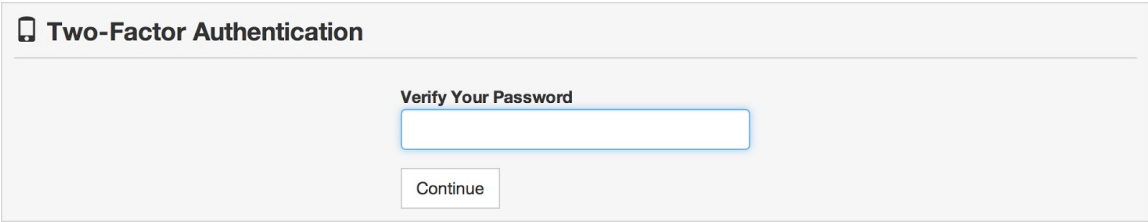
To enable two-factor authentication with the Fastly application, follow the steps below.

1. Log in to the Fastly application.
2. Click the **account** tab to access the account settings.

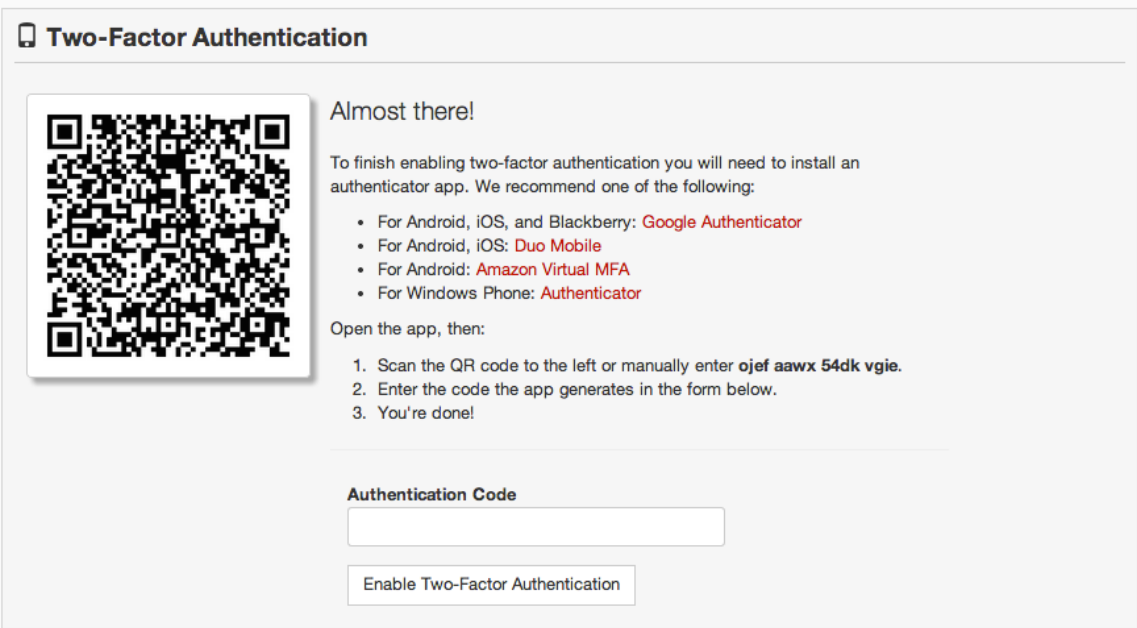


3. Click **Manage your security settings** from the **Account Security** area on the left. The Account Security settings appear.

4. Click **Set Up Two-Factor Authentication**. The password verification screen appears.



5. In the **Verify Your Password** field type your Fastly password and then click **Continue**. The authentication QR code appears.

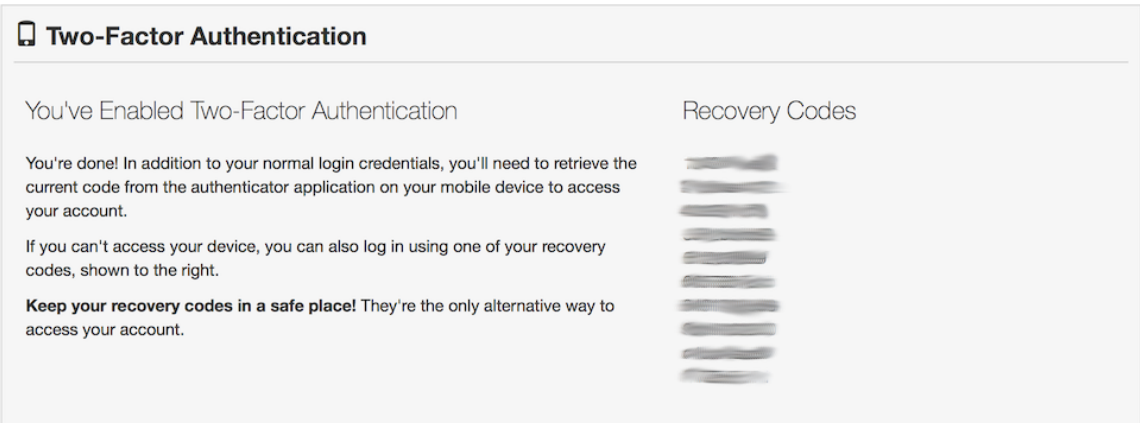


6. Launch the authenticator application installed on your mobile device and scan the displayed QR code or manually enter the key displayed in the setup window. A time-based authentication code appears on your mobile device.

7. In the **Authentication Code** field, type the time-based authentication code displayed on your mobile device.

   Android Users: Please note a common time syncing issue may cause your authenticator codes to fail. You can correct this using Google's instructions (https://support.google.com/accounts/answer/185834?hl=en#sync) for your authenticator app.

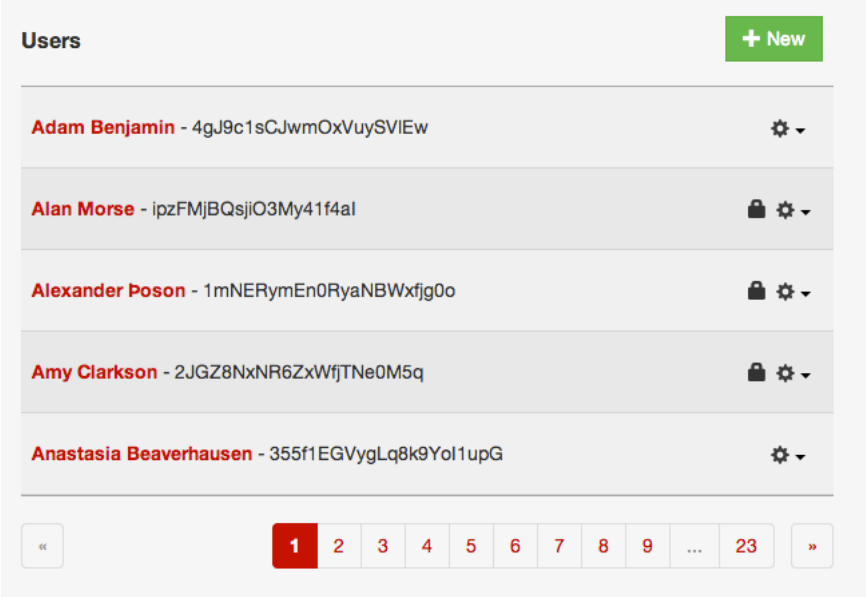
8. Click **Enable Two-Factor Authentication**. The confirmation screen appears along with your recovery codes.



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.

After you enable two-factor authentication, logging in to your Fastly account will require your email address and password, and then an authentication code generated by the authenticator application you've installed on your mobile device. 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 application. You may be required to authenticate more or less often depending on the timeframe specified by your account owner, however.

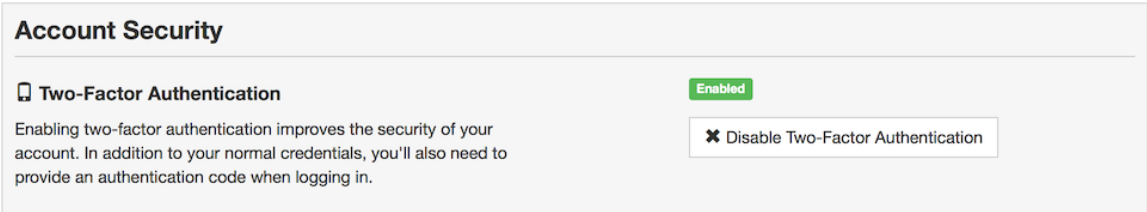
Account owners and super users: You can view who has enabled two-factor authentication at any time by viewing the Users area in the Account Settings on the account tab while logged into the Fastly application. Any user with two-factor authentication activated will appear in the users list with a padlock next to their name.



Disabling two-factor authentication

Two-factor authentication is disabled by default when you sign up for Fastly services. Once it's been enabled for your account, you can disable it at any time by following the steps below.

1. Log in to the Fastly application.
2. Click the **account** tab to access the account settings.
3. In the **Account Security** area of the settings sidebar, click **Manage your security settings**. The two-factor authentication controls appear.



4. Click **Disable Two-Factor Authentication**. The verification screen appears.

5. In the **Authentication Code** field, type the time-based authentication code displayed in the authenticator application on your mobile device, then click **Disable**.

What to Do If You Lose Your Mobile Device

If you've enabled two-factor authentication and you lose your mobile device, use a recovery code to log in to your 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 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.

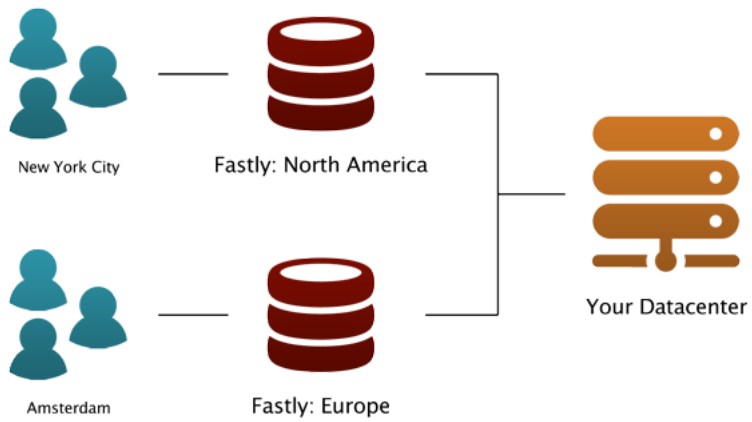
Locked out of your account? See our article on what you can do about it (/guides/customer-support/help!-im-locked-out-of-my-account!).

· How to Get Started with Fastly (/guides/getting-started/how-to-get-started-with-fastly)

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

How Fastly Works

In a nutshell, Fastly works by storing your website's content on servers all over the world and speedily delivering that content to your users.



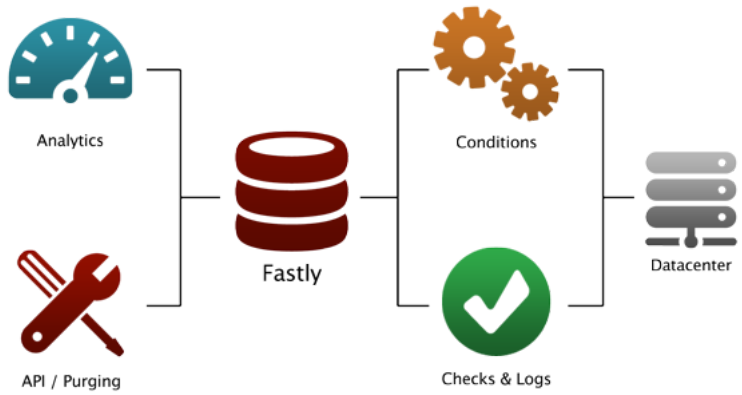
We track the geo-location of each user and make sure they are connecting to a server that is closest to them. This makes your site faster by reducing the time spent waiting for data to be sent from the server to the user.

We also give you full control over when and how we store content from your servers. You can easily 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 (/api/purge).

By using these tools you may only have to generate pages for the site once 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 will help you monitor how your data is being accessed and customize how content is delivered.



Instant Purging (/guides/purging/) allows you to have fine grained control over when and how content is updated for your users. Gone are the days of waiting up to 24 hours to change data at the edge, with Fastly you can update you data when you want and as often as you want.

With **Real Time Analytics** you always have 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.

Using **Conditions (/guides/tutorials/conditions-tutorial)** you can change how requests are routed, what headers to send, and how content is cached.

We make it easy to setup **Health Checks (/guides/backend-servers/health-checks-tutorial)** that monitor the status of one or many of your backend servers. This way if anything goes wrong with your servers you will know about it immediately.

System Logs (/guides/log-streaming/setting-up-remote-log-streaming) can be quickly and easily configured to send information from your servers anywhere and in the format you want.

Your entire configuration can even be handled programmatically via the **Fastly API (/api/)**. This allows you to write scripts to handle basic configuration tasks and even 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 then you should sign up now (https://www.fastly.com/signup/). Feel free to choose the developer plan so you can test how Fastly works on your site.

If you are excited to begin using Fastly but don't know where to begin then check out our Getting Started (/guides/getting-started/) documentation. You can learn everything you need to set up and configure your first service for your site. Once you're ready to explore more, check out the configuration Basics documentation to learn more about caching, as well as features like purging (/guides/purging/) and shielding (/guides/shielding/). And if you're looking for information on more advanced features, especially related to things like load balancing (/guides/load-balancing/) or the Varnish Control Language (/guides/vcl-faqs/) we support, you can check out our Advanced Configuration section of our help files or the API Reference (/api/), which includes a full reference to the Fastly API.

Finally, if you are having problems check out the list of Frequently Asked Questions (/guides/faqs/) or send us a message at support@fastly.com (mailto:support@fastly.com) and we'll help sort things out.

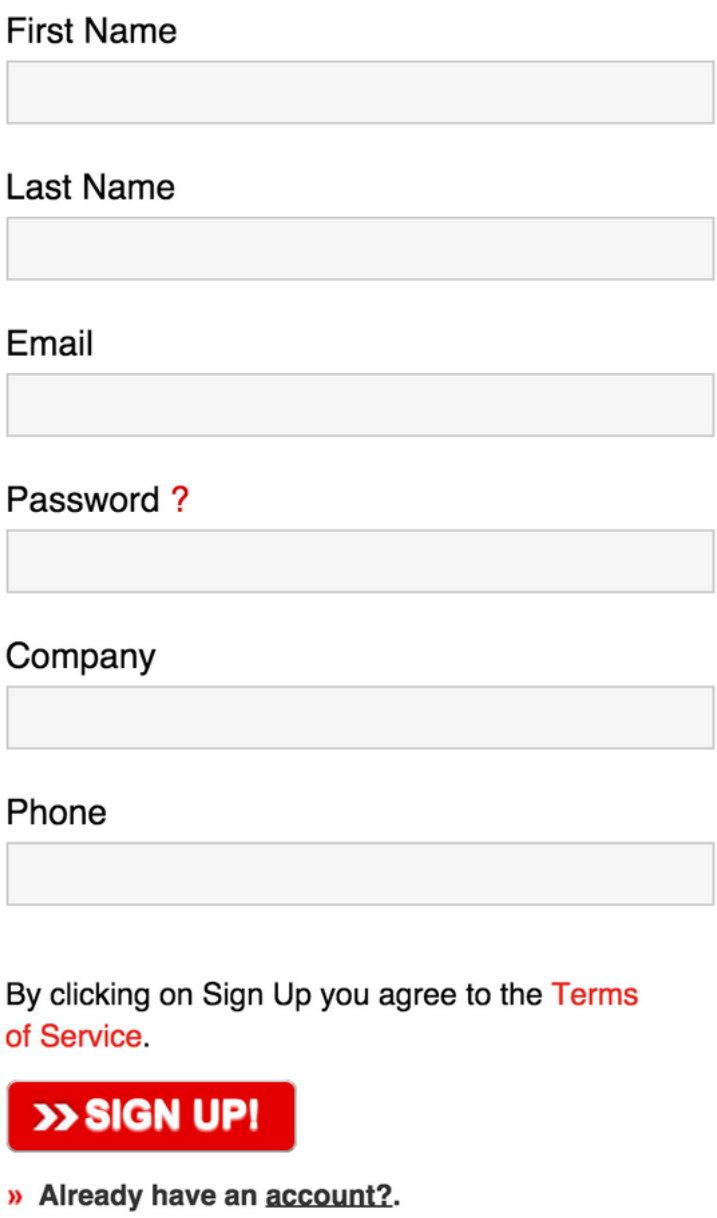
· Sign Up and Create Your First Service (/guides/getting-started/sign-up-and-create-your-first-service)

To create a Fastly account and set up your first service, follow the steps below.

Sign Up at Fastly.com

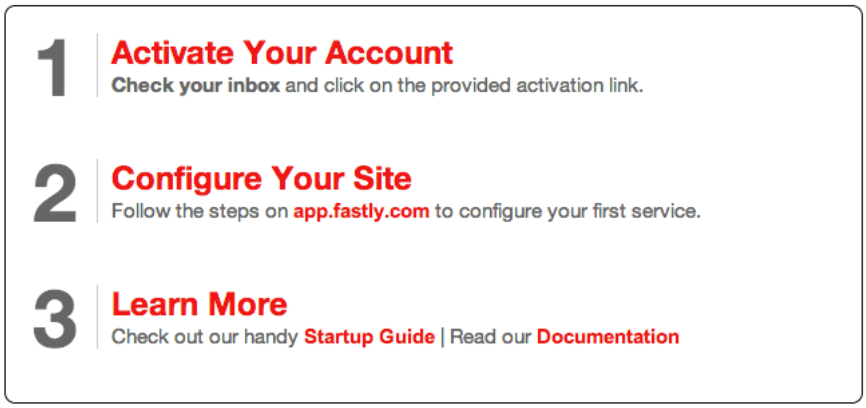
Before you do anything else, you must sign up for a Fastly account.

1. Click on any **Try Fastly Now** button on the Fastly.com website or simply point a browser to the signup form (https://www.fastly.com/signup/).
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 (/guides/customer-support/how-do-i-change-the-email-or-password-associated-with-my-account) any time after signup.

3. Click the **Sign Up** 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 the first step of the quick start process so you can create your first service.

Create Your First Service & Test It

Once you've verified your email, we log you into the application automatically and immediately take you through the quick start process to create your first service.

1. In the **Name Your Service** field, type the name of the first service you'd like to create. This name can be anything at all that will help you distinguish this service from any others you create in the future. You can rename this service (/guides/faqs/how-do-i-rename-my-service) at any time.
2. In the **Server Address** field, type the IP address (or hostname) and port number (usually 80) for your website's server. We need this to make sure your cache updates properly. If you have many servers you can configure the rest later.
3. In the **Domain Name** field, type the domain name of your website. We need this information to properly route requests to your website.
4. Click the **Configure** button to create your first service. The system configures your service and displays a link for you to test.
5. Click the configuration test link that appears. Your website should appear, but it may take up to 60 seconds as all the systems start talking to each other.

CNAME Your Domain

To complete the process of creating your first service, you must CNAME your domain to `global.prod.fastly.net`. CNAME-ing to Fastly will direct traffic to us. Each service provider does this a little differently, but we've written a separate guide that details the information most of the major providers (/guides/getting-started/cname-instructions-for-most-providers) will ask you for. Once you've completed this step, you should be all ready to go!

If you are having any problems, feel free to contact us at support@fastly.com (mailto:support@fastly.com).

· Transferring Account Ownership to Another User (/guides/getting-started/transferring-account-ownership)

To transfer account ownership to another user, please contact support@fastly.com (mailto:support@fastly.com) for assistance.

Remember that you can always add users to your account by inviting them (/guides/getting-started/adding-new-users-to-your-account), and then assigning them the appropriate role (/guides/customer-support/what-type-of-access-do-different-users-have).

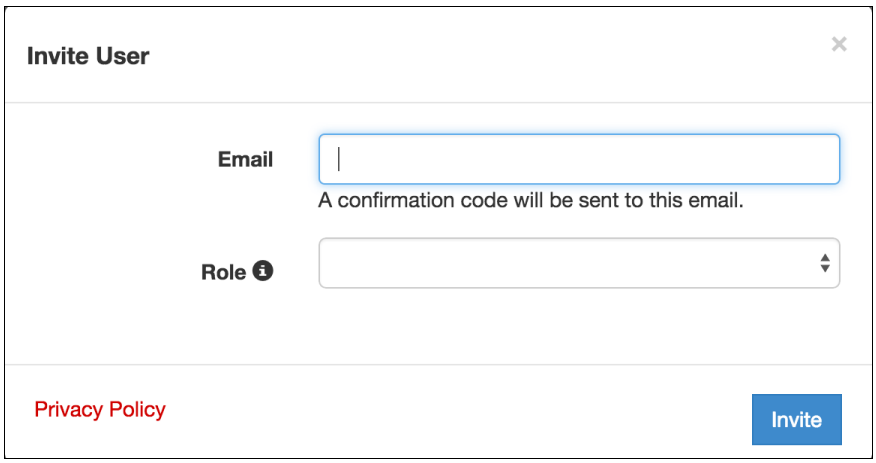
• Billing & Prices (/guides/billing-and-prices/)

· Can I get a copy of my bill sent to a person other than the owner of the account? (/guides/billing-and-prices/can-i-get-a-copy-of-my-bill-sent-to-a-person-other-than-the-owner-of-the-account)

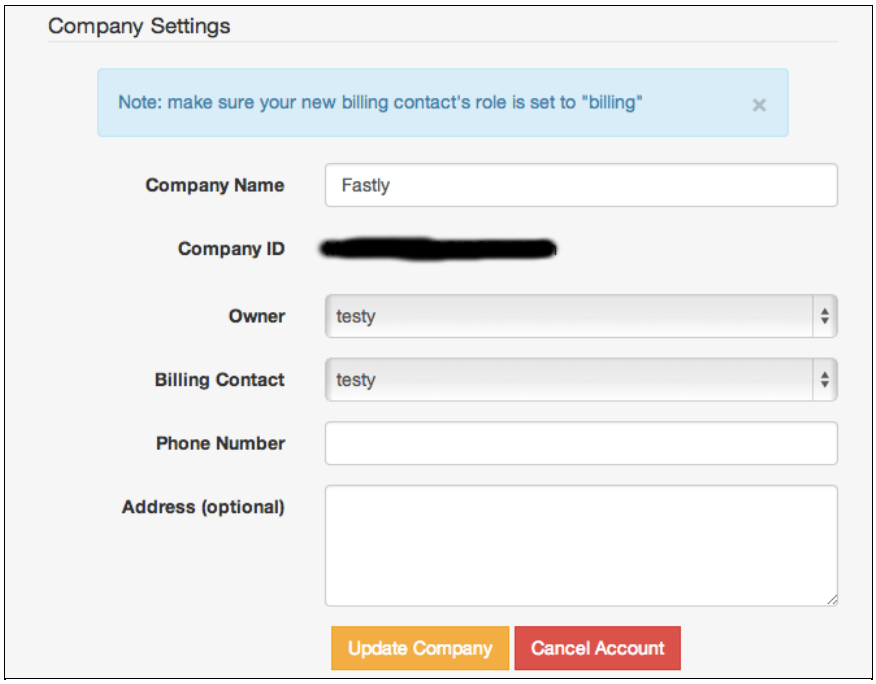
By default, billing invoices (/guides/billing-and-prices/what-types-of-billing-are-available-how-will-i-be-billed) are sent to the owner of a company's account with Fastly. That contact can be changed by anyone assigned the superuser role (/guides/customer-support/what-type-of-access-do-different-users-have) for an account. If you have been assigned the role of superuser, follow the steps below to have your billing invoice sent to a person other than the owner of your account with Fastly.

REMEMBER: Invoices are only sent to the email addresses of the account Owner and and the Billing Contact. Invoices are not sent to users with a "billing" role. To send invoices to multiple people, we recommend setting up an email group and setting that email group as your Billing Contact user.

1. Log in to the Fastly application (https://app.fastly.com/).
2. Click the **account** tab to access your Account Settings.
3. In the **User Invitations** area, click the **Invite** button. The Invite 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.
5. From the **Role** drop down menu select the role to assign (/guides/customer-support/what-type-of-access-do-different-users-have) once the user accepts the invitation.
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** tab in the application.
8. In the **Company Settings** area, select the user's name from the **Billing Contact** drop down menu, and then click **Update Company** to set the billing contact.



· Can I really test out your service for free? (/guides/billing-and-prices/can-i-really-test-out-your-service-for-free)

Our development plan allows you to try out our services free of charge and test everything to ensure it fits your requirements without needing to worry about costs while doing so. Our development plan isn't limited in functionality; however, it's not designed for you to use with your regular production traffic. For that we ask you to move over to a paid plan.

Conveniently, our paid plan is a pay-as-you-go option. No commitment is required. We bill monthly according to that month's usage (the combined total of your bandwidth and request usage), using the rates stated on our pricing page (http://www.fastly.com/pricing/). We do, however, include a $50 minimum each month on these usage fees once you're pushing production-level traffic to us. Once your testing is complete, we charge this monthly minimum fee to make sure we can fully support all of our customers. If your usage fees fall below this amount in any individual month, we round up to $50.

For example, say that most of your site's traffic is in North America and Europe and you're done testing out Fastly's services. You've begun to push production-level traffic through Fastly. If your site uses 10GB of traffic over 10 million requests, the charge per Fastly's pricing page would be $8.70. Because this amount falls below the $50 monthly minimum, we would charge you $50 for that month. The Asia-Pacific, Australia, and new Zealand prices are slightly higher, but at the above traffic levels they would still fall within the $50 monthly minimum.

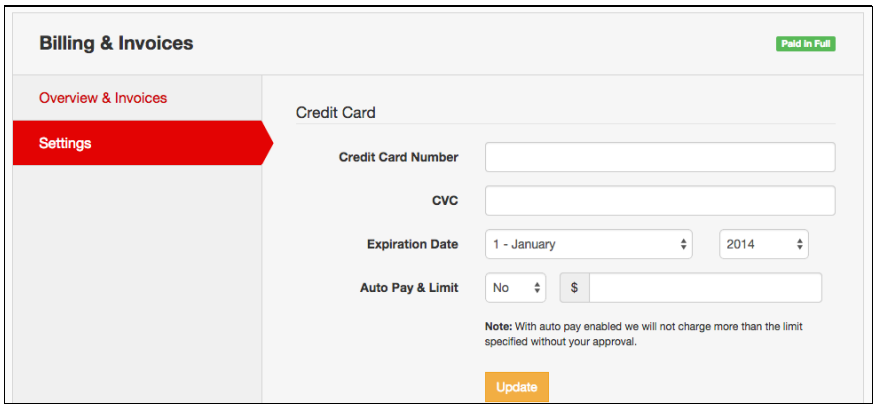
If you'd like any help getting set up with a development account (which you can move over to a paid account later if you like the service), please don't hesitate to reach out to us. Technical questions are best sent to the support@fastly.com (mailto:support@fastly.com) email address, and commercial ones through sales@fastly.com (mailto:sales@fastly.com) (just so you get a speedy response appropriately).

· Can I set up automatic payments? (/guides/billing-and-prices/can-i-set-up-automatic-payments)

To set up automatic payement of your bill (/guides/billing-and-prices/what-types-of-billing-are-available-how-will-i-be-billed) for Fastly services, you must have a credit card on file in the Billing & Invoices (/guides/fastlys-web-interface/about-the-billing-and-invoice-controls.html) of your account settings and specifically set your account.

1. Log into the Fastly application (https://app.fastly.com).

2. Click the **billing** tab. The Billings & Invoices controls appear.

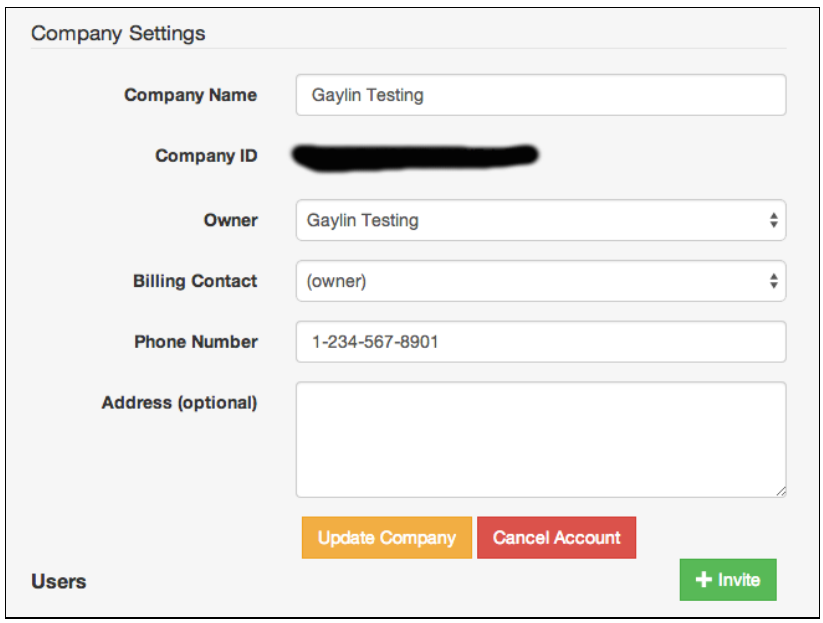


3. In the **Credit Card** area of the **Settings** pane, add the credit card number, card verification code, and expiration date for the credit card to be used for automatic billing.
4. From the **Auto Pay** drop down menu, select **Yes**.
5. In the **Limit** field, type the maximum dollar amount you are willing to charge to the credit card on file. If the total bill for Fastly services exceeds the limit you set for the card we have on file, the card will not be charged at all and you will be required to make your payment manually for the invoiced billing cycle.
6. Click **Update** to save your credit card and billing limit information.

· How do I cancel my account? (/guides/billing-and-prices/how-do-i-cancel-my-account)

To cancel your account, log into the Fastly application and click the account tab:

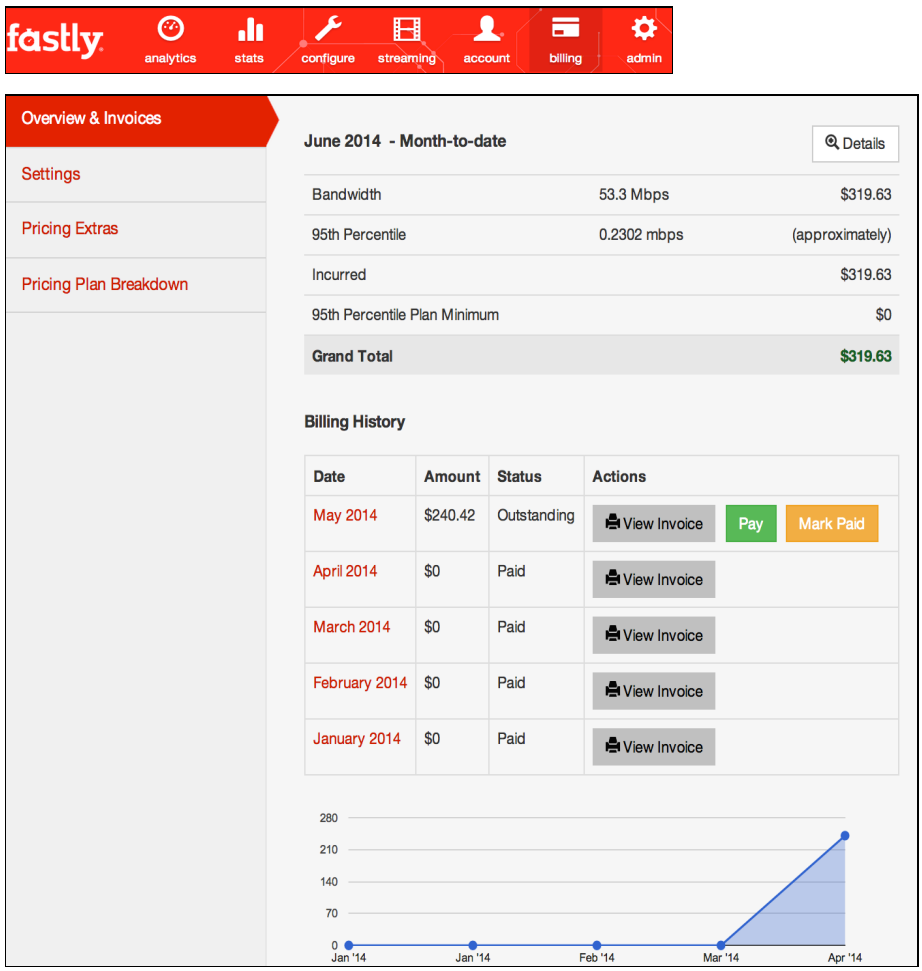
In the **Company Settings** area, click the **Cancel Account** button.



After your account is canceled, you'll be billed for any outstanding charges (/guides/billing-and-prices/what-types-of-billing-are-available-how-will-i-be-billed) accrued through the day you canceled. If you decide at a later date to reactivate your account, feel free to contact customer support (mailto:support@fastly.com) and request reactivation.

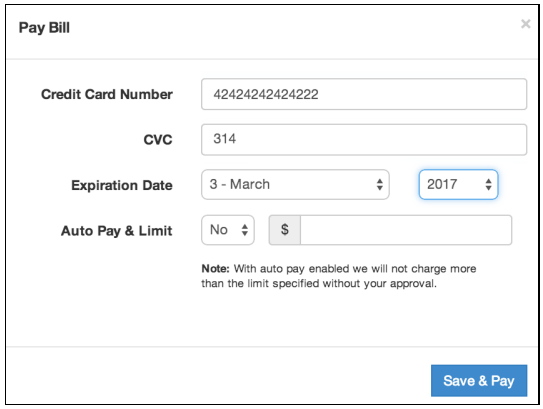
· How do I pay? (/guides/billing-and-prices/how-do-i-pay)

During the month you can keep up to date on what your current incurred amount is by logging into the Fastly application and accessing your billing page (https://app.fastly.com/#billing).



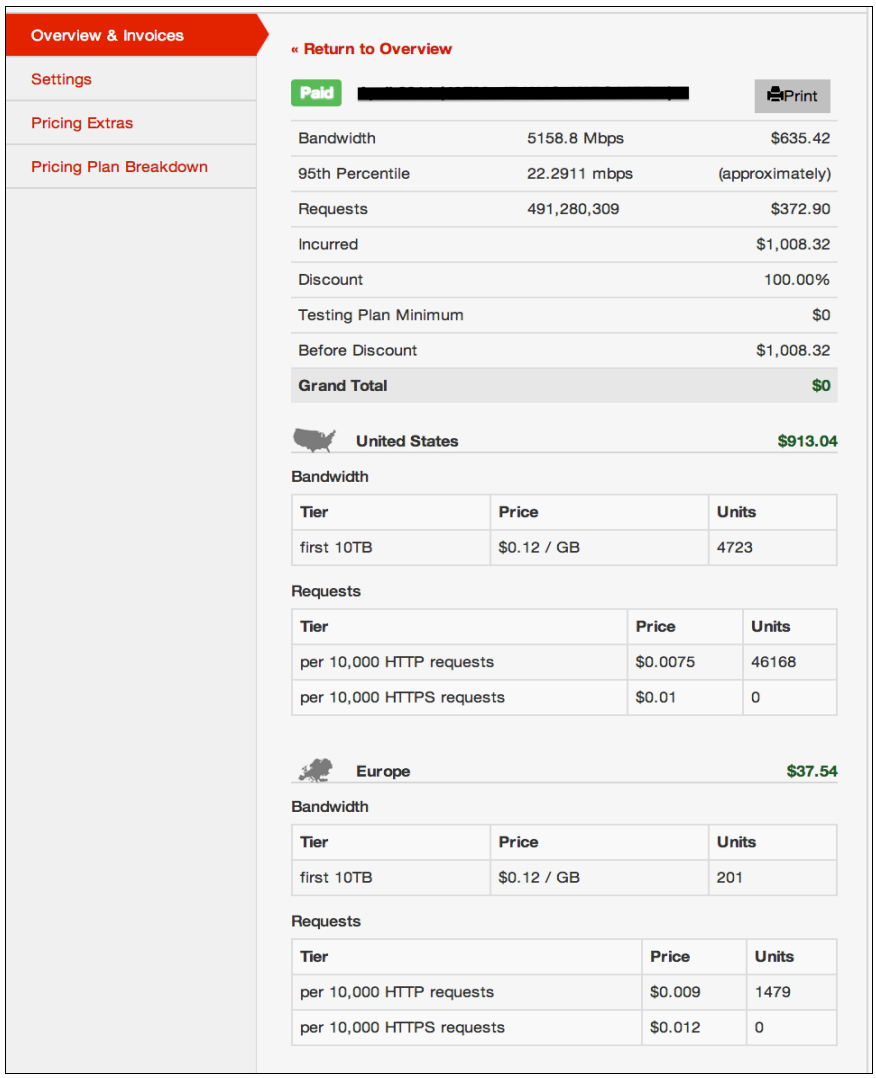
At the end of each month you'll be sent an email with how much you've incurred that month and a link to a specific page for that invoice.

From there you'll have a button to pay which will pop up a window for you to enter in your credit card details.

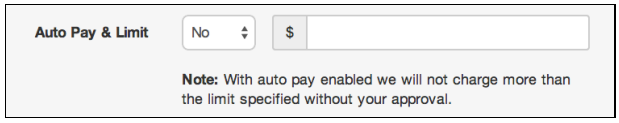


Fastly never sees your credit card number and all transactions are handled by our fully PCI compliant payment gateway (https://stripe.com/us/help/faq#security-and-pci), Stripe.

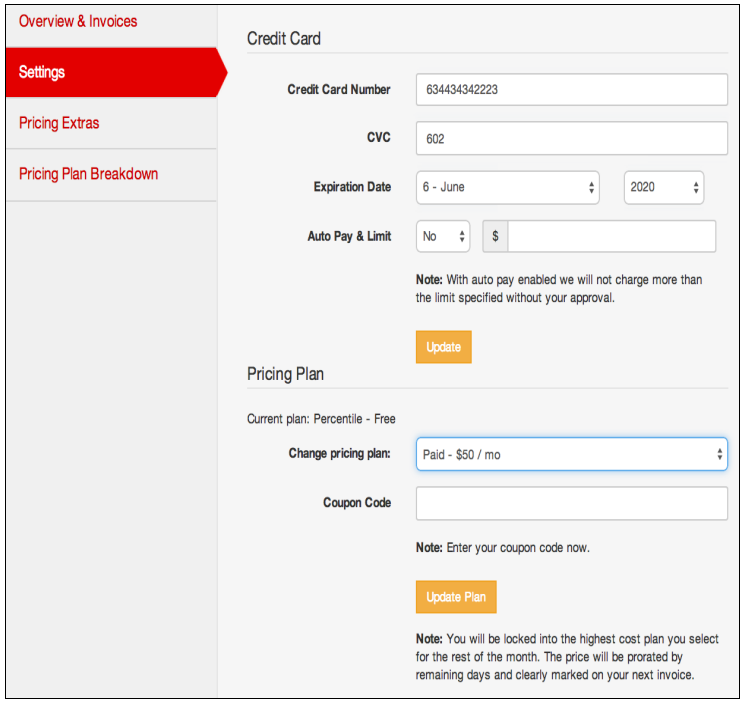
Once that's done the bill will update to indicate its been paid.



You also have the option to save your credit card details (again, Fastly never sees these details, they're stored with our payment gateway). When your card is stored it makes it easy for you to pay future invoices with one button. Alternatively you can set up Fastly so that your invoices are automatically charged to that credit card each month with an optional limit (leave the limit blank to have no limit).

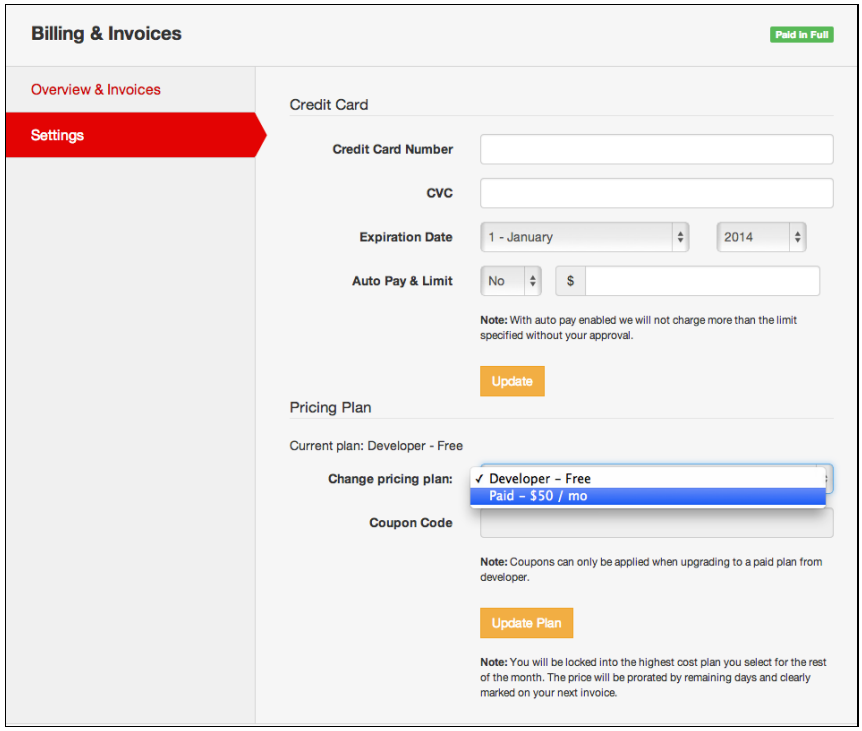


If you want to change any of your details later then you can just go to the settings pane of your billing tab.



· How do I switch from a Developers Plan to a Paid Plan? (/guides/billing-and-prices/how-do-i-switch-from-a-developers-plan-to-a-paid-plan)

Once you've completed any development testing (/guides/billing-and-prices/can-i-really-test-out-your-service-for-free), you can switch your account to a Paid Plan by selecting "Paid" from the Change pricing plan menu under the Pricing Plan area of the Settings section on the billing tab.



· How much do Fastly's services cost? (/guides/billing-and-prices/how-much-do-fastlys-services-cost)

Prices for Fastly's content delivery, streaming delivery, and SSL service options are detailed on the Fastly website pricing page (http://www.fastly.com/pricing/). Our SSL service options are also described in depth in our SSL pricing guide (/guides/ssl/which-ssl-options-are-available-and-how-much-do-they-cost). Prices for SSL service options are applied in addition to normal content delivery and streaming charges.

If you're using Fastly for content delivery via Heroku's cloud development services, please see Fastly's Heroku add-ons pricing plan (https://addons.heroku.com/fastly) for more details.

Do you have other questions about pricing? Please feel free to contact sales@fastly.com (mailto:sales@fastly.com) for more information.

· What types of billing are available? How will I be billed? (/guides/billing-and-prices/what-types-of-billing-are-available-how-will-i-be-billed)

What types of billing are available?

We generally bill for both content delivery and streaming services based on bandwidth used (measured in Gbytes) and number of requests. All pricing rates appear on the pricing page (http://www.fastly.com/pricing/) of our website.

We give discounts for larger pricing plans, for commitments of 12 months or longer, and for larger customers. For some customers we can also bill for bandwidth on the 95th percentile.

Are you an open source developer? We believe open source pulls the best thought from the best minds (in fact, we contribute heavily to Varnish, Chef, Ganglia and a number of other leading open-source initiatives). If you’re an open source developer, contact us. We'll host you for free.

Please contact sales@fastly.com (mailto:sales@fastly.com) for additional information.

How will I be billed?

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.

We send invoices out at the beginning of the month that allow you to pay by credit card. We offer the ability to have a credit card on file for automatic billing (/guides/billing-and-prices/can-i-set-up-automatic-payments) and we include the ability to set limits to prevent unexpected billing surprises. See our article about how to pay (/guides/billing-and-prices/how-do-i-pay) for more information.

If necessary we can also accept checks or wire transfers. Please contact billing@fastly.com (mailto:billing@fastly.com) for additional information.

Who receives the bill?

By default, the account owner will receive the bill for Fastly services. You can change your billing contact (/guides/billing-and-prices/can-i-get-a-copy-of-my-bill-sent-to-a-person-other-than-the-owner-of-the-account.html) at any time. If you ever delete your billing contact, billing will automatically revert to the account owner.

• FAQs (/guides/faqs/)

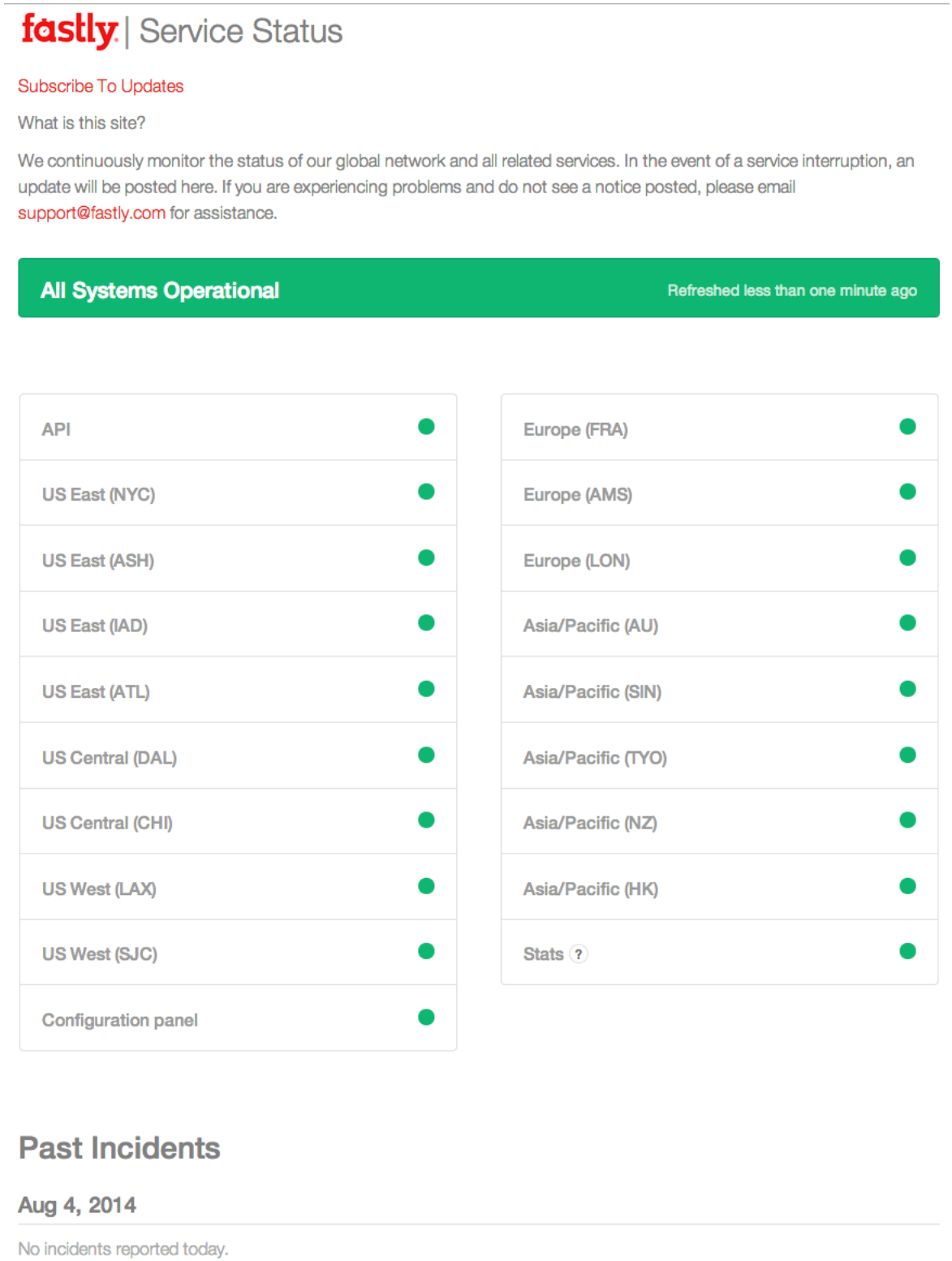
· Can Fastly host my content? (/guides/faqs/can-fastly-host-my-content)

We accelerate your site by caching both static assets and dynamic content by acting as a reverse proxy (http://en.wikipedia.org/wiki/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 Elastic Compute Cloud (EC2), Amazon Simple Storage Service (S3), and Google Cloud as your file origin. We cover topics related to origin servers in the Backend Servers (/guides/backend-servers/) section of our documentation.

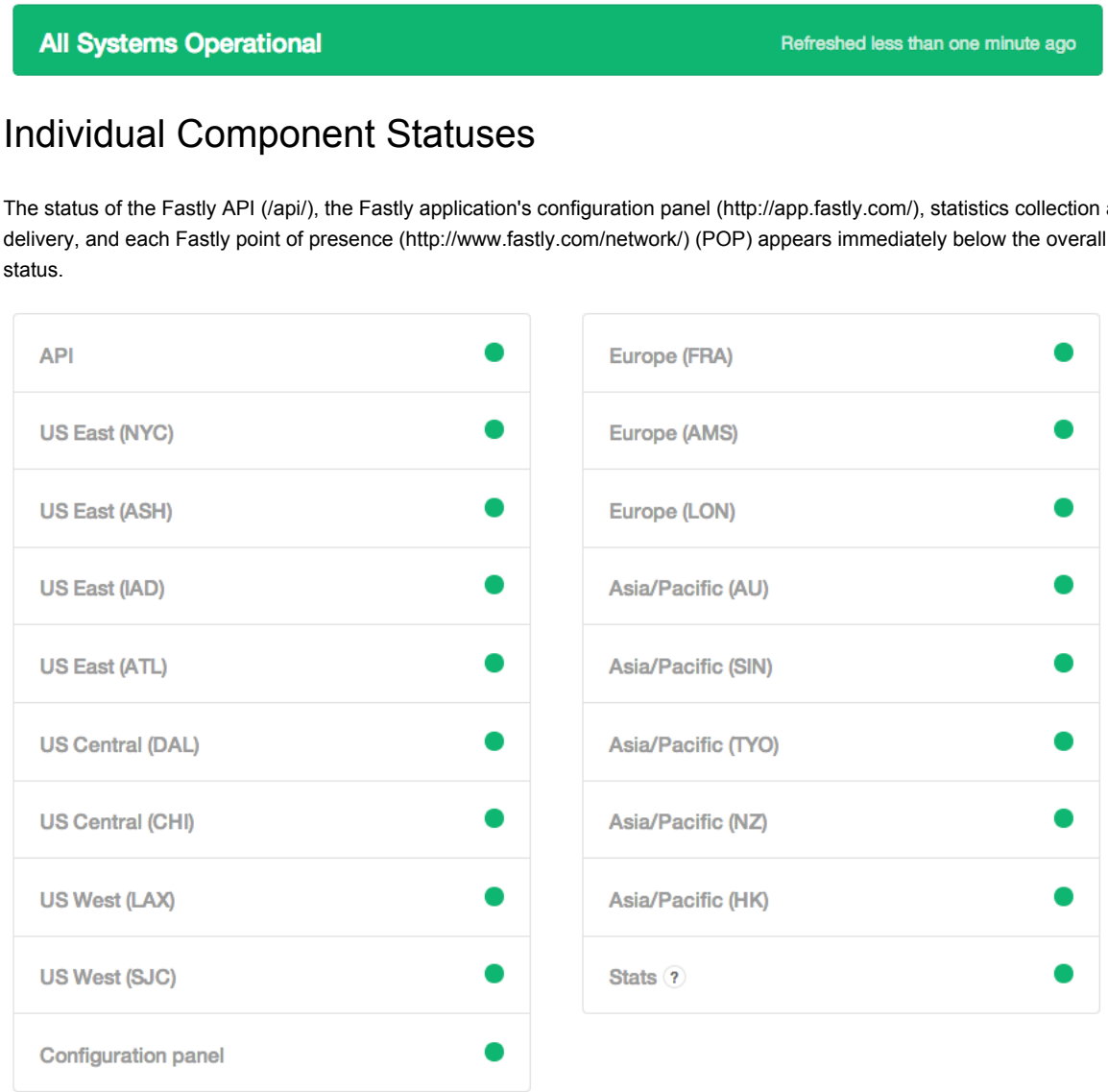
· Can I check on the status of Fastly's network and services? (/guides/faqs/can-i-check-on-the-status-of-fastlys-network-and-services)

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 (http://status.fastly.com). If you are experiencing problems and do not see a notice posted, please email support@fastly.com (mailto: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 (/api/), the Fastly application's configuration panel (http://app.fastly.com/), statistics collection and delivery, and each Fastly point of presence (http://www.fastly.com/network/) (POP) appears immediately below the overall status.

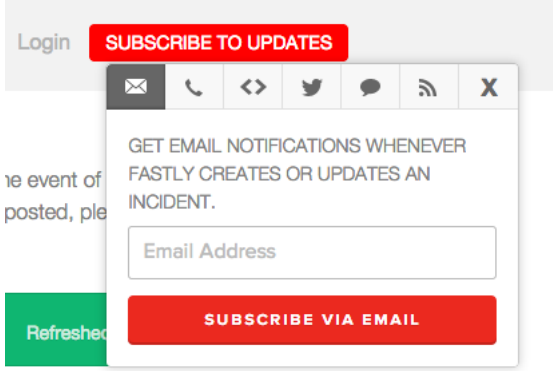
Past Incident Statuses

Fastly keeps track of past incidents. Past incidents, if any, for approximately the past two weeks appear immediately below the individual component statuses.

We also keep track of all past incidents in an incident history page (http://status.fastly.com/history).

Subscribing to Email Notifications

Fastly allows you to subscribe to status email notifications. Simply click the **Subscribe to Updates** button in the upper right corner of the status page screen, type your email address in the displayed field, and click **Subscribe Via Email**. Once subscribed, we'll email you any time we create or update an incident.



· Does Fastly support the SPDY protocol? (/guides/faqs/does-fastly-support-the-spdy-protocol)

SPDY (http://en.wikipedia.org/wiki/SPDY) is a protocol being explored as an alternative for transporting web content. It is specifically being designed with the intent to "reduce the latency of web pages" (you can read the Chromium Projects white paper on SPDY (http://dev.chromium.org/spdy/spdy-whitepaper) for more information). We're definitely considering SPDY, as it relates to our core goal of creating the fastest, most flexible, dev-friendly tool for scaling websites. At present, however, our existing tech doesn't support SPDY. Support will take time to develop and roll out. There's actually lower hanging fruit in our stack right now for optimization than SPDY alone.

We're also keeping track of the emerging HTTP 2.0 standard (http://tools.ietf.org/html/draft-tarreau-httpbis-network-friendly-00), which is based on SPDY (http://en.wikipedia.org/wiki/HTTP_2.0#SPDY_and_HTTP_2.0). We have plans to support HTTP 2.0 once the standard is solidified and browser support is available.

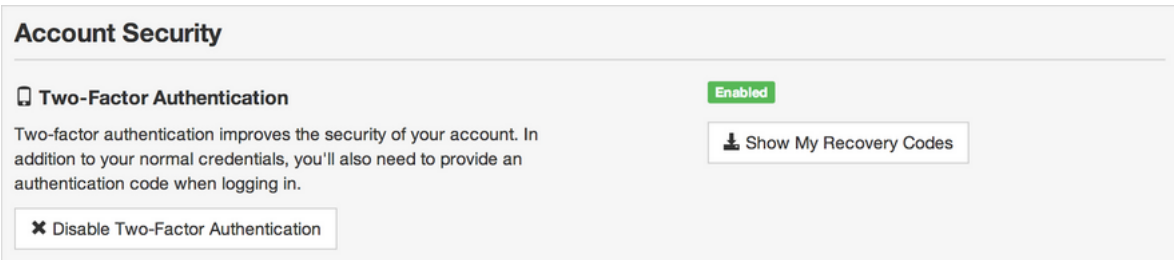
· Does Fastly work with Magento? (/guides/faqs/does-fastly-work-with-magento)

Fastly is based around the Varnish HTTP Cache and Magento has several different solutions for working with Varnish. Varnish Software company describes some of the different options in their blog post Doing the right thing: Magento + Varnish Cache (https://www.varnish-software.com/blog/doing-right-thing-magento-varnish-cache).

We haven't tested and do not officially support any of these specific methods. They may need some modifying to get them to work with our purging system (/guides/purging/), but if you are interested in working with us (mailto:support@fastly.com) then we'd love to help.

· Does the Fastly application support multi-factor authentication? (/guides/faqs/does-the-fastly-application-support-multifactor-authentication)

Fastly supports two-factor authentication, a two-step verification system, for logging in to the application. 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).



Two-factor authentication is disabled by default. You can enable and disable (/guides/getting-started/enabling-and-disabling-twofactor-authentication) it at any time in the Account Security area of the Account Settings for your Fastly account. Once enabled, an authentication code will be requested when you login at least every 14 days for each computer and browser you use to access the Fastly application.

IMPORTANT: If you enable two-factor authentication via the user interface, you will no longer be able to use a simple username+password when using the API, and must use the API key for authentication.

· How do I deactivate and reactivate services? (/guides/faqs/how-do-i-deactivate-and-reactivate-services)

To reactivate or deactivate a service, follow the steps below:

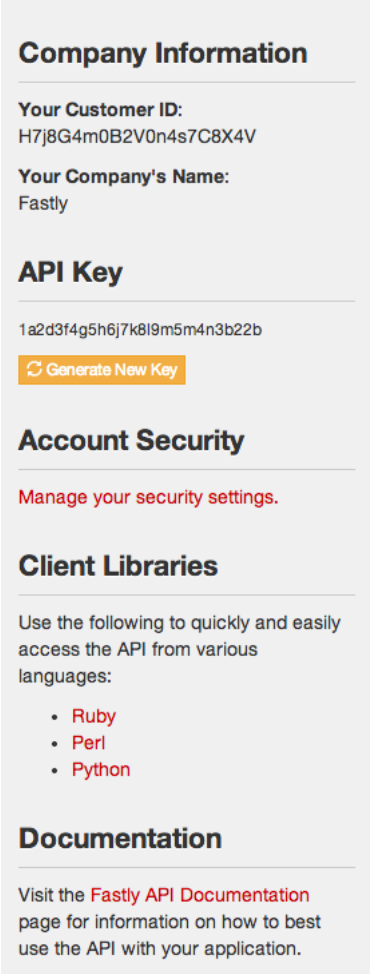
1. Log in to the Fastly application.
2. Click on the **configure** button (the wrench icon at the top of the application window).

3. Select the appropriate service from the **Service** drop down.
4. Click the yellow **Activate** or **Deactivate** button to the right of the service name as appropriate.

You can also activate or deactivate a service via the API (/api/config#version). Did you accidentally delete a service? We can help (/guides/customer-support/i-accidentally-deleted-a-service.-how-do-i-get-it-back).

· How do I regenerate my API key? (/guides/faqs/how-do-i-regenerate-my-api-key)

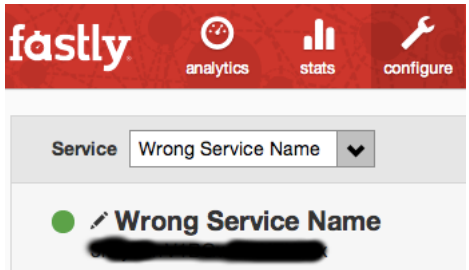
Log into the Fastly application (https://app.fastly.com/) and click on the **Account** button to view your account information. Then, click the **Generate New Key** button near the top left of the screen, immediately below your actual API key.



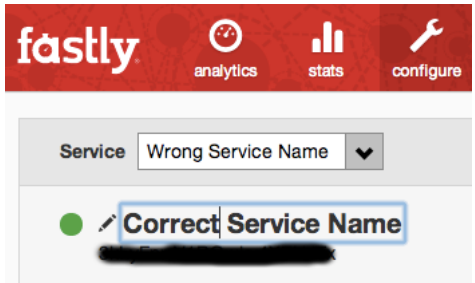
REMEMBER: Only account users with the superuser role (/guides/customer-support/what-type-of-access-do-different-users-have) can view and change the API key. If you don't see the API Key section, your role is lower than superuser and you must contact a superuser at your company to regenerate the API key.

· How do I rename my service? (/guides/faqs/how-do-i-rename-my-service)

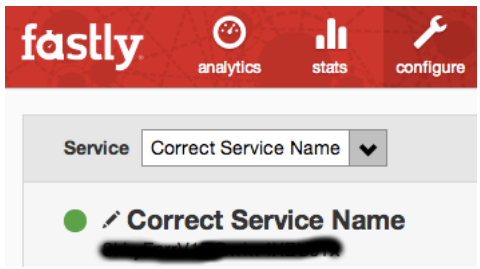
Different organizational schemes? Company mandate? Fat fingers and typos? All of these may contribute to a service name that just doesn't work any more. You can rename your service at any time by clicking your service name next to the pencil icon in the Fastly application (https://app.fastly.com/):



Your service name will become editable and you can immediately begin typing the new service name:



Simply press the return key so your changes will take effect:

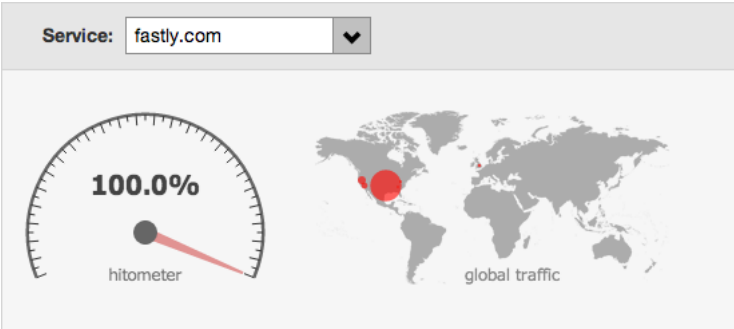


· What do the red dots on the global traffic map mean? (/guides/faqs/what-do-the-red-dots-on-the-global-traffic-map-mean)

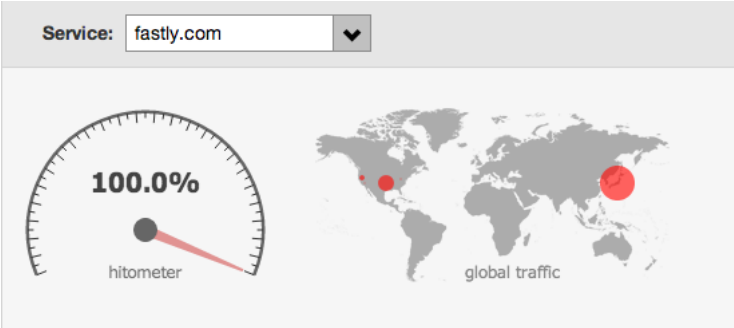
The global traffic map at the top of the Fastly Analytics Dashboard (/guides/fastlys-web-interface/about-the-analytics-dashboard) represents a real-time visual representation of the general regions of the world in which Fastly's points of presence (POPs) are receiving requests for your service. As each moment passes, the traffic changes, and these changes are reflected in the map visually by red dots that increase or decrease in size as your request traffic ebbs and flows.

For example, the following images show how requests to the Fastly.com website changed over the course of three seconds, at one second intervals:

The first image displays traffic coming into Fastly's POPs on the west coast of the United States (Los Angeles, to be exact), the central region of the US (in this case, Dallas), and in Europe via our Amsterdam POP.



The second image, taken one second later, shows how the global traffic has changed, with a great deal more traffic coming into our central US region, a smaller amount requested from two POPs in the western US region (Los Angeles and San Jose), and a still smaller amount going to the Europe via our London POP.



Finally, our third image, again taken one second later, shows how the global traffic has changed once again with traffic going through our Asia/Pacific POP in Tokyo, and smaller amounts going through Dallas and Los Angeles.

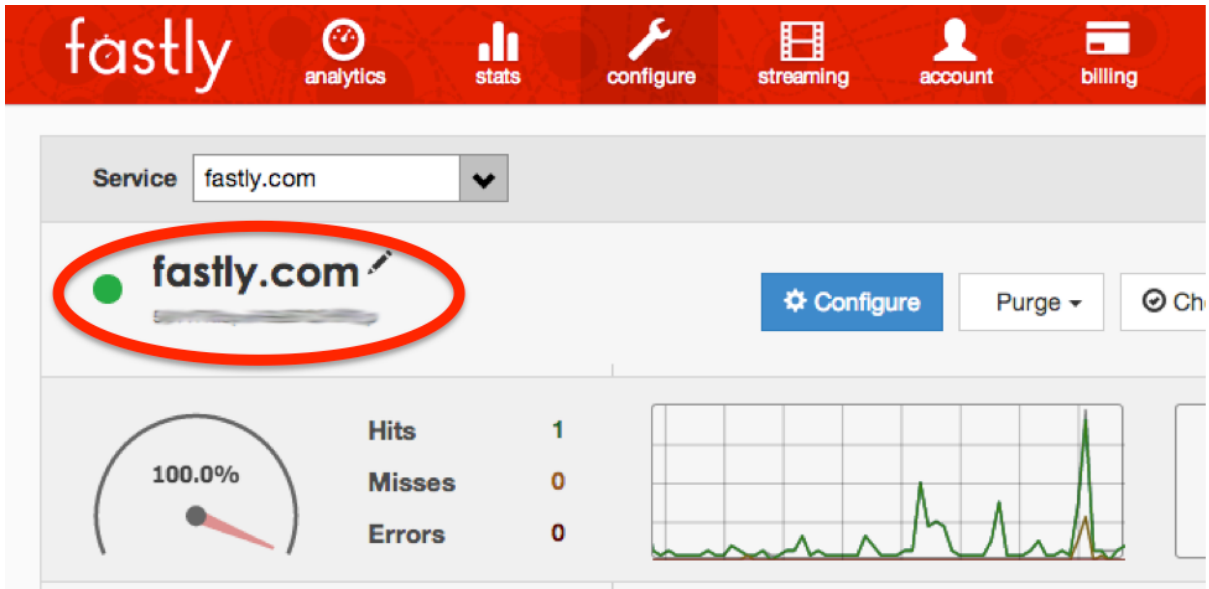
· Where are Fastly's POPs located? (/guides/faqs/where-are-fastlys-pops-located)

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 (http://www.fastly.com/network/) shows a detailed view of the current and planned locations of all Fastly POPs. For more information, about how we choose these locations, check out our blog post How Fastly Chooses POP Locations (http://www.fastly.com/blog/how-fastly-chooses-pop-locations/).

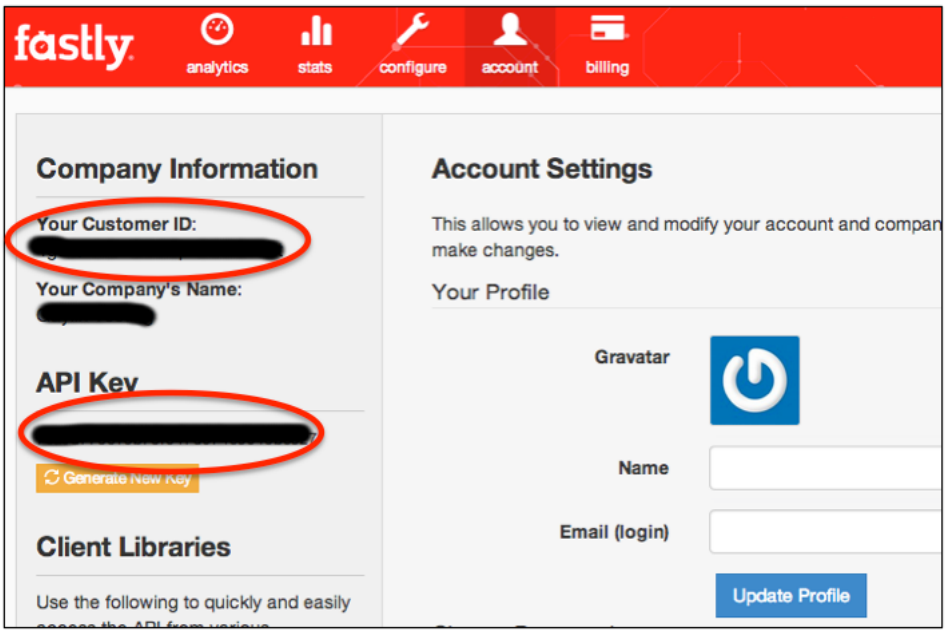
Once you're signed up for Fastly service (either through a test account (/guides/billing-and-prices/can-i-really-test-out-your-service-for-free) or a paid plan) you can a see a live, real-time visual representation (/guides/faqs/what-do-the-red-dots-on-the-global-traffic-map-mean) of the general regions of the world in which Fastly's points of presence (POPs) receive requests for your service.

· Where can I find my Service ID, Customer ID, or API Key? (/guides/faqs/where-can-i-find-my-service-id-customer-id-or-api-key)

Your **Service ID** is located on the **configure** tab at the top left of the screen in the Fastly application (https://app.fastly.com/):



Your **Customer ID** (also called your Company ID) and your **API Key** are located on the **account** tab to the left of the **Account Settings** area:



REMEMBER: Only account users with the superuser role (/guides/customer-support/what-type-of-access-do-different-users-have) can view and change the API key. If you don't see the API Key section, your role is lower than superuser and you must contact a superuser at your company to regenerate the API key.

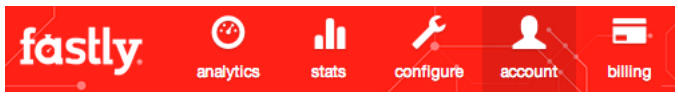
· Why did my Googlebot crawl stats drop so radically after I signed up with Fastly? (/guides/faqs/why-did-my-googlebot-crawl-stats-drop-so-radically-after-i-signed-up-with-fastly)

When you set up Fastly (/guides/getting-started/sign-up-and-create-your-first-service) to run production traffic, many of your site objects will acquire new entity tags (http://en.wikipedia.org/wiki/HTTP_ETag) (ETags). Web servers assign these identifiers as part of the normal process of uniquely identifying resource content each time it changes at a URL (http://en.wikipedia.org/wiki/Uniform_Resource_Locator). Fastly set up is no exception to this normal process.

Occasionally, Google's webcrawlers misidentify changes like these as malicious, and stop crawling specific sites as a protection measure. If you notice any major changes in your SEO stats, indexing, or crawler behavior after completing your Fastly production setup, we recommend taking advantage of Google's Webmaster Tools (https://www.google.com/webmasters/), specifically the Fetch as Google Tool (https://support.google.com/webmasters/answer/158587)article and the sections on Additional Support (https://support.google.com/webmasters/answer/1249981?hl=en).

• Fastly's Web Interface (/guides/fastlys-web-interface/)

· About the Account Settings Controls (/guides/fastlys-web-interface/about-the-account-settings-controls)



The Account settings controls allow customers to view and modify overall account access settings as well as personal profile settings for a logged in user

What Customers Can Control with the Account Settings

If their logged in permissions (/guides/customer-support/what-type-of-access-do-different-users-have) allow it, users can also change their password (/guides/customer-support/how-do-i-change-the-email-or-password-associated-with-my-account) to the account and administer the access of multiple users on that account. The Account settings area also displays the logged in user's customer ID, company name, and API key (/guides/faqs/where-can-i-find-my-service-id-customer-id-or-api-key) (which can be regenerated (/guides/faqs/how-do-i-regenerate-my-api-key) from this location as well). Customers can cancel their account (/guides/billing-and-prices/how-do-i-cancel-my-account) via the Account Settings page if they have the appropriate permissions at the time they're logged in.

When the Account Settings Controls Appear

Account settings controls appear automatically for logged in users with the appropriate access permissions. (/guides/customer-support/what-type-of-access-do-different-users-have)

· About the Analytics Dashboard (/guides/fastlys-web-interface/about-the-analytics-dashboard)

The Analytics Dashboard, sometimes simply referred to as "the Dashboard," allows customers to monitor caching for each of their services, one at a time in real time, as they operate on a second-by-second basis.

What the Analytics Dashboard Tells Users

The Dashboard specifically displays the following information about a service:

- the percentage of requests per second (via a Fastly hitometer)
- a global traffic profile (via the global traffic map (/guides/faqs/what-do-the-red-dots-on-the-global-traffic-map-mean))
- the time-stamped metrics specific to requests, hits, and misses (via the metrics table)

In addition, the Dashboard provides two scrolling graphs. The first graph displays either the number of requests, the bandwidth, or the hit ratio over a two-minute window for any single service as selected by the user. The two-minute rolling average for each of these measurements can also be displayed as an overlay. In the second graph displays the miss latency for any single service over the two-minute window. One minute after Analytics Dashboard measurement data in these graphs has rolled off the screen, it becomes available for retrieval by the Historical Stats controls (/guides/fastlys-web-interface/about-the-historical-stats-tool).

When the Analytics Dashboard Appears

The Dashboard appears automatically for logged in users with the appropriate access permissions (/guides/customer-support/what-type-of-access-do-different-users-have); however, it may appear grayed out and 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.

· About the Billing and Invoice Controls (/guides/fastlys-web-interface/about-the-billing-and-invoice-controls)

The Billing and Invoices controls provide customers with an overview of charges accrued to date (/guides/billing-and-prices/what-types-of-billing-are-available-how-will-i-be-billed) for the current month. These charges are displayed both in aggregate and detailed by POP region (/guides/faqs/where-are-fastlys-pops-located). Customers can also access their complete billing history, payment statuses, payment amounts, and viewable invoices.

What Customers Can Control via Billing and Invoices

Customers can change their payment settings, including their payment method, the credit card that should be charged for payment of the account with Fastly, whether or not automatic payments (/guides/billing-and-prices/can-i-set-up-automatic-payments) should be made for them, and which pricing plan they currently participate in.

In addition to being able to tell at a glance whether or not their account is paid in full when they view these controls, customers can use them to switch between plan types (/guides/billing-and-prices/how-do-i-switch-from-a-developers-plan-to-a-paid-plan), as well as cancel Fastly services (/guides/billing-and-prices/how-do-i-cancel-my-account) at any time.

When the Billing and Invoices Controls Appear

Billing controls appear automatically for logged in users assigned the billing or super user roles (/guides/customer-support/what-type-of-access-do-different-users-have).

· About the Configuration Control Panel (/guides/fastlys-web-interface/about-the-configuration-control-panel)

The configuration Control Panel allows customers to define exactly how each instance of their cache should behave and deliver content from each data sources. Customers use the Control Panel to create versions of each service's configuration settings and then use the controls to deploy or deactivate (/guides/faqs/how-do-i-deactivate-and-reactivate-services) them.

What Customers Can Control with the Configuration Control Panel

Specifically, customers with the appropriate permissions can configure and manage:

- the domains (/guides/domains-and-dns/) used to route requests to a service
- the hosts used as backends (/guides/backend-servers/) for a site and how they should be accessed
- the health checks (/guides/backend-servers/health-checks-tutorial) that monitor backend hosts and the alerts that Fastly sees when there is trouble with them
- various request and cache settings (/guides/caching/), headers, and responses that control how Fastly caches and serves content for a service
- how logging (/guides/log-streaming/setting-up-remote-log-streaming) should be performed and where server logs should be sent (as specified by an RSYSLOG endpoint)

- various third-party plugins available for use with Fastly services (e.g., WordPress (https://github.com/fastly/WordPress-Plugin))
- custom Varnish configuration language (/guides/vcl-faqs/) (VCL) files if custom VCL is enabled (/guides/vcl-faqs/where-can-i-upload-custom-vcl)
- how conditions (/guides/tutorials/conditions-tutorial) 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)

Users can activate configuration changes immediately and roll back those changes just as quickly should they not have the intended effect. The Control Panel also allows users to compare differences (http://www.fastly.com/blog/introducing-version-diff/) between two configuration versions.

When the Control Panel Appears

The Control Panel appears automatically for logged in users with the appropriate access permissions (/guides/customer-support/what-type-of-access-do-different-users-have).

· About the Historical Stats Tool (/guides/fastlys-web-interface/about-the-historical-stats-tool)

Fastly's Historical Stats tool provides you with statistical information about your website. We display the information as a series of graphs derived from your sites metrics starting from the moment you first start using Fastly.

Using Fastly's Historical Stats tool allows you to view two core categories of metrics:

- **Caching and Performance**. These metrics help you optimize your website's speed.
- **Traffic**. These metrics help you analyze your website's traffic as it evolves over time.

Taken together, these statistics can help you not only optimize the overall performance of your website but also expose issues that may be hidden by caching, such as degradation of origin performance or unusually high numbers of errors and redirects for a particular timeframe. By analyzing metrics beginning from the very moment you start using Fastly, you can see how your traffic has grown over the lifetime of using Fastly. Use the information to not only diagnose non-trivial issues, but also make informed marketing and business development decisions by identifying clear traffic growth and patterns for your site.

About Each of the Core Metrics

The Historical Stats tool provides a graphical interface to our popular Historical Stats API (/api/stats). Two types of metrics are displayed: caching-performance metrics and displayed traffic metrics.

The displayed caching and performance 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 displayed traffic metrics include the following:

- **Requests** metrics show you the total number of requests over time that were received for your site by Fastly.
- **Bandwidth** metrics show you the amount of bandwidth (measured in bytes) served for your site by Fastly.
- **Header & Body Bandwidth** metrics show the relative proportion of bandwidth (measured in bytes) used to serve the

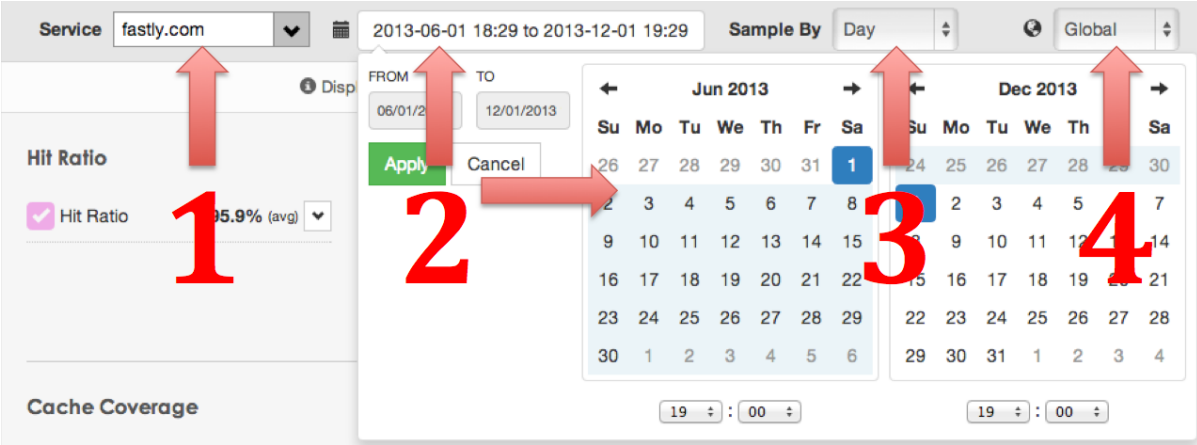
body portion of HTTP requests and the header portion of the requests for your site.

- **Origin Latency** metrics show you the average amount of time to first byte (measured in milliseconds) on a cache miss. High origin latency means that your backends are taking longer to process requests.
- **Error Ratio** metrics show you the ratio of error responses (4XX & 5XX status code errors (http://en.wikipedia.org/wiki/List_of_HTTP_status_codes)) compared to the total number of requests for your site. This metric allows you to 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.
- **HTTP Client and Server Errors** metrics shows the number of HTTP Client Errors (4XX), and Server Errors (5XX) served for your site by Fastly.

REMEMBER: Display of historical statistics are delayed between 15 to 30 minutes. and there must be a statistically measurable amount of traffic to see the graphs.

Controlling the Amount of Historical Information Displayed

You have full control over how you view the historical information.

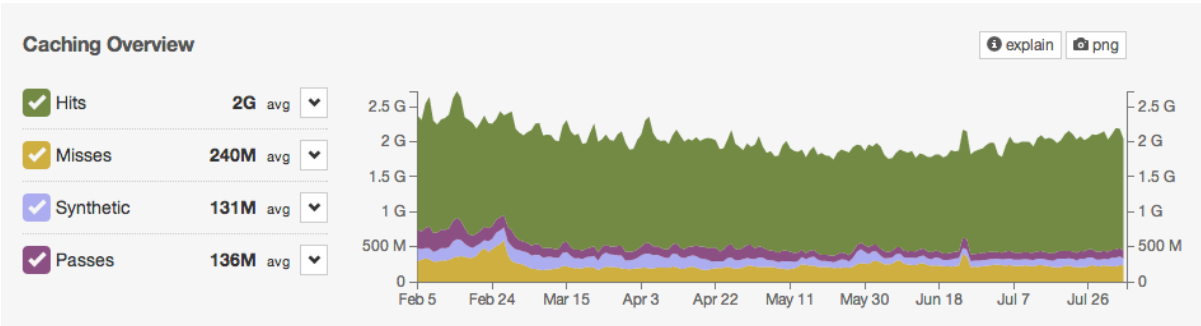


Specifically you can choose:

1. A single service or all services aggregated together
2. The exact date and time range you wish to see
3. How often to sample the data (by Day, Hour, or Minute)
4. Whether to view worldwide data or only data from a specific region (USA, Europe, Australia/New Zealand, or Asia Pacific)

About the Graph Controls

All historical information, regardless of which metric you're viewing, appears as a graph with three major sections.

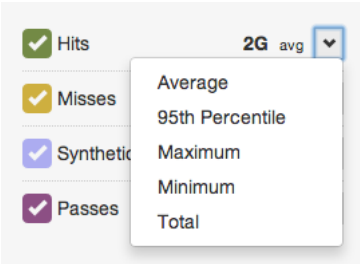


Specifically, you can interact with:

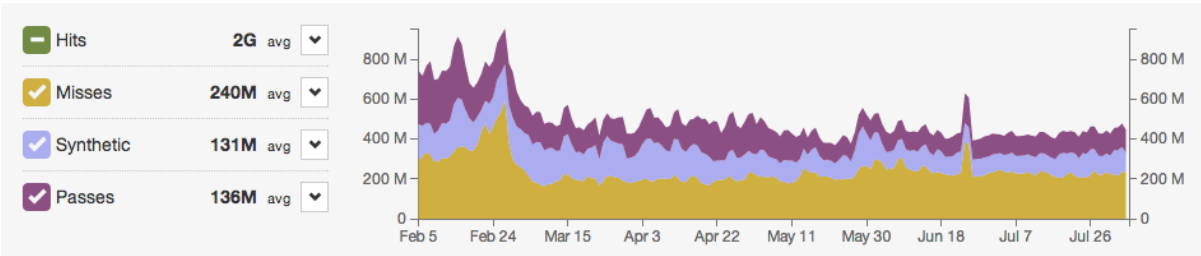
1. The Statistics Display controls
2. The time-series graph
3. The utility buttons

Use the Statistics Display Controls to view other information about the statistics displayed in each time-series graph. For example, the Caching Overview graph includes the statistics for hits, misses, synthetic responses, and passes:

Note the word “avg” and the drop down button to the right of each colored label. The drop down buttons control the statistical function used to display the values in each graph. You can choose to display each statistic independently as an average, as a 95th percentile, as a minimum, as a maximum, or their total. Simply select the drop down button next to each statistical value and set them to the desired function:

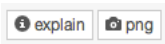


You can even exclude certain statistics entirely. For example, in this Caching Overview graph, we’ve completely hidden Hits from the display by clicking the checkmark button next to the word “Hits” in the controls.



Notice that the checkmark in the statistical controls changes to a minus sign when a statistic is hidden. The actual numbers still appear in the controls but the graph doesn’t display it.

Two utility buttons appear at the top, far right of each time-series graph:



When clicked, the explain button provides a quick explanation of the displayed graph in its default state and how to interpret the information it presents. Clicking the png button downloads the currently displayed time-series graph as a .png file, which can then be used in presentations and reports.

When the Historical Stats Tool Appears

Historical Stats controls appear automatically for logged in users with the appropriate access permission; however, the general display of historical stats are delayed between 15-30 minutes and require a statistically measurable amount of traffic to appear in the graphs at all.

· About the Streaming Controls (/guides/fastlys-web-interface/about-the-streaming-controls)

The Streaming controls allow customers to define how each instance of Fastly's streaming services will transcode and package an incoming RTMP feed and serve it to the customer's end users. If your account is enabled for video delivery (/guides/streaming/how-do-i-enable-streaming-for-my-account), see Fastly's streaming guides (/guides/streaming/) for associated services and features.

When the Streaming Controls Appear

Streaming controls are hidden by default until customers contact Fastly Sales staff at sales@fastly.com and specifically request access. Sales then arranges the activation of streaming controls by the Customer Support team, at which time they become viewable for logged in users with the appropriate permissions (/guides/customer-support/what-type-of-access-do-different-users-have).

Related Documentation

The following documents provide additional information about Fastly's streaming features:

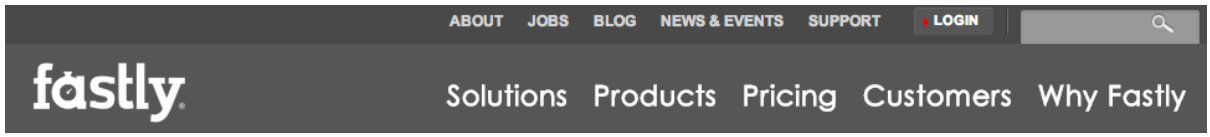
- About Fastly's Live Streaming Video Service (/guides/how-fastly-services-work/about-fastlys-live-streaming-video-service)
- About Fastly's Video Caching Service (/guides/how-fastly-services-work/about-fastlys-video-caching-service)
- About Fastly's Video on Demand On-the-Fly Packaging Service (/guides/how-fastly-services-work/about-fastlys-video-on-demand-onthefly-packaging-service)

· About the Web Interface Controls (/guides/fastlys-web-interface/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-based user interface for users with the appropriate access permissions.

Access to Fastly's User Interface Controls

Access to Fastly's user interface controls require users sign up for a Fastly account. Signup is free. Once signed up, customers access the user interface controls via the Fastly login page. The Login page (https://app.fastly.com/) can be accessed using any standard web browser or by clicking the Login button at the top right of almost all pages at Fastly's website (http://www.fastly.com).



Once logged in to a Fastly account via a web browser, the user interface controls appear based on the user's roles and permissions (/guides/customer-support/what-type-of-access-do-different-users-have). The controls are grouped by like functions and customers access each functional set by clicking on an appropriate icon at the top of the window once they are logged in to the Fastly application. Groups of controls may sometimes be referred to as "tabs" (e.g., the Analytics tab). The default control groups appear as follows:

Not all Fastly services are enabled by default. Some services (e.g., Streaming (/guides/streaming/how-do-i-enable-streaming-for-my-account)) must be specifically requested by contacting Fastly Customer Support at support@fastly.com. Once enabled, the appearance of the user interface controls will change to include these services, as shown below

Related Documentation

The following documents provide additional information about Fastly's user interface controls:

- About the Analytics Dashboard (/guides/fastlys-web-interface/about-the-analytics-dashboard)
- About the Historical Stats Tool (/guides/fastlys-web-interface/about-the-historical-stats-tool)
- About the Configuration Control Panel (/guides/fastlys-web-interface/about-the-configuration-control-panel)
- About the Streaming Controls (/guides/fastlys-web-interface/about-the-streaming-controls)
- About the Account Settings Controls (/guides/fastlys-web-interface/about-the-account-settings-controls)
- About the Billing and Invoice Controls (/guides/fastlys-web-interface/about-the-billing-and-invoice-controls)

• How Fastly Services Work (/guides/how-fastly-services-work/)

· About Fastly's Application Programming Interface (API) (/guides/how-fastly-services-work/about-fastlys-application-programming-interface-(api))

Fastly provides an application programming interface (API) that can be accessed via a number of popular interactive clients. The Fastly API allows customers to manage Fastly services via remote procedure calls instead of the web-based user interface (/guides/fastlys-web-interface/about-the-web-interface-controls). This currently includes features such as:

- Authentication (/api/#Auth)
- Configuration (/api/config)
- Historical Stats (/api/stats)
- Purging (/api/purge)
- Remote Logging (/api/logging)
- Streaming (/api/streaming)

The API features do not include customer account setup, which can only occur through the user interface controls. For examples of each API call in action, including full descriptions of the fields used and examples of requests and responses, see Fastly's API Reference (/api/).

Available API Clients

The API's main entry point is https://api.fastly.com (https://api.fastly.com). It can be accessed via the following interactive clients:

- a Perl module
- a Ruby gem
- two different Python libraries
- a Node.js client
- a Scala client

Fastly's API Client web page (/api/clients) contains links to GitHub repositories where these clients can be found. When third-party organizations have supplied these clients, we've noted so on the web page.

DISCLAIMER: Fastly makes no warranty on third-party software. We assume no responsibility for errors or omissions in the third-party software or documentation available. Using such software is done entirely at your own discretion and risk.

Authentication via the API

Nearly all API calls require requests to be authenticated. The Fastly API provides two methods for authenticating API calls: API key and username-password. Most API calls can be authenticated using an API key alone. Those calls that don't use an API key alone (usually at the account level) typically require using an authenticated session cookie instead.

Authentication Method	Description
API Key	Allows the API key (/guides/faqs/how-do-i-regenerate-my-api-key) located on a customer's account page (https://app.fastly.com/#account) to be included as a Fastly-Key header.
Username and Password	Allows a POST HTTP command to be issued with the user and password parameters, which returns a cookie that can be stashed and used in subsequent requests.

· About Fastly's Cached Content Control Features (/guides/how-fastly-services-work/about-fastlys-cached-content-control-features)

Fastly has no set hard limit on how long objects will remain cached (/guides/caching/how-long-does-content-remain-cached). Instead, Fastly supports customer-configurable Time to Live (TTL) settings and customer controlled content purging.

Time to Live Support

Fastly supports the expiration of content via customer-configurable Time to Live (TTL) settings. TTL settings work as timers on cached customer content. When content has resided in the cache for the entire TTL interval, that content is given the status of "expired." Before Fastly delivers requested content that is expired, the cache checks to see if the content is still valid by checking with the customer's application server first.

If the application server says the content remains unchanged, the cache sets the content's status to "valid" and resets its TTL value. If the object has been changed, it is declared "invalid" because the content has expired. The application server delivers updated content. Fastly CDN Service caches the updated content with the status of "valid", and its TTL timer begins to run.

The fetch and refresh process may take a second or more, and during that time, a Fastly cache may receive dozens or hundreds of end-user requests for that content. Fastly's request collapsing feature groups these requests and fulfills them at once when the application server response is received.

Fastly offers customers the option of setting a global, default TTL for cached content control. When set, Fastly's CDN service caches objects in a consistent manner even when applications are inconsistent in doing so.

Instant Purge Support

Fastly supports an Instant Purge feature that allows customers to actively invalidate content (/guides/purging/). Rather than requiring a customer's network operations and application staff to guess how frequently each bit of content may change, Fastlly allows customers to generate an HTTP Purge method that is sent to the CDN Service whenever an application changes or deletes data in its database. The Fastly CDN Service invalidates the associated content throughout the service's cache network, causing a new version of that content to be retrieved from the application server the next time it is requested.

The invalidation process takes less than 300 milliseconds, making it possible to deliver dynamic content that changes rapidly and unpredictably. Fastly's CDN Service can process thousands of changes per second, with no limit imposed on the number of purges any customer can submit. Using Instant Purge, customers can eliminate cache-to-origin HTTP traffic that all other CDN services generate to determine if expired objects are still valid.

Fastly allows URL-based and key-based purging, as well as purging of all content at once via specific, configurable purging commands (/api/purge). Fastly currently supports Ruby, Python, PHP, and Perl libraries (/api/clients) for instant purging.

Related Documentation

The following documents provide information related to Fastly's Cached Content Control features:

- How long does content remain cached? (/guides/caching/how-long-does-content-remain-cached)
- How do I mix and match Fastly VCL with custom VCL? (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl)
- How Can I Serve Stale Content on Errors (/guides/caching/how-can-i-serve-stale-content-on-errors)
- How can I override default caching based on a backend response? (/guides/caching/how-can-i-override-default-caching-based-on-a-backend-response)
- How does Fastly support the Edge-Control header? (/guides/vcl-faqs/how-does-fastly-support-the-edgecontrol-header)
- How can I purge content on Fastly? (/guides/purging/how-can-i-purge-content-on-fastly)
- How can I force Fastly to fetch a new copy of a page? (/guides/)
- How do I do wildcard purges with Fastly? (/guides/purging/how-do-i-do-wildcard-purges-with-fastly)

· About Fastly's Cloud Accelerator (/guides/how-fastly-services-work/about-fastlys-cloud-accelerator)

Fastly's Cloud Accelerator is an integration between Fastly and Google services. Specifically, the integration allows customers of Google's Cloud Platform (https://cloud.google.com/) service to directly connect via peered network interconnections (direct PNIs) to Fastly's content delivery network services.

When customers configure a Google Cloud Platform service as their origin and sign up for Fastly services (/guides/getting-started/sign-up-and-create-your-first-service), they designate a specific point of presence (POP) to serve as an Origin Shield (/guides/how-fastly-services-work/about-fastlys-origin-shielding-features) that handles cached content (/guides/how-fastly-services-work/how-fastlys-cdn-service-works) from their servers.

Requests from Fastly POPs (/guides/faqs/where-are-fastlys-pops-located) to the Cloud Accelerator Origin Shields are routed over Fastly's network, which leverages optimized TCP connection handling, quick-start and opened connections to ensure fast response times between POPs and through to the end-user. Fastly ensures that 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.

· About Fastly's Delivery Acceleration Features (/guides/how-fastly-services-work/about-fastlys-delivery-acceleration-features)

Request Collapsing

Cached content sometimes must be refreshed when that content becomes "stale" or expires. When multiple end-users request content that is in the process of being refreshed, request collapsing groups those requests to be satisfied together, as soon as the content is received. This accelerates content delivery by keeping Fastly's CDN Service from repeating duplicate requests to a customer's origin server. Request collapsing is enabled by default.

Grace Mode

When an application server becomes unavailable for any reason, end users will normally receive error messages indicating the content they've requested cannot be retrieved. When enabled, grace mode instructs Fastly's CDN Service to accelerate content delivery to end users by continuing to serve stale or expired (but likely still valid) content for a set amount of time. This allows customers to return otherwise unavailable application servers to normal operations and still serve content rather than error messages to end users. By default, grace mode is not configured. Customers must specifically configure their service to serve stale content (/guides/caching/how-can-i-serve-stale-content-on-errors).

HTTP Request Fulfillment

The Fastly CDN Service responds to HTTP GET requests initiated from end users' using a customer's website, or from a program making calls to an Internet-hosted API.

Header Support

Fastly's CDN Service supports forwarding HTTP headers to end users when they are received from a customer's origin server. Alternatively, headers can be added, removed, or modified using our edge scripting language either before or after caching a response from the origin. This includes the Cache-Control and Expires headers as well as the Surrogate-Control header. HTTP header support allows customers to send one set of instructions to the Fastly cache servers and another set of instructions to downstream caches, such as proxies or browsers. In particular, the Surrogate-Control header allows customers to specify how to forward and transform specific header types.

Related Documentation

The following documents provide information related to Fastly's Delivery Acceleration features:

- How long does content remain cached? (/guides/caching/how-long-does-content-remain-cached)
- How Can I Serve Stale Content on Errors (/guides/caching/how-can-i-serve-stale-content-on-errors)
- How can I force absolutely no caching in the browser? (/guides/caching/how-can-i-force-absolutely-no-caching-in-the-browser)
- How do I conditionally prevent pages from caching? (/guides/caching/how-do-i-conditionally-prevent-pages-from-caching)
- Adding or Modifying Headers on HTTP Requests and Responses (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses)
- Which HTTP status codes are cached by default? (/guides/caching/which-http-status-codes-are-cached-by-default)
- Maintaining Separate HTTP and HTTPS Requests to Backend Servers (/guides/backend-servers/maintaining-separate-http-and-https-requests-to-backend-servers)
- Cache Control Tutorial (/guides/tutorials/cache-control-tutorial)
- Removing headers from backend response (/guides/backend-servers/removing-headers-from-backend-response)
- Is it possible to tell Fastly to cache for a different amount of time than for end users? (/guides/caching/is-it-possible-to-tell-fastly-to-cache-for-a-different-amount-of-time-than-for-end-users)
- How can I add Surrogate-Key headers when my backend is S3? (/guides/purging/how-can-i-add-surrogatekey-headers-when-my-backend-is-s3)
- How do I set a Surrogate-Key header based on the URL? (/guides/purging/how-do-i-set-a-surrogatekey-header-based-on-the-url)
- What is the difference between PASS action set in Request Settings and Cache Settings? (/guides/caching/what-is-the-difference-between-pass-action-set-in-request-settings-and-cache-settings)

· About Fastly's Live Streaming Video Service (/guides/how-fastly-services-work/about-fastlys-live-streaming-video-service)

In addition to caching and delivering content packaged by the customer's infrastructure, Fastly also offers a Live Streaming Service that can perform this critical step. In order to provision this service, customers create instances of streaming services in the Fastly Control Panel. They begin by naming the streams and selecting various packaging settings for the stream instance they create. They then specify the resolution and bit rates or each service, called "renditions," that will ultimately best match a viewer's screen resolution and connection speed. As soon as these provisioning steps are complete, Fastly assigns a stream key for authentication and edge URLs for each service.

Customers publish their RTMP feeds to one or more of Fastly's ingest POPs (/guides/streaming/advanced-encoding-settings), identifying their streams with the stream name they selected during provisioning, and the stream key assigned through the Fastly Control Panel. Presently the following Fastly POPs are specifically designated as ingests:

Location	Publishing Address
San Jose	`rtmp://sjc.ingest.fastly-streams.com/live/`
Ashburn	`rtmp://ash.ingest.fastly-streams.com/live/`
Amsterdam	`rtmp://ams.ingest.fastly-streams.com/live/`
Tokyo	`rtmp://tyo.ingest.fastly-streams.com/live/`

Fastly's RTMP ingests support up to 5Mbps standard on the input stream and up to 10Mbps can be enabled. When publishing an RTMP stream to these servers, the stream name and key are validated before the server will accept the stream.

Fastly ingest servers transcode (if transcoding is enabled) a customer's incoming RTMP streams and package them into containers that can be viewed on different devices based on the settings that customers provide. Fastly supports the H.264 video codec and AAC or MP3 audio codecs.

Once these transcoding and packaging processes are complete, Fastly servers deliver the customer's packaged content worldwide, via Fastly's CDN Service (/guides/how-fastly-services-work/how-fastlys-cdn-service-works), which viewers can then access.

Customers make video content available to users by configuring their video players to pull the content from the provided edge URLs. The ingest POP delivers the video content (the manifests and video segments) as requested. After the first response, content requests are fulfilled from the cached content copy, effectively allowing the ingest POP to operate as the origin server for requests.

Related Documentation

The following documents provide information related to Fastly's Live Streaming Video features:

- How do I enable streaming for my account? (/guides/streaming/how-do-i-enable-streaming-for-my-account)
- How much do Fastly's streaming services cost? (/guides/streaming/how-much-do-fastlys-streaming-services-cost)
- Can I stream to multiple ingests for redundancy? (/guides/streaming/can-i-stream-to-multiple-ingests-for-redundancy)
- How do I add or remove transcoding renditions for live streams? (/guides/streaming/how-do-i-add-or-remove-transcoding-renditions-for-live-streams)
- How do I publish a live stream? (/guides/streaming/how-do-i-publish-a-live-stream)
- How is streaming bandwidth billed? (/guides/streaming/how-is-streaming-bandwidth-billed)
- How to create, edit, or delete a live stream (/guides/streaming/how-to-create-edit-or-delete-a-live-stream)
- What are wildcard streams? (/guides/streaming/what-are-wildcard-streams)
- Advanced Encoding Settings (/guides/streaming/advanced-encoding-settings)

· About Fastly's Origin Shielding Features (/guides/how-fastly-services-work/about-fastlys-origin-shielding-features)

When configuring Fastly's CDN Service during the self-provisioning process (/guides/customer-support/selfprovisioned-fastly-services), customers can choose an "origin shield" as a specific point of presence (POP) designated to host cached content from their servers. This server is referred to as a "shield" because it protects a customer's application servers from continuous requests for content.

Shield POPs

Customers can designate a specific POP to serve as a "shield" (/guides/shielding/how-do-i-enable-shielding) for their origin servers. If Fastly's caches do not have the content being requested, they fetch it from the shield server instead of the customer's origin servers. Fastly caches fetch content from a customer's origin server only when the shield server does not have the content being requested.

Load Balancing

Customers can designate multiple servers as their origin servers. When two or more application servers are provisioned as origin servers, Fastly's CDN Service will distribute fetch requests for content across those application servers using the round-robin method of distribution. This type of load balancing (/guides/load-balancing) is enabled by default; customers must explicitly disable it.

Health Checks

Customers have the option to configure Fastly's CDN Service to perform health checks (/guides/backend-servers/health-checks-tutorial) on their application servers and measure their responsiveness. Health checks are not enabled by default; the customer must specifically enable them. Customers can use health check responsiveness measurements to fine-tune the distribution of fetch requests. Request collapsing is enabled by default.

Request Collapsing

Cached content sometimes must be refreshed when that content becomes "stale" or expires. When multiple end-users request cached content that is in the process of being refreshed from origin servers, request collapsing groups those requests to be satisfied together. This protects customer application servers by keeping Fastly's CDN Service from sending duplicate requests to the origin information.

Grace Mode

When an application server becomes unavailable for any reason, end-users will normally receive error messages indicating the content they've requested cannot be retrieved. When enabled, grace mode shields application servers by instructing Fastly's CDN Service to continue to serve stale or expired (but likely still valid) content to end-users for a set amount of time. This allows customers to return otherwise unavailable application servers to normal operations and still serve content rather than error messages to end-users. By default, grace mode is not configured. Customers must specifically configure their service to serve stale content (/guides/caching/how-can-i-serve-stale-content-on-errors).

Related Documentation

The following documents provide information related to Fastly's Origin Shielding features:

- How do I enable Shielding? (/guides/shielding/how-do-i-enable-shielding)
- Caveats of Shielding (/guides/shielding/caveats-of-shielding)
- Understanding Cache HIT and MISS headers with Shielded Services (/guides/shielding/understanding-cache-hit-and-miss-headers-with-shielded-services)
- How do I configure a failover backend server? (/guides/load-balancing/how-do-i-configure-a-failover-backend-server)
- How do I implement sticky load balancing? (/guides/load-balancing/how-do-i-implement-sticky-load-balancing)
- How do I set up load balancing between my backends? (/guides/load-balancing/how-do-i-set-up-load-balancing-between-my-backends)
- How do I set up load balancing between my directors via the API? (/guides/load-balancing/how-do-i-set-up-load-balancing-between-my-directors-via-the-api)

- What does '503 Quorum weight not reached' error mean? (/guides/load-balancing/what-does-503-quorum-weight-not-reached-error-mean)
- Health Checks Tutorial (/guides/backend-servers/health-checks-tutorial)
- How long does content remain cached? (/guides/caching/how-long-does-content-remain-cached)
- How Can I Serve Stale Content on Errors (/guides/caching/how-can-i-serve-stale-content-on-errors)
- How can I force absolutely no caching in the browser? (/guides/caching/how-can-i-force-absolutely-no-caching-in-the-browser)
- How do I conditionally prevent pages from caching? (/guides/caching/how-do-i-conditionally-prevent-pages-from-caching)
- What is the difference between PASS action set in Request Settings and Cache Settings? (/guides/caching/what-is-the-difference-between-pass-action-set-in-request-settings-and-cache-settings)

· About Fastly's Real-Time Log Streaming Features (/guides/how-fastly-services-work/about-fastlys-realtime-log-streaming-features)

Fastly's log streaming features allow logging configuration information to be sent over TLS (Transport Layer Security). This means that logging information can be encrypted, which allows customers to send potentially sensitive information to log files without exposing data.

Supported Protocols and Logging Providers

Fastly supports syslog-compatible logging providers such as Splunk, Papertrail, and Logentries. We also support other methods of sending logs besides the syslog protocol. We provide support for Sumologic and allow pushing of log files to Amazon S3 buckets, including any S3-compatible providers (such as DreamHost's DreamObjects). Finally, Fastly supports FTP uploading.

How Real-Time Logging Works

As Varnish Cache software processes transactions, it writes instances to a memory log for engineers' diagnostic use. These memory instances are quickly overwritten, but selected data may be formatted into log lines and those can be continually transmitted or "streamed" in real time to a customer's chosen logging service or any other standard syslogd server for reporting purposes.

Transaction Types

The log lines reflect three types of transactions.

Transaction Type	Descriptions
Session transactions	may be multiple client transactions
Client transactions	data on how end user connections were handled
Backend transactions	data on backend connection activity

Syslog Object Setup

Customers enable log streaming (/guides/log-streaming/setting-up-remote-log-streaming) using Fastly's Control Panel or the Fastly API (/api/logging) by defining syslog objects that specify how Fastly caches should respond in various conditions.

Object Field	Description
Name	The name of the syslog object
Address	IPv4, IPv6, or hostname address of the syslogd receiving streamed log lines
Port	The port number
Format	The data fields the customer selects to receive (Apache style log formatting)

Response condition	Conditions under which to execute the syslog (if empty, always execute)

Once a customer deploys one or more syslog objects, Fastly caches will select the log lines that pass each of the syslog object's response conditions. The selected log lines will be sent in a continuous manner (streamed) to the location (address and port) specified in the syslog object, using the format specified in syslog object.

All values entered in the Fastly Control Panel (syslogd destination and data fields) are entered as arbitrary text into the Varnish memory log. Multiple endpoints can be specified. All logs receive all records (each syslog receives all records).

The format of the log line consists of two parts: a header that identifies the service and endpoint, and data fields derived from the Apache-style format string defined for each endpoint. This allows some customization of the output, but no business logic. Each line from the Varnish logs is examined to look for the response condition. Log lines from Varnish Log records that meet the response condition are sent to the syslogd server at the address and port specified in the syslog object.

Customers can configure the log format using a superset of the Apache logging format noted below.

Log Formatting Specifications

The following options are available for log formatting.

Option	Description
`-a`	When writing to a file, append to it rather than overwrite it.
`-b`	Include log entries that result from communication with a backend server. If neither `-b` nor `-c` is specified, varnishlog acts as if they both were.
`-C`	Ignore case when matching regular expressions.
`-c`	Include log entries that result from communication with a client. If neither `-b` nor `-c` is specified, varnishlog acts as if they both were.
`-D`	Daemonize.
`-d`	Process old log entries on startup. Normally, varnishlog will only process entries that are written to the log after it starts.
`-Iregex`	Include log entries that match the specified regular expression. If neither `-I` nor `-i` is specified, all log entries are included.
`itag`	Include log entries with the specified tag (see below). If neither `-I` nor `-i` is specified, all log entries are included.
`knum`	Only show the first num log records.
`-n`	Specifies the name of the varnishd instance to get logs from. If `-n` is not specified, the host name is used.
`-o`	Group the log entries by request ID. This has no effect when writing to a file using the `-w` option.
`-Pfile`	Write the process's PID to the specified file.
`-rfile`	Read log entries from file instead of shared memory.
`-snum`	Skip the first num log records.
`-u`	Do not buffer the output.
`-V`	Display the version number and exit.
`-wfile`	Write log entries to file instead of displaying them. The file will be overwritten unless the -a option was specified. If varnishlog receives a SIGHUP while writing to a file, it will reopen the file, allowing the old one to be rotated away.
`-Xregex`	Exclude log entries that match the specified regular expression.
`-xtag`	Exclude log entries with the specified tag (see below).

Log Entry Tags

The following log entry tags are defined.

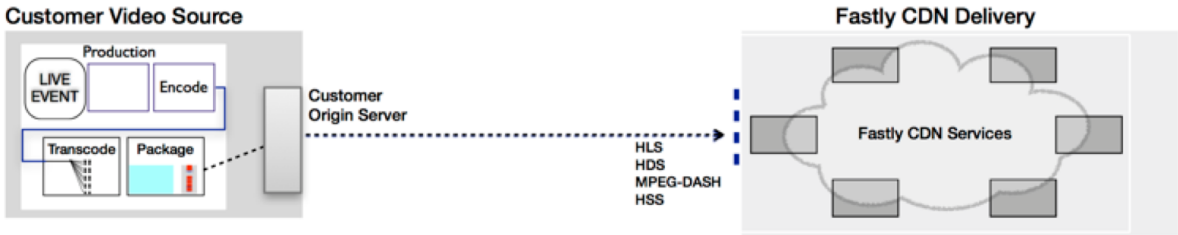
Tag	Tag
Backend	ReqStart
BackendClose	RxHeader
BackendOpen	RxLostHeader
BackendReuse	RxProtocol
BackendXID	RxRequest
CLI	RxResponse
ClientAddr	RxStatus
Debug	RxURL
Error	SessionClose
ExpBan	SessionOpen
ExpKill	StatAddr
ExpPick	StatSess
Hit	TTL
HitPass	TxHeader
HttpError	TxLostHeader
HttpGarbage	TxProtocol
Length	TxRequest
ObjHeader	TxResponse
ObjLostHeader	TxStatus
ObjProtocol	TxURL
ObjRequest	VCL_acl
ObjResponse	VCL_call
ObjStatus	VCL_return
ObjURL	VCL_trace
ReqEnd	WorkThread

Related Documentation

The following documents provide information related to Fastly's Real-Time Log Streaming features:

- Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming)
- How do I escape characters in logs? (/guides/vcl-faqs/how-do-i-escape-characters-in-logs)

· About Fastly's Video Caching Service (/guides/how-fastly-services-work/about-fastlys-video-caching-service)



For customers with their own video packaging infrastructure, Fastly can act as a globally distributed HTTP cache to improve quality of service and increase viewer capacity. When a manifest or video segment is requested by a customer's player, a Fastly edge or shield POP will pull the requested content from the customer's origin media server. Subsequent requests for that content will be served from Fastly's cache servers instead of the customer's origin (read *How Fastly's CDN Services Work* (/guides/how-fastly-services-work/how-fastlys-cdn-service-works) for more information).

Fastly can cache and deliver any HTTP based media streaming protocol including:

- HTTP Live Streaming (HLS),
- HTTP Dynamic Streaming (HDS),
- HTTP Smooth Streaming (HSS), and
- Dynamic Adaptive Streaming over HTTP (MPEG-DASH).

In addition, both live and on-demand streams are supported.

Related Documentation

The following documents provide information related to Fastly's Video Caching Service features:

- How do I enable streaming for my account? (/guides/streaming/how-do-i-enable-streaming-for-my-account)
- How much do Fastly's streaming services cost? (/guides/streaming/how-much-do-fastlys-streaming-services-cost)
- Advanced Encoding Settings (/guides/streaming/advanced-encoding-settings)

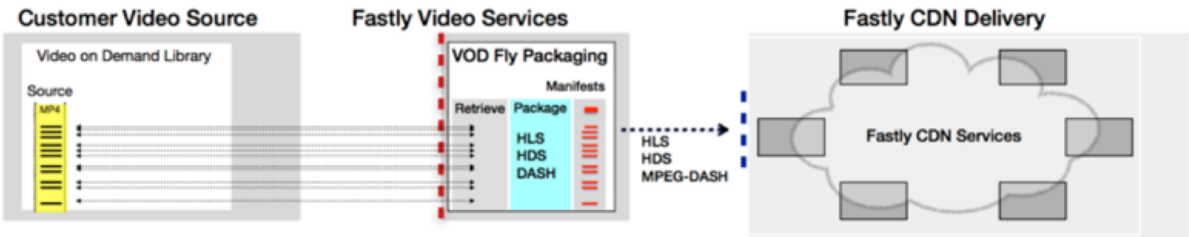
· About Fastly's Video on Demand On-the-Fly Packaging Service (/guides/how-fastly-services-work/about-fastlys-video-on-demand-onthefly-packaging-service)

Fastly offers "on the fly," dynamic, video content packaging service. Rather than having to pre-package all profiles of every video in all streaming protocols, customers simply package content in real time. Video content then becomes immediately available to viewers.

For example, say a user starts viewing a 60-minute video and then immediately activates the seek-forward controls to move forward in the video to the 5-minute mark. The user then watches two more minutes of the video (to the 7-minute mark) and stops.

Fastly's VOD on-the-fly packaging service only packages video segments that an end user requests for viewing. In the above example, the service would package only the 2 minutes of viewed content. The service would not package the 58 minutes of video that was not watched and viewers would not have to wait for packaging in order to start viewing. Seek-forward and seek-backward controls would work immediately.

How Fastly's VOD On-the-Fly Packaging Service Works



Customers enable the VOD on-the-fly packaging service through headers defined on backend requests, which are set up when the service is created through Fastly's Control Panel user interface. The packaging process starts when customers receive HTTP requests from viewers' player software. Through the browser, the Customer website gives the viewer's video player the host server address, and the name of the VOD file with its extension denoting the packaging format desired.

The ingest POP (/guides/streaming/advanced-encoding-settings) that the customer designates then puts the data in the HTTP request it sends to Fastly's VOD on-the-fly packaging service (via the shield POP). Fastly processes these requests for video segments. Specifically, the service parses the data to get the VOD host server address and the name of the MP4 source file requested.

The service then retrieves the requested segments in MP4 format from a customer's libraries on their VOD host server. The VOD host server can be any server that supports HTTP byte range requests, or a storage service such as Amazon S3, or Google Cloud Storage. Once retrieved, the service muxes (http://www.techopedia.com/definition/24124/multiplexer-mux) the segment into HLS or HDS format and then sends it with an HTTP response to the shield POP. The shield POP, in turn, sends it to the edge POP, which sends it to the video player of the viewer who made the original request.

Packaged segments are cached, and subsequent requests are fulfilled from cache to avoid repeatedly packaging the same requests. Fastly caches the packaged video chunks while the customer maintains only the original MP4 source files in their VOD library. Customers do not have to store a duplicate VOD library with every packaging option for every MP4 source file they control.

· Content Sources Supported by Fastly (/guides/how-fastly-services-work/content-sources-supported-by-fastly)

Fastly caches deliver various types of content (/guides/how-fastly-services-work/types-of-content-delivered-by-fastly) 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 (http://en.wikipedia.org/wiki/Representational_state_transfer) 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.

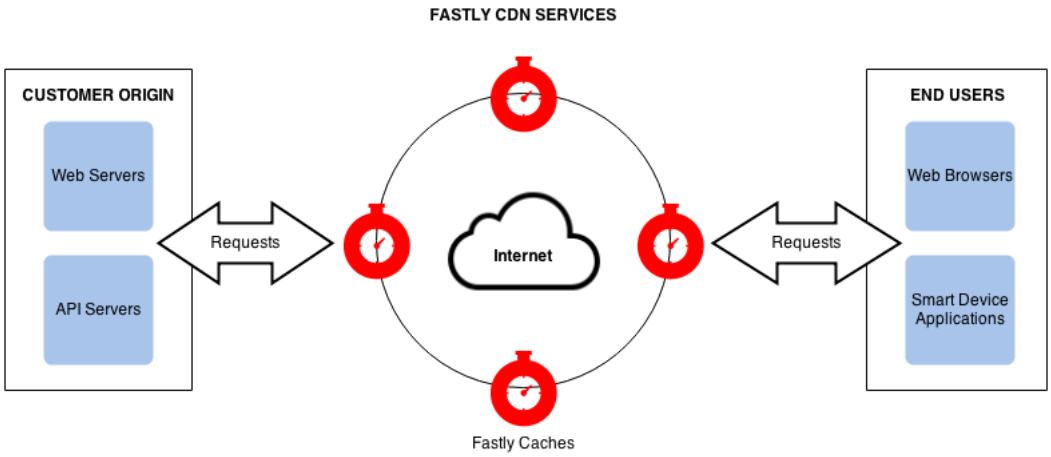
“Pre-packaging” in this context refers to the process of receiving the video contents, converting or “transcoding” the stream into segments (chunks) for presentation at a specific dimension and transmission rate, and then packaging it so a video player can identify and request the segments of the live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

If your account is enabled for video delivery, see Fastly’s streaming guides (/guides/streaming/) for associated services and features. To request video delivery on your account, please contact your Fastly Account Representative at sales@fastly.com (mailto:sales@fastly.com).

· How Fastly's CDN Service Works (/guides/how-fastly-services-work/how-fastlys-cdn-service-works)

Fastly is a content delivery network (http://en.wikipedia.org/wiki/Content_delivery_network) (CDN). We serve as an Internet intermediary and offer the Fastly CDN Service to make our customers’ transmission of their content to their end users more efficient.

Our customers make content available through their websites and their Internet-accessible (hosted) application programming interfaces (APIs). A customer can create content (customer-generated content), as can a customer’s 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.”

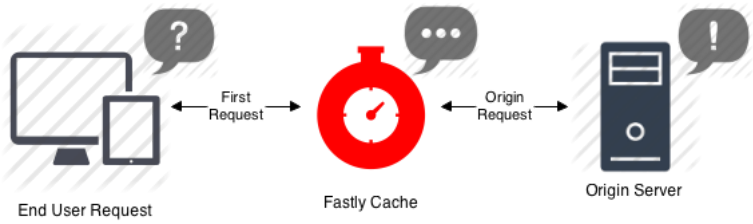


Fastly's delivers its CDN service from key access points to the Internet called "points of presence" (POPs). Fastly places POPs (http://www.fastly.com/blog/how-fastly-chooses-pop-locations/) where their connectivity to the Internet reduces network transit time when delivering content to end-users. Each POP has redundant Fastly cache servers. When end user's request a customer's content objects, Fastly delivers them from whichever of the cache locations can deliver the objects fastest.

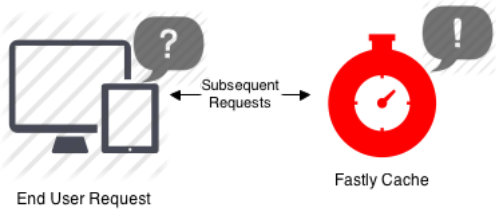
Fastly's caches only receive and process customers' end user requests for content objects. Customers decide which objects will be cached, for how long, who can access them, whether they are to be encrypted when transmitted over the Internet, what level of encryption shall be used, and when the objects will be deleted from the caching service. Customers 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 (/guides/getting-started/sign-up-and-create-your-first-service), customers must identify which of their application servers will provide the original content objects for each of their various domains (e.g., company.com, myco.com). Their application servers can be physical servers in a customer 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 have that object, the cache will distribute the processing load for the fetch across all of them (based on the configuration criteria set by the customer). 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.



· Intro to Caching and CDNs (/guides/how-fastly-services-work/intro-to-caching-and-cdns)

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 - ours especially! - are optimized to serve small files *very very* quickly. CDNs typically have caches placed in datacenters 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 (either 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 quite 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 or even more efficient handling of your traffic.

Remember: 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 config 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.

If none of the following headers are found in the request, Fastly will respect the `Expires` header value.

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 store this content. Ever. At all. Please.
- `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 (/guides/tutorials/cache-control-tutorial).

(Note: For more information than you need on the rest of the Cache-Control headers, see the relevant section in Mark Nottingham's Caching Tutorial (http://www.mnot.net/cache_docs/#CACHE-CONTROL).)

Surrogate Headers

`Surrogate` headers are a relatively new addition to the cache management vocabulary (described in this W3C tech note (http://www.w3.org/TR/edge-arch)). 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 (/api/purge) 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 (/guides/tutorials/cache-control-tutorial). 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 application (/guides/shielding/) 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 (http://en.wikipedia.org/wiki/Reverse_proxy)
- Fastly's Cache-Control docs (/guides/tutorials/cache-control-tutorial)
- Mark Nottingham's Caching Tutorial (http://www.mnot.net/cache_docs/#CACHE-CONTROL)
- Surrogate header W3C tech note (http://www.w3.org/TR/edge-arch)

· Types of Content Delivered by Fastly (/guides/how-fastly-services-work/types-of-content-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. For more information about the content sources Fastly supports, see *Content Sources Supported by Fastly* (/guides/how-fastly-services-work/content-sources-supported-by-fastly).

Static Content

Static content includes content that remains relatively unchanged. Fastly can control static content (/guides/caching/) 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 (/guides/purging/) that invalidates the content.

Classic examples of static content include images and logos.

Dynamic Content

Dynamic content includes content that changes at unpredictable intervals. We serve 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 (/guides/purging/) 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. If your account is enabled for video delivery, see Fastly's streaming guides (/guides/streaming/) for associated services and features.

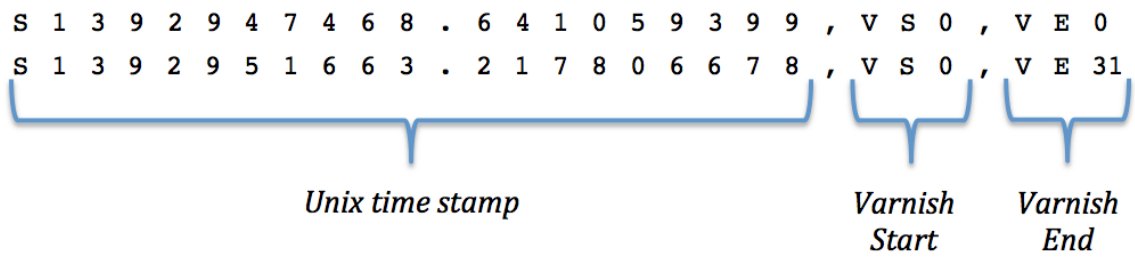
· Understanding the X-Timer header (/guides/how-fastly-services-work/understanding-the-xtimer-header)

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 an X-Timer headers:

- `S1392947468.641059399,VS0,VE0` (a cache HIT)
- `S1392951663.217806578,VS0,VE31` (a cache MISS)

Lets 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 (http://en.wikipedia.org/wiki/Unix_time) 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.

• Customer Support (/guides/customer-support/)

· Availability SLA (/guides/customer-support/availability-sla)

Support availability and response times vary depending on the type of account you have and the level of support you have purchased.

	Unpaid Account	**Month-to-Month Account**	**Termed Contract**	**Premium Support**
Service Level Agreement	None	None	Termination Option	Invoice Credits + Termination Option

Definitions

"Degraded Performance" for the Services means the Services are experiencing Error Conditions that are (1) caused by issues under Fastly Control, (2) observable or reproducible by Subscriber or Fastly, (3) requiring Subscriber to redirect traffic off the Services. Degraded Performance does not include any reduction on availability of the Application User Interface or API due to planned maintenance.

"Error Condition" means the Services are (1) not responding to end user requests, (2) incorrectly sending end users error condition messages or (3) sending incorrect partial content to end users and these conditions are observable or reproducible by Subscriber or Fastly.

"Fastly Control" means elements entirely under Fastly's control and not a consequence of (a) Subscriber hardware or software failures, (b) Subscriber or end user connectivity issues, (c) Subscriber operator errors, (d) Subscriber traffic amounts that exceed Subscriber's Permitted Utilization, (e) corrupted Subscriber content, (f) acts of god (any) or war, or earthquakes, or terrorist actions.

Termination

Any Subscriber that has a contract with a term and a minimum commitment shall have thirty (30) days to terminate their subscription agreement if the Services experience Degraded Performance (a) for longer than 7.2 hours in any one month, or (b) for longer than 43.8 minutes each month in any three contiguous months. Subscriber shall have thirty (30) days to terminate their contract following the third failure

Availability Invoice Credits

Subscribers who purchase Premium Support shall be entitled to Invoice Credits according to the following table.

Availability Percent	Period of Degraded Performance	Monthly Credit Percent
Below 100% - 99.99%	Up to 4.32 minutes	1%
99.99% – 99.9%	Up to 43.8 minutes	5%
99.89% – 99.0%	Up to 7.2 hours	10%

Invoice Credits for unavailability will accrue on a monthly basis. The Credit Amount for a month is equal to the monthly usage charge multiplied by Monthly Credit Percent.

Credit Terms

- Requests for Invoice Credits for Degraded Performance must be made within 30 days of the period of Degraded Performance.
- The maximum amount of any credit is the Invoice Amount for the month the Degraded Performance occurred.
- A pending credit does not release Subscriber from its obligation to pay Fastly's submitted invoices in full when due.
- Credits will be applied to the Invoice two months following the month an invoice credit was incurred.

· Compliance Services (/guides/customer-support/compliance-services)

Subscribers who purchase Compliance Services will have access to the following:

- Access to library of third-party audit reports and certification attestations (most recent 12 months),
- Access to executive summary reports for penetration tests and network scans (most recent 12 months),
- Access to library of security related policies and procedures,
- Access to library of executive summaries of annual risk assessments (most recent 12 months),
- Unlimited Audit rights at $500 per hour of Fastly time, with 10 hours included per year. Audit to be on 10 business days' advance notice.
- Access to library of historical reason for outage (RFO) documents (most recent 12 months).
- Ability to be added as an Additional Insured on Fastly's General Commercial Liability Insurance for a one-time fee of $1,000.

Subscribers who wish to purchase Compliance Services must also purchase Premium Support.

· Help! I'm locked out of my account! (/guides/customer-support/help!-im-locked-out-of-my-account!)

Why is my account locked?

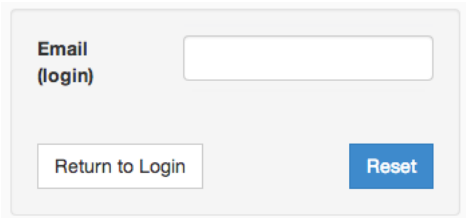
For security reasons, Fastly limits the number of times someone can try logging in to an account. We don't want to give people unlimited attempts at guessing your password, so we stop them from trying after a limited number of failed attempts to sign in.

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:

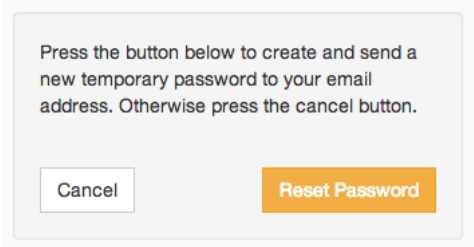
1. Point any standard web browser (/guides/getting-started/browser-recommendations-when-using-the-fastly-application) to the Fastly application login page (https://app.fastly.com/#login). The login controls appear.

2. Click the **Forgot Password** button underneath the login controls. The Reset Password Controls appear.



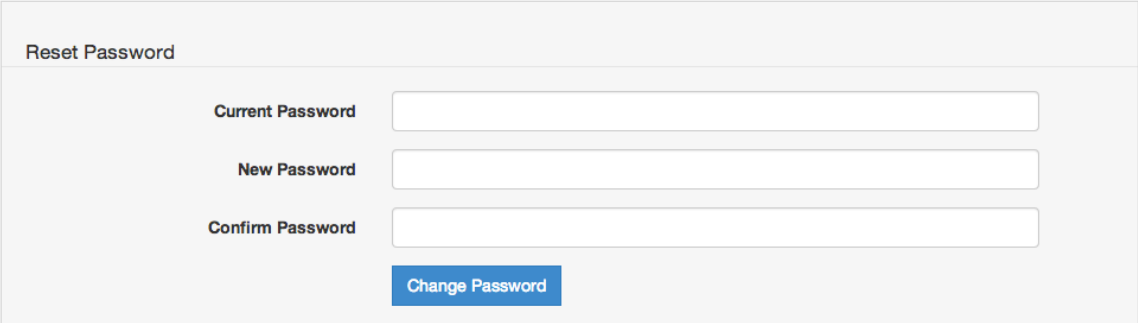
3. In the **Email (login)** field, type the email address you normally use to log in to your Fastly account, and then click the **Reset** button. Password reset instructions will be emailed to you.

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 send 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.
- 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. 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 list of codes to avoid running out before you recover your mobile device.

If you don’t believe you will be able to recover a 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 superuser role contact Customer Support at support@fastly.com (mailto: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.

Please contact Customer Support at support@fastly.com (mailto: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 in order 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, please follow the instructions in the email.

If you think your account has been hacked or phished, please contact Customer Support at support@fastly.com (mailto: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 optional IP whitelisting, 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 whitelist 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 whitelisting is enabled and you forget to update your whitelist configuration, you will be locked out of your account. You will need to contact support@fastly.com (mailto: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.

· How can I merge accounts? (/guides/customer-support/how-can-i-merge-accounts)

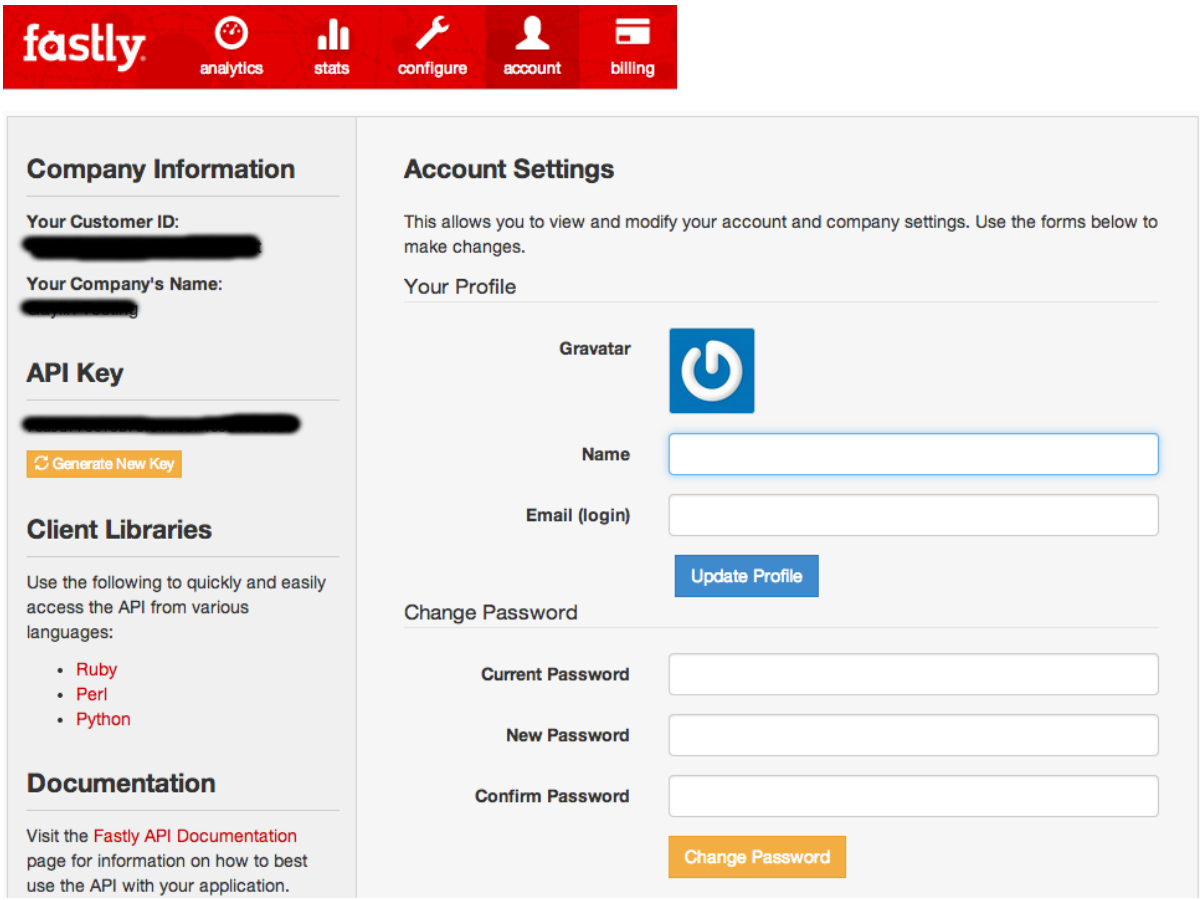
If several employees at your company independently create testing accounts when learning about how Fastly's services work (/guides/how-fastly-services-work/), you can have those testing accounts merged into a single account by emailing the Customer Support team at support@fastly.com (mailto:support@fastly.com) with the following information:

- the Customer IDs (/guides/faqs/where-can-i-find-my-service-id-customer-id-or-api-key) of the accounts to be merged
- which account should be considered the primary account (any other accounts will be merged into the primary)

After you contact us, we'll reach out to verify the ownership of each account. If we can confirm ownership, we'll initiate a merge.

· How do I change the email or password associated with my account? (/guides/customer-support/how-do-i-change-the-email-or-password-associated-with-my-account)

Log in to the Fastly application and click the **account** tab to access the Account Settings information.



Changing Your Name or Email

To change the name or email associated with the account, update the information that appears in the **Name** and **Email (login)** fields in the **Your Profile** section. Then click **Update Profile** to save the changes.

Changing Your Password

To change the password currently associated with your account, type your current password in the **Current Password** field, then type your new password twice, once in the **New Password** field and once in the **Confirm Password** field. Save the changes by clicking **Change Password**.

When choosing a password remember the following:

- Passwords must contain at least six letters.
- Passwords cannot solely contain sequences of letters or numbers (e.g., `12345678`, `abcdefg`).
- Passwords cannot solely contain repeated characters (e.g., `222222`, `aaaaaa`).
- Passwords cannot solely contain adjacent key placements on a standard keyboard (e.g., `QWERTY`).
- Passwords cannot match commonly used or easily guessed words (e.g., `password123`, `changeme`).
- Passwords cannot match your user name.

· How do I delete a service version or an entire service? (/guides/customer-support/how-do-i-delete-a-service-version-or-an-entire-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 a historic log of the changes that were made to a service. To undo changes to a particular service, you can always go back to a previous version and activate that one. You also can clone a new service version based on any old version.

IMPORTANT: When a service is deleted from your configuration, Fastly maintains a copy in its databases in case the deletion was accidental (/guides/customer-support/i-accidentally-deleted-a-service.-how-do-i-get-it-back). Any domains that were associated with that service will remain associated with it even after deletion. Before deleting a service, you should delete the domains within it first and then delete the service itself.

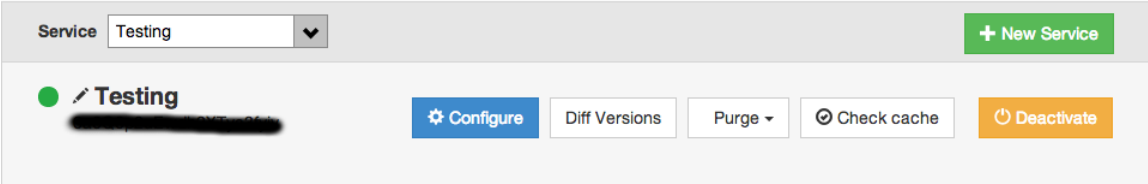
To delete any service along with all of its versions, follow the steps below.

1. Log in to the Fastly application.
2. Click the **configure** tab to access the control panel.

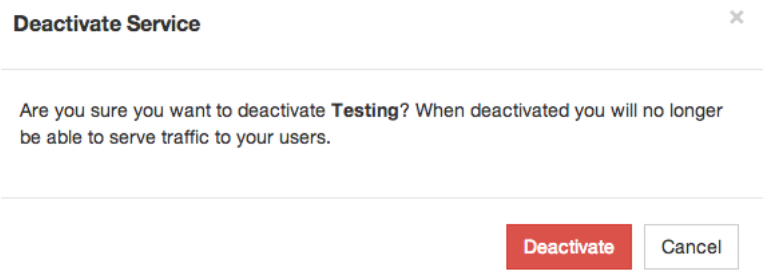
3. Select the appropriate service from the **Service** drop down.
4. In the **Domains** area, click the gear icon to the right of each domain associated with this service and then select **Delete** from the menu that appears.

   IMPORTANT: Be sure to repeat this step until all domains associated with the service have been deleted. If any domains remain associated with a deleted service, the system may not allow you to associate them with a new service at a later date.

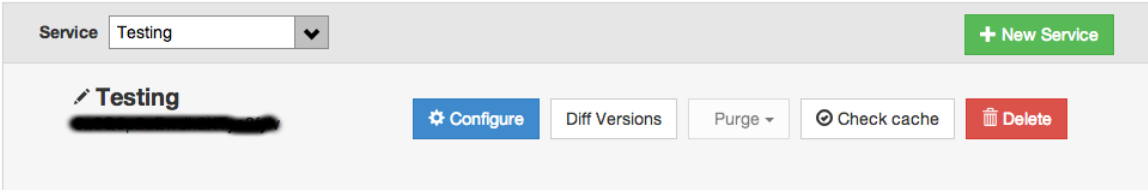
5. Click the yellow **Deactivate** button to the right of the service name.



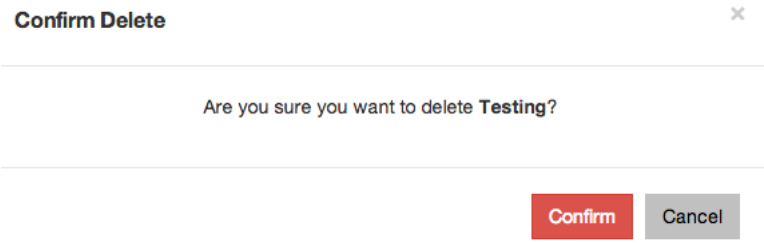
The **Deactivate Service** warning appears.



6. Click the red **Deactivate** button to confirm you want to deactivate your service and acknowledge that you no longer want to serve traffic with it. The **Delete** button becomes active on the control panel.



7. Click the red **Delete** button to the right of the service name. The **Confirm Delete** window appears.



8. Click the red **Confirm** button to confirm that you want to delete the service.

REMEMBER: You can also deactivate an entire service (/guides/faqs/how-do-i-deactivate-and-reactivate-services) instead of deleting it.

· How to Submit a Support Request (/guides/customer-support/how-to-submit-a-support-request)

Fastly offers technical support for all accounts. The method that customers use to submit requests and the response time by Fastly to those requests depends on the level of technical support they purchase for their accounts.

For all levels of technical support, Fastly's customer support ticketing system automatically generates and assigns each request a unique service ticket number, which is then sent within minutes to the customer at the email address used when submitting the ticket. This automated response also contains a direct link to the service ticket within Fastly's customer support system. Fastly then responds to these requests as appropriate for the level of support purchased by the customer.

How Accounts Without Premium Support Submit Requests

Customers who do not pay for Premium Support support submit support requests either via email to support@fastly.com or directly in the Fastly application interface (via the Support tab at the right side of every page in the interface) any time they require assistance with self-provisioned tasks (/guides/customer-support/selfprovisioned-fastly-services). Fastly then responds to these requests as appropriate for the level of support purchased by the customer.

How Accounts with Premium Support Submit Requests

To submit emergency support tickets, customers must pay for a Premium Support plan that includes support for severe incidents. Customers with Premium Support plans can submit requests for support any time

- a service that has been working stops working.
- a production deployment experiences a drastic change in performance that either is customer- or end-user-affecting.

Customers with Premium Support accounts submit emergency tickets to a uniquely assigned, emergency support email addresses (instead of via the Support tab on the Fastly application interface or via the standard customer support email address).

· I accidentally deleted a service. How do I get it back? (/guides/customer-support/i-accidentally-deleted-a-service.-how-do-i-get-it-back)

Services can be deactivated (/guides/faqs/how-do-i-deactivate-and-reactivate-services) or deleted (/guides/customer-support/how-do-i-delete-a-service-version-or-an-entire-service). Deactivated services can be reactivated (/guides/faqs/how-do-i-deactivate-and-reactivate-services) at any time, but once they've been deleted you must contact Customer Support (mailto:support@fastly.com) to have them restored. When sending your request, please remember to include:

- your customer ID (/guides/faqs/where-can-i-find-my-service-id-customer-id-or-api-key)
- your company name (/guides/faqs/where-can-i-find-my-service-id-customer-id-or-api-key)
- your service ID (/guides/faqs/where-can-i-find-my-service-id-customer-id-or-api-key) (the name of the service you want restored)

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

· Self-Provisioned Fastly Services (/guides/customer-support/selfprovisioned-fastly-services)

Fastly customers configure or “provision” Fastly caching and video services personally, independent of Fastly staff, via the Fastly application interface (https://app.fastly.com). Fastly calls this “self-provisioning.” Self-provisioning tasks include things like:

- creating and deploying services
- adding domains and origin servers
- configuring load balancing
- modifying how services handle HTTP headers
- submitting purge commands

Once provisioned, Fastly services can be deployed immediately. If self-provisioned tasks fail to operate in an appropriate or expected manner, customers can find answers to a variety of frequently asked questions in Fastly’s guides and tutorials (/guides/). They can also receive personalized assistance by submitting requests (/guides/customer-support/how-to-submit-a-support-request) directly to Fastly’s Customer Support Engineering staff.

· Support Description and SLA (/guides/customer-support/support-description-and-sla)

Support availability and response times vary depending on the type of account you have and the level of support you have purchased. The following table summarizes those offerings:

	Unpaid Account	**Month-to-Month Account**	**Termed Contract**	**Premium Support**
Online Forums	Yes	Yes	Yes	Yes
Email Support Response Time Commitment	Best Effort	Best Effort	Next Business Day	Severity 1 Incidents: 15 minutes* All Others: Next Business Day
Severe Incident Response Email Address	No	No	No	Yes
Support SLA	None	None	Termination Option	Invoice Credits + Termination Option

Technical Support

The following section applies to all subscribers.

Definitions

- **"Business Hours"** are 8AM-6PM Monday through Friday Pacific Time.
- **"Business Days"** are Monday through Friday excluding US and UK national and banking holidays.
- An **"Incident"** is an occurrence during which end users' use Subscriber's services is adversely impacted.
- A **"Severity 1 Incident"** is an incident resulting in a major service outage requiring Subscriber to redirect all traffic from Fastly to another CDN.
- A **"Severity 2 Incident"** is an incident resulting in minor or intermittent outage not requiring Subscriber to redirect traffic to another CDN.
- "Fastly Control" means elements entirely under Fastly's control and not a consequence of (a) Subscriber hardware or software failures, (b) Subscriber or end user connectivity issues, (c) Subscriber operator errors, (d) Subscriber traffic amounts that exceed Subscriber's Permitted Utilization as defined in the Terms and Conditions, (e) corrupted Subscriber content, (f) acts of god (any) or war, or earthquakes, or terrorist actions.

Subscriber Responsibilities

Subscriber is responsible using and configuring services according to the Documentation available at https://docs.fastly.com.

Support Requests

Subscribers submit support requests by sending email to support@fastly.com, or by selecting the Control Panel "SUPPORT" tab. Subscribers receive a system-generated response within minutes containing the ticket number and a direct link to the ticket.

Incident reports should include at the least the following:

- Services are not responding to end user requests.
- Services incorrectly send end users error condition messages.
- Services send incorrect or partial content to end-users.

Incident reports should include all relevant information such as:

- Subscriber's determination of the Severity Level of the incident,
- Subscriber hardware failures,
- Subscriber operator errors,
- Services configuration errors made by Subscriber employees,
- Potential Excess Utilization (as defined in the Terms of Use or master services agreement)
- Corrupted Subscriber content,
- DDOS attacks, and
- Relevant force majeur acts such as extreme weather, earthquakes, strikes or terrorist actions.

Communications

Communications between Fastly support engineers and subscriber staff are conducted using the ticketing application, which maintains a time-stamped transcript of all communications, and sends emails to Subscriber and Fastly staff as tickets are updated.

Response Time

Fastly shall use best efforts to respond in a timely fashion.

Termed Contracts

The following applies to any subscriber that has a contract with a term and a minimum commitment.

Response Times

Fastly commits to acknowledging receipt of a support ticket within the next business day following submission of a support request.

Termination

In any three-month period where three (3) or more support Response Time objectives are not met and the failure to meet the objectives materially adversely impacted Subscriber, Subscriber shall have thirty (30) days to terminate their subscription agreement following the third failure.

Premium Support

The following applies to subscribers who have purchased Premium Support.

Incident Reporting

Severity 1 Incidents: Fastly will provide Subscriber an Incident Support Email address for Subscriber to report Incidents. Subscriber should report Incidents promptly using the Incident Support email.

Severity 2 Incidents: Subscriber should report Severity 2 Incidents by submitting a Support Request.

Response Times

Fastly will respond to the report of an Incident by troubleshooting the cause(s) of the Incident and resolve them if caused by factors within Fastly’s control, or provide information to those who can resolve the factors if the factors are within others’ control, as follows:

For a Severity 1 Incident:

- Fastly support staff will acknowledge receipt of the email within 15 minutes.

- Fastly will start actively troubleshooting within 30 minutes of receipt of the email.
- Fastly will perform its tasks on a 24 / 7 basis.
- Fastly and Subscriber will immediately communicate upon learning new information that may be useful in troubleshooting the incident, and status updates between Fastly and Subscriber staff will take place no less frequently than every 30 minutes for the first two hours, and no less frequently than every hour thereafter.
- Fastly staff will work until (a) the incident is resolved or (b) the incident is believed to be outside of Fastly's control.

For a Severity 2 Incident:

- During Business Hours Fastly support staff will acknowledge receipt of the email within two hours or within two hours of the start of the next business day if the Incident does not come in during a Business Day.
- Fastly engineers will be actively troubleshooting within one business day, will work on the Incident during Business Hours, and will provide status updates to Subscriber daily on each subsequent Business Day.

Charges for Incident Response

For Severity 1 Incidents caused by factors within Subscriber's control, a flat fee of $1500 will be assessed, and any time spent beyond three hours will be invoiced at Subscriber's undiscounted Professional Services rates.

For Severity 2 Incidents caused by factors within Subscriber's control, Subscriber will be invoiced at Subscriber's undiscounted Professional Services Rates.

For all incidents,

- If the Incident-causing factors are within Fastly's control, there will be no hourly charges for Fastly engineering staff time.
- If the factors are within Subscriber's control, Subscriber agrees to pay Fastly its hourly charges for Fastly engineering staff time. If it appears likely the factors are within Subscriber's, Subscriber may tell Fastly staff to stop working on troubleshooting the Incident (thereby stopping the hourly charges from being incurred). Subscriber agrees to tell Fastly to stop working on an Incident via an email sent to Fastly's Incident Support email address. The timestamp on the email will be the time charges cease to be incurred.

Support Invoice Credits

In the event a Severity 1 Incident occurs, Subscriber has purchased Premium Support, the cause of the Incident is within Fastly's control, and any of the communication or response timeframes are materially not met, a one-time credit of $500 per incident will be credited to Subscriber's account.

Credit Terms:

- Requests for Invoice Credits must be made within 30 days of the incident which triggered the service credit.
- In no event shall Invoice Credits exceed the invoice value of the month in which they are accrued.
- A pending credit does not release Subscriber from its obligation to pay Fastly's submitted invoices in full when due.
- Credits will be applied to the Invoice two months following the month an invoice credit was incurred.

· What type of access do different users have? (/guides/customer-support/what-type-of-access-do-different-users-have)

Your Fastly account can be managed by multiple users. These users can have minimal, partial, or full access to your account based on the role you've assigned to them. This role can be changed by **superusers** at any time via the web interface or via the API (/api).

User Roles

Fastly allows you to assign one of four different roles to each user who has access to your account:

- **user**: Minimal access. They can view analytics (the pretty graphs) and change personal profile details (such as name and

email),

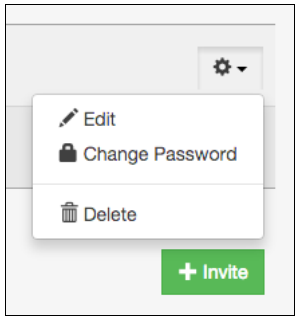
- **billing**: Partial access. They can view billing information, pay invoices, view and change the pricing plan, plus perform all **user** actions.
- **engineer**: Partial access. They can create and configure services, plus perform all **user** actions.
- **superuser**: Full access. They can manage users (invite, create, edit, or delete), view and change organization settings (name, account owner, billing primary, and price plan), plus perform all **engineer**, **billing**, and **user** actions.

By default, the system assigns the role of **superuser** to the person who signs up for your organization's account.

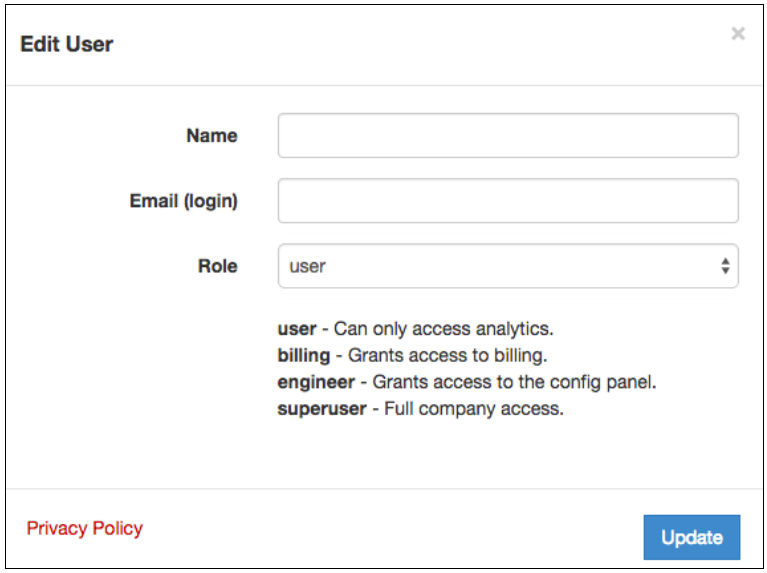
Changing Roles for Existing Users

Clicking on the **account** icon at the top of the interface takes you to the account management section.

Scrolling down the page to the **User** section will show all the users associated with the account. A gear icon with an attached drop menu appears for each user to the right of their name.



Selecting **Edit** will open an **Edit User** window for a selected user.



To edit the user's role, simply select the new role from the **Role** drop down menu and click the **Update** button.

· Why am I receiving a service or domain limit message? How do I fix it? (/guides/customer-support/why-am-i-receiving-a-service-or-domain-limit-message-how-do-i-fix-it)

We currently limit the maximum number of services and domains (/guides/domains-and-dns/) you can configure with the Fastly application. 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 numb
er of domains: 20"}OK
```

If you're receiving a limit message and need to create more services or domains, please contact support@fastly.com (mailto:support@fastly.com) so we can help you plan as you ramp up. We can probably increase the limit for you but there may be an easier way to achieve your goals with a smaller number of services or domains.

• Legal & Compliance (/guides/legal-and-compliance/)

· Legal and Compliance FAQ (/guides/legal-and-compliance/legal-and-compliance-faq)

This FAQ is designed to answer questions we often get about our Terms of Service, our Documentation, our Acceptable Use Policy, and how they all work together. It does not replace any of those, nor is it legally binding on its own. If you have questions or would like to suggest an update, check out our support page (http://www.fastly.com/support/).

Why does Fastly have a Privacy Policy, and how does it affect my business?

The Privacy Policy (http://www.fastly.com/privacy/) applies to information we collect about visitors to our website and the information we collect from interactions with our subscribers. It also summarizes our commitments to the U.S. – E.U. Safe Harbor framework (http://www.export.gov/safeharbor/eu/eg_main_018365.asp) and the U.S. - Swiss Safe Harbor framework (http://export.gov/safeharbor/swiss/) as set forth by the U.S. Department of Commerce.

Note that Fastly’s Privacy Policy (http://www.fastly.com/privacy/) doesn’t apply to subscribers’ use of Fastly’s caching services. It also doesn’t apply to the way our subscribers' end users access websites or applications that are configured to use Fastly — these commitments are covered by the Terms of Service (http://www.fastly.com/terms/) and our documentation (/).

What data do you collect from Fastly services and how do you use it?

As you’ll find with any Internet company, when Fastly provides services to our subscribers, we receive some data about their websites and applications. This includes information on how our subscribers’ end users interact with Fastly services. We collect data that is operationally necessary to deliver our services, and retain it in anonymized and aggregated forms. Details about how we use subscriber data can be found in our documentation (/guides/legal-and-compliance/security-and-technology-compliance).

I think my business has some specific needs and requirements. What should I do?

Please reach out to your sales representative if you think you have needs or requirements that aren’t addressed by our terms, data privacy, documentation, or this FAQ. Feel free to email sales@fastly.com (mailto:sales@fastly.com) to be connected with a sales team member for assistance.

What does “Acceptable Use” mean?

Fastly is an intermediary between our subscribers and their users. This means we have to put a few rules in place to ensure that we’re not held accountable for illegal or harmful things that subscribers (or their users) do while using Fastly services. We expect subscribers and their end users to respect the law while using Fastly. You can find more information in our Acceptable Use Policy (http://www.fastly.com/acceptable-use/), which hosts DMCA information and process.

Why do subscribers need to give you an Abuse Contact and post a DMCA or takedown policy?

In addition to being an intermediary between our subscribers and their end users, we often appear to be the hosting provider for subscriber content. If someone has a complaint about content being cached on our network, we want to enable the parties involved to resolve their issues as quickly as possible. The fastest way to do that is for Fastly to send a subscriber contact email and link to a subscriber's DMCA policy to the person who alerted us to the potential violation of our Acceptable Use Policy.

What are non-Fastly services and what am I giving Fastly permission to do?

Non-Fastly services are services that our subscribers elect to use outside of Fastly. Examples include your origin, a DNS provider, a CMS like Wordpress, or something similar. In many cases, Fastly is actually configured to work with these third-party services. The Non-Fastly Services provision gives us permission to work with these services, makes it clear that subscribers are directly responsible for their relationships with third-party service providers, and lets subscribers know that we are not responsible if something goes wrong with a service that Fastly does not provide.

What are "Beta Services"?

Fastly often offers subscribers early access to services or features before they are publicly launched. This helps us collect feedback and improve our products, and ensures that we're working on the things that matter most for our subscribers. Because beta services are offered before they're ready for the general public, we can't make any promises about how they will perform or whether they will eventually be released — but we do promise to be very clear about what the beta entails when we send out invitations.

What does "feedback" mean?

At Fastly, we take pride in our customer support and how effectively we are able to respond to subscriber requests. We also want to build an awesome CDN service that continues to evolve, and we are very open to suggestions for features and improvements. Our Feedback provision ensures that if a subscriber gives us a suggestion about how to improve our service, we can integrate it efficiently.

How/why is Fastly documentation integrated into the agreement?

We believe in transparent communication. That's why we built out a public documentation portal that describes how our services work, our security features, and the certifications we've achieved. Fastly documentation is available to anyone (including all subscribers and prospective subscribers) at http://docs.fastly.com. We contractually stand behind all of our documentation; anyone can sign up for a free account and confirm that what they see on http://docs.fastly.com matches our services.

What if Fastly documentation changes over time?

We work hard to improve and evolve our services, and we'll update the documentation as that happens. From time to time, we may release new features or services that are subject to additional terms and conditions — those updates will be posted on our documentation pages. Anyone can subscribe to an RSS feed of those changes here (/changes/rss).

Where are your Service Level Agreements (SLAs) and support commitments?

Our Availability SLA (/guides/customer-support/availability-sla) describes our service level commitments for our network. The type of SLA a subscriber receives depends on the type of contract and level of support service the subscriber has purchased from us. Our Support Description and SLA (/guides/customer-support/support-description-and-sla) describes the hours of

availability and response times for all our levels of customer support. The support offering a subscriber receives depends on the level of service a subscriber has purchased from us.

Which version of the Terms of Service, Acceptable Use Policy, and Privacy Policy apply to me?

We updated our Terms of Service on December 8, 2014 and we had previously updated them on November 5, 2013. The Terms of Service applicable to you are determined by the date you agreed to them, and stay in effect unless you agree to a new version, which means that updates to our terms of service don't affect you until you take significant action with your account. You will be asked to agree to our Terms of Service when you first create an account (usually a trial account). If after that, you switch to a paid month-to-month account, or sign a service order with us, you will be asked to agree to the terms then in effect. If you are a trial subscriber prior to an update to the terms, you won't have to accept new terms of service until you convert to a paid account. Some subscribers have other arrangements with us, and updates to the terms of service on our website do not affect the terms of their arrangement with us.

We believe in transparency and want to make sure you have the opportunity to see the changes we have made over time. The historical versions of our Terms of Service are available here (/guides/legal-and-compliance/tos-archive). Updates to our Acceptable Use Policy are effective for everyone upon posting, but existing subscribers will have thirty days before they become effective for them. Updates to our Privacy Policy are effective thirty (30) days after being posted.

What are utilization spikes?

Our infrastructure is well-equipped to handle large variations in traffic. In the event that a subscriber has an exceptionally large but brief spike in traffic, Fastly may incur additional costs that we need to pass on to our subscribers. Our Utilization Spike clause (http://www.fastly.com/terms/) allows us to charge extra in the rare event that a subscriber's peak bandwidth utilization (measured in megabits per second on a 95th percentile basis) is more than five times their average traffic for that month.

· Security and Technology Compliance (/guides/legal-and-compliance/security-and-technology-compliance)

Information about the processes and safeguards we have to protect your data is organized by category:

- Business Continuity and Operational Resilience
- Cache Data and End User Information Management
- Datacenter and Physical Security
- Encryption and Key Management
- Human Resources Security
- Identity and Access Management
- Incident Management
- Network and Infrastructure Security
- System and Configuration Change Management

Business Continuity & Operational Resilience

Service failover. If any of our points of presence (POPs (/guides/faqs/where-are-fastlys-pops-located)) experience issues serving content, we can redirect traffic seamlessly to a neighboring POP until those issues are resolved.

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.

Environmental safeguards. We compensate for environmental disruptions at our datacenters with systems that control backup power, temperature and humidity, and fire suppression.

Cache Data and End User Information Management

Cache Data. Subscribers manage which content is cached, where, and for how long by setting policies that control that content. Please see our introduction to caching (/guides/how-fastly-services-work/intro-to-caching-and-cdns) for more information.

Logging. Subscribers may stream syslog activity (/guides/streaming/) to a remote endpoint for analysis and use. Fastly does not retain subscriber syslog activity.

End User IP Addresses. Fastly retains anonymized and aggregated end user IP address information on a very limited basis to provide and improve its services. This includes creating internal statistics that combine the anonymized and aggregated IP addresses with similar data from other Fastly subscribers. Complete end user 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. Fastly may also retain indefinitely end user IP addresses associated with malicious activity that threatens the network or our subscribers.

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.

Encryption and Key Management

Secure data transmission. The Fastly service supports SSL configurations (/guides/ssl/) 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 procedures and technology to secure private keys throughout their lifecycle.

Key storage and access security. We store private keys in encrypted repositories. Access to key storage is restricted to personnel who support key management processes.

Human Resources Security

Employee background screening. We screen new employees as part of the hiring process. Screening activities are dependent on applicable local regulations and may include procedures such as criminal background checks and reference checks.

Confidentiality agreement. Our employees 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 on an annual basis.

Identity and Access Management

User management lifecycle. Our personnel receive user access appropriate for their job responsibilities. That access is updated or removed as an employee’s job function or employment status changes.

User access review. We periodically inspect access privileges to our systems and data to make sure that access is appropriate for all personnel.

Multi-factor authentication. We enforce multi-factor authentication to better control access to our technology environment.

Incident Management

Incident monitoring. We monitor multiple internal and external reporting channels to detect service-related issues. Personnel are available 24x7x365 to confirm and respond to incidents.

Incident response plan. We maintain an incident response plan with established roles and responsibilities, communication protocols, and standard response procedures.

Communication and reporting. We update impacted customers using various communication methods (such as status.fastly.com (http://status.fastly.com/)), depending on an incident's scope and severity.

Network and Infrastructure Security

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.

Network security reviews. We use proactive network scans and third-party penetration testing to confirm and enhance the security of our network.

Vulnerability management. We monitor various public and private distribution lists, as well as other communication channels, for up-to-date security vulnerability notices. We validate and implement critical security patches within one day of discovery.

System and Configuration Change Management

Change management process. We have defined a set of procedures that we follow consistently when we develop and deploy technology changes. We use these procedures each time we modify the software, configurations, and devices that support the Fastly service.

Testing and approval. We review, test, and approve technology changes at various stages of the development process before deploying them fully into the Fastly service.

Change and configuration monitoring. We have multiple logging, monitoring, and alert mechanisms in place to enhance the visibility of changes and ensure adherence to our change management process.

· Terms of Service Archive (/guides/legal-and-compliance/tos-archive)

Fastly updated its Terms of Service (http://www.fastly.com/terms/) on the past dates listed below.

- Fastly Terms of Service: November 5th, 2013 (/tos_archive/2013-11-05-tos)

Basic Configuration

• Caching (/guides/caching/)

· Can Fastly handle tracking cookies when caching content? (/guides/caching/can-fastly-handle-tracking-cookies-when-caching-content)

Tracking cookies can be difficult to manage because they are inherently personal and user specific. Fastly can handle tracking cookies when caching site content through the use of custom VCL (/guides/vcl-faqs/where-can-i-upload-custom-vcl). You can read more about this in Fastly's blog post on _How to Cache with Tracking Cookies (http://www.fastly.com/blog/how-to-cache-with-tracking-cookies/). _

Keep in mind that the ability to upload custom VCL code (/guides/vcl-faqs/where-can-i-upload-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

· Can I use a POST request body as a cache key? (/guides/caching/can-i-use-a-post-request-body-as-a-cache-key)

As long as the body of a POST request is less than 2K in size and the content type is `application/x-www-form-urlencoded`, then we allow you to use it as part of the cache key. Your VCL (/guides/vcl-faqs/) should look something like:

```
sub vcl_hash {
  set req.hash += req.url;
  set req.hash += req.http.host;
  if (req.request == "POST" && req.postbody) {
    set req.hash += req.postbody;
  }
  return (hash);
}
```

You'll also need to force a cache lookup, but only for requests that can be cached, by doing something like this in **vcl_recv**:

```
  if (req.request == "POST" && req.postbody ~ "(^|&)action=list(&|$)") {
    return (lookup);
  }
```

TIP: To refine this, you could add only the important parts of `req.postbody` to the hash using `regsub()`.

· Creating Error Pages with Custom Responses (/guides/caching/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. This tutorial shows you how to set up your service configuration to serve a custom page or a synthetic response when we receive an error code from your backend.

We assume you are already accustomed to editing and deploying configurations using the web-based configuration application (https://app.fastly.com/). If you are not familiar with basic editing using the app, please see our Getting Started (/guides/getting-started/) information to learn more before you continue.

Overview

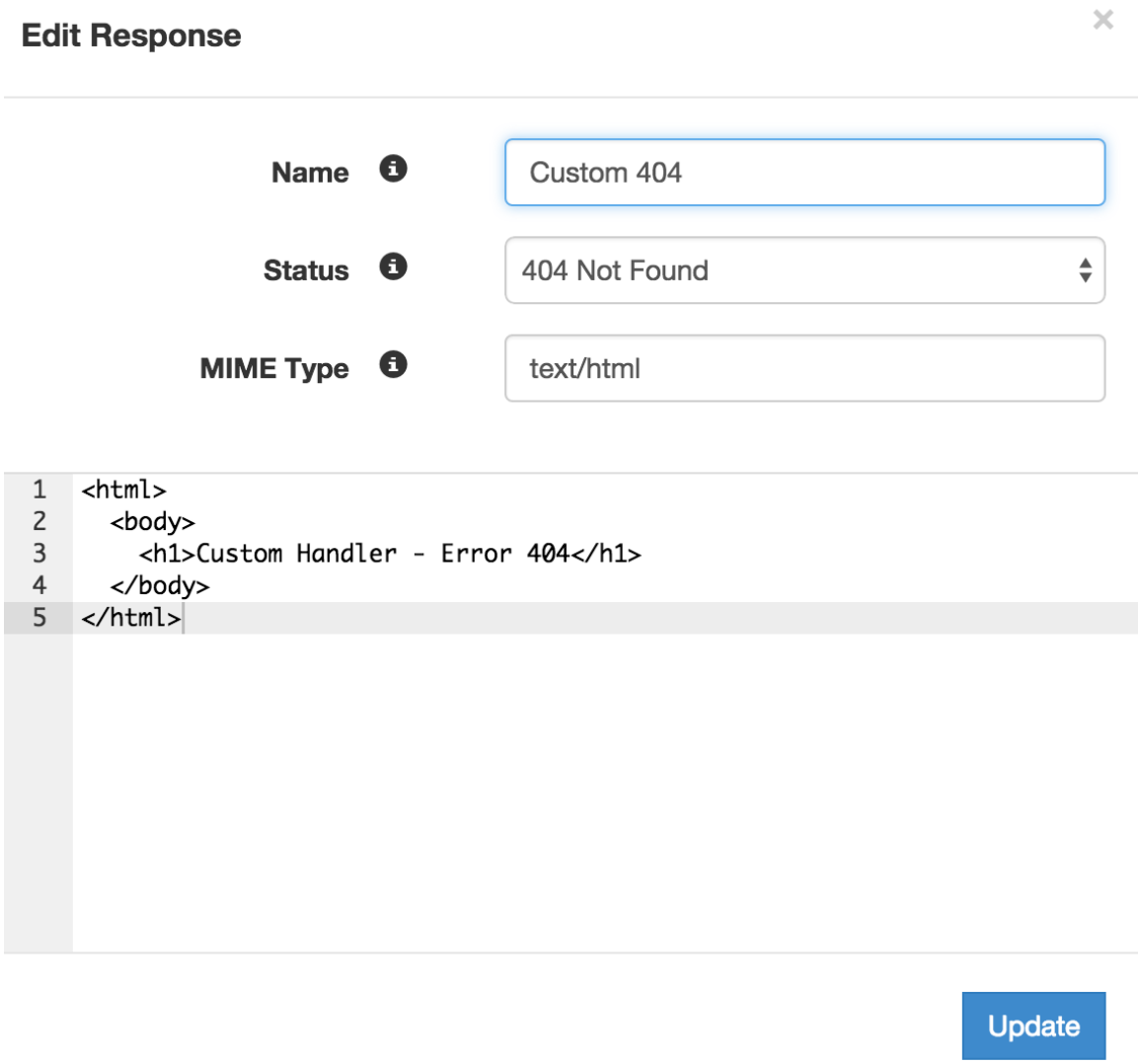
In this tutorial, we will create a response object that contains the HTML you want to serve for your error page. We provide example HTML, but you can use any HTML you see fit. The response object will require that you use a condition in order for it to be served. If you are not familiar with conditions or responses we suggest you read the following:

- Responses Tutorial (/guides/tutorials/responses-tutorial)
- Conditions Tutorial (/guides/tutorials/conditions-tutorial)

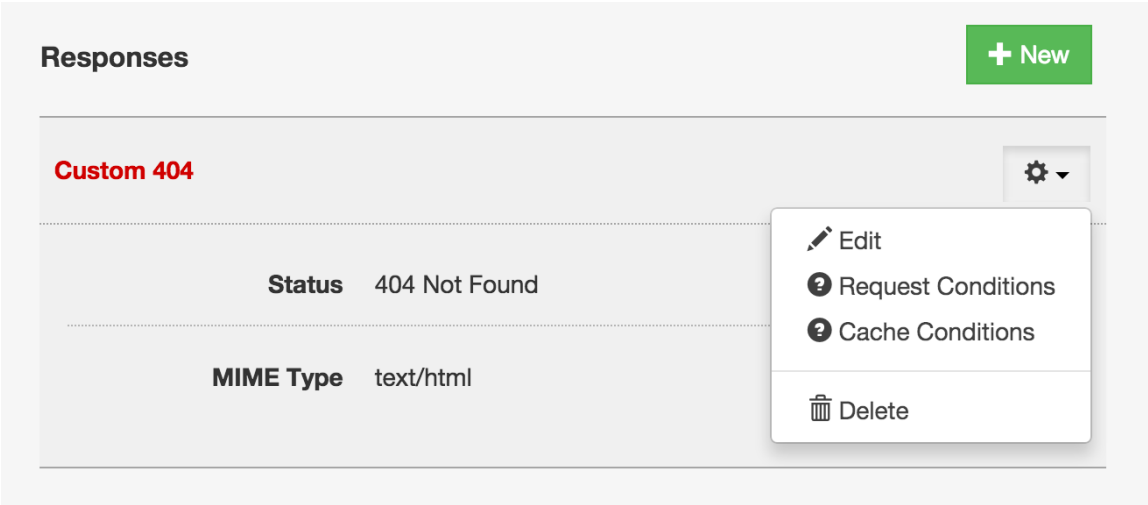
Creating the Custom Response

1. Log in to the Fastly application and click the **configure** tab (wrench icon).

2. From the **Service** drop down menu, select the appropriate service and then click the blue **Configure** button. The main controls for your selected service appear.
3. Click **Content** from the section list on the left.
4. In the **Responses** area at the bottom of the page click the **New** button. The New Response window appears.



5. Fill out the **New Response** controls as follows:
   - In the **Name** field, give the response you're creating a name (e.g., Custom 404).
   - From the **Status** drop down 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).
6. Click the **Create** button to create your custom response. Your new response appears in the list of responses.
7. Click the gear icon to the right of the name of your new response and select **Cache Conditions** from the drop down menu that appears.



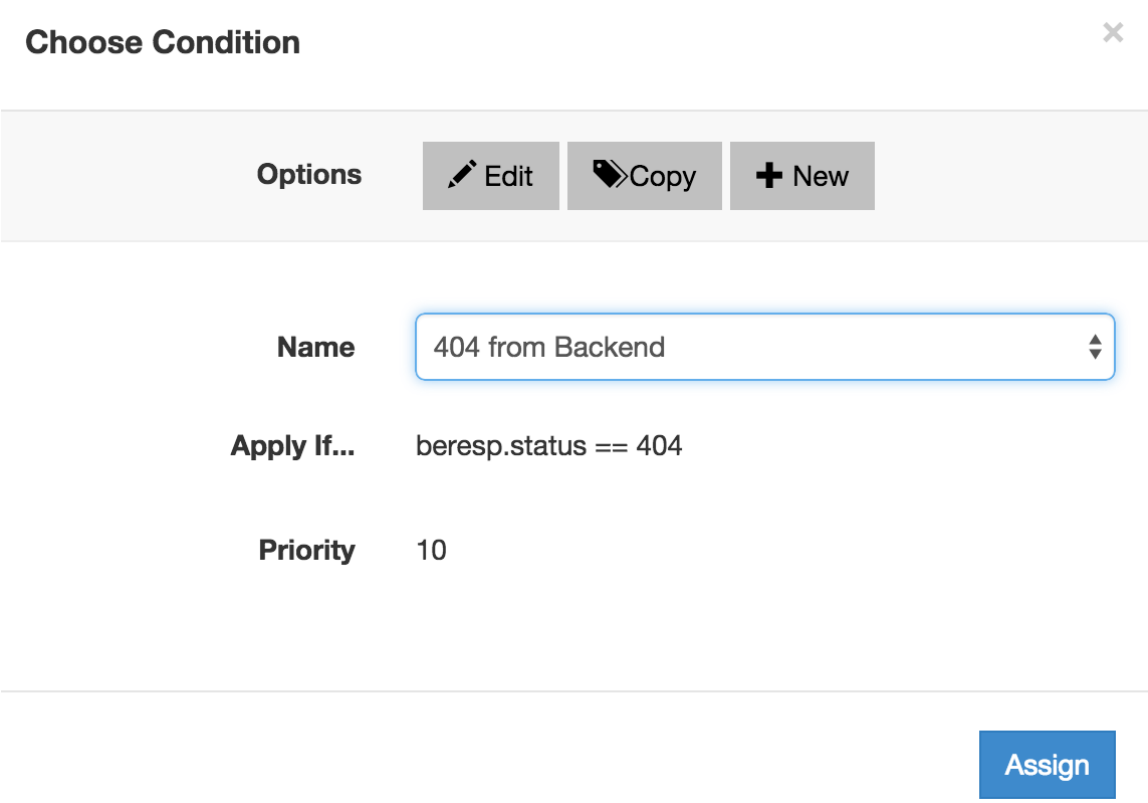
The **New Condition** window appears (if you have previously created a cache condition, click the new button at the top of the edit condition modal that appears).

8. In the **Name** field, give the condition you're creating a name (e.g., 404 Not Found).

9. In the **Apply If** field, type the condition under which the new response occurs. The condition should take the following format:

```
beresp.status == ###
```

where ### equals the status condition you're creating the response for. For instance, using our 404 error example above, your **New Condition** window would look like this:



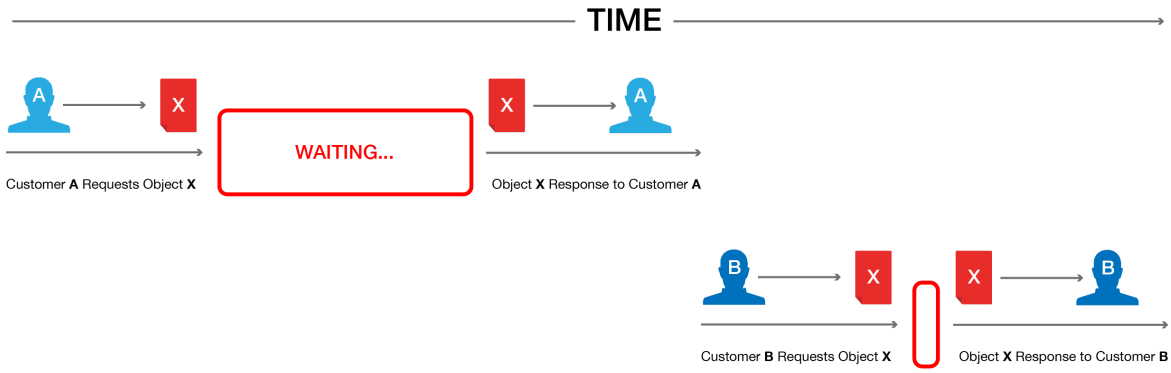
Using the value of `beresp.status == 404` in the **Apply If** field here tells Fastly to use this response object whenever origin servers return a 404 status. (Remember: The Conditions Tutorial (/guides/tutorials/conditions-tutorial) has more detailed information on conditions.)

10. Click the **Create** button and the condition will be created and applied to the custom response object you made earlier.
11. Deploy your service. Fastly will now serve your custom HTML error page when required.

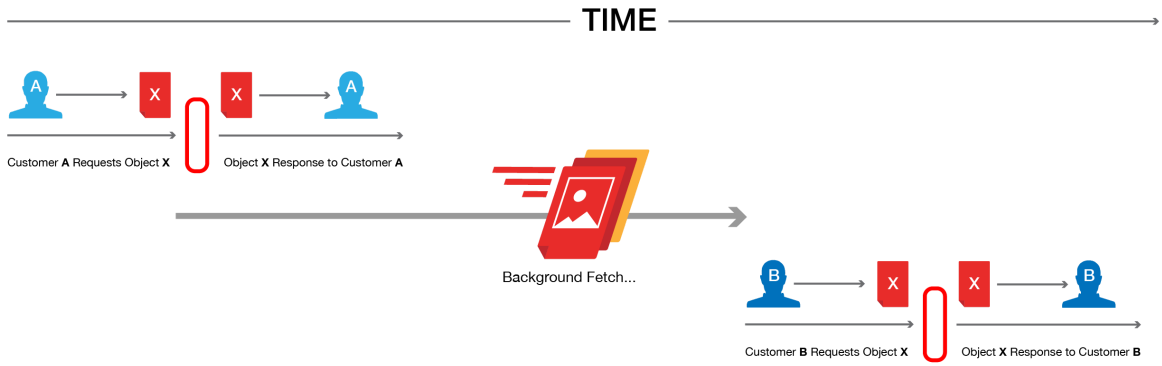
· Does Fastly allow serving old content while fetching new content? (/guides/caching/does-fastly-allow-serving-old-content-while-fetching-new-content)

This help document is a part of the basic setup process for serving "stale" or expired content to users (/guides/caching/how-can-i-serve-stale-content-on-errors).

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 but 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://tools.ietf.org/html/rfc5861) “HTTP Cache-Control Extensions for Stale Content” by Mark Nottingham, which is under consideration for inclusion in Google’s Chrome browser (https://www.mnot.net/blog/2014/06/01/chrome_and_stale-while-revalidate).

Usage

To activate this behavior simply add a `stale-while-revalidate` or `stale-if-error` statement 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-while-revalidate` or `stale-if-error` statements in any Cache-Control or Surrogate-Control response headers.

Additional reading

- RFC 5861 (http://tools.ietf.org/html/rfc5861)
- Mark Nottingham's blog post (https://www.mnot.net/blog/2014/06/01/chrome_and_stale-while-revalidate)
- Chrome discussion (https://groups.google.com/a/chromium.org/forum/#!msg/chromium-dev/zchogDvIYrY/ZqWSdt3LJdMJ)

· Does the gzip widget exclude old browsers automatically? (/guides/caching/does-the-gzip-widget-exclude-old-browsers-automatically)

Some web browsers should not receive gzipped content. Fastly's default gzip configuration intentionally excludes old browsers. Specifically, we run the following VCL before any custom VCL:

```
if (req.http.Accept-Encoding) {
  if (req.http.User-Agent ~ "MSIE 6") {
    unset req.http.Accept-Encoding;
  } elsif (req.http.Accept-Encoding ~ "gzip") {
    set req.http.Accept-Encoding = "gzip";
  } elsif (req.http.Accept-Encoding ~ "deflate") {
    set req.http.Accept-Encoding = "deflate";
  } else {
    unset req.http.Accept-Encoding;
  }
}
```

For more information about gzipping, see our article on how to enable automatic gzipping (/guides/caching/how-do-i-enable-automatic-gzipping) via the Fastly widget.

· How are request settings applied? (/guides/caching/how-are-request-settings-applied)

Requests settings are applied based on the Action you select in the Request Settings window. You can choose any one of the following settings:

- **N/A** - Applies the request settings within the policy and continues through to other request configurations.
- **Lookup** - Immediately searches the cache for content. If none is found (a miss), then the request is sent to the origin. Fastly executes this setting prior to other request configurations.
- **Pass** - Sends the request to the origin each time. Fastly executes this setting immediately and will ignore additional request configurations. See "What is the difference between PASS action set in Request Settings and Cache Settings?" (/guides/caching/what-is-the-difference-between-pass-action-set-in-request-settings-and-cache-settings) to learn more.

· How can I force absolutely no caching in the browser? (/guides/caching/how-can-i-force-absolutely-no-caching-in-the-browser)

Theoretically, all browsers should follow some fairly simple rules from the HTTP standard. In practice, however, various browsers (such as Internet Explorer 8) don't strictly follow the rules. The following combination of headers seems to force absolutely no caching with every browser we've tested.

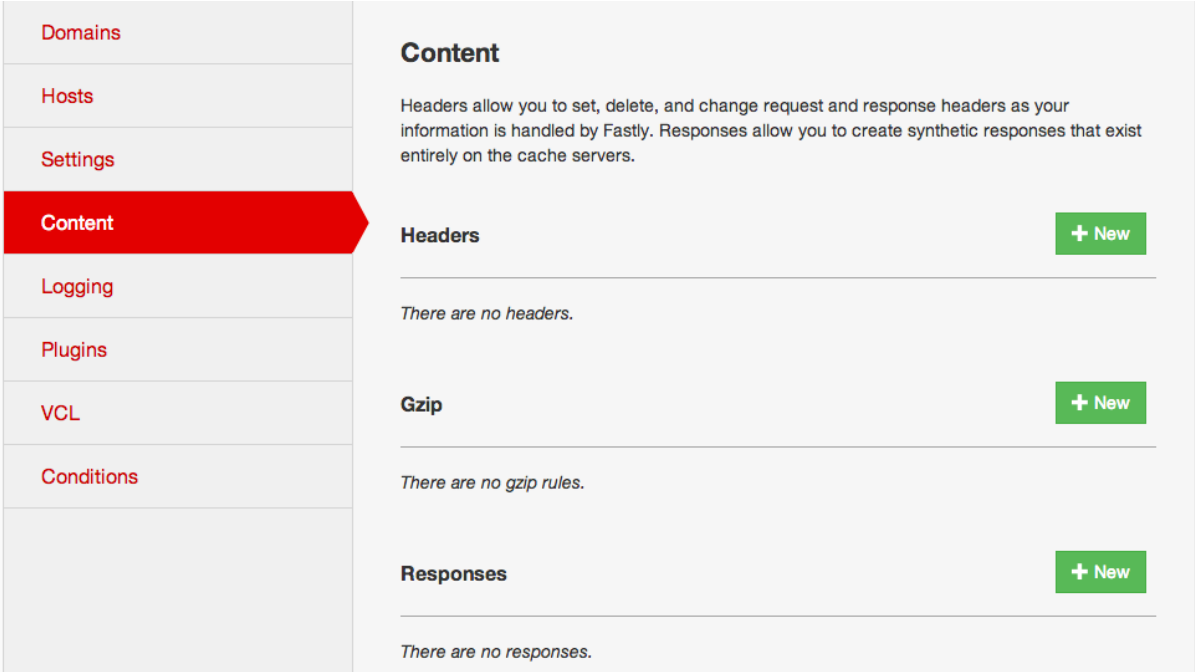
```
Cache-Control: no-cache, no-store, private, must-revalidate, max-age=0, max-stale=0, post-check=0, pre-check=0
Pragma: no-cache
Expires: 0
```

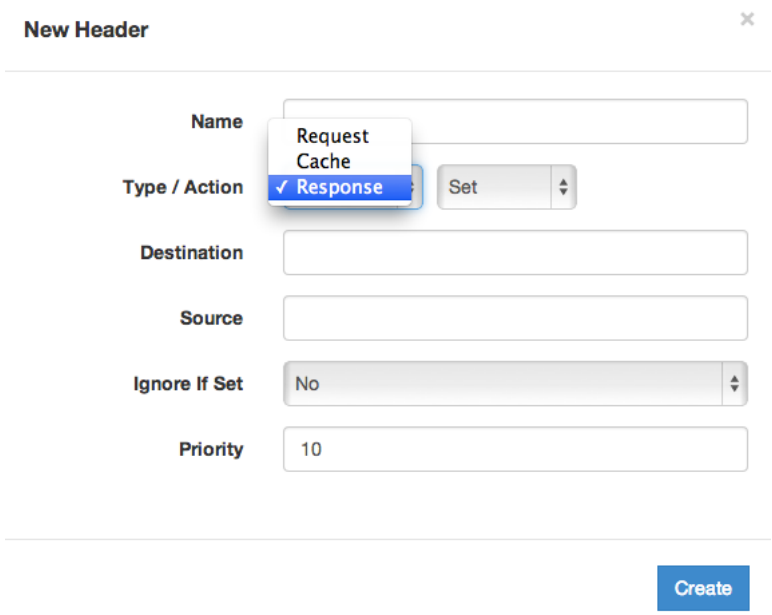
In addition, IE8 has some odd behavior to do with the back button. Adding

`Vary: *`

seems to fix the problem.

REMEMBER: 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 of type Response (/guides/tutorials/responses-tutorial) under the Content pane of your control panel.





· How can I manipulate the X-Forwarded-For header? (/guides/caching/how-can-i-manipulate-the-x-forwarded-for-header)

You can control what happens to the X-Forwarded-For HTTP header via the **Request Settings** window in the **Settings** pane on the **configure** tab. From the **X-Forwarded-For** drop down 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 present.
- **Overwrite** - Overwrites the X-Forwarded-For header with just the client IP.

For more information about requests and responses, see our tutorial (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses).

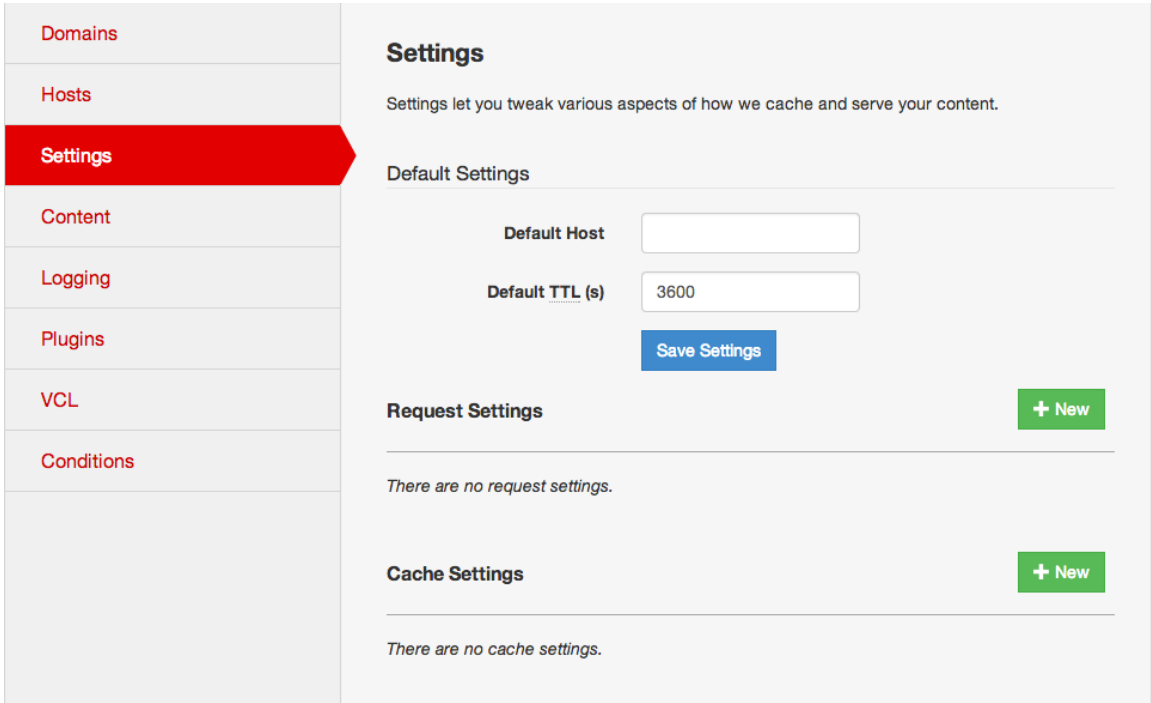
· How can I override default caching based on a backend response? (/guides/caching/how-can-i-override-default-caching-based-on-a-backend-response)

In certain situations you may want to conditionally apply a different caching policy (/guides/caching/how-do-i-conditionally-prevent-pages-from-caching) 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 creation conditions for it.

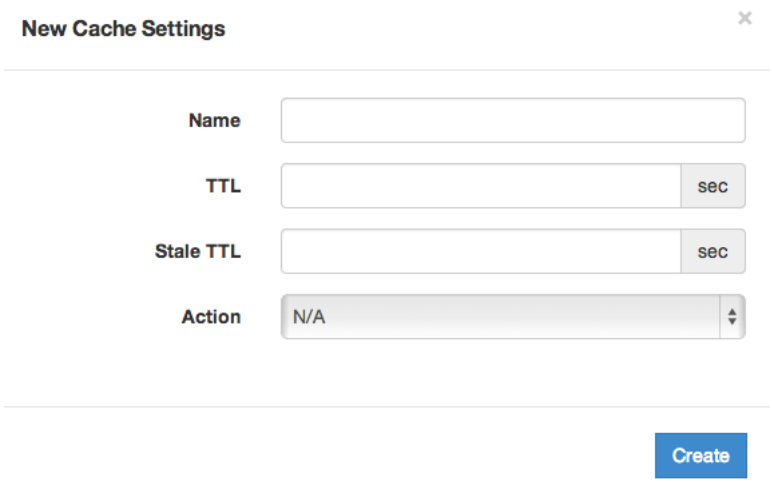
Creating the New Cache Object

1. Log in to the Fastly application.
2. Click the **configure** tab to access the control panel.

3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **Settings** section.



6. In the **Cache Settings** area, click **New** button. The New Cache Settings window appears.

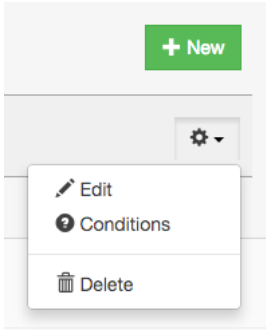


7. Fill out the **New Cache Settings** window as follows:
   - In the **Name** field, type a descriptive name for the new cache settings.
   - In the **TTL** field, type the amount of time, in seconds, to cache the objects (e.g., `300`).
   - In the **Stale TTL** field type the amount of time to serve stale or expired responses, in seconds, should the backend become unavailable (e.g., `300`).
   - From the **Action** menu, select **Cache**.
8. Click the **Create** button. A new cache object appears in the Cache Settings area of the Settings section.

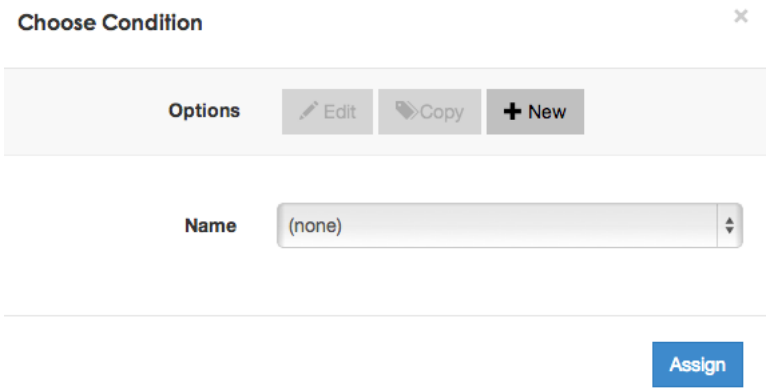
Creating an Override Condition for the new Cache Object

Once the object has been created you will need to add a condition to it.

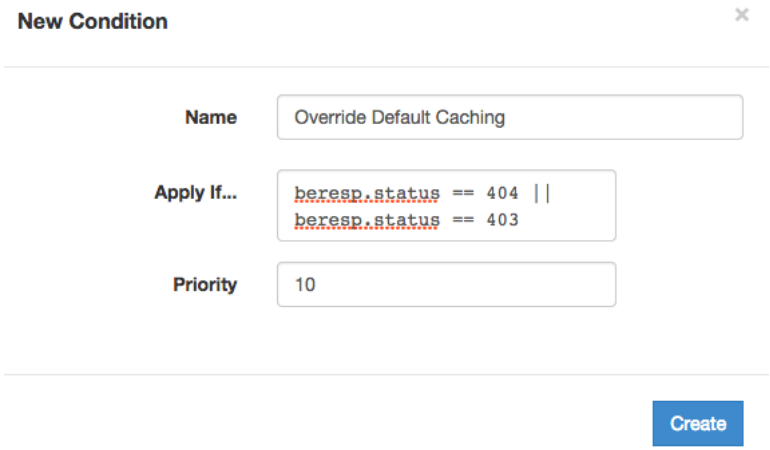
1. Click the gear icon to the right of the object and select Conditions.



The Choose Condition window appears with nothing selected.



2. Click **New** at the options at the top of the window. The New Condition window appears.



3. Fill out the **New Condition** window as follows:
   - In the **Name** field, type a descriptive name for the new condition.
   - In the **Apply If** field type an appropriate backend response header to specify when the condition will be applied. For example, `beresp.status == 404 || beresp.status == 403`
   - Leave the **Priority** field set at its default value.
4. Click **Create** to create the condition.

Once you create the condition, remember to assign it, then deploy the new config version and you are done.

Other notes

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

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

· How Can I Serve Stale Content on Errors (/guides/caching/how-can-i-serve-stale-content-on-errors)

This help document is a part of the basic setup process for serving stale or expired content to users. 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. These instructions require that custom VCL be enabled (/guides/vcl-faqs/where-can-i-upload-custom-vcl). Email support@fastly.com (mailto:support@fastly.com) if this feature is not enabled for your account.

In the context of Varnish, there are three ways an origin can fail:

- The origin can be marked as unhealthy by failing healthchecks.
- 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.

Although not strictly necessary, healthchecks (/guides/backend-servers/health-checks-tutorial) should be enabled in conjunction with this VCL. Without healthchecks 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 healthchecks 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, you should customize the following values to suit your 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.
- `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.
- `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 {
#FASTLY recv

   if (req.request != "HEAD" && req.request != "GET" && req.request != "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.request == "GET" || req.request == "HEAD")) {
      restart;
    }

    /* else go to vcl_error to deliver a synthetic */
    error 503;

  }

  /* 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.request == "GET" || req.request == "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 */

   /* optional, rewrite status code */
   /* set obj.status = 200; */
   /* set obj.response = "OK"; */

   /* include your HTML response here */
   synthetic {"<!DOCTYPE html>Your HTML!</html>"};
   return(deliver);
 }

}

sub vcl_pass {
#FASTLY pass
}
```

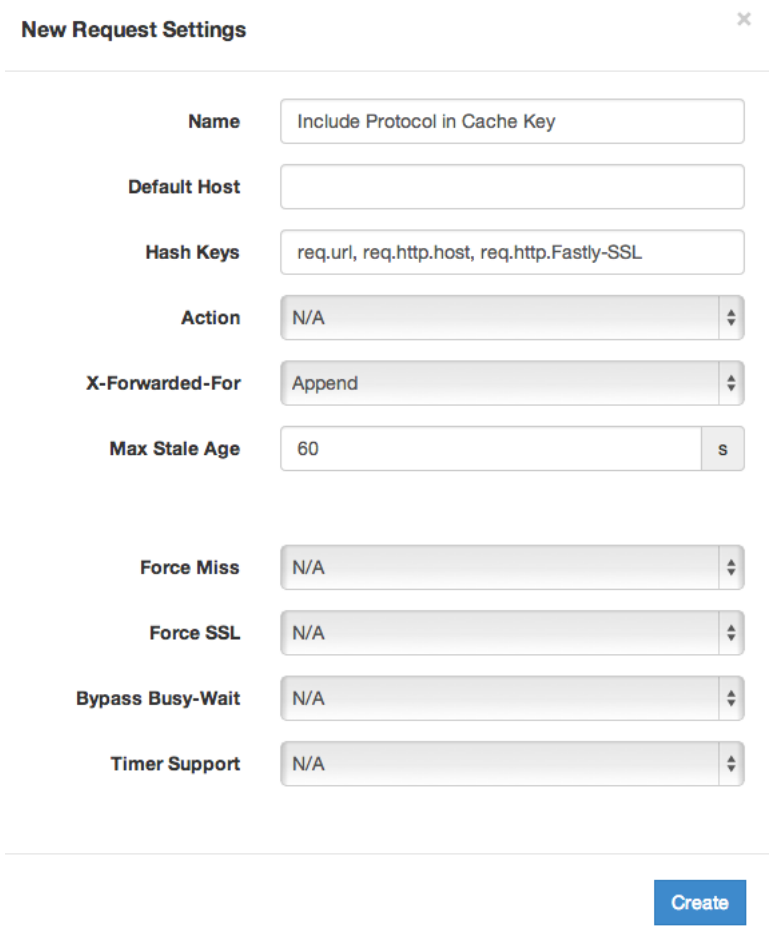
· How do I change what the cache key is defined as? (/guides/caching/how-do-i-change-what-the-cache-key-is-defined-as)

Fastly allows you to define precisely what makes an HTTP object unique. By default we use the URL and the host of a request (plus a special, internal Fastly variable called GENERATION that allows us to make Purge All (/guides/purging/how-can-i-purge-content-on-fastly) work).

However, you can set the cache key explicitly (including attaching conditions) by adding a request setting from the Settings pane of your configuration control panel (/guides/fastlys-web-interface/about-the-configuration-control-panel) and including a comma separated list of cache keys.

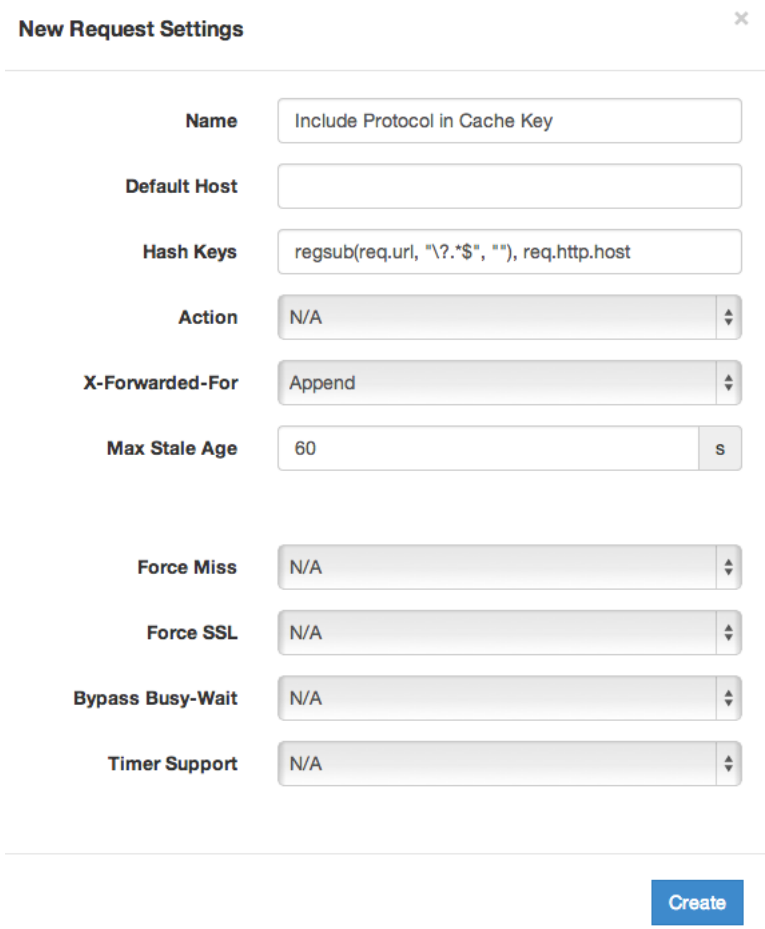
For example, for CORS compliance you could include the protocol in the cache key by setting it to this text in the Hash Keys field of the Request Settings:

`req.url, req.http.host, req.http.Fastly-SSL`



or alternatively if you don't want the query params to be part of the cache key you could use this text in the Hash Keys field:

```
regsub(req.url, "\?.*$", ""), req.http.host
```



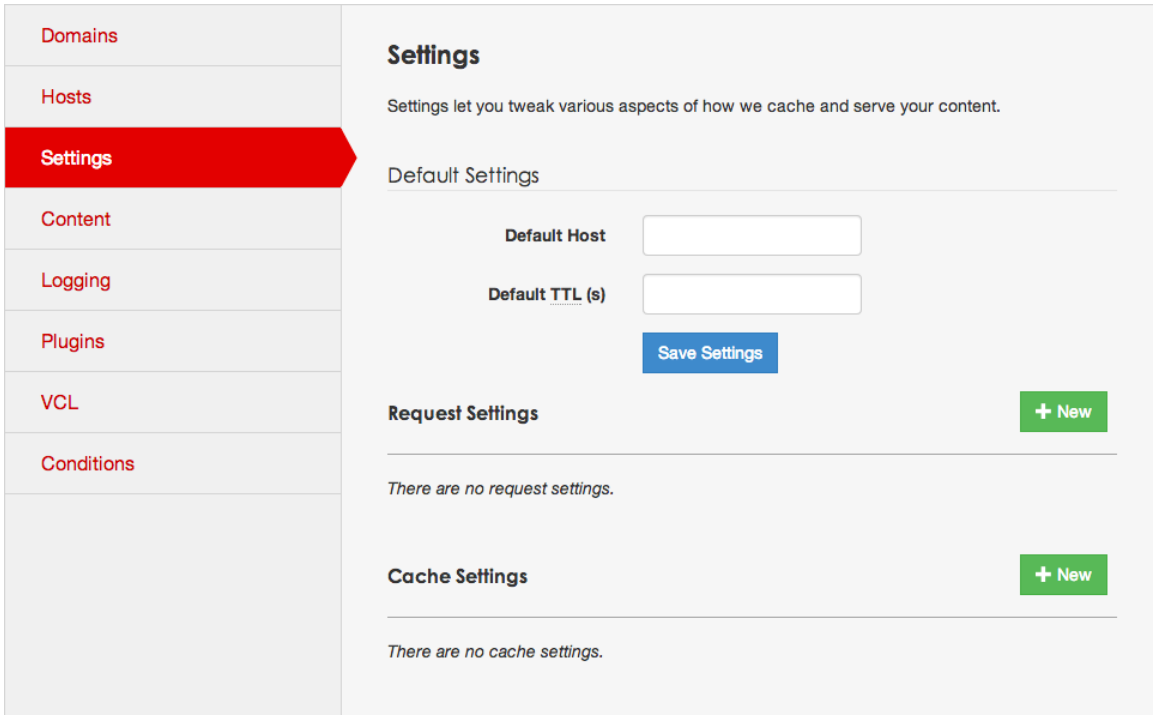
NOTE: If you put something in the cache key then purges will also need to contain that. For example, if you add Fastly-SSL to the cache key then you'll need to do a purge on both the http and https versions of the URL.

· How do I conditionally prevent pages from caching? (/guides/caching/how-do-i-conditionally-prevent-pages-from-caching)

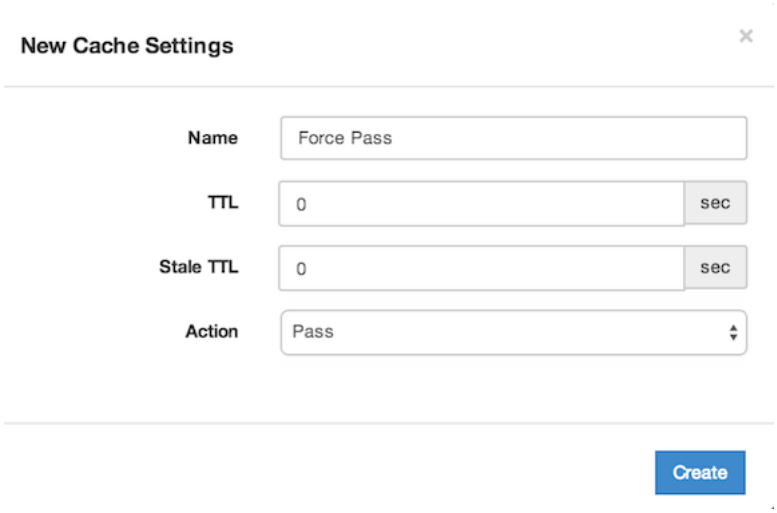
To conditionally prevent pages from caching, follow the steps below.

1. Log in to the Fastly application.
2. Click on the **configure** button (the wrench icon at the top of the application window).

3. Select the appropriate service from the **Service** drop down menu.
4. Click the blue **Configure** button to the right of the displayed service name.
5. Click **Settings** from the section list on the left.

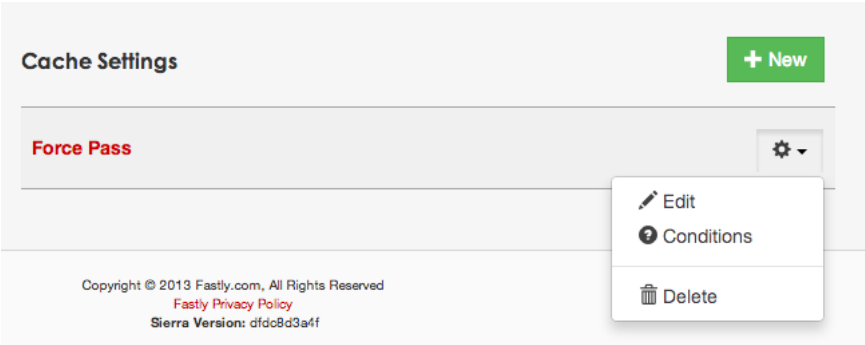


6. In the **Cache Settings** area, click the green **New** button to create a new cache setting. The New Cache Settings window appears.

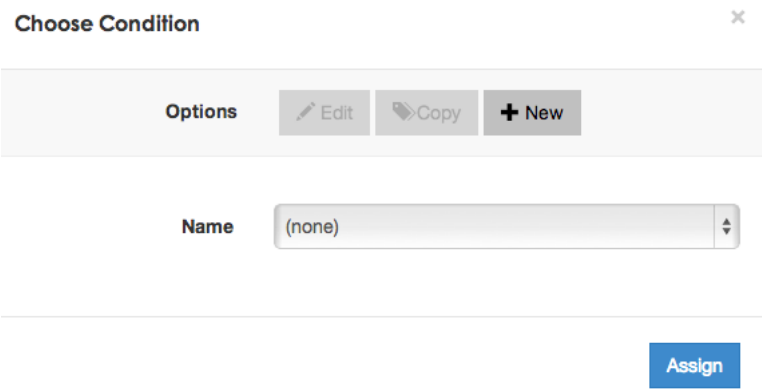


7. Create a new cache setting and click then click the **Create** button. The new cache setting you created appears in the Cache Settings area of the Settings pane on the Configuration tab.

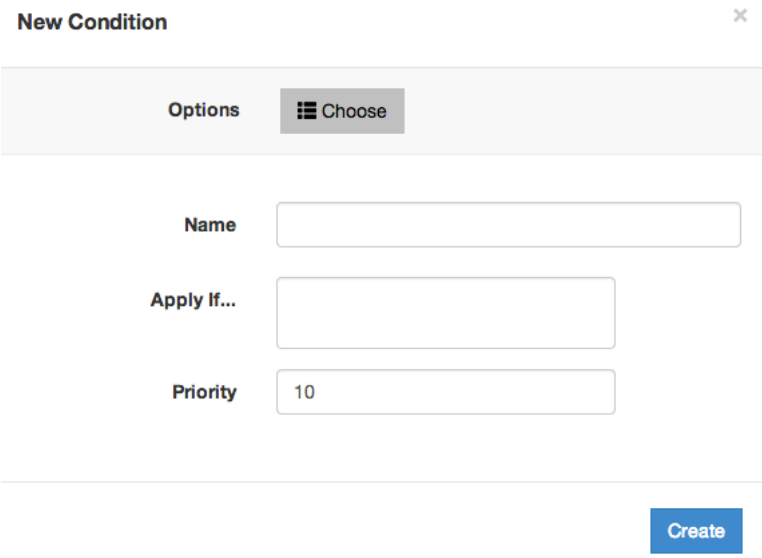
8. Click the gear icon to the right of the newly created cache setting and then select **Conditions** from the drop down menu.



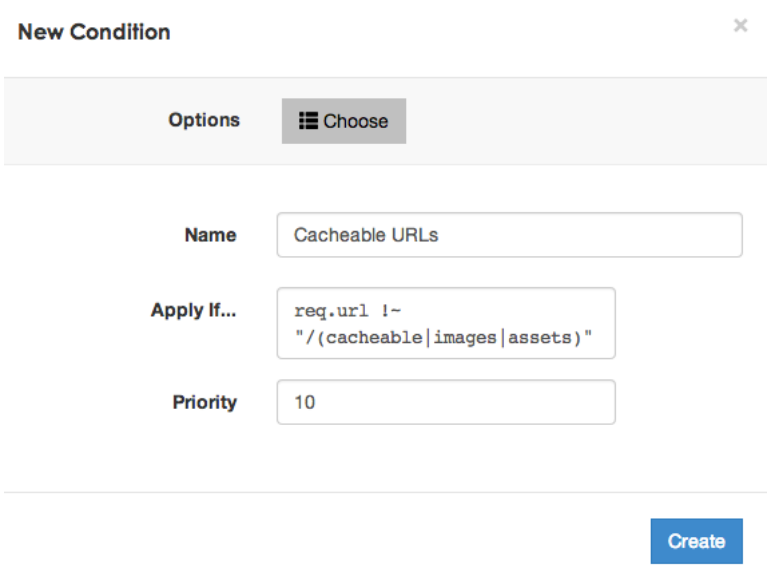
The Choose Condition window appears.



9. Click the **New** button on the **Choose Condition** window. The New Condition window appears.



10. Create a condition that matches against the URLs you want and then click the **Create** button. In this example, we set the condition to look for URLs containing '/cacheable', '/images', or '/assets'. If it finds them, the URLs should be cached. If it doesn't find them, the URLs are explicitly not cached by the Apply If statement shown below.



TIP: You can use these steps to override default caching based on a backend response (/guides/caching/how-can-i-override-default-caching-based-on-a-backend-response).

· How do I deliver different content to different devices? (/guides/caching/how-do-i-deliver-different-content-to-different-devices)

The easiest way 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 (/guides/vcl-faqs/how-can-i-change-the-url-based-on-the-domain) using our user interface but in pure VCL it would look something like this:

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

Obviously the code fragment above doesn't contain a comprehensive list of mobile and tablet devices. Google has an official blog post (http://googlewebmastercentral.blogspot.com/2011/03/mo-better-to-also-detect-mobile-user.html) on detecting Android mobile versus tablet and t (https://github.com/varnish/varnish-devicedetect/blob/master/devicedetect.vcl)his VCL fragment (https://github.com/varnish/varnish-devicedetect/blob/master/devicedetect.vcl) 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" ||
            req.http.User-Agent ~ "(?i)midp-2.0" ||
            req.http.User-Agent ~ "(?i)u990" ||
            req.http.User-Agent ~ "(?i)netfront" ||
            req.http.User-Agent ~ "(?i)opera mini" ||
            req.http.User-Agent ~ "(?i)palm" ||
            req.http.User-Agent ~ "(?i)nintendo wii" ||
```

```
            req.http.User-Agent ~ "(?i)playstation portable" ||
            req.http.User-Agent ~ "(?i)portalmmm" ||
            req.http.User-Agent ~ "(?i)proxinet" ||
            req.http.User-Agent ~ "(?i)sonyericsson" ||
            req.http.User-Agent ~ "(?i)symbian" ||
            req.http.User-Agent ~ "(?i)windows\ ?ce" ||
            req.http.User-Agent ~ "(?i)winwap" ||
            req.http.User-Agent ~ "(?i)eudoraweb" ||
            req.http.User-Agent ~ "(?i)htc" ||
            req.http.User-Agent ~ "(?i)240x320" ||
            req.http.User-Agent ~ "(?i)avantgo") {
            set req.http.X-UA-Device = "mobile";
        }
    }
}
```

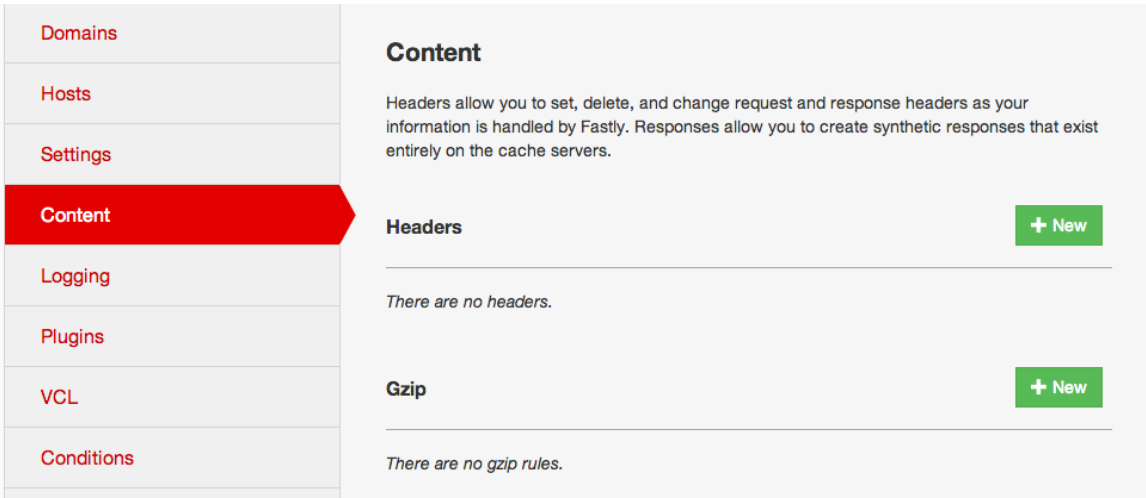
· How do I enable automatic gzipping? (/guides/caching/how-do-i-enable-automatic-gzipping)

WARNING: Because dynamic gzipping fetches content from origin, compresses it, and then caches it, this feature doesn't work with our ESI feature (/guides/caching/which-language-elements-of-the-esi-specification-does-fastly-support). If you enable gzipping, Fastly will stop processing ESI language elements.

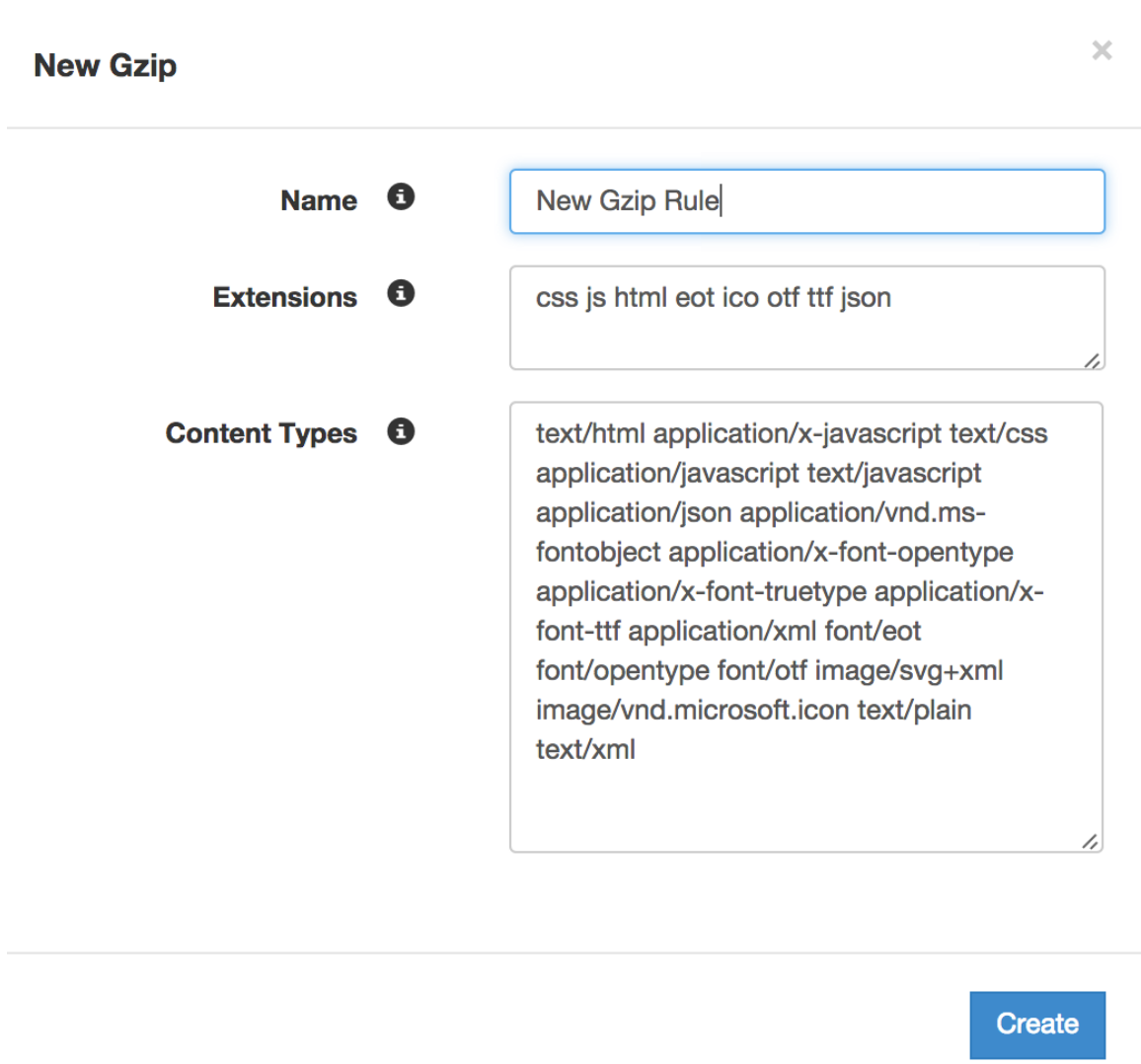
To dynamically gzip content based on file extension or content-type following the steps below. (NOTE: We also have a historical article that describes how to manually enable gzipping (/guides/caching/how-do-i-manually-enable-dynamic-gzipping)).

1. Log in to the Fastly application.
2. Click the **configure** tab to access the control panel.

3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **Content** section.



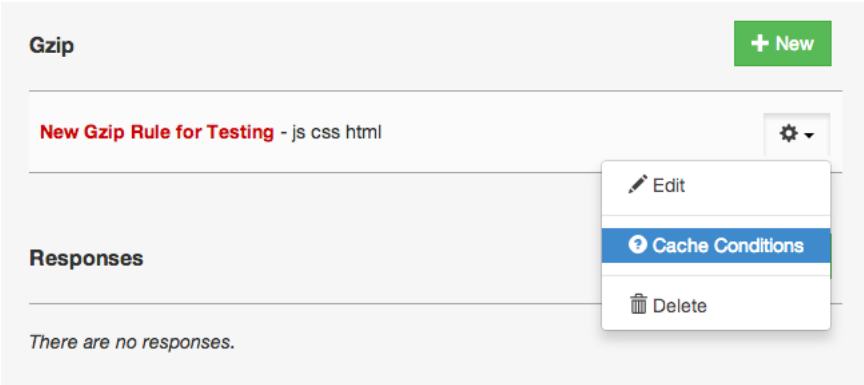
6. In the **Gzip** area, click the **New** button to create a new gzip rule. The New Gzip window appears.



7. Fill in the fields of the **New Gzip** window as follows:
   - In the **Name** field, type an arbitrary name for your new gzip rule.
   - In the **Extensions** field, type the file extensions for each file type you wish to have dynamically gzipped, separated by spaces. Only type the file extension. The dot (`.`) and regular expressions are not necessary. We recommend setting the Extensions to `html js css eot ico otf ttf json`.
   - 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. We recommended setting the Content Types to the following: `text/css text/html text/javascript text/plain text/xml application/javascript application/json application/vnd.ms-fontobject applicat`
8. Click the **Create** button. The new GZIP rule appears in the GZIP area of the Content section
9. Click **Activate** at the top of the window to apply the changes to your service.

Adding Caching Conditions to GZIP Rules

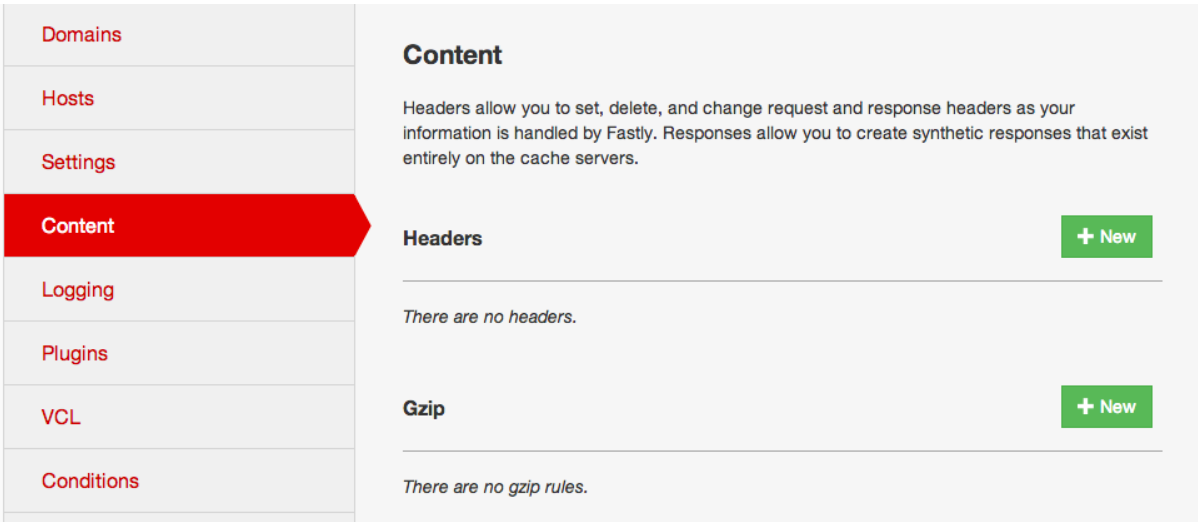
You can add conditions to GZIP via the Cache Conditions menu associated with any rule you create.



See our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial) and our Conditions Tutorial (/guides/tutorials/conditions-tutorial) for specific information.

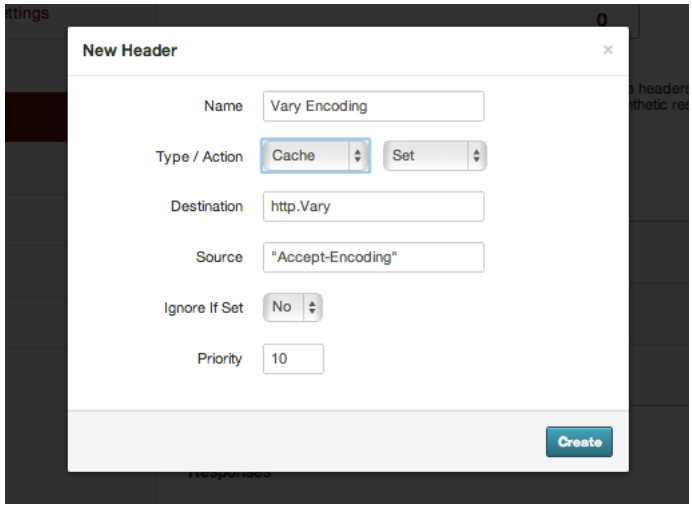
· How do I manually enable dynamic gzipping? (/guides/caching/how-do-i-manually-enable-dynamic-gzipping)

UPDATE: We've added a widget (/guides/caching/how-do-i-enable-automatic-gzipping) to make gzipping much easier. It's located on the configure tab in the new Gzip section of the Content pane:



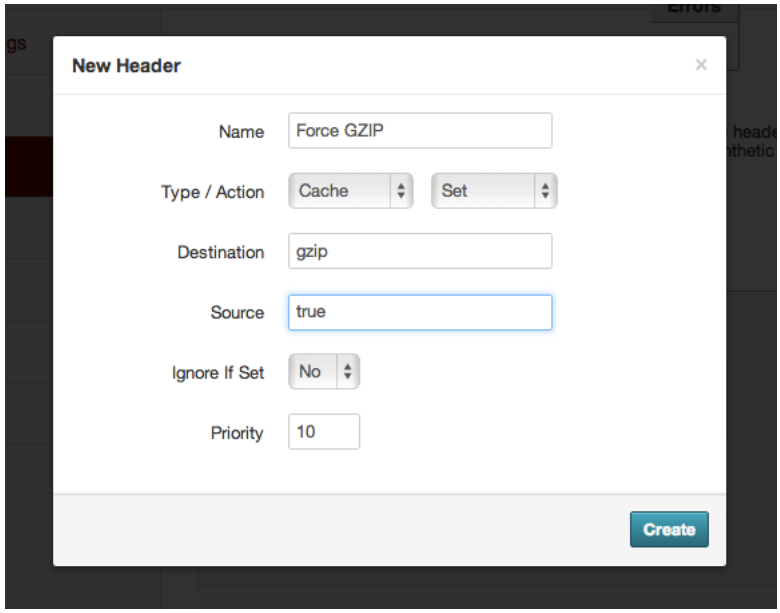
The widget has the ability to dynamically gzip content based on file extension or content-type. However we've left this tutorial here for historical purposes (it might also be useful to help with other configurations). If you want to do dynamic gzipping manually or have more control then read on!

First go to the "Content" pane of the control panel for your service and create a new "Header". The important parts are that it's a "Cache" type, that it's "Set", that the destination is "http.Vary" and the source is "Accept-Encoding".



This forces the cache to respect different encodings so things don't get confused.

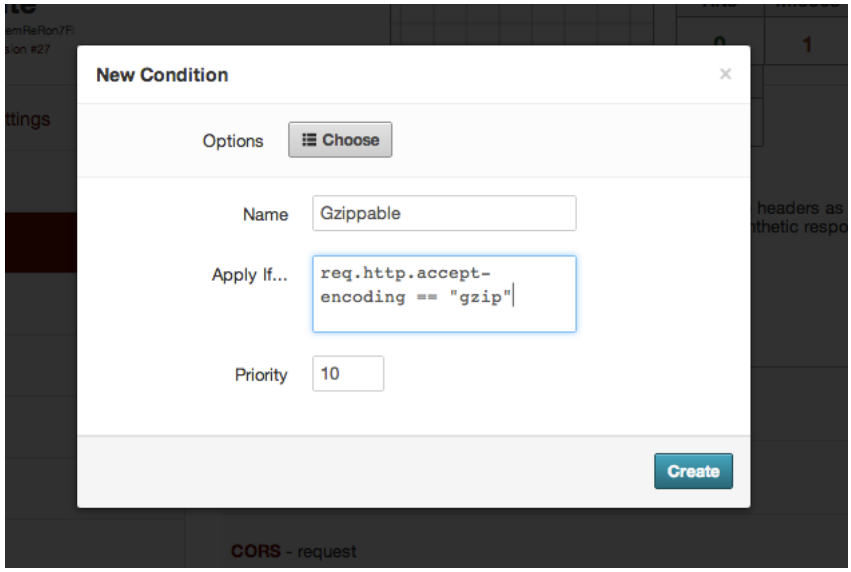
Next create another new "Cache" type header with a "Set" action



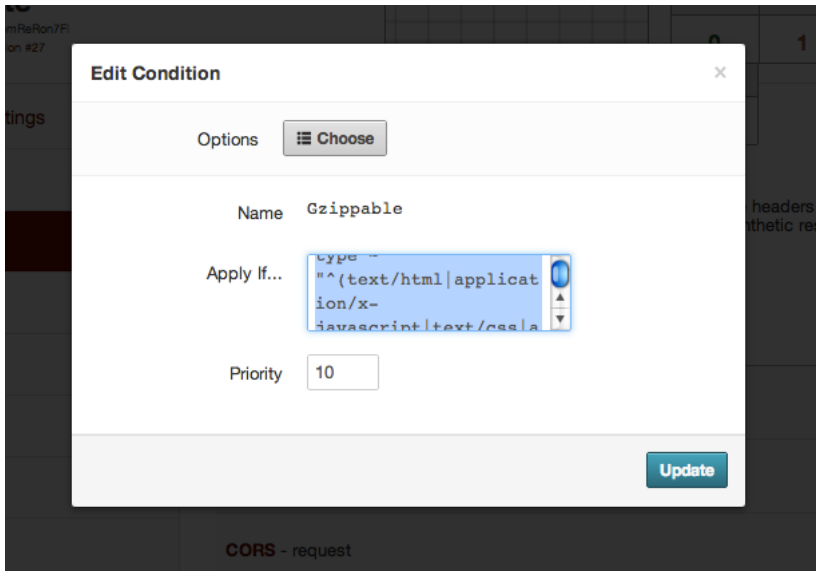
This forces everything to be gzipped. However first we want to check that the browser can actually accept Gzipped content so go back to the GZIP cache condition and click on the arrow on the right

now click on "cache conditions" to create a new one with the "Apply If ... " condition of

```
req.http.accept-encoding == "gzip"
```



If you deploy your service now, everything should work and be gzipped by default. However for images that doesn't tend to be very useful - gzipping works best for HTML, Javascript and CSS. So edit the "Gzippable" condition



the "Apply if ..." is hard to read but what it now says is

```
req.http.accept-encoding == "gzip" && beresp.http.content-type ~ "^(text/html|application/x-javascript|text/css|application/javascript|text/java
```

Click "update".

Now go and add a new condition for "Vary Encoding" which just has the

```
beresp.http.content-type ~ "^(text/html|application/x-javascript|text/css|application/javascript|text/javascript)"
```

and the conditions for both "Vary Encoding" and for "Force GZIP" should be updated.

If you check your VCL there should now be a section in vcl_fetch that looks like this

```
if ( req.http.accept-encoding == "gzip" && beresp.http.content-type ~ "^(text/html|application/x-javascript|text/css|appli
cation/javascript|text/javascript)" ) {
 # Header rewrite Force GZIP : 10
 set beresp.gzip = true;
}
if ( beresp.http.content-type ~ "^(text/html|application/x-javascript|text/css|application/javascript|text/javascript)" )
{

# Header rewrite Vary Encoding : 10
 set beresp.http.Vary = beresp.http.Vary ",Accept-Encoding";
}
```

If you deploy again and inspect the headers (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses) for some CSS versus an image you'll see that the CSS is gzipped and the image isn't.

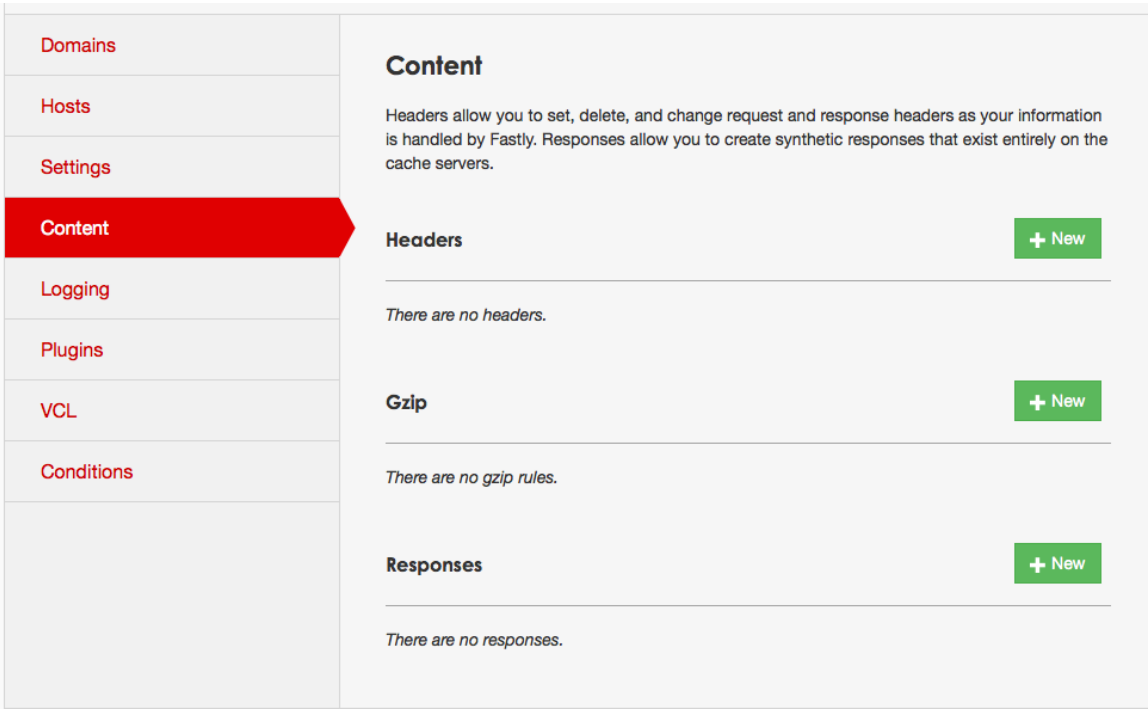
· How do I use a cookie as a cache key? (/guides/caching/how-do-i-use-a-cookie-as-a-cache-key)

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 (/guides/tutorials/conditions-tutorial). 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=`

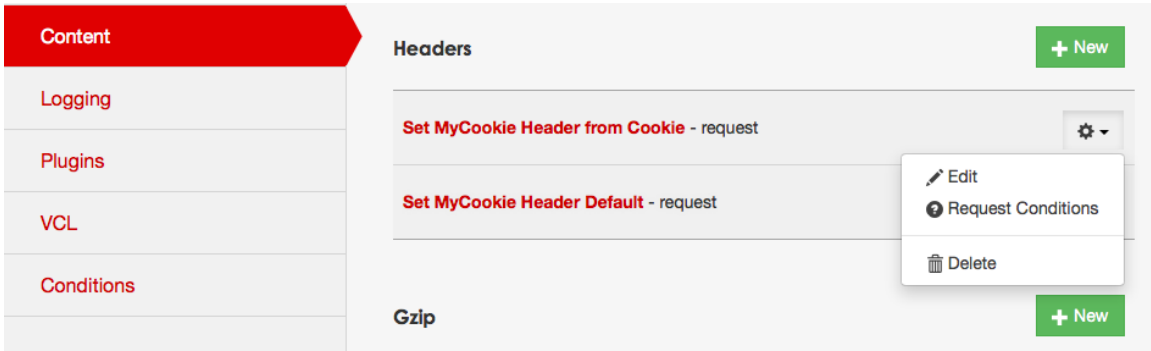
1. Go to the **configuration** tab, select your service, and select the **Content** pane.



2. Click the **New** button next to the **Headers** section and create a new header called `Set MyCookie Header Default`.
   - Set the **Type** to `Request`
   - Set the **Action** to `Set`
   - Set the **Destination** to `http.X-MyCookie`
   - Set the **Source** to `"0"` (with quotes)
3. Create another new header called `Set MyCookie Header from Cookie`.

- Set the **Type** to `Request`
- Set the **Action** to `Set`
- Set the **Destination** to `http.X-MyCookie`
- Set the **Source** to `regsub(req.http.cookie, ".*mycookie =([^;]+);.*", "\1")`
- Set the **Priority** to be larger than the default header you just created

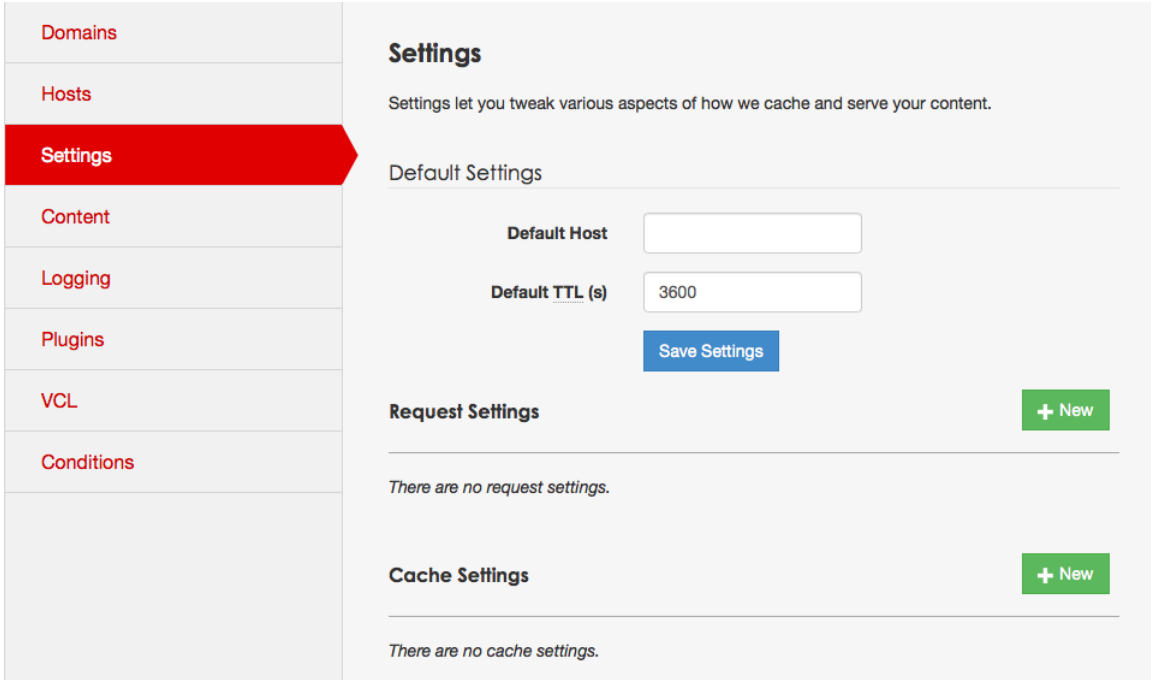
4. Click the gear icon to the right of the newly created `Set MyCookie Header from Cookie` header and select **Request Conditions**.



5. Choose the request conditions.

- Set the **Name** to `Has MyCookie cookie`
- Set the **Apply If** to `req.http.cookie ~ "mycookie="`

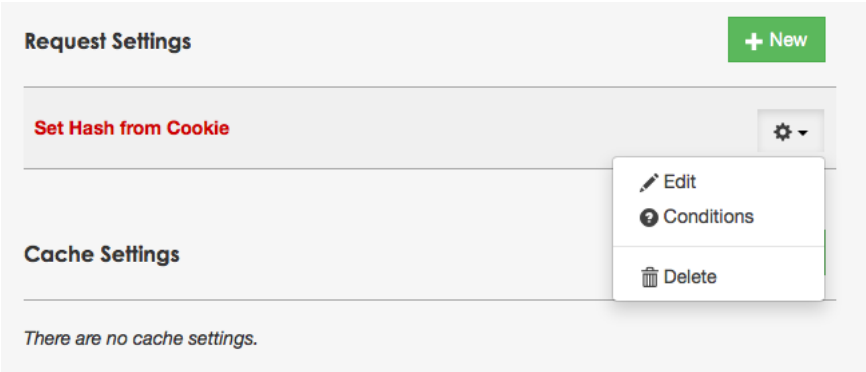
6. Now select the **Settings** pane.



7. Click the **New** button next to the **Request Settings** section and create a new request setting.

- Set the **Name** to `Set Hash from Cookie`
- Set the **Hash Keys** to `req.url, req.http.host, req.http.X-MyCookie`

8. Click the gear icon to the right of the newly created request setting and select **Conditions**.



9. On the **Choose Conditions** window, click **New** to add a new condition.

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 Hash Key is `req.url, req.http.host, "Has mycookie"`
2. Add a condition to that setting where the "Apply if ..." is `req.http.cookie ~ ".*mycookie =1;.*"`

· How long does content remain cached? (/guides/caching/how-long-does-content-remain-cached)

We have no set hard limit on how long objects will be cached for. The maximum cache time depends on a number of factors including the TTL (Time To Live) and Grace Period, how often the object gets accessed, and how busy other customers are.

There's also a chance we have to reboot our servers (we don't have a persistent cache store, although that is on the roadmap). Setting TTL and Grace Period to a week, possibly even two weeks should be absolutely fine. For more information about controlling how long Fastly caches your resources, start with our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial).

· Is it possible to tell Fastly to cache for a different amount of time than for end users? (/guides/caching/is-it-possible-to-tell-fastly-to-cache-for-a-different-amount-of-time-than-for-end-users)

This is possible using the Surrogate-Control headers defined by the W3C (http://www.w3.org/TR/edge-arch). If, for example, you wanted Fastly to cache something for a month (clearing with API purges, if necessary) but you also wanted to set a maximum age of a single day for users viewing that object in a browser, then you could return the HTTP header:

```
Surrogate-Control: max-age=2629744
Cache-Control: max-age=86400
```

The Surrogate Control header in this example tells Fastly to cache the object for a maximum of 2629744 seconds (one month). The Cache Control header in this example tells the browser to cache the object for a maximum of 86400 seconds (1 day).

For more information about controlling caching, see our guides on caching (/guides/caching/) or our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial).

· Is the cache on a Fastly query string agnostic? (/guides/caching/is-the-cache-on-a-fastly-query-string-agnostic)

Under normal circumstances, Fastly would consider these URLs as different objects that are cached separately:

- http://example.com
- http://example.com?asdf=asdf
- http://example.com?asdf=zxcv

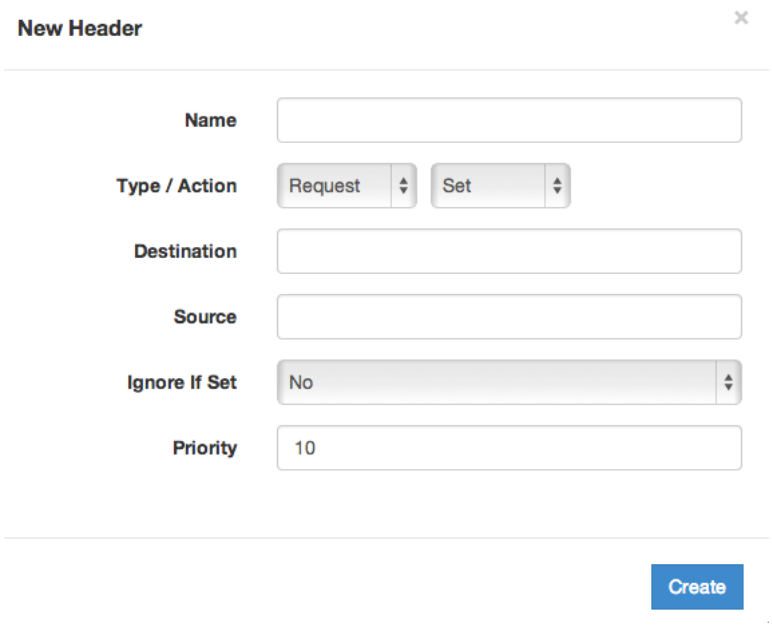
It is possible, however, to have them all ignore the query string and return the same cached file.

1. Log in the Fastly application and click the **configure** tab (wrench icon).

2. From the **Service** drop down menu, select the appropriate service and then click the blue **Configure** button. The main controls for your selected service appear.

3. Click **Content** from the section list on the left.

4. In the **Headers** area, click the **New** button. The New Header window appears.



5. Fill out the **New Header** window as follows:

   - Set the **Name** field to whatever you want (e.g., "Chop Off Querystring")
   - Select **Request** from the **Type** menu.
   - Select **Set** from the **Action** menu.
   - In the **Destination** field type `url`.
   - In the **Source** field type `regsub(req.url, "\?.*$", "")`
   - Select **No** from the **Ignore If Set** menu.
   - Set the **Priority** field to whatever priority you want.

6. Click the **Create** button to create the new header. The new header you created appears in the list of headers on the Content page.

This will chop off the query string if it's there and not use it when looking it up in the cache or when requesting from the backend.

For more information about controlling caching, see our guides on caching (/guides/caching/) or our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial).

· What is the default caching behavior of HTTP verbs other than GET? (/guides/caching/what-is-the-default-caching-behavior-of-http-verbs-other-than-get)

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.

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
```

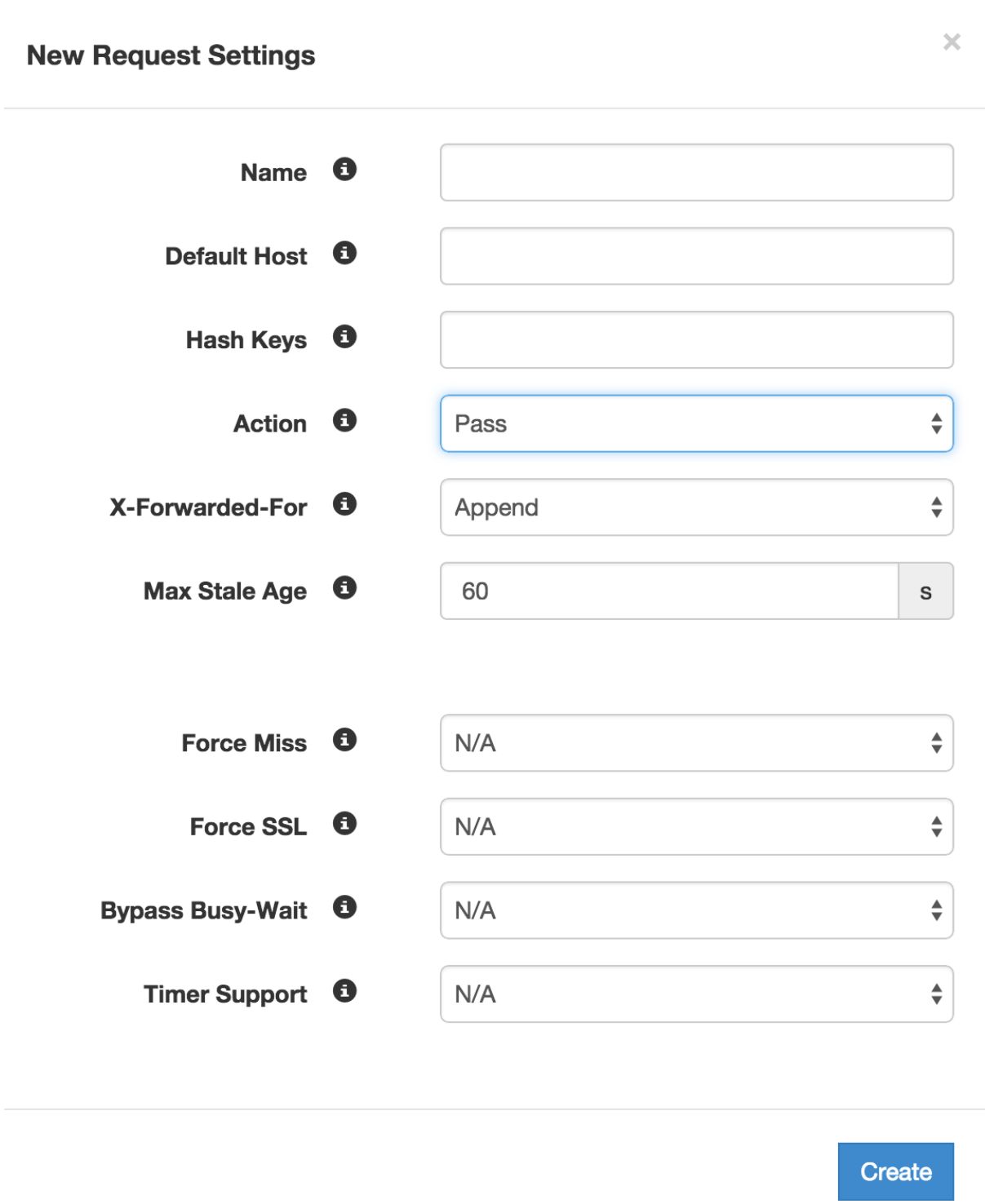
Anything other than HEAD or GET requests are proxied and not cached by default. However, POSTs *can* be cached if required, and caching of that can be keyed off of the contents of the POST request body if that is below 2K. We've posted an example of how to do this (/guides/caching/can-i-use-a-post-request-body-as-a-cache-key) in our caching guides (/guides/caching/).

· What is the difference between PASS action set in Request Settings and Cache Settings? (/guides/caching/what-is-the-difference-between-pass-action-set-in-request-settings-and-cache-settings)

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)`.

Different Approaches to Passing

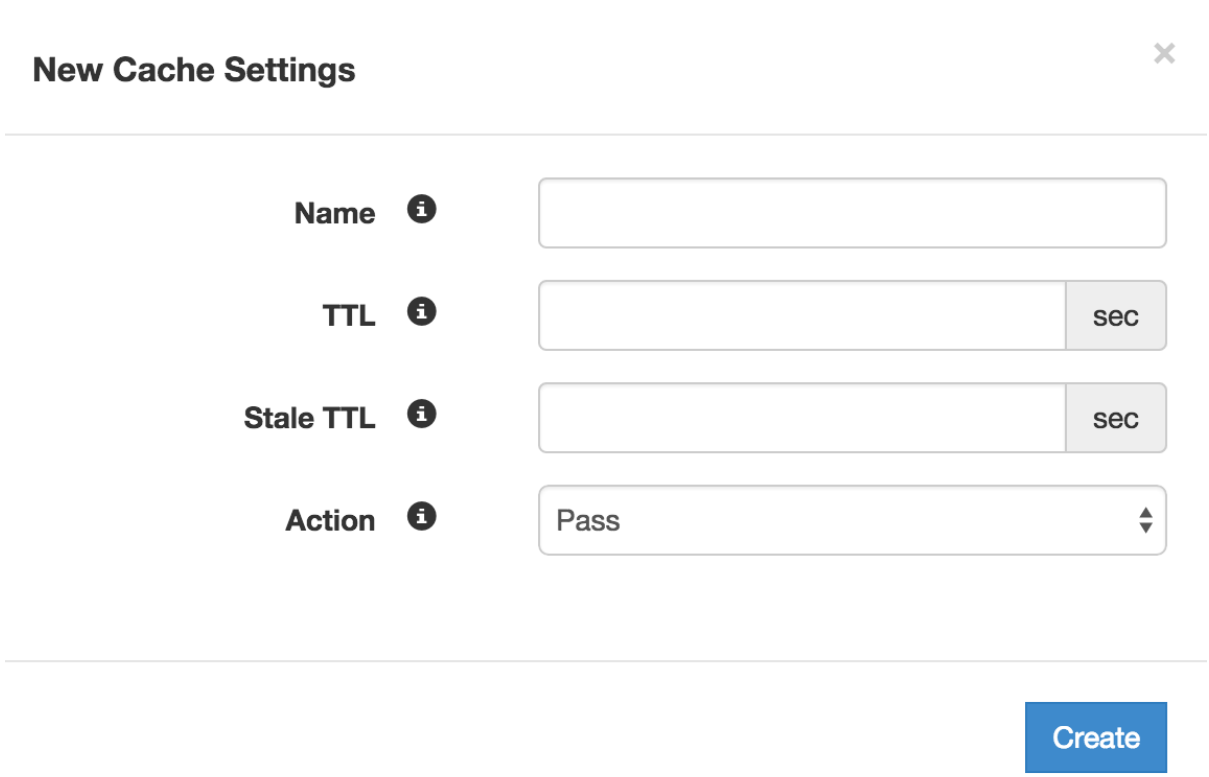
Using a Request Setting



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 in 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 from 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. Note that 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.

For more information about controlling caching, see our guides on caching (/guides/caching/) or our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial).

· What support does Fastly have for large files? (/guides/caching/what-support-does-fastly-have-for-large-files)

Although Fastly is ideal for caching small objects we have added two features to enhance performance specifically for large files up to 5GB: Stream-on-Miss and Large File Support.

Stream-on-Miss

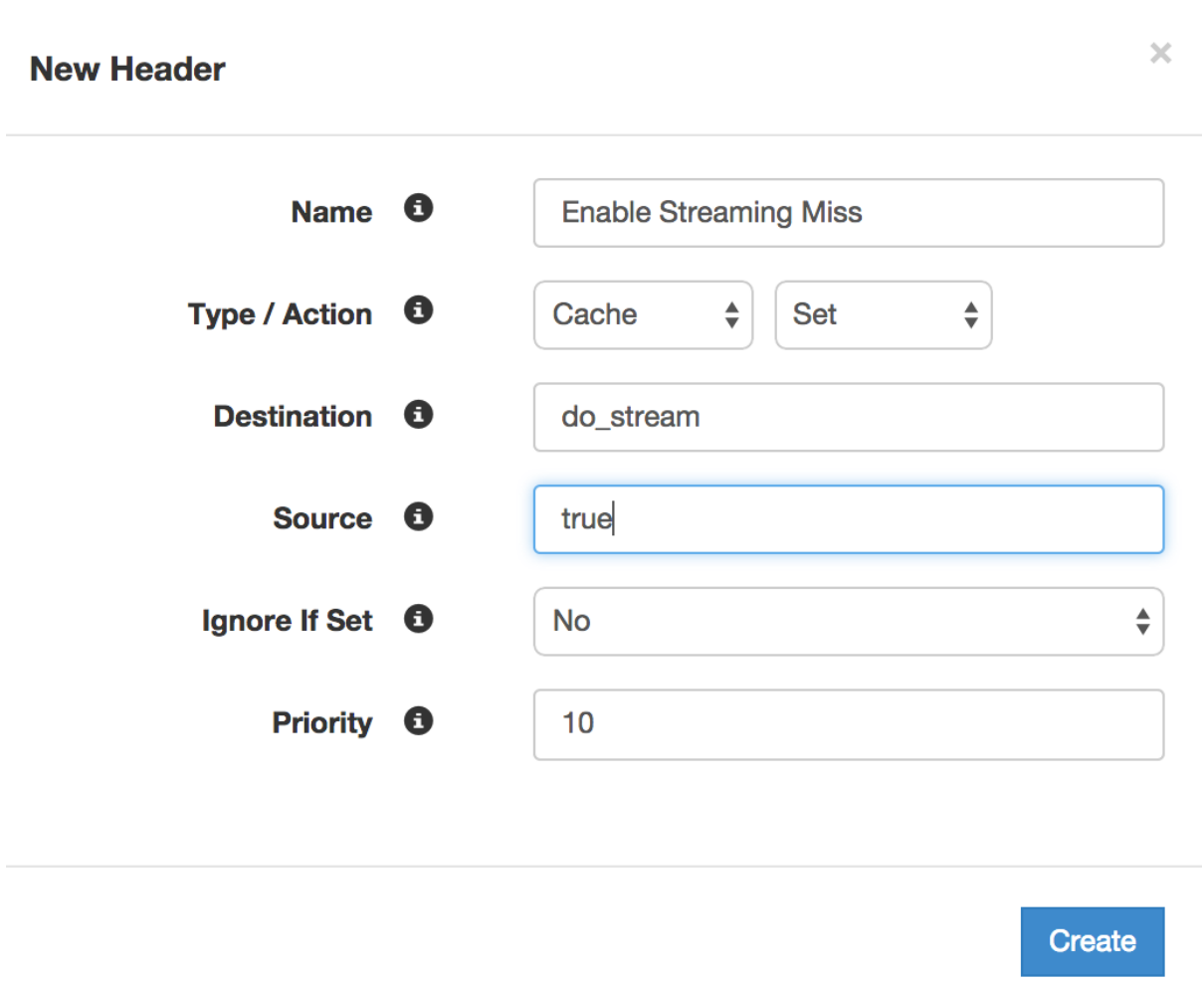
Our Stream-on-Miss support means that, when fetching an object from the origin the response is streamed back to the client immediately and is written to cache only after the whole object has been fetched. This reduces the first-byte latency, which is the time that the client must wait before it starts receiving the response body. The larger the response body, the more pronounced the benefit of streaming.

Configuration

Configuration is simple - in VCL simply set `beresp.do_stream` to true in `vcl_fetch`.

```
  sub vcl_fetch {
    ...
    set beresp.do_stream = true;
    ...
    return(deliver);
}
```

The same can be achieved by creating a new header of type “Cache”, action “Set”, Destination `do_stream` and Source `true` (this can, of course, be controlled with conditions (/guides/tutorials/conditions-tutorial)).



Limitations

Origins cannot use SSL/TLS

Stream-on-Miss does not currently support https (SSL/TLS) origin servers. The content can be served to the client over https, but it cannot be fetched with Stream-on-Miss over https.

Objects fetched from https origins are therefore limited to the non-Stream-on-Miss size of 2GB.

Stream-on-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 Stream-on-Miss will be disabled for the fetch and the fetched object will be subject to the non-streaming-miss object size limit.

If an HTTP/1.0 request is received while a Stream-on-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 Stream-on-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.

Stream-on-Miss is not compatible with on-the-fly gzip compressing of the fetched object

Stream-on-Miss can handle large files whether or not they are compressed.

However, on-the-fly compression of objects that are not already compressed is not compatible with Stream-on-Miss. If the VCL sets `beresp.gzip` to true, Stream-on-Miss will be disabled.

Stream-on-Miss is not compatible with ESI (Edge-Side Includes)

Responses that are processed through ESI cannot be streamed. Responses that are included from an ESI template cannot be streamed.

When ESI is enabled for the response or when the response is fetched using `<esi:include>`, then Stream-on-Miss will be disabled and the fetched object will be subject to the non-streaming-miss object size limit of 2GB.

Large File Support

Maximum File Size

If Stream-on-Miss is enabled then the maximum size is slightly below 5GB (specifically 5,368,578,048 bytes). With Stream-on-Miss disabled the maximum size is still higher than the previous maximum but is limited to a little under 2GB (specifically 2,147,352,576 bytes)

Failure Modes

What happens when the maximum object size limit is exceeded?

If the response from the origin has a Content-Length header field which exceeds the maximum object size Fastly will immediately generate a 503 response to the client unless specific VCL is put in place to act on the error.

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 specific VCL is in place to act on the error).

If no Content-Length header field is present and Stream-on-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 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.

What happens when an origin read fails?

A failure to read the response header from the origin, regardless of Stream-on-Miss, causes a 503 response (which can be acted on in VCL).

If reading the response body from the origin fails or times out, the problem will be reported differently depending on whether Stream-on-Miss is in effect for the fetch. Without Stream-on-Miss a 503 response will be generated as above. With Stream-on-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 again abruptly terminate the client connection and the client will detect a protocol violation either because, in the case when the response was chunked, it will see its connection close without a properly terminating 0-length chunk or, in the case when Content-Length was known, it will see the connection close before the number of bytes given by the

Incidentally this is the reason why HTTP/1.0 clients cannot be supported by Stream-on-Miss in the cases when the Content-Length is not yet known or available: without the client receiving a Content-Length and without support for chunking, the client cannot distinguish the proper end of the download from an abrupt connection breakage anywhere upstream from it.

· Where can I learn more about caching via API? (/guides/caching/where-can-i-learn-more-about-caching-via-api)

We have a great series of blog posts (http://www.fastly.com/blog/api-caching-part-1/) on this topic that you might be interested in. And if you have any questions, you can always contact support@fastly.com (mailto:support@fastly.com%20)for help.

· Which HTTP status codes are cached by default? (/guides/caching/which-http-status-codes-are-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 (/guides/tutorials/conditions-tutorial) and responses (/guides/tutorials/responses-tutorial).

Code	Message
`200`	OK
`203`	Non-Authoritative Information
`300`	Multiple Choices
`301`	Moved Permanently
`302`	Moved Temporarily
`404`	Not Found
`410`	Gone

· Which language elements of the ESI specification does Fastly support? (/guides/caching/which-language-elements-of-the-esi-specification-does-fastly-support)

Fastly supports the following ESI language elements:

- include
- comment
- remove

We don't support the following ESI language elements:

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

For more information about the ESI language, see the latest version of the ESI Language Specification 1.0 (http://www.w3.org/TR/esi-lang) submitted to the World Wide Web Consortium. For examples detailing how you can use ESI with Fastly, see our blog series on using ESI (http://www.fastly.com/blog/using-esi-part-1-simple-edge-side-include/).

· Why am I getting cache MISSes with Google Pagespeed Module? (/guides/caching/why-am-i-getting-cache-misses-with-google-pagespeed-module)

If you are using the Google Pagespeed module and notice constant MISSes for HTML pages, check your setting in the module's .htcaccess 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 the .htcaccess file with:

```
ModPagespeedModifyCachingHeaders off
```

More details about the Pagespeed Module (https://developers.google.com/speed/pagespeed/module/install) can be found within Google Developers directory. For more information about controlling how long Fastly caches your resources, start with our Cache Control Tutorial (/guides/tutorials/cache-control-tutorial)

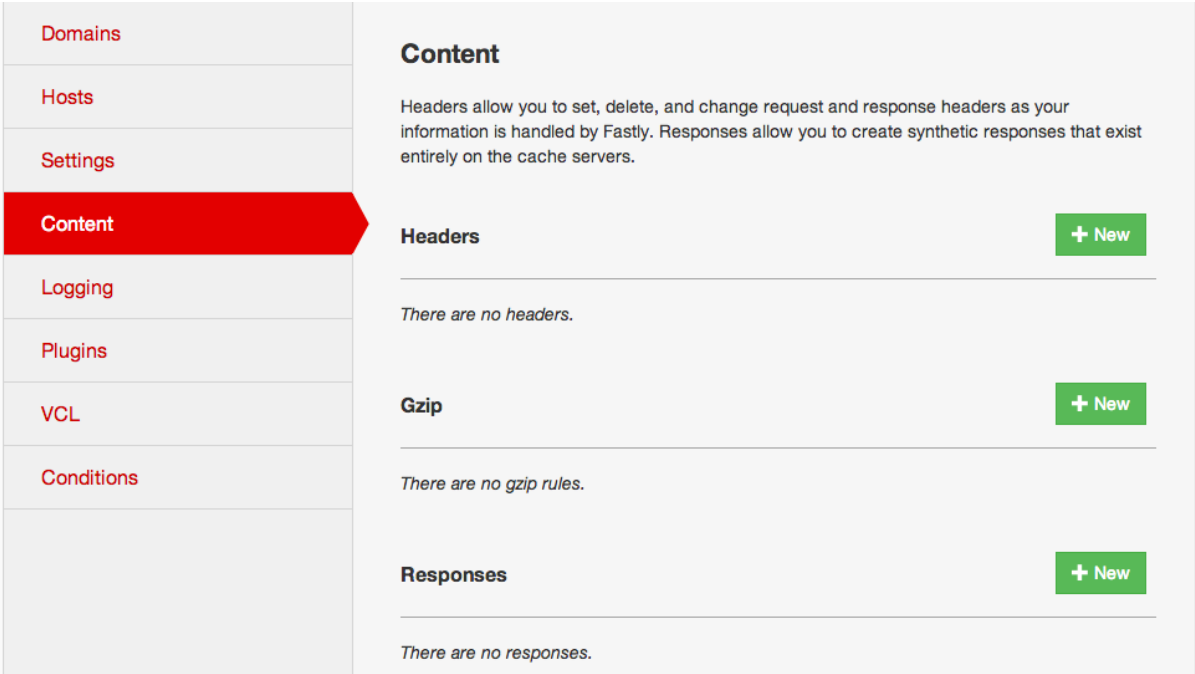
• Purging (/guides/purging/)

· How can I add Surrogate-Key headers when my backend is S3? (/guides/purging/how-can-i-add-surrogatekey-headers-when-my-backend-is-s3)

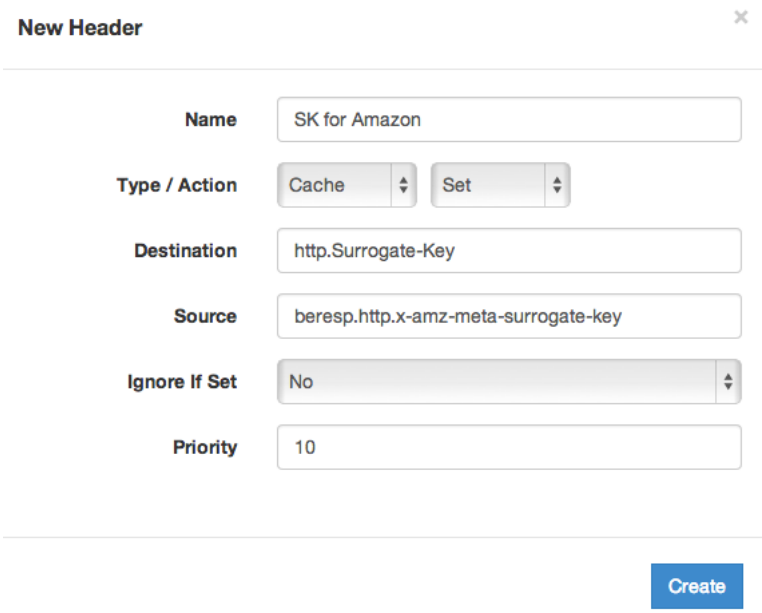
You can mark content with a key using a header and use that key to purge groups of specific URLs (/guides/purging/how-can-i-purge-content-on-fastly) at once without purging everything, or purge each URL singularly.

On the 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. (Please note that capitalization is important here, as S3 only accepts all lowercase header names.)

Inside the Fastly web interface, find the service you are setting this up for and then click on the Content section on the left side of the screen:



Then click the green New button next to the Headers section which will display the New Header window. You'll want to fill it out like this:



The **Name** field can be anything that is memorable to you.

The **Type** and **Action** controls should be selected as **Cache** and **Set** respectively.

The **Destination** field should be set to `http.Surrogate-Key`.

The **Source** field should be set to `beresp.http.x-amz-meta-surrogate-key`.

The remaining fields can be left set to their defaults.

Click the **Create** button to create the new header and remember to deploy the version you are editing for the new header to become live.

· How can I purge content on Fastly? (/guides/purging/how-can-i-purge-content-on-fastly)

Unlike the DNS (Domain Name Service) system, which relies on TTL (Time to Live) lifetimes expiring and allowing new records to be fetched, Fastly empowers you to actively expel content out of our caches at any time. This lets you set long cache times on your side, letting our servers take the load for longer, but leaves you safe in knowing that, should you need to update a resource at any time, it is only a few clicks away.

We offer varying levels of cache purging. You can purge something as small as a single URL via our "Purge URL" command or as large as a full purge of all content under your service via our "Purge All" command. Need something more selective? We also offer tag-based purging via "Purge Key" command.

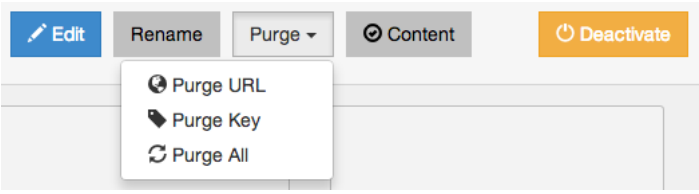
Tag-based purging of cache information allows you to emit special tag headers with your content, which our caches strip out and index. You can then issue purge commands to selectively flush out anything with a specific tag. For extra usefulness, items can have multiple tags that allow you to setup your own purging hierarchy.

All three purge commands are available through our web interface and via our API (however, please note that key purging requires some changes on your origin servers).

Purging Cached Content

To purge content using the web app, follow the steps below.

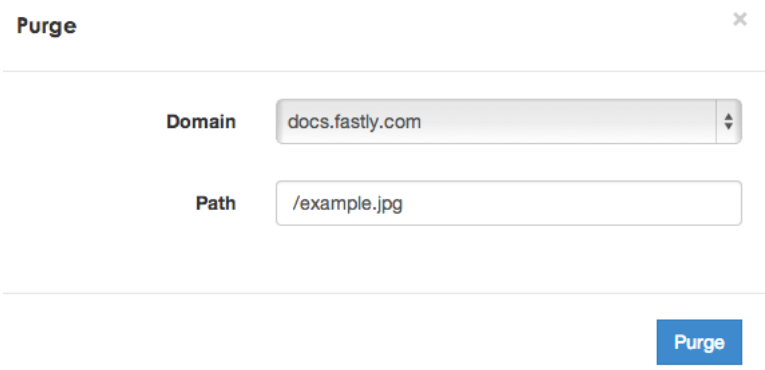
1. Log in to the Fastly application.
2. Click the **configure** button (the wrench icon at the top of the window).
3. From the **Service** drop down menu, select the service you want to work on.
4. Select the appropriate purge type from the **Purge** drop down button. All three purge functions are available through this button.



5. In the window that appears, provide the appropriate information to purge your content (see the examples below) and then click the **Purge** or **Purge All** button.

Purging a URL

To purge a single URL, select the domain the URL is on from the **Domain** drop down. Then, type the path to the content in the **Path** field. For example, to purge http://docs.fastly.com/example.jpg, we would select the “docs.fastly.com” domain from the drop down menu and enter `/example.jpg` as the path.



The syntax for purging a URL through the API can be found under the Purge section (/api/purge) of the Configuration API (/api/config) documentation.

Purging a Key

To purge content tagged with a specific key, type that key in the **Key** field.

Remember: To enable key purging to work, your server must also send a special response header along with the content being cached. Have your server send the `Surrogate-Key` header, with a list of tags for this URL. If you're using multiple tags, separate them with spaces.

For example, to tag a response with both the tags `foo` and `bar` you would send this header:

```
Surrogate-Key: foo bar
```

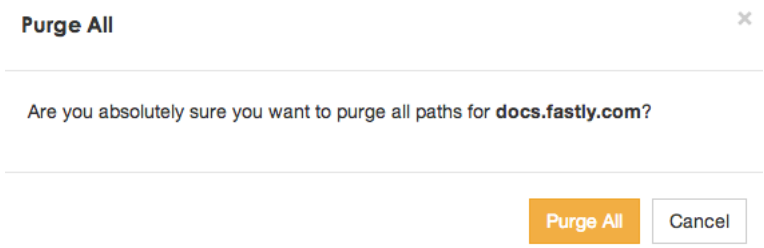
The tags sent by your server can be as simple or complex as you need, but keep in mind that:

- headers are limited to a 1024 characters (1k) total length including spaces,
- the key that exceeds the limit will be dropped instead of truncated, and
- all logical keys past the one that exceeds the limit will be dropped.

The syntax for purging content tagged with keys through the API can be found under the Purge section (/api/purge) of the Configuration API (/api/config) documentation.** **

Purging a Service

To purge all content under your service, selecting the Purge All option will clear all cached content for all domains on this service. Selecting this option displays a window asking you to confirm your decision.



The syntax for purging a service through the API can be found under the Purge section (/api/purge) of the Configuration API (/api/config) documentation.

· How do I do wildcard purges with Fastly? (/guides/purging/how-do-i-do-wildcard-purges-with-fastly)

Wildcard purging is a useful tool that 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.

Allow us to (re)introduce the idea of Surrogate keys (http://www.fastly.com/blog/surrogate-keys-part-1/). 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 through our Fastly application or API. Keep in mind, however, the purging will only apply to new objects as they're being put into the cache after you set up this configuring change, not for objects already in the cache when this configuration is being applied.

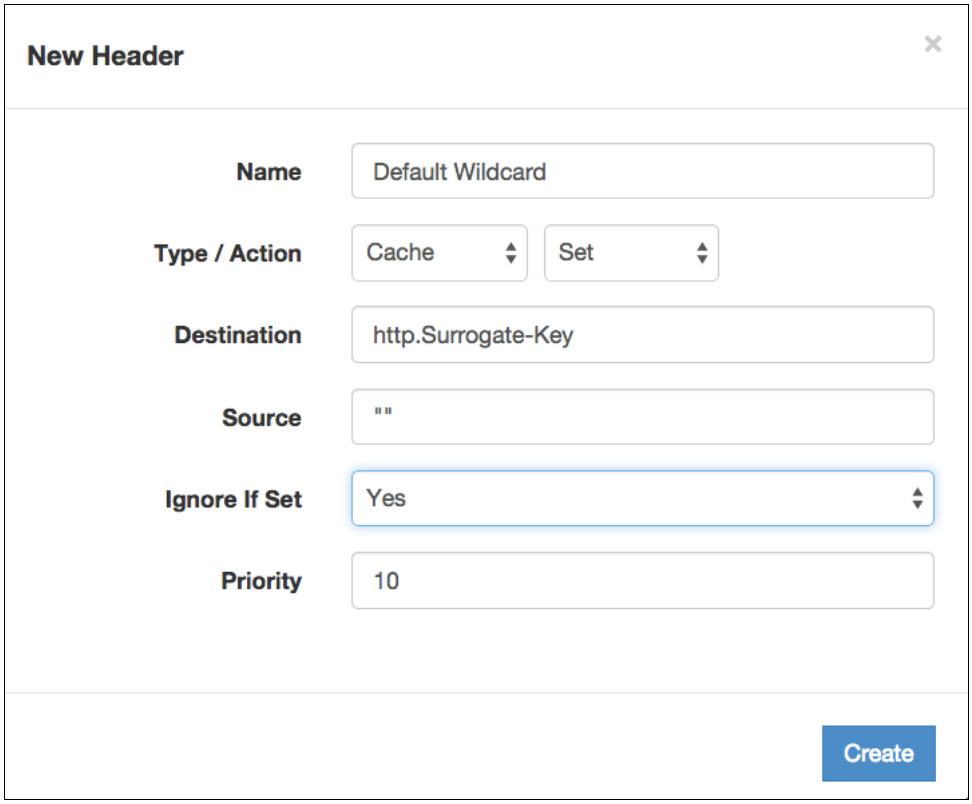
To purge content based on wildcard paths, follow the steps below.

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 application.
2. Click the **configure** tab to access the control panel.

3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **Content** section.
6. In the **Headers** area, click the **New** button to create a new header. The New Header window appears.

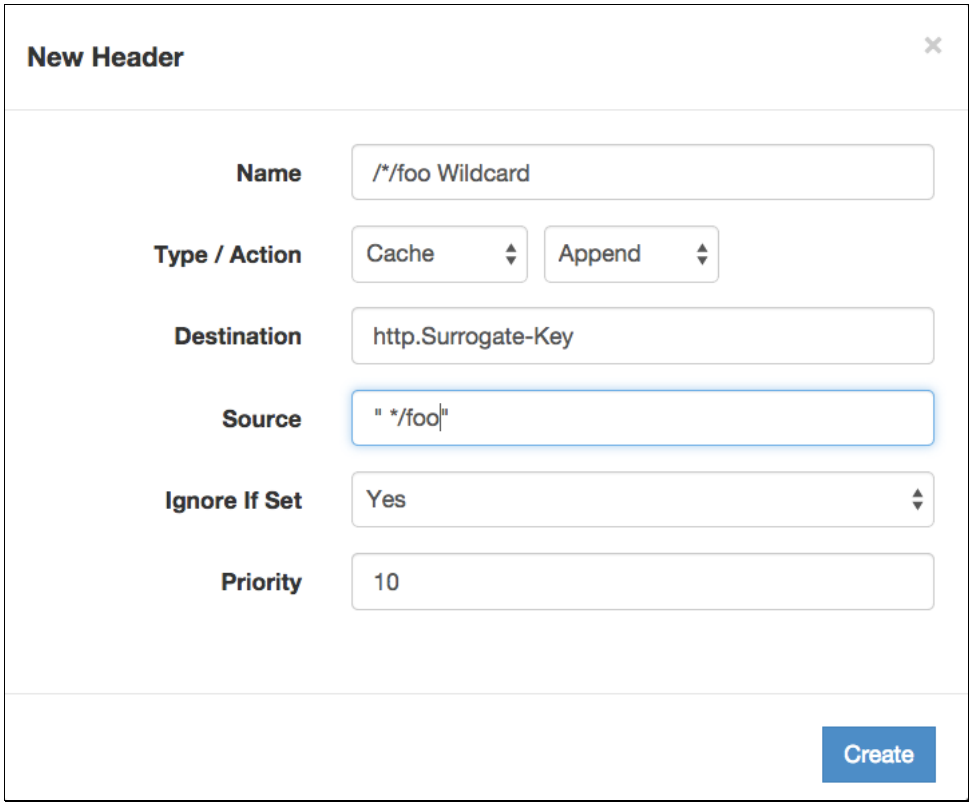


7. Fill out the **New Header** window as follows:
   - In the **Name** field, type `Default Wildcard`.
   - From the **Type/Action** drop down menus, select **Cache** and **Set**.
   - In the **Destination** field type `http.Surrogate-Key`.
   - In the **Source** field type `""`.
   - From the **Ignore if Set** menu, select **Yes**.
   - In the **Priority** field, type `10`.
8. Click the **Create** button. A new header appears in the Headers area of the Content section.

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 **New** button again to create another new header.



2. Fill out the next **New Header** window as follows:
   - In the **Name** field, type `/*/foo Wildcard`.
   - From the **Type/Action** drop down menus, select **Cache** and **Append**.
   - In the **Destination** field type `http.Surrogate-Key`.
   - In the Source field type `" */foo"`.
   - From the **Ignore if Set** menu, select **Yes**.
   - In the Priority field, type `10`.
3. Click the **Create** button. A new header appears in the Headers area of the Content section.

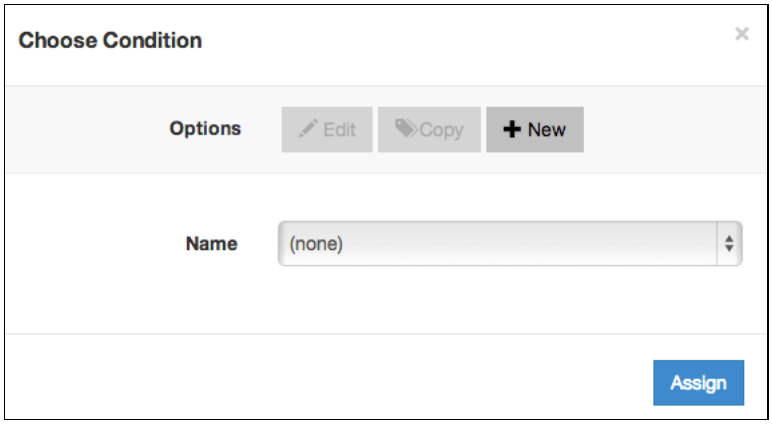
Notice the Action is set to Append to add to the default wildcard surrogate key. Also, there is a space before the asterisk in the Source field, which is important when appending multiple surrogate keys to a URL. Finally, the Priority is set to 20 so that the Default Wildcard header is executed first and then the wildcard path appends.

Create Conditions for Each Wildcard Path Being Purged

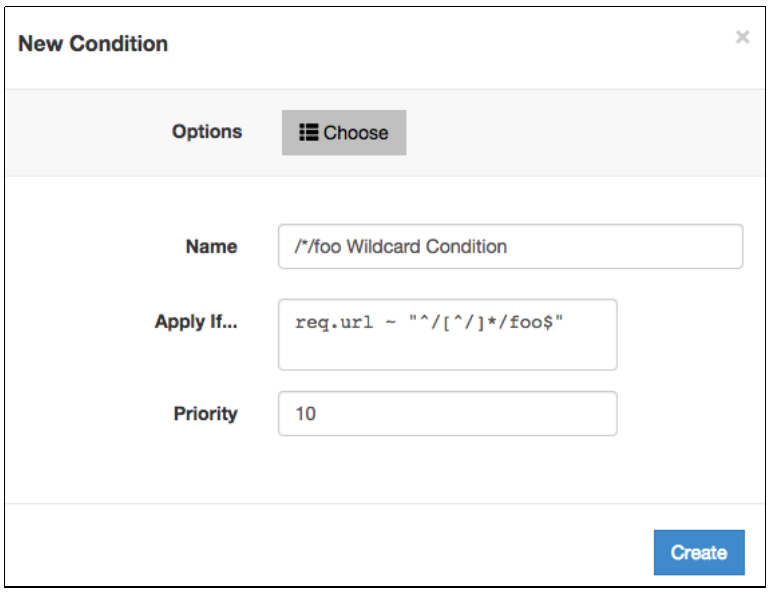
Finally, create a condition for each of the wildcard paths you need the ability to purge. Once you add the new wildcard path header, a gear icon appears to the right of its name in the Header area of the Content section.

This is where you'll create a new condition following the steps below.

1. Click the gear icon to the right of the wildcard path header name and select **Response Conditions** from the drop down menu. The Choose Condition window appears.



2. In the **Options** section, click the **New** button to add a new condition. The New Condition window appears.



3. Fill out the **New Condition** controls as follows:
   - In the **Name** field, type `/*/foo Wildcard Condition`.
   - In the **Apply If** field, type `req.url ~ "^/[^/]*/foo$"`.
   - In the **Priority** field, type `10`.
4. Click **Create** 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
- 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: `req.url ~ "/[^/]*/foo"`
- Matched Pattern: `/delta/wow/a/foo/neat/cool/img.gif`

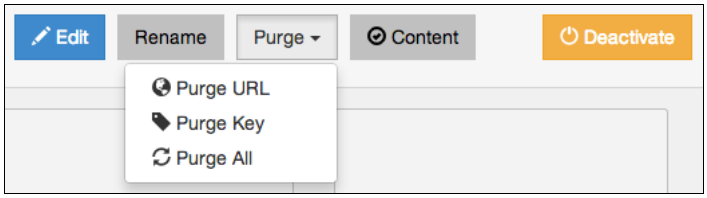
- Apply If: `req.url ~ "^/.*/foo$"`

- Matched Pattern: `/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. Select the appropriate service from the **Service** drop down.
2. Click the **Purge** button to the right of the service name and then select **Purge Key** from the drop down menu that appears.



   The Purge Key window appears.
3. In the **Key** field, type the Source name of the surrogate key you want to purge, without the quotations.

   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 Key** button.

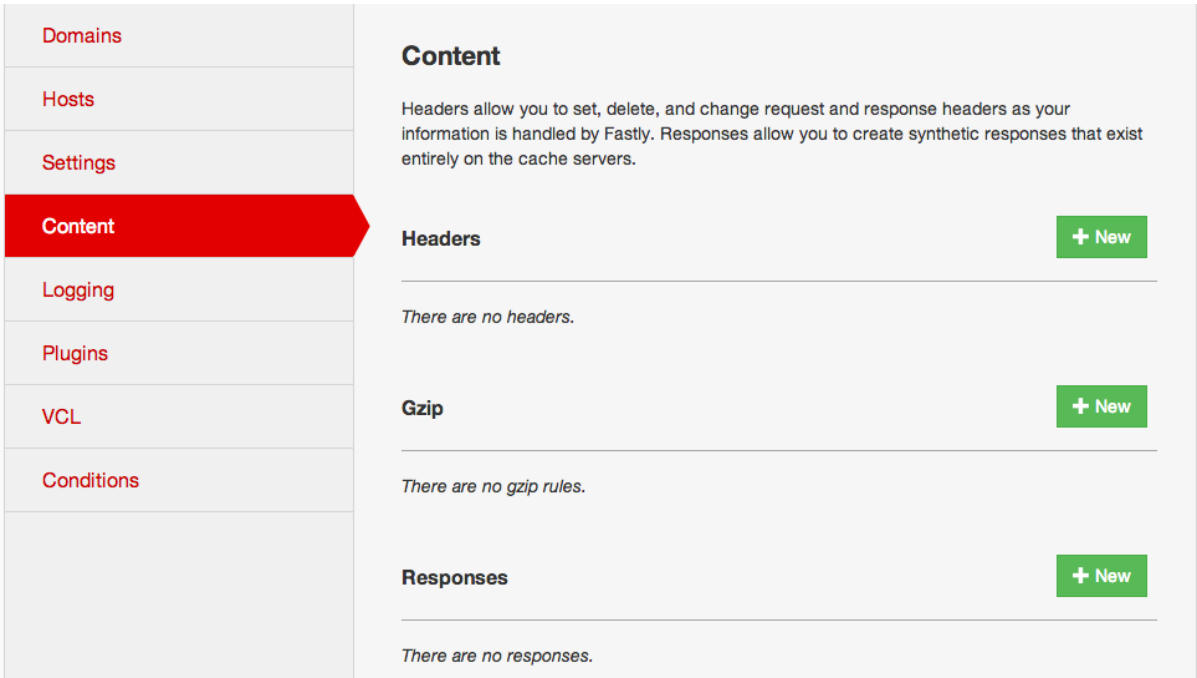
You can also use our key-based purging via the API using an HTTP request like so:

```
POST /service/<Fastly Service ID>/purge/*/foo
Fastly-Key: <Fastly Key>
```

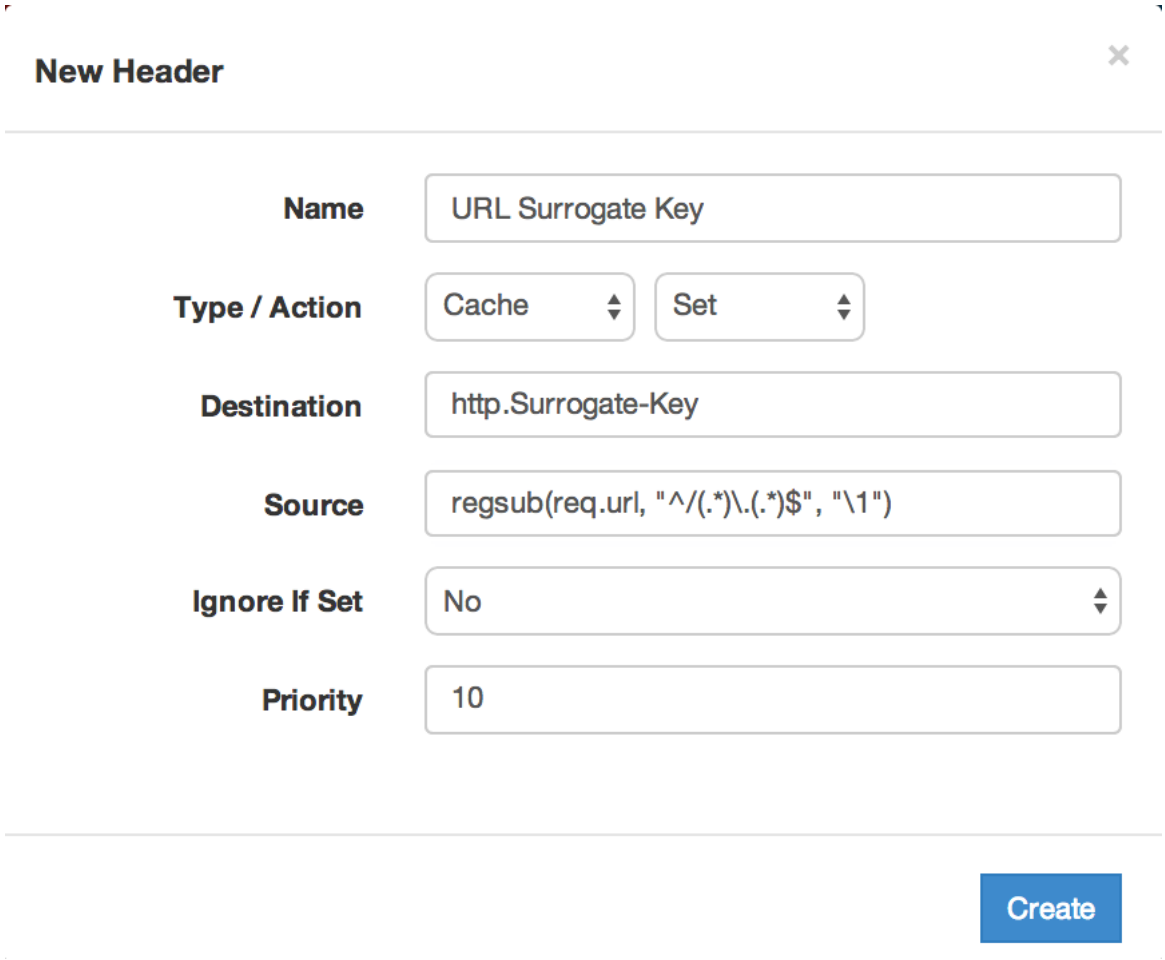
Both the UI version and the API version will purge any content that was associated with the `"*/foo"` surrogate key according to the setup in your header rules.

· How do I set a Surrogate-Key header based on the URL? (/guides/purging/how-do-i-set-a-surrogatekey-header-based-on-the-url)

To set a Surrogate-Key header based on a URL, access the Content area of the configuration settings:



Then, create a new Header (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses) via the **New Header** window:



Give the new header any name that helps you identify it properly, but be sure to select **Cache** for the **Type** and **Set** for the **Action**.

Then make sure the regex in the **Source** field appears as follows:

```
regsub(req.url, "^/(.*)\.(.*)$", "\1")
```

which will accept a URL that looks like this:

`/foo.html`

and will create the surrogate key 'foo'.

When creating a surrogate key, keep in mind that:

- headers are limited to a 1024 characters (1k) total length including spaces,
- the key that exceeds the limit will be dropped instead of truncated, and
- all logical keys past the one that exceeds the limit will be dropped.

• Domains & DNS (/guides/domains-and-dns/)

· About Domain Names and Fastly's CDN Service (/guides/domains-and-dns/about-domain-names-and-fastlys-cdn-service)

A domain name is a component of a Uniform Resource Locator (http://en.wikipedia.org/wiki/Uniform_Resource_Locator) (URL). A domain name represents an Internet Protocol (http://en.wikipedia.org/wiki/Internet_Protocol) (IP) resource and, in this context, refers to the IP address of a server computer hosting a website (http://en.wikipedia.org/wiki/Web_site), the website itself, or any other service communicated via the Internet. Customers associate their domain names with their origin (content source) when provisioning a Fastly service.

Domain names are made up of components as follows:

Domain Name Component	Example
URL	http://www.example.com/index.html
Top-Level Domain Name	com
Second-Level Domain Name	example.com
Host Name	www.example.com

Domain names are registered with a domain name registrar. Fastly is not a domain name registrar.

Fastly supports the use of multiple subdomains for the same origin server, and allows the specification of any number of subdomains for each origin. Some customers use only their second level or apex domain (/guides/domains-and-dns/can-i-use-fastly-with-a-second-level-domain) (e.g., example.com rather than www.example.com) as their canonical domain. Unfortunately, due to limitations in the DNS specification, we don't recommend using Fastly with domains this way.

· Can I use Fastly with a second level domain? (/guides/domains-and-dns/can-i-use-fastly-with-a-second-level-domain)

Some customers use only their second level or apex domain (e.g., example.com (http://example.com/) rather than www.example.com (http://www.example.com/)) as their canonical domain. Due to limitations in the DNS specification, we don't recommend using Fastly with domains this way.

The DNS instructions in RFC1033 (http://www.ietf.org/rfc/rfc1033.txt) require the root domain to be an "A Record," not a CNAME. This means that with most DNS providers you can set up a subdomain CNAME to point to a service like Amazon Web Services (AWS), but you cannot set up your root domain as a CNAME. You also cannot reliably point your root A Record to an IP address within the AWS network because Amazon reserves the right to reallocate the IP address dedicated to your instance.

Also, the DNS instructions in RFC1034 (https://www.ietf.org/rfc/rfc1034.txt) (section 3.6.2) state that, if a CNAME RR is present at a node, no other data should be present. This ensures that the data for a canonical name and its aliases cannot be different. Setting a CNAME for a root domain record (e.g., example.com (http://example.com/)) will also have ns or mx records to make it work, which breaks the “no other data should be present” rule.

In general, the problem with root domains happens when they fail to redirect to their www equivalents (example.com (http://example.com/) points nowhere instead of pointing to www.example.com (http://www.example.com/)). There are several workaround options:

- Only using Fastly for API/AJAX calls, images, and other static assets
- Having the second level domain show an IFrame to the Fastly proxied version of the site
- Redirecting from the second level domain to the Fastly proxied version (e.g., redirect any requests for example.com (http://example.com/) to www.example.com (http://www.example.com/))

None of these workarounds, however, are ideal.

Another solution exists that involves using A NAMES, a proprietary solution from DNS Made Easy (http://www.dnsmadeeasy.com/services/aname-records/). This solution, however, requires you to use that service to host your DNS. In addition, there are some performance tradeoffs that customers may wish to avoid.

The only way to correctly set up Fastly with a second level or apex domain involves having Dyn host your domain. In this case, special arrangements can be made to make everything work. The steps for that are:

1. Create a sub-account for Fastly in your Dyn account.
2. Give the sub-account permissions to Create GLSBs and Manage DLSBs.
3. Send the credentials to us - we can provide you with PGP keys if necessary. Feel free to contact support@fastly.com (mailto:support@fastly.com) if you have additional questions that aren't answered in our Domains & DNS guides (/guides/domains-and-dns/).

· Do you support IPv6? (/guides/domains-and-dns/do-you-support-ipv6)

Fastly currently supports IPv4 (http://en.wikipedia.org/wiki/IPv4) (Internet Protocol version 4), not IPv6 (http://en.wikipedia.org/wiki/IPv6). Feel free to contact support@fastly.com (mailto:support@fastly.com) if you have additional questions that aren't answered in our Domains & DNS guides (/guides/domains-and-dns/).

· Enabling Global POPs (/guides/domains-and-dns/enabling-global-pops)

The sun never sets on the Fastly empire, but how can you take full advantage?

Simply set your CNAME (/guides/getting-started/cname-instructions-for-most-providers) to `global.prod.fastly.net`. You'll now have access to all of our worldwide POPs (http://www.fastly.com/network/) as they come online. We don't restrict POP access. Instead, you control it.

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

Run the following command in your terminal:

```
$ host www.example-site.com
```

Your output should appear similar to the following:

```
www.example-site.com is an alias for global.prod.fastly.net.
global.prod.fastly.net is an alias for global-ssl.fastly.net.
global-ssl.fastly.net is an alias for fallback.global-ssl.fastly.net.
fallback.global-ssl.fastly.net has address 199.27.77.185
fallback.global-ssl.fastly.net has address 199.27.77.184
```

If you don't see `global.prod.fastly.net` in your output, then your CNAME isn't properly set. We have instructions for setting your CNAME (/guides/getting-started/cname-instructions-for-most-providers) 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 t (http://www.opendns.com/support/cache/)he OpenDNS Cache Check (http://cachecheck.opendns.com/).

What if I want to limit my POP use to only POPs in the United States and European Union?

You can. Simply set your CNAME (/guides/getting-started/cname-instructions-for-most-providers) to `a.prod.fastly.net` instead of `global.prod.fastly.net` and you should be set.

· How can I create multiple domains all at once? (/guides/domains-and-dns/how-can-i-create-multiple-domains-all-at-once)

It is possible to create multiple domains within in a single service programmatically using Fastly's API (/api/). Before you add domains, however, consider that it may be easier to serve multiple domains through custom VCL (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl). You'll be able to finely tune requests in this manner, and you'll be flexible to implement all the processes you could ever want.

Are there domain creation limits?

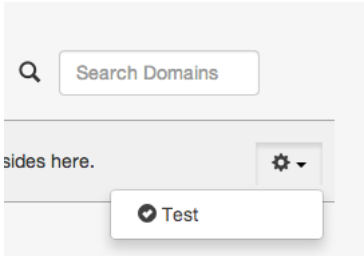
We set a limit of 20 domains per service by default. However, if you email support@fastly.com (mailto:support@fastly.com), we can work with you to set up and fine-tune domain handling in your service and we may be able to adjust this number for you.

· How can I test my Fastly setup before changing my domain? (/guides/domains-and-dns/how-can-i-test-my-fastly-setup-before-changing-my-domain)

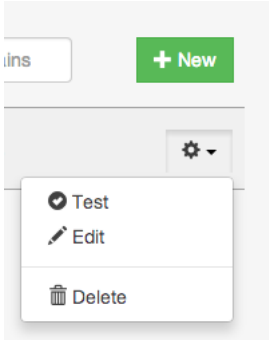
After you deploy your service, but before you change your DNS entries to send your domain to our servers, you can check via a special URL to see how your service is pulled through our network and help you identify if a problem is a DNS issue, or a Fastly configuration problem.

You can test this URL by following these steps:

1. Log in to the Fastly application.
2. Click the **configure** button (the wrench icon at the top of the window).
3. From the **Service** drop down menu, select the service you want to test.
4. Click the **Configure** button.
5. Click **Domains** to view the Domain settings.
6. In the **Domains** area click the gear icon next to the name of the domain you wish to test. One of the following will be true:
    - If the service is currently deployed, only a **Test** selection appears under the drop down menu.



- If the service is not deployed, both **Test** and **Edit** appear as selections under the drop down menu.

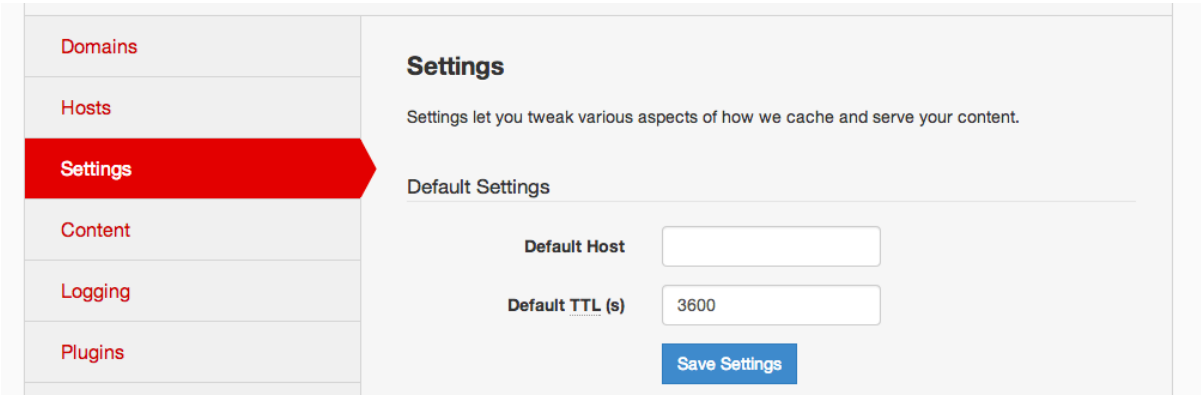


7. Select **Test** from the drop down menu. The domain you are testing will appear in a new browser window. The test URL generally appears in the format `$domain.global.prod.fastly.net`. For example, if your website were `example.com`, your test URL would appear as `example.com.global.prod.fastly.net`.

NOTE: There are a few special instances where this format is not the case. If the message "unknown domain" appears, this indicates there are no currently deployed services set up with the specified domain, and Fastly's network does not know how to route the traffic. Contact support@fastly.com (mailto:support@fastly.com) for additional help if one of these errors appears.

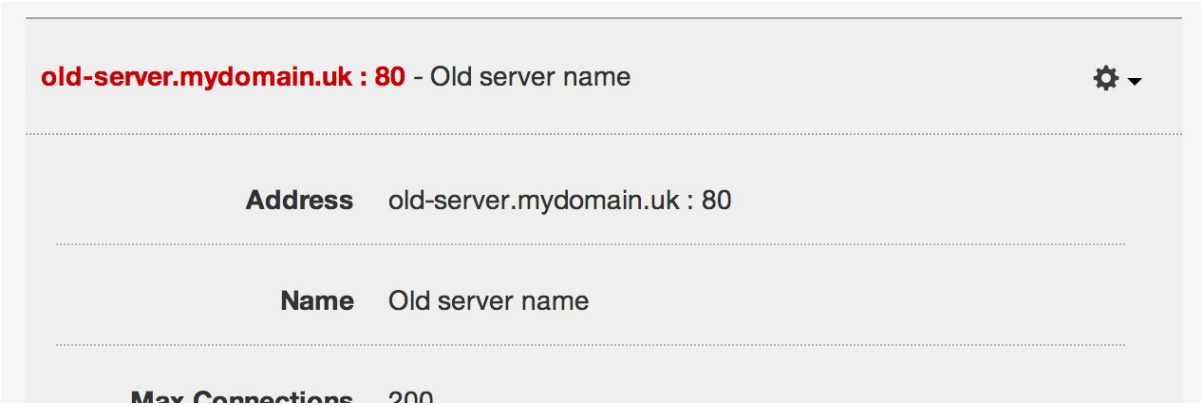
· I host multiple sites on one server. How do I specify which one to use? (/guides/domains-and-dns/i-host-multiple-sites-on-one-server.-how-do-i-specify-which-one-to-use)

Default host settings are used when responding to requests to your backend or origin server (/guides/backend-servers/). You can specify the name of your default host on the **configure** tab under the **Settings** section for a specific service.

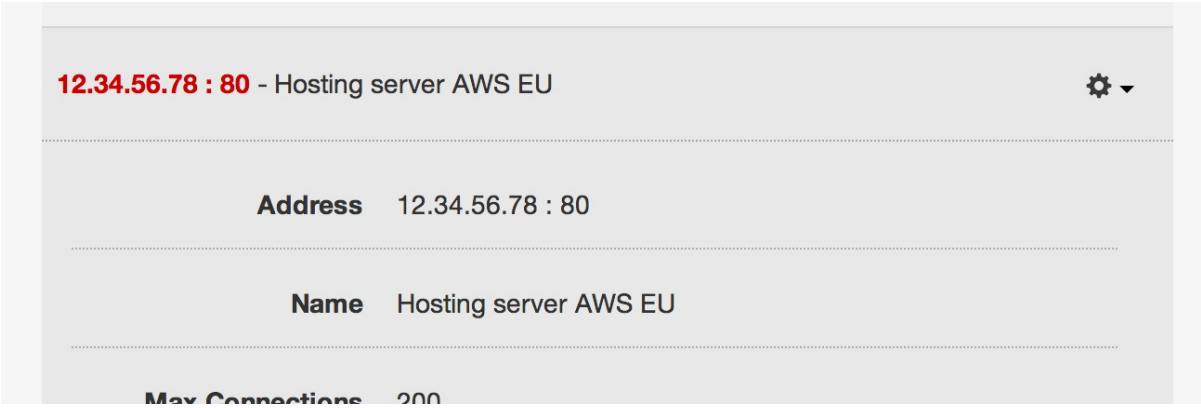


· Why am I getting an Error 1000 with my CloudFlare DNS setup? (/guides/domains-and-dns/why-am-i-getting-an-error-1000-with-my-cloudflare-dns-setup)

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 (global.prod.fastly.net) 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";
}
```

• SSL (/guides/ssl/)

· Can I use SSL to communicate to my backend servers? (/guides/ssl/can-i-use-ssl-to-communicate-to-my-backend-servers)

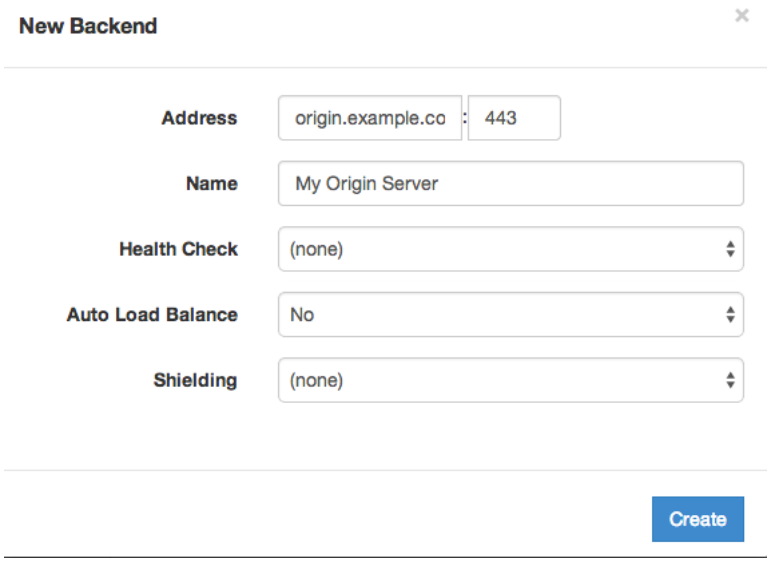
We offer two ways to use SSL to communicate to backend servers (/guides/backend-servers/).

The Easy Way

1. Log in to the Fastly application.

2. Click the **configure** tab to access the control panel.

3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **Hosts** section.
6. In the **Backends** area click the **New** button to create a new backend for your non-secure server. The New Backend window appears.



7. Fill out the **New Backend** window as follows:
   - In the **Address** field, type the address of your secure server (for example, `origin.example.com`).
   - In the **Port** field type `443`.
   - In the **Name** field, type the name of your server (for example, `My Origin Server`).
   - Leave the **Health Check**, **Auto Load Balance**, **Weight**, and **Shielding**. controls set to their default values.
8. Click the **Create** button. The server appears in the Backends area.

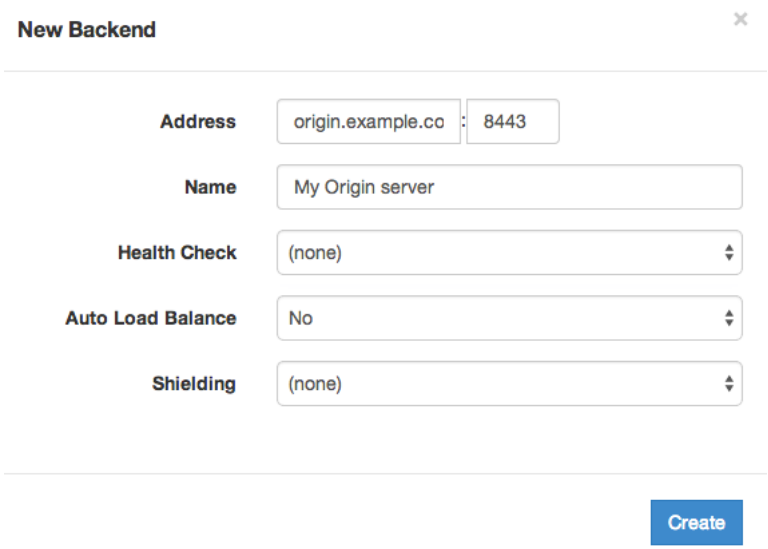
For port 443, and only port 443, we will in our system enable SSL automatically. No other configuration is needed. If you find that setting the port specifically to 443 doesn't produce the results you're hoping for or simply isn't working, consider trying the more complicated settings detailed below.

The Swiss Army Knife Way

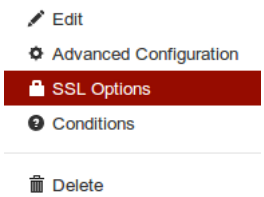
To use a different port, or for a little more control over the SSL connections, there are the SSL Options.

1. Log in to the Fastly application.
2. Click the **configure** tab to access the control panel.

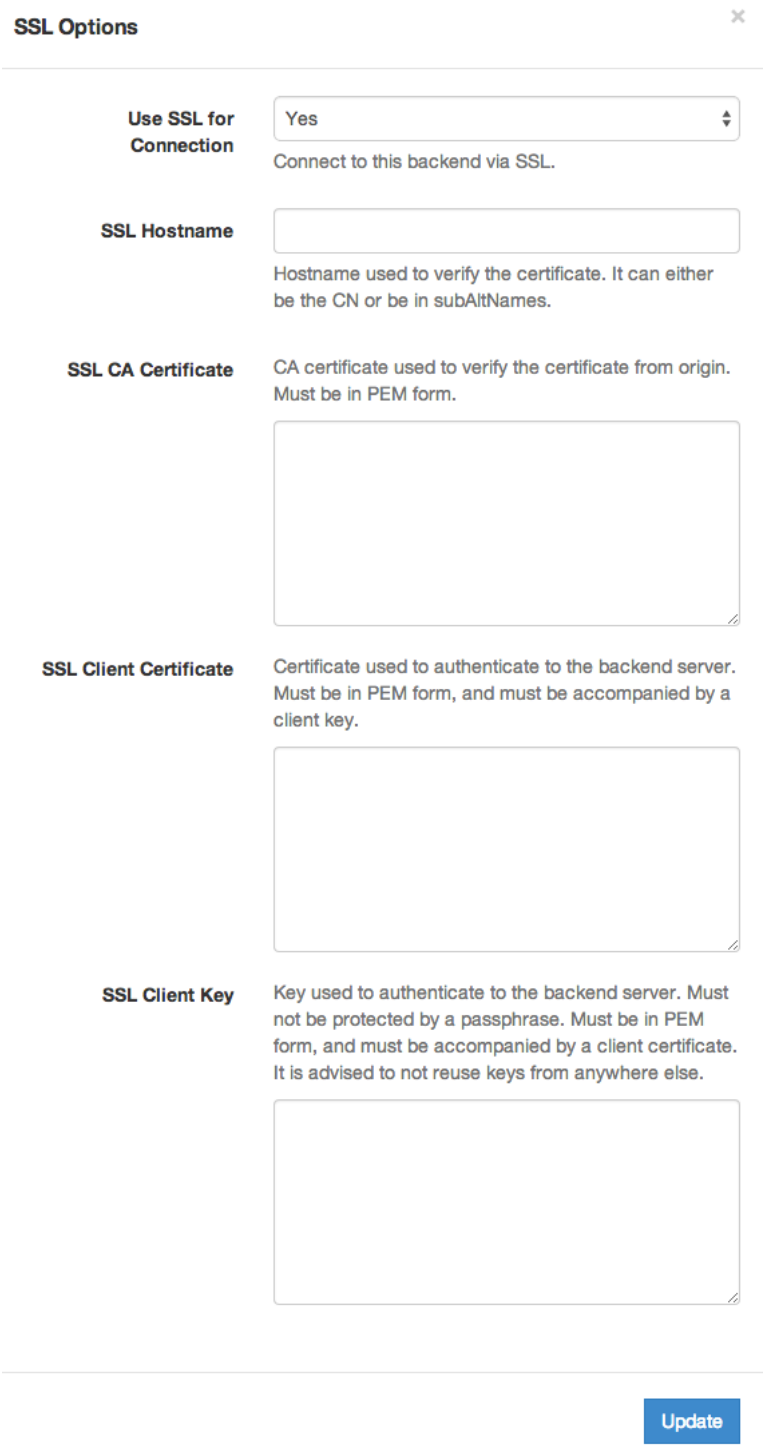
3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **Hosts** section.
6. In the **Backends** area click the **New** button to create a new backend for your non-secure server. The New Backend window appears.



7. Fill out the **New Backend** window as follows:
   - In the **Address** field, type the address of your secure server (for example, `origin.example.com`).
   - In the **Port** field type `443`.
   - In the **Name** field, type the name of your server (for example, `My Origin Server`).
   - Leave the **Health Check**, **Auto Load Balance**, **Weight**, and **Shielding**. controls set to their default values.
8. Click the **Create** button. The server appears in the Backends area.
9. Click the gear icon next to the Backend you just created.
10. From the menu that appears, select **SSL Options**.



The SSL Options window appears.



11. From the **Use SSL for Connection** menu.
12. Select **Yes** and then click **Update**.

And that's all you really 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 SSL Hostname

Normally we check the server certificate against the hostname part of the backend address. If you would like us to verify the certificate using a different hostname, you can enter it here.

This hostname will also be sent to the server in the TLS handshake, so if you are using Server Name Indication (http://en.wikipedia.org/wiki/Server_Name_Indication) (SNI) to put multiple certificates on your backend, this will select which one is used.

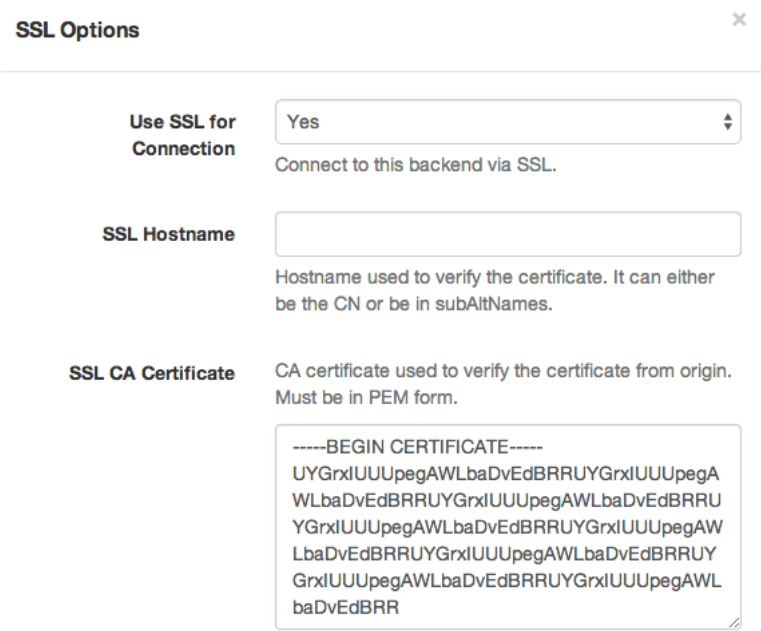
Specifying an SSL CA Certificate

If you're using a certificate that is either self-signed, or signed by a Certificate Authority that is not commonly recognized by major browsers (and unlikely to be in the Ubuntu bundle that we use), you can provide the CA certificate here.

This certificate must be in PEM form, which 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-----
```


You can just paste this into the text box, like so:



Specifying an SSL Client Certificate & Key

If you want to make sure the incoming SSL connections on your backend are coming from Fastly, and not some random person making requests directly to your backend, you can set up your backend to verify the client using a client certificate. For the Fastly side of things, all you have to do is paste the certificate and private key in PEM form into the appropriate text boxes:

SSL Client Certificate

Certificate used to authenticate to the backend server. Must be in PEM form, and must be accompanied by a client key.

```
UYGrxIUUUpegAWLbaDvEdBRRUYGrxIUUUpegA
WLbaDvEdBRRUYGrxIUUUpegAWLbaDvEdBRRU
YGrxIUUUpegAWLbaDvEdBRRUYGrxIUUUpegAW
LbaDvEdBRRUYGrxIUUUpegAWLbaDvEdBRRUY
GrxIUUUpegAWLbaDvEdBRRUYGrxIUUUpegAWL
baDvEdBRR
-----END CERTIFICATE-----
```

SSL Client Key

Key used to authenticate to the backend server. Must not be protected by a passphrase. Must be in PEM form, and must be accompanied by a client certificate. It is advised to not reuse keys from anywhere else.

```
UYGrxIUUUpegAWLbaDvEdBRRUYGrxIUUUpegA
WLbaDvEdBRRUYGrxIUUUpegAWLbaDvEdBRRU
YGrxIUUUpegAWLbaDvEdBRRUYGrxIUUUpegAW
LbaDvEdBRRUYGrxIUUUpegAWLbaDvEdBRRUY
GrxIUUUpegAWLbaDvEdBRRUYGrxIUUUpegAWL
baDvEdBRR
-----END PRIVATE KEY-----
```

Update

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 (http://httpd.apache.org/docs/2.4/ssl/ssl_howto.html#accesscontrol)
- Nginx (http://wiki.nginx.org/HttpSslModule#ssl_client_certificate)
- IIS (http://ondrej.wordpress.com/2010/01/24/iis-7-and-client-certificates/)

· How can I find the original IP when using SSL termination? (/guides/ssl/how-can-i-find-the-original-ip-when-using-ssl-termination)

As mentioned in our article "How do I tell if this is an SSL request or not? (/guides/ssl/how-do-i-tell-if-this-is-an-ssl-request-or-not)," Fastly uses a SSL terminator (http://en.wikipedia.org/wiki/SSL_termination_proxy), separate from the caching engine for performance. This means, however, that the engine overwrites the original IP 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.

VCL

If you're using shielding, you should use should use `req.http.Fastly-Client-IP`.

If you're not using shielding, or if you're building an ACL, you should use `client.ip`.

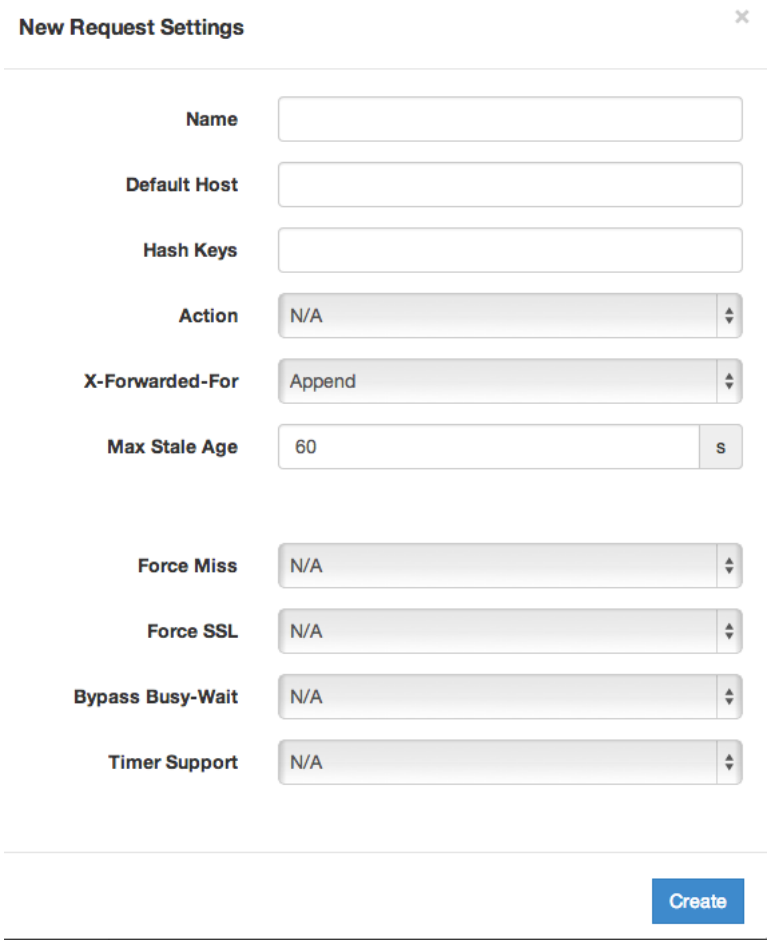
Other

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.

· How do I force unencrypted requests to use SSL? (/guides/ssl/how-do-i-force-unencrypted-requests-to-use-ssl)

If you want to only allow SSL on your site, we have you covered. There is a switch built into the "Request Settings" that will allow you to force unencrypted requests over to SSL. It works by returning a **301 Moved Permanently** response to any unencrypted request, which redirects to the SSL equivalent. For instance, making a request for '**http**://www.example.com/foo.jpeg' would redirect to '**https**://www.example.com/foo.jpeg'.

1. Log in to the Fastly application and click the **configure** button (wrench icon).
2. From the **Service** drop down menu, select the appropriate service and then click the blue **Configure** button (with the gear icon) to the right of the service name.
3. Click the **Settings** pane and find the **Request Settings** section.
4. Click the **New** button to the right of the **Request Settings** section. The New Request Settings window appears.



5. Type a name in the **Name** field. ('Default' works fine, if it's your only one.)
6. Select **Yes** from the **Force SSL** drop down.
7. Click **Create** to save your request setting changes.

Once you deploy your changes, you're done!

For more information about SSL-related issues, see our SSL guides (/guides/ssl/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

· How do I order a paid SSL option or set up Free Shared Domain SSL for my site? (/guides/ssl/how-do-i-order-a-paid-ssl-option-or-set-up-free-shared-domain-ssl-for-my-site)

Ordering a Paid SSL Option from Fastly

To order one of Fastly's SSL paid service options, simply send email to support@fastly.com (mailto:support@fastly.com) or open a ticket in Zendesk via the support tab in the Fastly application and we'll walk you through the ordering and setup process. You can read more about about our SSL options (/guides/ssl/which-ssl-options-are-available-and-how-much-do-they-cost) to determine which of our SSL options is right for you.

Setting Up Fastly's Free Shared Domain SSL for Your Site

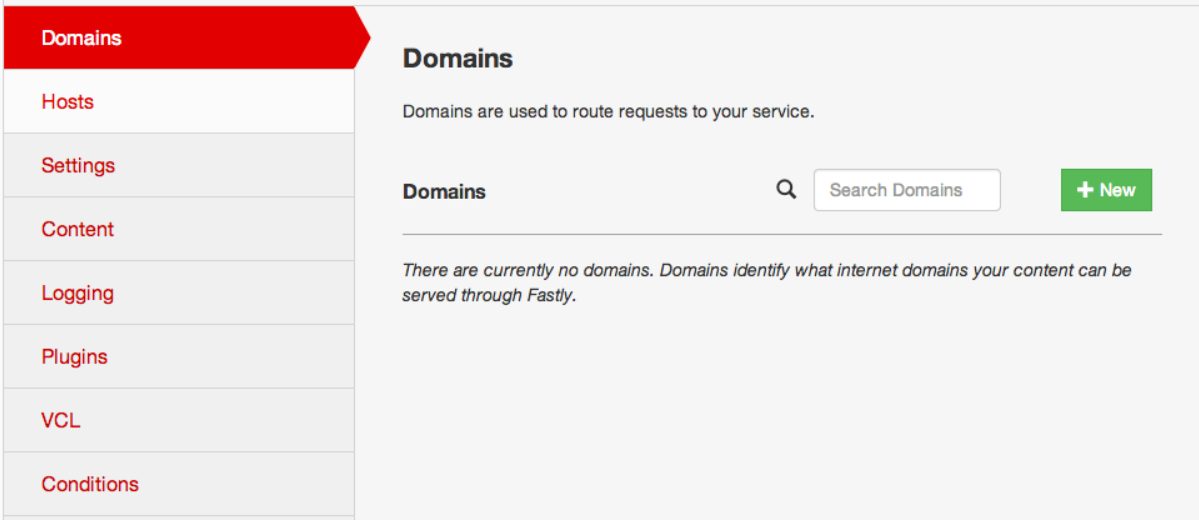
To use Fastly's free shared domain SSL, follow the steps below.

Log in to the Fastly application and under your service add a new domain called `<name>.global.ssl.fastly.net`. (**Important:** `<name>` can only be a single word. You cannot use dot-separated name such as `www.domain.com.global.ssl.fastly.net` because multiple nesting in SSL certificates is not supported.)

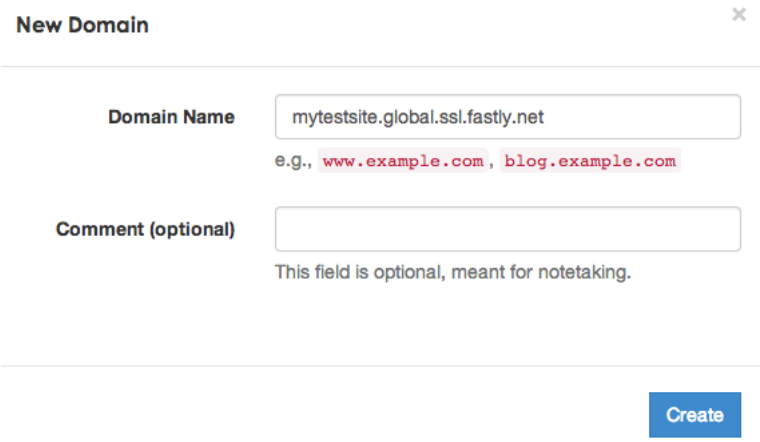
Deploy your service changes and you should be able to access the domain via the URL:

`https://<name>.global.ssl.fastly.net/`

For example, in the Domains area of the configuration settings, you'll see a New button to create a new domain:



Clicking on the New button will make the New Domain window appear:



Type the Domain Name you wish to create. If it's already taken you will need to pick a different one. When done click the Create button and then remember to deploy your changes. You can now access your site via the URL:

`https://mytestsite.global.ssl.fastly.net/`

Please note that if you DNS alias your own domain (`www.mytestsite.com`) to that name (`mytestsite.global.ssl.fastly.net`) an SSL name mismatch warning will appear in the browser. The only way to fix the mismatch is to add your domain to our SSL certificate. You can find more information about that on our SSL Options (/guides/ssl/which-ssl-options-are-available-and-how-much-do-they-cost) page.

· How do I tell if this is an SSL request or not? (/guides/ssl/how-do-i-tell-if-this-is-an-ssl-request-or-not)

To maintain optimal caching performance, Fastly uses an SSL terminator (http://en.wikipedia.org/wiki/SSL_termination_proxy) separate from the caching engine. This means, however, that the caching engine doesn't know that it was originally an SSL request. As a result, we set the `Fastly-SSL` header when fetching the content from your servers.

Because we set this header, you can check for its presence on your backend by doing something like:

```
if (req.http.Fastly-SSL) {
    set resp.http.X-Is-SSL = "yes";
}
```

and that should tell you if the request was an SSL request or not.

WordPress

If you're using Fastly SSL services with Wordpress, you'll want to add a check for the `HTTP_FASTLY_SSL` header so that WordPress can build URLs to your CSS or JS assets correctly. Do this by placing a check in your `wp-config.php` file to override the SSL flag that is checked later:

```
if( !empty( $_SERVER['HTTP_FASTLY_SSL'] ) ) {
  $_SERVER['HTTPS'] = 'on';
}
```

As usual, this must be placed anywhere before the `require_once` line with `wp-settings.php`.

For more information about SSL-related issues, see our SSL guides (/guides/ssl/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

· Which SSL options are available and how much do they cost? (/guides/ssl/which-ssl-options-are-available-and-how-much-do-they-cost)

Fastly has four SSL service options:

- Shared Domain
- Shared Certificate
- Wildcard on Shared Certificate
- Customer Certificate Hosting

SSL connections are conducted using HTTPS requests. The rate for these requests is higher than the rate for HTTP requests because they use more CPU resources.

Shared Domain

This service option uses the Fastly SAN certificate's wildcard entry for global.ssl.fastly.net. Specifically:

- Customers using this service prepend their domain name to global.ssl.fastly.net (for example, `<example domain>.global.ssl.fastly.net`), then set up a backend for that domain and use that domain name.
- More information about how to set up shared domain SSL appears on the setup page (/guides/ssl/how-do-i-order-a-paid-ssl-option-or-set-up-free-shared-domain-ssl-for-my-site).

For this option, the costs are as follows:

- The usage fee for HTTPS requests applies.
- There are no initial setup or monthly, recurring fees.

Shared Certificate

This service option uses the Fastly SAN certificate (you can read about SAN certificates here). (http://www.digicert.com/subject-alternative-name.htm) Specifically:

- You get to use your domain, but Fastly does the certificate administration.
- You provide your domain name list to Fastly and we add those names to the Certificate SAN field.

For this option, the costs are as follows:

- The usage fee for HTTPS requests applies.
- The Initial Setup fee is $500.
- A Change fee of $500 applies to later add or remove domains.
- The monthly, recurring fee is $100 for 1 domain.
- Multiple domains can be added at the same time for discounted pricing. Contact sales@fastly.com (mailto:sales@fastly.com) for current rates.

Shared Wildcard Certificate Service

This service option uses the Fastly SAN certificate (you can read about SAN certificates here). (http://www.digicert.com/subject-alternative-name.htm) Specifically:

- You get to use your domain, but Fastly does the certificate administration.
- Your provide one or more wildcard domain name entries to Fastly and we add those names to the Certificate SAN field.

For this option, the costs are as follows:

- The usage fee for HTTPS requests applies.
- The Initial Setup fee is $500.
- A Change fee of $500 applies to later add or remove entries.

- The monthly, recurring fee is $275 for 1 wildcard domain.
- Multiple domains can be added at the same time for discounted pricing. Contact sales@fastly.com (mailto:sales@fastly.com) for current rates.

Customer Certificate Hosting

This service option uses the customer's SSL certificates, which includes Extended Verification (EV) certificates. Specifically:

- You provide the certificates to Fastly.
- Fastly installs them into the caches and allocates IP addresses on each cache.
- Fastly creates a new (customer-specific) DNS Global Domain Map that associates the certificate with the allocated IP addresses.
- Fastly maintains the domain map and the certificate on the caches.

For this option, the costs are as follows:

- The usage fee for HTTPS requests applies.
- The Initial Setup fee is $2000.
- A Change fee of $1200 applies to later add or remove certificates.
- The monthly, recurring fee is $1,500 for one certificate.

Note: If you present multiple certificates at one time – such that all Certificates can be loaded into the Caches at the same time – then you will incur a single Setup fee for those certificates. You will be charged the full monthly fee for the first hosted certificate, and an additional 50% of the monthly fee for each additional certificate hosted.

Ordering SSL and Getting It Set Up

Simply open a ticket in Zendesk, via the support tab or by mailing support@fastly.com (mailto:support@fastly.com), and we'll walk you through the process. Please put "SSL Certificate Request" in the subject.

Our registrar explains the shared (SAN) certificate (http://www.digicert.com/subject-alternative-name.htm) this way:

"It's basically a way to conserve IP addresses by putting multiple hostnames or domains on one certificate. There are no security implications... Addition of your name to the certificate still needs to be authorized by you."

· Why do I see SSL certificate errors when using Wget? (/guides/ssl/why-do-i-see-ssl-certificate-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 SSL 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 (https://bugzilla.redhat.com/show_bug.cgi?id=674186) or this Debian one (http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=409938). For more information about SSL-related issues, see our SSL guides (/guides/ssl/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

• Streaming (/guides/streaming/)

· Advanced Encoding Settings (/guides/streaming/advanced-encoding-settings)

Where are your ingests?

We ingest RTMP at the following locations:

- `rtmp://ash.ingest.fastly-streams.com/live/` (Ashburn)
- `rtmp://ams.ingest.fastly-streams.com/live/` (Amsterdam)
- `rtmp://sjc.ingest.fastly-streams.com/live/` (San Jose)
- `rtmp://tyo.ingest.fastly-streams.com/live/` (Tokyo)

Which audio and video codecs do you support for live streaming?

We support H.264 video and AAC or MP3 audio across all formats. HDS outputs also support Nellymoser Asao.

Keyframe intervals are extremely important for HTTP streaming formats. Segments can only be sliced on keyframes, so your chosen segment length should be divisible by your keyframe interval. By default all segment lengths are 5 seconds, which means your keyframe interval should be 1s, 2.5s, or 5s. For instance, a 4s keyframe interval would cause your segments to be 8 seconds long.

Which rendition dimensions do you support?

Renditions used for viewing video content by end users must be specifically selected. A rendition represents a version of a stream that's transcoded by Fastly. These versions are specified using numbers that indicate the vertical resolution of the video in pixels, none of which are selected by default.

Fastly supports renditions with the following dimensions: audio only, 240p, 360p, 480p, and 720p. You can add or remove renditions for a stream at any time. Popular dimensions for standard and widescreen aspect ratios include the following:

Aspect Ratio	Popular Dimensions
Standard 4:3	320 x **240** and 640 x **480**
Widescreen 16:9	640 x **360** and 1280 x **720**

Each dimension is transcoded at a standard bitrate, although custom bitrates are supported. The standard bitrates for each dimension are as follows:

Dimensions	Default Mbit Rates
720p	2.5 Mbit
480p	0.8 Mbit
360p	0.5 Mbit
240p	0.25 Mbit

The Mbit rates shown in the table below represent the bandwidth (in Megabits – not Megabytes) per second that the stream is prepared for. The amount of content transmitted per second (Megabytes here) depends entirely on the Mbit rate, not the dimensions or anything else.

Mbit Rate	Mbytes / sec	Mbytes / min	Mbytes / hour
2.5 Mbit / sec	0.3125	18.75	1,125
2 Mbit / sec	0.250	15.00	900
1 Mbit / sec	0.125	7.50	450

For more information about streaming-related issues, see our streaming guides (/guides/streaming/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

· Can I stream to multiple ingests for redundancy? (/guides/streaming/can-i-stream-to-multiple-ingests-for-redundancy)

You can send your RTMP feed to two ingests at the same time as long as you're not using the wildcard streams (/guides/streaming/what-are-wildcard-streams) feature. If your stream goes offline at one of the two ingests you specify, we'll automatically pull from the other. Whichever ingest comes online with your stream first will become the primary ingest.

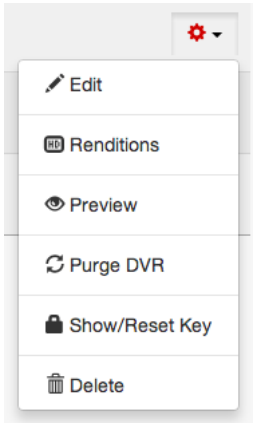
For more information about streaming-related issues, see our streaming guides (/guides/streaming/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

· How do I add or remove transcoding renditions for live streams? (/guides/streaming/how-do-i-add-or-remove-transcoding-renditions-for-live-streams)

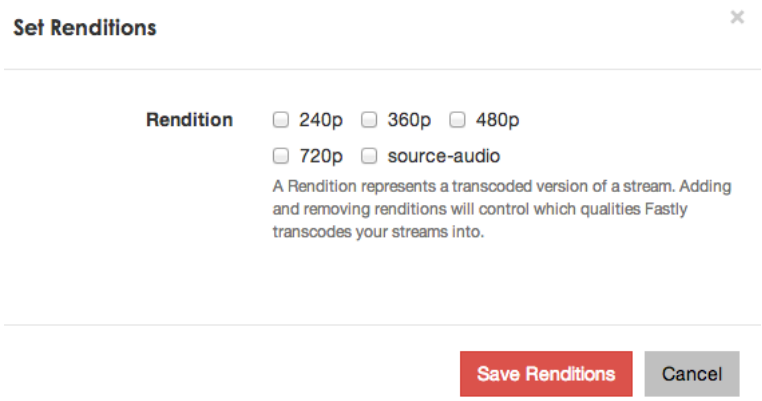
Each stream Fastly serves for you can have renditions that control the resolutions we'll transcode your streams into. There is no need to send multiple bitrates to us over RTMP. Just send the highest quality you can produce and we'll handle transcoding.

To add or remove renditions once streaming is enabled (/guides/streaming/how-do-i-enable-streaming-for-my-account) for your account, follow the steps below.

1. Log in to the Fastly application (and create a new live stream (/guides/streaming/how-to-create-edit-or-delete-a-live-stream) if you haven't already done so).
2. Find the stream you want to update.
3. Click the **gear menu icon** to the right of the stream name and select **Renditions** from the drop down menu.



The Set Renditions window appears:



4. Select the resolutions for the stream and then click **Save Renditions**.

Once you have this set up, make sure you allow ~1 minute for the transcodes to start after you start publishing your stream.

· How do I enable streaming for my account? (/guides/streaming/how-do-i-enable-streaming-for-my-account)

To enable streaming for your account simply contact sales@fastly.com (mailto:sales@fastly.com) and request the service. It only takes a few minutes for us to set things up so that you can get started.

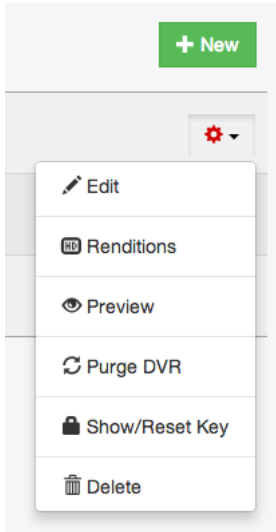
For more information about streaming-related issues, see our streaming guides (/guides/streaming/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

· How do I publish a live stream? (/guides/streaming/how-do-i-publish-a-live-stream)

To publish a live stream you'll need to enable streaming (/guides/streaming/how-do-i-enable-streaming-for-my-account) on your account and then point your encoding software at one of our ingest servers. We ingest RTMP at the following locations:

- `rtmp://ash.ingest.fastly-streams.com/live/` (Ashburn)
- `rtmp://ams.ingest.fastly-streams.com/live/` (Amsterdam)
- `rtmp://sjc.ingest.fastly-streams.com/live/` (San Jose)
- `rtmp://tyo.ingest.fastly-streams.com/live/` (Tokyo)

Our ingest servers authenticate publishers by using a stream key. Find your stream key by first finding your stream on the streaming window, and then clicking the gear icon to the right of the stream name. The Show/Reset Key selection displays your stream key and allows you to reset your key, which is the equivalent of changing your password for streaming.



In order to use your key, you must append it to the stream name. For example, if you were pointing your encoding software to our Ashburn ingest server, your full RTMP URL would look similar to this:

`rtmp://ash.ingest.fastly-streams.com/live/yourstreamname?key=YPXDaFYomBHUymvOkYeyvVsuvBpjUrNVLGVthWrEOlFezOSTzx`

If your encoder separates the stream name and RTMP server, break them out like this:

Server:

`rtmp://ash.ingest.fastly-streams.com/live/`

Stream:

`yourstreamname?key=YPXDaFYomBHUymvOkYeyvVsuvBpjUrNVLGVthWrEOlFezOSTzx`

If your stream key is not present or invalid, our servers will respond with `NetStream.Publish.AuthenticationFailed`.

For redundancy, you can publish to two ingests simultaneously as long as you're not using the wildcard streams (/guides/streaming/what-are-wildcard-streams) feature. Whichever ingest comes online with your stream first will become the primary.

· How is streaming bandwidth billed? (/guides/streaming/how-is-streaming-bandwidth-billed)

Streaming is generally billed based on bandwidth used (measured in Gbytes) and number of requests, just like content delivery. All pricing rates appear on the pricing page of our website (http://www.fastly.com/pricing/).

We offer discounts for larger pricing plans, for commitments of 12 months or longer, and for larger customers. For some customers we can also bill for bandwidth on the 95th percentile. Please contact sales@fastly.com (mailto:sales@fastly.com) for additional information.

· How much do Fastly's streaming services cost? (/guides/streaming/how-much-do-fastlys-streaming-services-cost)

Prices for Fastly's streaming delivery (as well as content delivery and SSL service options (/guides/ssl/which-ssl-options-are-available-and-how-much-do-they-cost)) are detailed on the Fastly website pricing page (http://www.fastly.com/pricing/). Please note that Fastly does not provide object storage (/guides/faqs/can-fastly-host-my-content). You must use a third-party provider (e.g., DreamObjects (http://www.dreamhost.com/cloud/dreamobjects/), Amazon S3 (http://aws.amazon.com/s3/)) and we can use those object stores as origin (/guides/backend-servers/) for your content.

Do you have other questions about pricing? Please feel free to contact sales@fastly.com (mailto:sales@fastly.com) for more information.

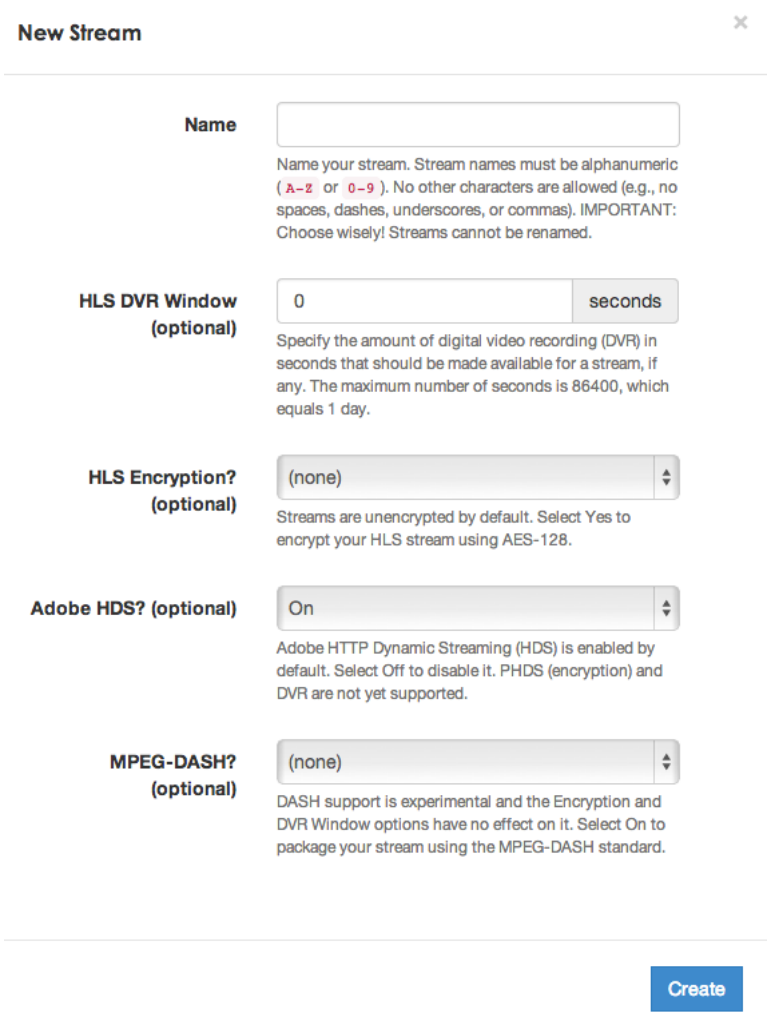
· How to create, edit, or delete a live stream (/guides/streaming/how-to-create-edit-or-delete-a-live-stream)

Once streaming has been enabled for your account (/guides/streaming/how-do-i-enable-streaming-for-my-account), you can create new, live streams or edit existing ones in just a few simple steps.

Creating a new live stream

To create a new live stream, follow the steps below.

1. Log in to the Fastly application.
2. Click on the **streaming** button (the film icon at the top of the application window). The Streaming overview page appears.
3. Click the **New** button at the right of the page. The New Stream window appears.



4. In the **Name** field, type the name of the stream you wish to create.

   IMPORTANT: Choose the name of your stream carefully. Once the stream has been created, the name cannot be edited. The name must be all lowercase alphanumeric characters (only letters and numbers). No other characters are allowed.

5. Decide whether or not to specify the amount of digital video recording (DVR) that should be made available for a stream by typing the amount, in seconds, in the **HLS DVR Window **field. Digital recording is optional and turned off by default. Specify DVR amounts in seconds. The maximum number of seconds is 86400, which equals 1 day.

6. Decide whether or not your stream will be encrypted by selecting the appropriate response from the **HLS Encryption** drop down. Encryption is optional and streams remain unencrypted by default. To encrypt streams using the AES-128 standard, select Yes from the drop down.

7. Decide whether or not to disable Adobe HTTP Dynamic Streaming (HDS) by selecting the appropriate response from the Adobe HDS drop down.

   Dynamic streaming is optional and is enabled by default. To turn off dynamic streaming, select Off from the drop down. Remember: Protected HTTP Dynamic Streaming (encryption) and DVR are not yet supported.

8. Decide whether or not to package your stream using the MPEG-DASH standard by selecting the appropriate response from the **MPEG-DASH** drop down.

   MPEG-DASH is optional and turned off by default. To package streams using the experiment DASH standard, select On from the drop down. Remember: Because DASH support is experimental, the Encryption and DVR Window options have no effect on it.

9. Click the **Create** button. The system creates a live stream using the name you have selected and places it at the top of the list of streams on the Streaming page

Editing an existing live stream

To edit an existing live stream you've already created, follow the steps below. REMEMBER: Only edit streams that aren't actively streaming!

1. Log in to the Fastly application.

2. Click on the **streaming** button (the film icon at the top of the application window). The Streaming overview page appears with a list of all the streams you've created to date.

   NOTE: When you first create a stream, it appears at the top of the list of streams on the Streaming page. If you refresh the page or visit the page again later, the list of streams will appear in alphanumeric order.

3. Find the stream to you want to update.

4. Click the **gear menu icon** to the right of the stream name and select **Edit **from the drop down menu. The Edit Stream window appears:

5. Make changes as necessary to any menu or field except the **Name** field. REMEMBER: The Name field can't be changed once a stream has been created but you can always create a new stream with a new name.

6. Click the **Update** button. The system updates your stream with the information you've edited.

Deleting a live stream

To delete an existing live stream you've already created, follow the steps below.

1. Log in to the Fastly application.

2. Click on the **streaming** button (the film icon at the top of the application window). The Streaming overview page appears with a list of all the streams you've created to date.

3. Find the stream to you want to delete.

4. Click the **gear menu icon** to the right of the stream name and select **Delete **from the drop down menu.

5. On the confirmation window that appears, click the **Confirm** button to delete the stream and make it immediately unavailable.

For more information about streaming-related issues, see our streaming guides (/guides/streaming/) or contact support@fastly.com (mailto:support@fastly.com) with questions.

· What are wildcard streams? (/guides/streaming/what-are-wildcard-streams)

Fastly streaming supports a feature we call "wildcard streams" that allows you to send multiple streams without having to create each one in the user interface or API first. This feature is available by default for any stream you create.

In order to use wildcard streams, simply prepend any prefix to your stream name followed by an underscore. For example, if your stream name is `test`, you could publish streams named `test1_test`, `test2_test`, and `test3_test`, without creating additional streams in the interface.

REMEMBER: All wildcard streams and their top-level stream must be published to a maximum of two ingests.

In order to play back wildcard streams, change the name of the manifest in your edge URL. If your edge URL is:

`http://test.edge.fastly-streams.com/hls/test.m3u8`

you could play back `test1_test` at:

`http://test.edge.fastly-streams.com/test1_test.m3u8`

Don't change the domain; the domain will be that of the top level stream.

• Backend Servers (/guides/backend-servers/)

· 503 Error Explanations (/guides/backend-servers/503-error-explanations)

Varnish (/guides/vcl-extensions/guide-to-vcl), the software that runs on the Fastly CDN, will sometimes return standardized 503 responses due to various issues that can occur when attempting to fetch data from your backend origins. The following list provides the most common types of 503 responses and basic explanations for each.

503 Backend Read Error

This happens if there is a timeout issue with your backends when requesting content to be stored in the cache. It can also be due to any other network error that might cause a read to fail when our cache servers attempt to fetch content from your origins. These errors occur periodically due to transient networks issues, such as router failovers, packet loss, and origin overload.

503 Backend Write Error

Similar to "Backend Read Error," but when sending information to a backend when performing a POST via Fastly.

503 Client Read Error

Similar to "Backend Read Error" but when reading information from a client. Generally as this has to do with network issue between the client and Fastly this will not reach the client (but it can occur).

503 Connection Refused

We attempted to make a TCP connection to your origin IP and the server refused the connection. This usually occurs when a webserver is not running or a firewall is misconfigured.

503 Network Unreachable

We are unable to find a route to the given IP range. This generally occurs when routers are misconfigured or down.

503 Backend Is Unhealthy

This occurs when an edge server receives a client request and must make a request to your origin, but your custom health checks are reporting the backend as down (and thus we do not even try). We currently have a bug where this error is thrown instead of "Backend.max_conn Reached" (see below). This only happens when the maximum connections is first encountered.

503 Backend.max_conn Reached

Each backend in your Fastly service has an associated maximum number of connections. Varnish will respond with this error when it makes a request to a backend that has reached the defined limit.

503 Illegal Vary Header From Backend

A Vary header tells an intermediate cache which headers control whether or not a different version of an object is to be served. A common example is "Accept-Encoding", for each variation of the header the cache will keep a distinct copy (such as gzip vs. no gzip). This error occurs when a backend, which is responsible for letting the cache know which vary headers are accepted, returns a malformed list (e.g., incorrect whitespace, comma separation issues).

503 No Stale Object Available

The Fastly CDN can be configured to serve stale objects (/guides/caching/how-can-i-serve-stale-content-on-errors) in the event of backend failure. This error occurs when the stale cache object has expired but your backend is still failing for some reason (and thus no stale object is available).

503 Quorum Wait not Reached

This error occurs in conjunction with the "Director" object that is only available via the Fastly API. In general this error will occur when a director does not have enough backends available in the group to serve traffic based on the Director's configuration.

503 No Healthy Backends

This error also occurs in conjunction with a Director configuration object. It simply means that there are no healthy backends under the auspices of the director which handles the caching of the specified content.

503 All Backends Failed or Unhealthy

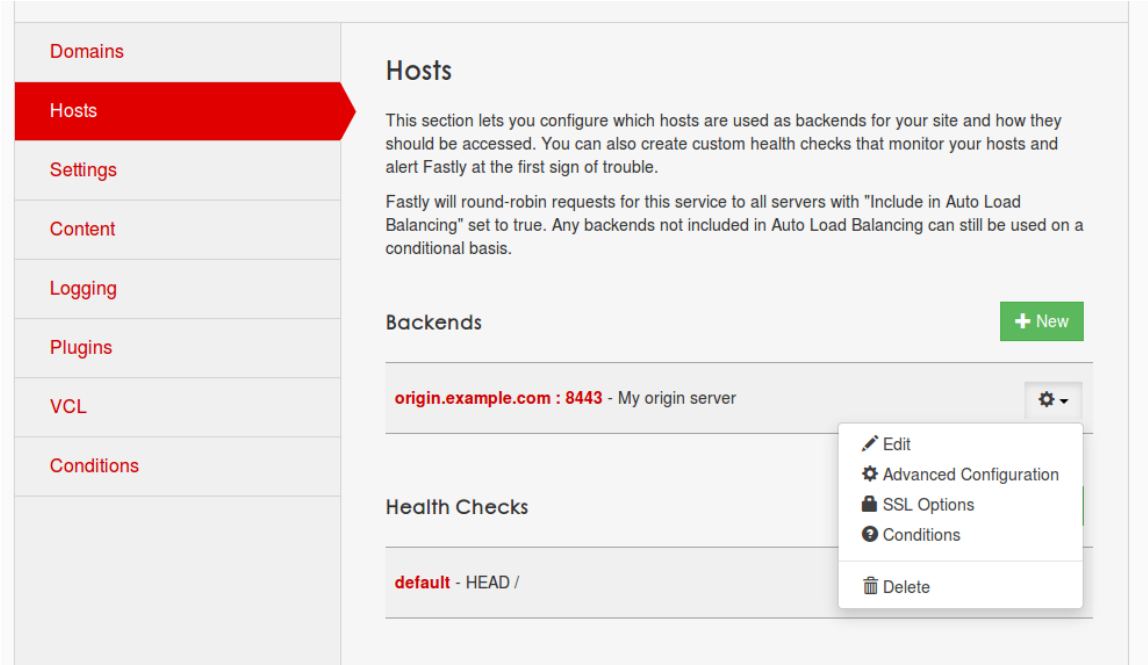
This occurs when either all the backends associated with a director are unhealthy or we tried to fetch information from multiple backends in a director and they all failed with the same error.

· Can I change the timeouts for my backend configuration? (/guides/backend-servers/can-i-change-the-timeouts-for-my-backend-configuration)

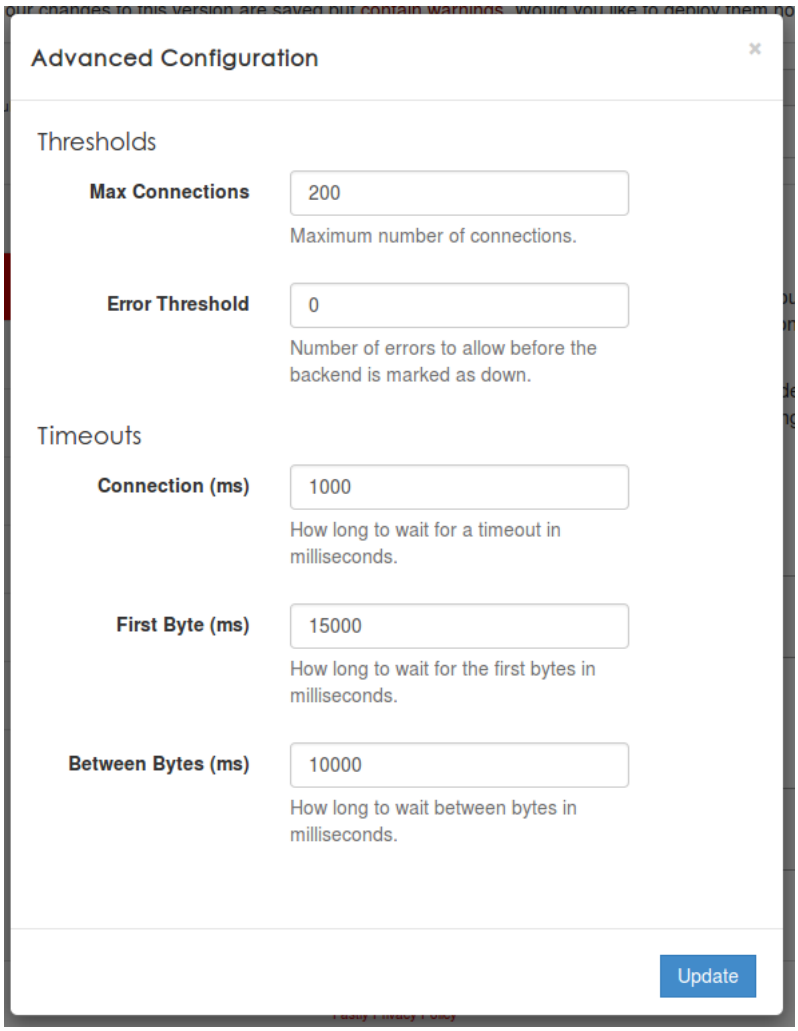
Timeouts can be changed by following the steps below.

1. Log in to the Fastly application.
2. Click on the **configure** button (the wrench icon at the top of the application window).
3. Select the appropriate service from the **Service** drop down menu.
4. Click the blue **Configure** button to the right of the displayed service name.
5. Click the **Hosts** pane from the list on the left.

6. In the list of backends, click the gear icon to the right of the named backend you wish to edit, then select **Advanced Configuration** from the drop down menu.



The Advanced Configuration menu appears.



7. Type the new timeout in the appropriate field of the **Timeouts** section.
8. Click the **Update** button.

Note: For additional information about backend configurations, see our Backend Servers Help Guides (/guides/backend-servers/) *or contact* support@fastly.com (mailto:support@fastly.com) *with questions*.

· Can I route assets to different backends based on specific criteria I supply? (/guides/backend-servers/can-i-route-assets-to-different-backends-based-on-specific-criteria-i-supply)

Some customers have assets stored on multiple backend 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 power to set conditions on their backends, which simply adds an if statement block to your VCL.

Basic Setup: Create Conditions for Each Backend

To add a condition on your backend, navigate to **Hosts** → **Backends** ****within the Fastly application. Click the gear icon next to a specific backend, and select* ***Conditions** from the menu that appears to create the new condition you wish to apply to that backend. Here are some example criteria and the conditions that would be placed in the **Apply If** field of the **New Condition** you create:

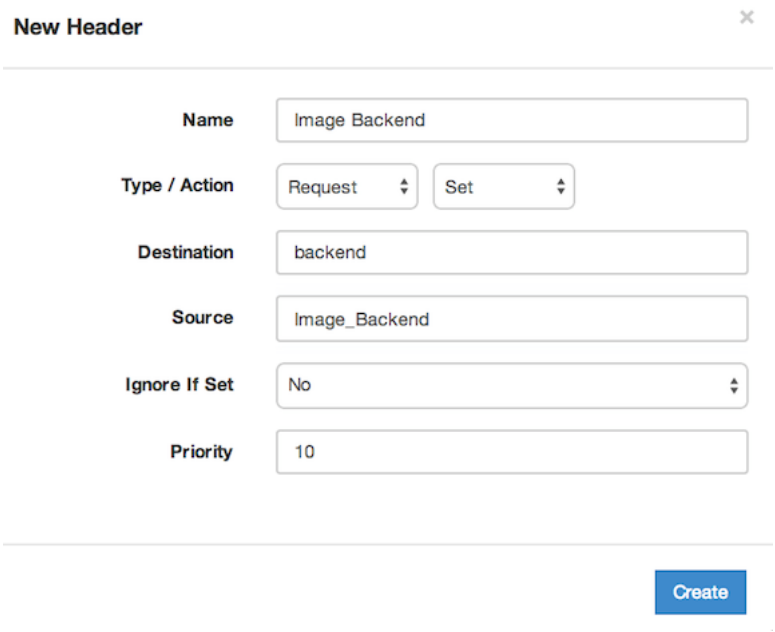
Criteria	Sample Conditions

hosts	`req.http.host ~ "www.example.com"`
content-type / url	`req.url ~ ".(jpg|png|gif)($|\?)`

Backup Setup: Create a Header

What if you have a condition already assigned to your backend? 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 backend. If you want to separate your request conditions instead of grouping them, you can use header rules to route assets to different backends instead.

Within the Fastly application, navigate to **Content** → **Headers**. Then select the **New** button to create the first header that will set the default backend.

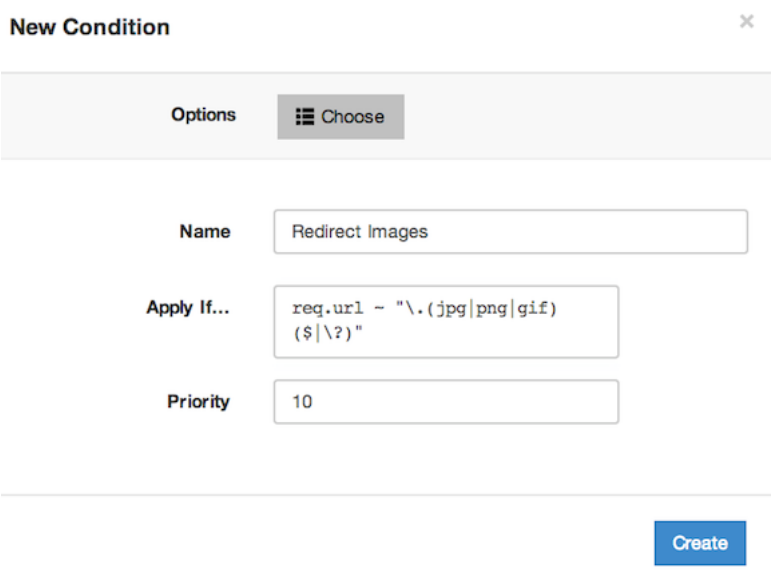


Set the first header's controls as follows:

- **Name:** `Image Backend` (or any meaningful, preferred name)
- **Type/Action:** Request/Set
- **Destination:** `backend`
- **Source:** `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.)
- **Ignore If Set:** `No`
- **Priority:** `10`

Once you've set the controls, click **Create** to add the first new header.

Next click the gear icon to add a condition. Select **New** to add a new condition.



Enter the condition settings as follows:

- **Name:** `Redirect Images` (or any meaningful, preferred name)
- **Apply if**: `req.url ~ "\.(jpg|png|gif)($|\?)"`

Once you have finished, click **Create** to assign the condition to your header. Then click the **Activate** button, near the top right corner of the window, to deploy the new version of the service you just edited.

Note: Learn more about Conditions with our Conditions Tutorial (/guides/tutorials/conditions-tutorial).

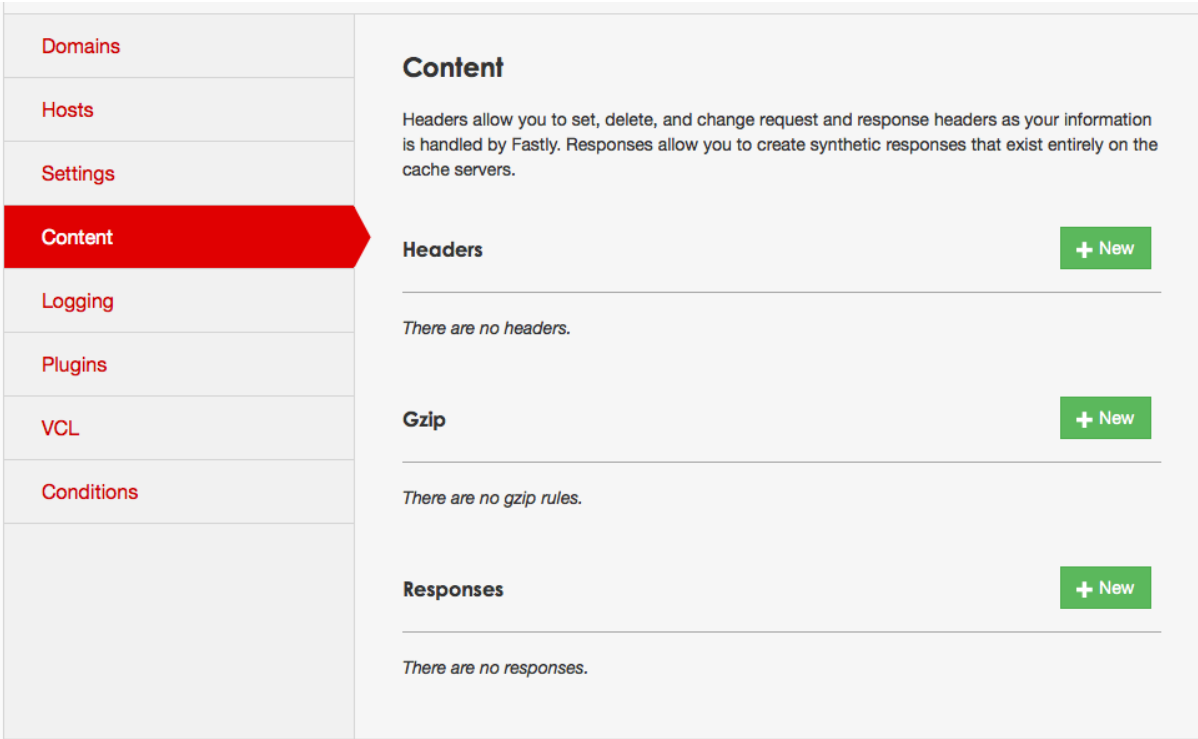
· Can I use my EC2 public domain name as the address for a backend? (/guides/backend-servers/can-i-use-my-ec2-public-domain-name-as-the-address-for-a-backend)

You can use your EC2 public domain name instead of an IP address.

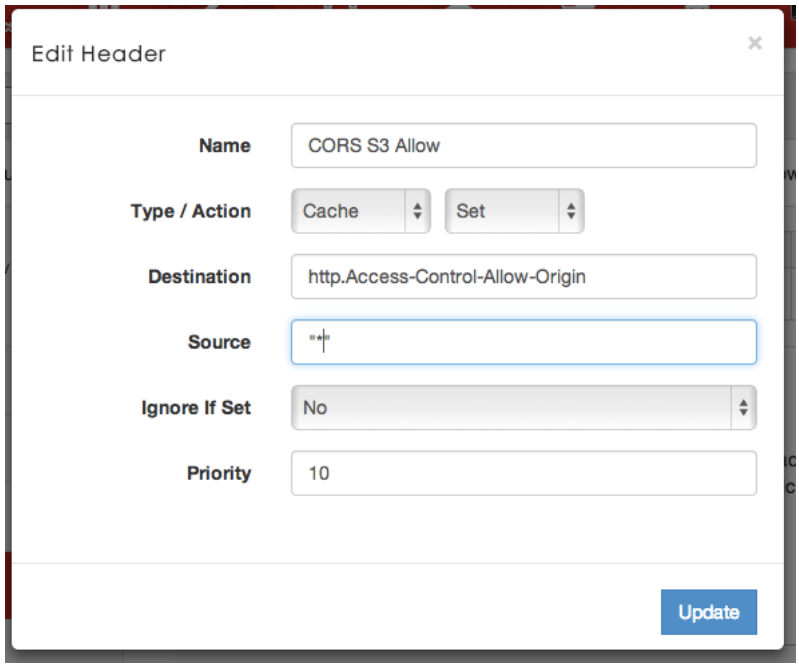
Note: *There are several other ways to set up origins detailed in our* Backend Servers Help Guides (/guides/backend-servers/).

· Enabling Cross-Origin Resource Sharing (CORS) (/guides/backend-servers/enabling-crossorigin-resource-sharing-(cors))

We recommend enabling CORS (Cross-Origin Resource Sharing (http://docs.aws.amazon.com/AmazonS3/latest/dev/cors.html)) when using Amazon S3 (/guides/backend-servers/using-amazon-s3-as-a-backend) as your backend server. To enable CORS, set up a custom HTTP header via the Content pane for your service.



Then, create a custom header with the following information that adds the required "Access-Control-Allow-Origin" header for all requests:



Test it out

Running the command `curl -I your-hostname.com/path/to/resource` should include similar information to the following in your header:

```
Access-Control-Allow-Origin: http://your-hostname.com
Access-Control-Allow-Methods: GET
Access-Control-Expose-Headers: Content-Length, Connection, Date...
```

· Generating HTTP redirects at the edge (/guides/backend-servers/generating-http-redirects-at-the-edge)

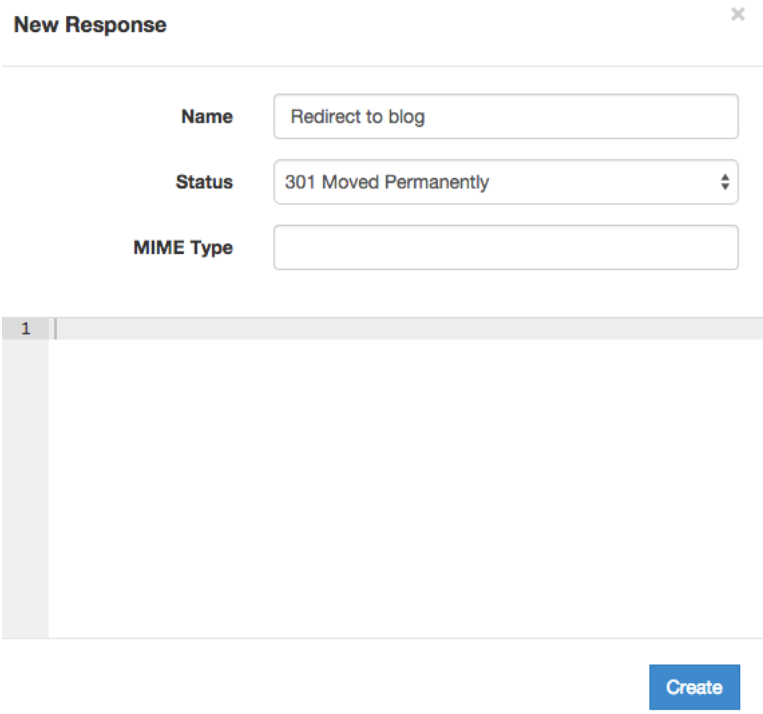
When users request information from your origin servers, you may want to redirect them for various reasons. For example, you may want to redirect them to either a secure or non-secure version of a web page, or to pages that have been moved or updated since the last time they were requested.

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.

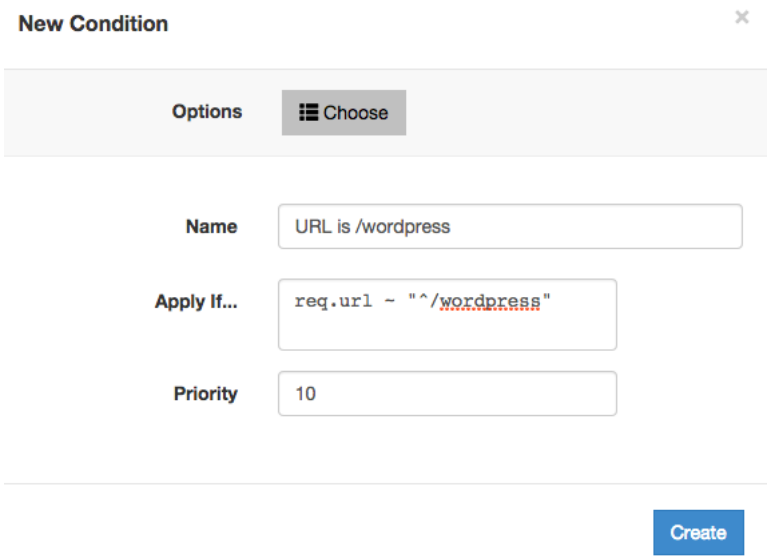
Create a New Response & Condition

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. Click the **Content** section.
2. In the **Responses** area, click the **New** button to create a new response. The New Response window appears.



3. Fill out the fields of the **New Response** window as follows:
   - In the **Name** field type a meaningful name for your response (e.g., `Redirect to blog`).
   - 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.
4. Click **Create** to create the new response.
5. Click the gear icon to the right of the new response you just created and select **Request Conditions** from the drop down menu. The Choose Condition window appears.
6. In the **Options** section, click the **New** button. The New Condition window appears.



7. Fill out the fields of the **New Condition** window as follows:
    - In the **Name** field type a meaningful name for your condition (e.g., `URL is /wordpress`).
    - 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"`).
    - Leave the **Priority** set to 10.
8. Click **Create** to create the new condition.

Create a New Header & 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. In the **Headers** area, click the **New** button to create a new header. The New Header window appears.
2. Fill out the fields of the **New Header** window as follows:
    - In the **Name** field type a meaningful name for your header (e.g., `Location for wordpress redirect`).
    - From the **Type** and **Action** drop downs, select **Response** and **Set** respectively.
    - In the **Destination** field type `http.location`.
    - In the **Source** field, type the source location of the new content (e.g., `"http://www.my-site.com/new-location/of/item"`).
    - Leave the **Ignore If Set** and **Priority** fields at their default settings.
3. Click **Create** to create the new header.
4. Click the gear icon to the right of the new header you just created and select **Response Conditions** from the drop down menu. The Choose Condition window appears.
5. In the **Options** section, click the **New** button. The New Condition window appears.
6. Fill out the fields of the **New Condition** window 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.
    - Leave the **Priority** set to 10.
7. Click **Create** to create the new condition.

· Health Checks Tutorial (/guides/backend-servers/health-checks-tutorial)

In this tutorial we will show you how to create health check that will periodically contact your backends to make sure they are still working.

We will assume that you are already accustom to editing and deploying configurations using the web-based configuration application (https://app.fastly.com). If you are not familiar with basic editing using the app please see our Help Guides (/guides/) to learn more before moving forward.

Overview

Health checks, while simple in principal, are a little more involved than most other configuration objects. Let's run through the options you have when creating a health check so you have a place to start when working through the tutorial.

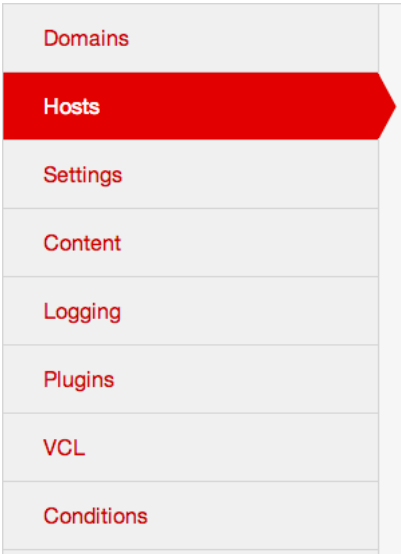
- Name - A human-readable identifier for the health check (e.g., "West Coast Origin Check")
- Request - An HTTP Verb and path to visit on your backends when performing the check
- HTTP Host Header - the HTTP Host Header to set when making the request (e.g. "example.com")
- Expected Response - The HTTP status code the backend must respond with for the check to pass (usually "200 OK")
- Check Frequency - How often the backend is checked with additional parameters that determine if the backend is up or down

The way a health check works is that we construct and then send an HTTP request using the "Request" and "HTTP Host Header" parameters of the health check. When the backend responds we check the header of the response and ensure the appropriate response code was sent back using the "Expected Response" parameter. If the backend doesn't respond after a reasonable amount of time or does not respond with the correct code the check fails.

If a certain number of checks fail over the last few checks we made then the backend is marked as down and a notification is sent. The number of checks made before we can determine status is called the "Window" and the number of checks that must pass in order for the backend to be considered "up" is called the "Threshold." The default user interface for health checks hides these messy details and gives you a few easy-to-use options along with explanations that work for most of our customers. It also allows you to customize your checks fully using the "Custom..." option of the "Check Frequency" parameter.

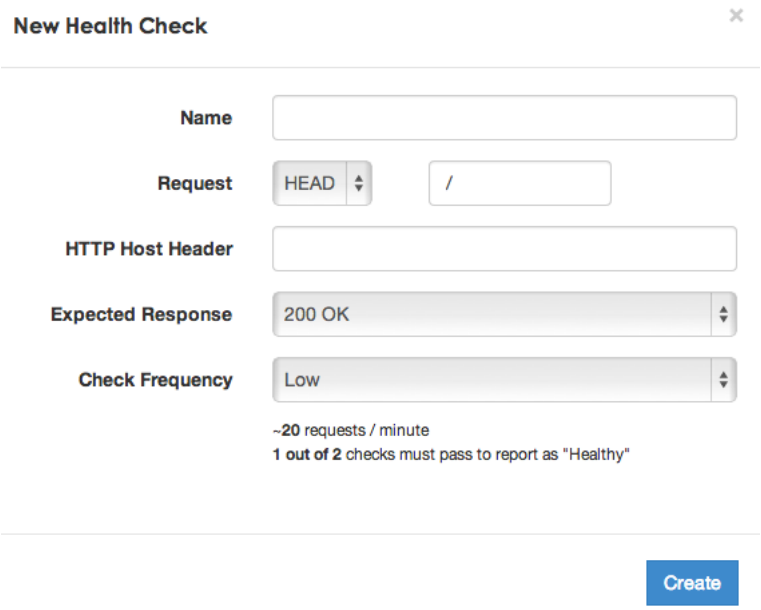
Creating a Health Check

Open up the fastly application (https://app.fastly.com) in your browser and navigate to the configuration editor for your service. Find the "Hosts" tab (previously called "Servers") on the left side navigation menu.



At the bottom you should see a section named "Health Checks" with a green button entitled "New" to the right of the title. Click the button.

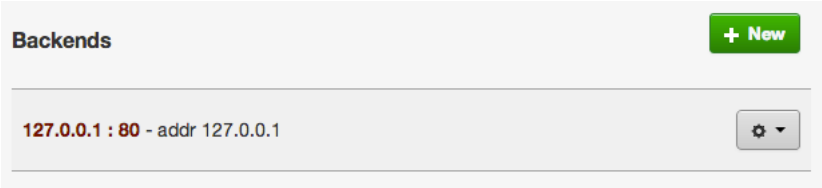
The new health check window should appear. Give your health check a name and set the appropriate path (many users simply use "/" to check the root request on a backend). The default HTTP verb used by the interface is "HEAD" but some servers do not support that verb and you will have to set the verb to something more sensible for you (such as "GET" or "POST"). Finally set the appropriate check frequency. Notice that the description at the bottom will change depending on what you select. If you are feeling particularly adventurous, you can set the check frequency to "Custom..." and fiddle with the advanced configuration parameters.



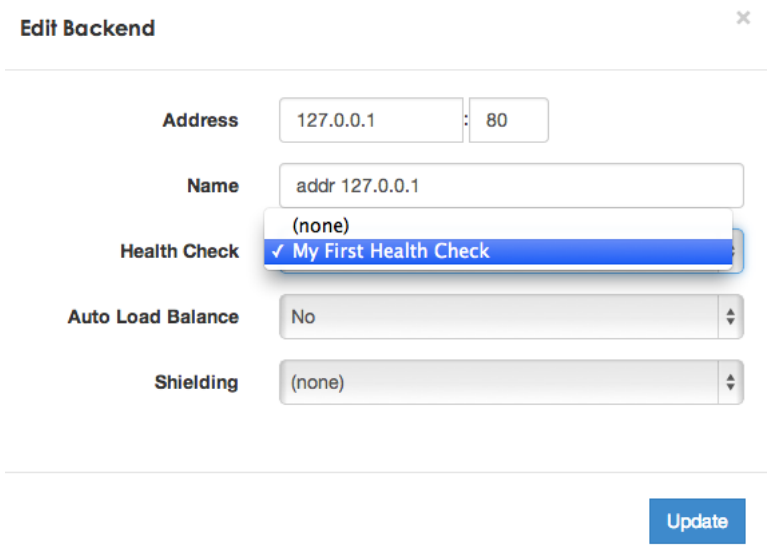
When you're all done setting up the health check object click the "Create" button and your new health check should appear in the list under the "Health Checks" heading. Health checks do nothing on their own but can be added as a special parameter to a backend in your configuration. Let's move on and see how one accomplishes this daunting task.

Assigning a Health Check

To begin find the section entitled "Backends" near the top of the body of the page.



Edit one of your existing backends by selecting the Edit option from the gear to the right of the backend name. The Edit Backend window should appear. Find the "Health Check" field and select the health check you just created from the drop down menu, then click the Update button.



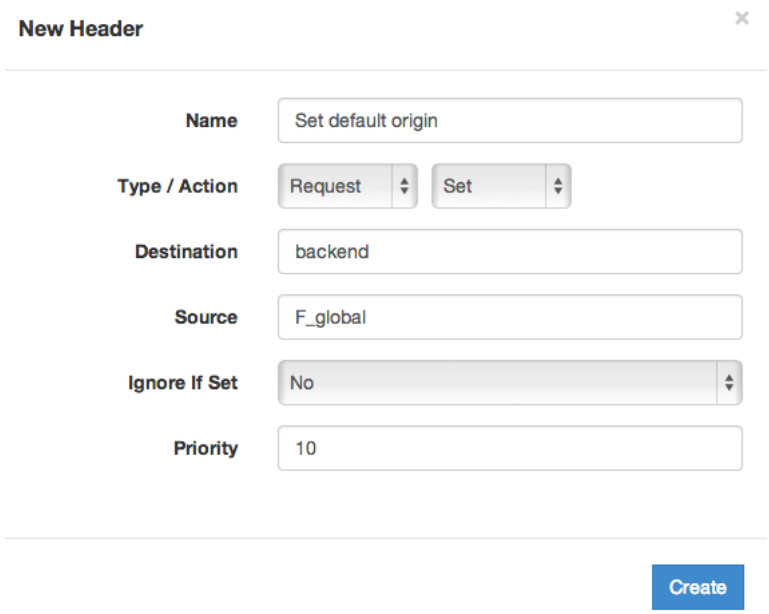
· How can I change my origin server based on a user's location? (/guides/backend-servers/how-can-i-change-my-origin-server-based-on-a-users-location)

The VCL looks something like this:

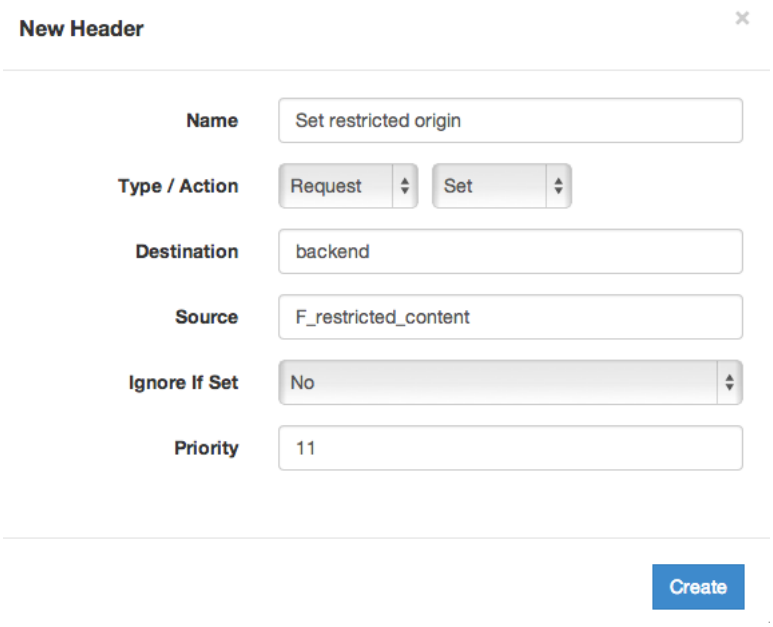
```
# default conditions
set req.backend = F_global;

# Use restricted content if the user is in Asia, France or Germany
if (geoip.continent_code == "AS" || geoip.country_code == "fr" || geoip.country_code == "de") {
      set req.backend = F_restricted_content;
 }
```

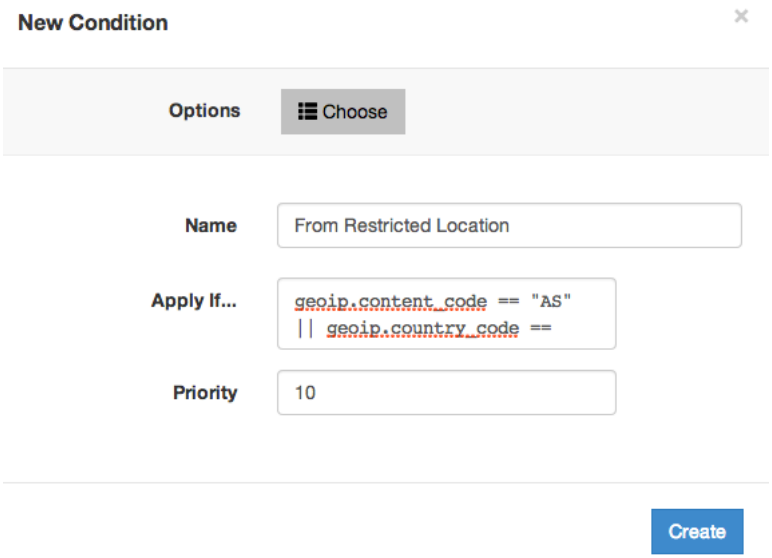
So, how does this translate to the Fastly UI? First, create a new Header (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses) under the Content pane of your configuration control panel (/guides/fastlys-web-interface/about-the-configuration-control-panel). The Type is Request and the Action is Set. Note that the priority is 10. The backend (here it's F_global should be the name of whatever your global origin server is. You can see the name if you look at your VCL).



Now create another header almost exactly the same with two differences: it sets the backend to be your restricted origin server and the priority is higher than than the previous header.



Finally, attach a condition (/guides/tutorials/conditions-tutorial) to the restricted origin header which checks the GeoIP header (/guides/vcl-extensions/geoiprelated-vcl-features):



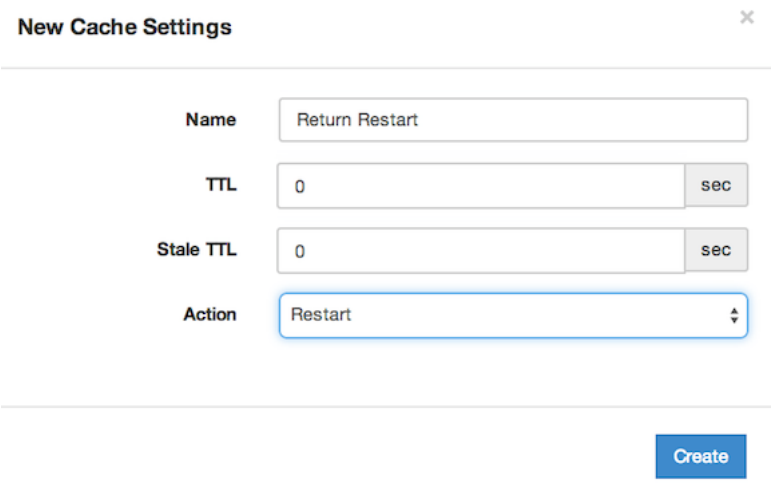
Note: There are several other ways to set up origins detailed in our Backend Servers Help Guides (/guides/backend-servers/).

· How can I check multiple backends for a single request? (/guides/backend-servers/how-can-i-check-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

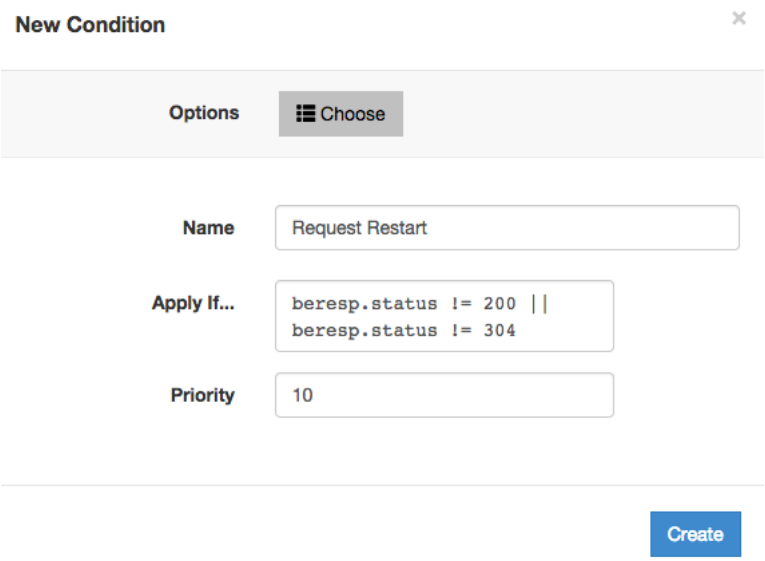
Within the Fastly application, navigate to **Settings** → **Cache Settings**. Then select the **New** button to create a cache restart within vcl_fetch.



Set the cache settings as follows:

- **Name:** `Return Restart` (or any meaningful, preferred name)
- **TTL:** `0`
- **Stale TTL:** `0`
- **Action:** `Restart`

Click **Create** to add the new cache setting (it will appear in the Cache Settings area). Then click the gear icon next to this new cache setting to add a new condition.



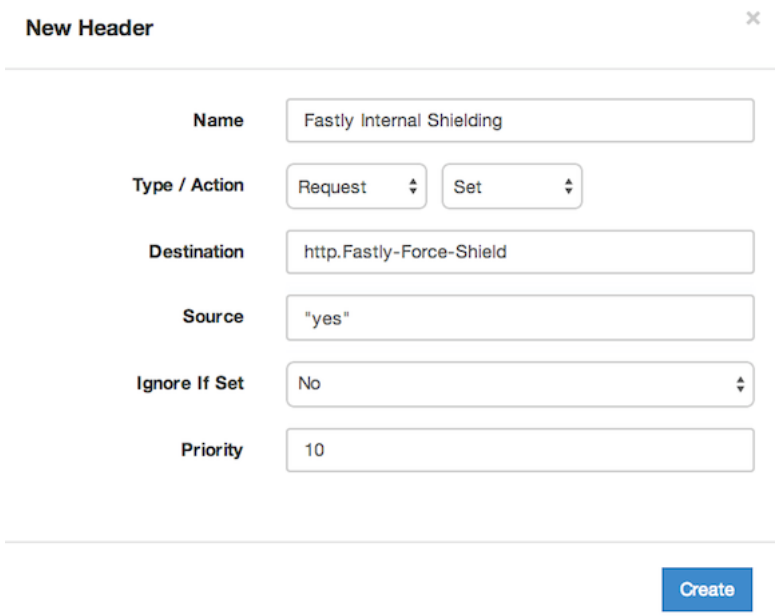
Set the new condition settings as follows:

- **Name:** `Restart Request` (or any meaningful, preferred name)
- **Apply if**: `beresp.status != 200 || beresp.status != 304`

Click **Create** to assign the condition.

Create New Request Headers

Within the Fastly application, navigate to **Content** → **Headers**. Then select the **New** button to create a request header within vcl_recv.

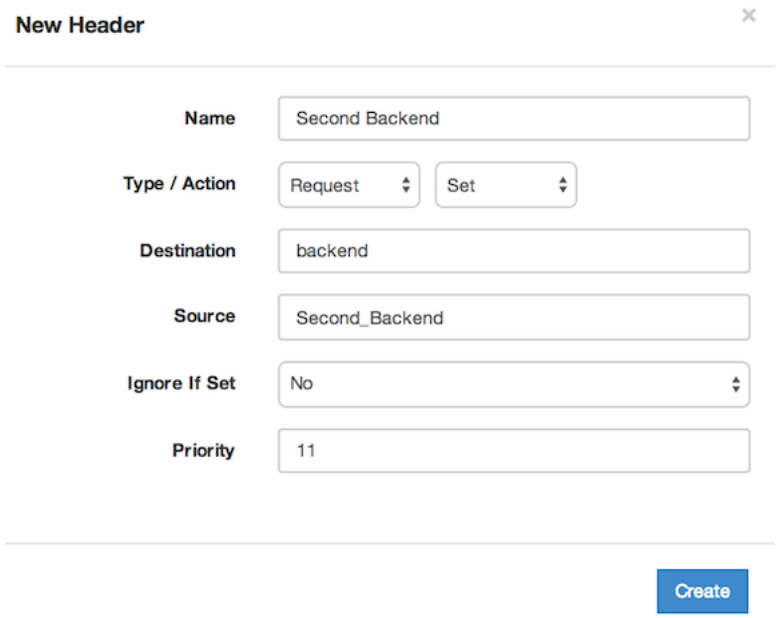


Set the header controls as follows:

- **Name:** `Fastly Internal Shielding` (or any meaningful, preferred name)
- **Type/Action:** `Request/Set`
- **Destination:** `http.Fastly-Force-Shield`
- **Source:** `"yes"`
- **Ignore If Set:** `No`
- **Priority:** `10`

Click **Create** to add the new header (it will appear in the Headers area).

Add another header to switch to the next backend.



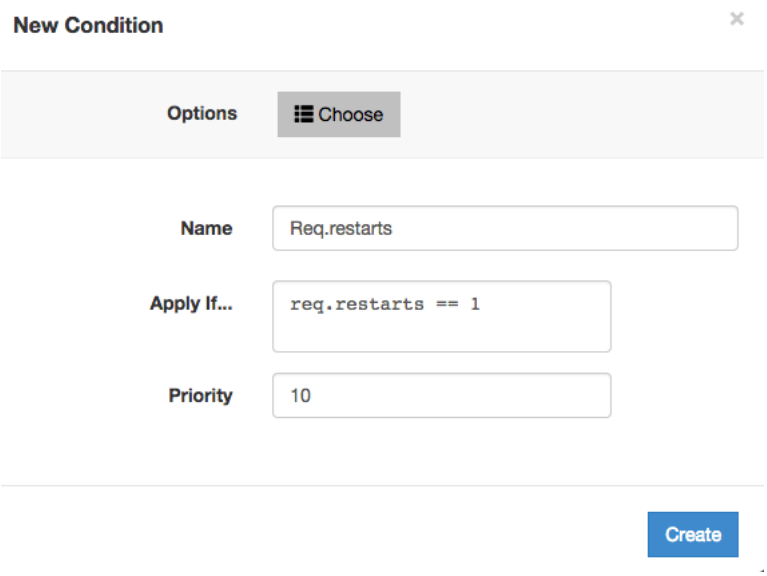
Set the next header controls as follows:

- **Name:** `Second Backend` (or any meaningful, preferred name)
- **Type/Action:** `Request/Set`
- **Destination:** `backend`
- **Source:** `Second Backend` (this should match the name of your other backend)
- **Ignore If Set:** `No`
- **Priority:** `11` (Note that the priority is higher to set after the Fastly Internal Shielding)

Click **Create** to add the new header (it will appear in the Headers area).

Create New Header Conditions

Once you've created the new headers, click the gear icon next to each new header rule in the Headers area to add a new condition.



Set the new condition settings as follows:

- **Name:** `Req.request` (or any meaningful, preferred name)
- **Apply if**: `req.restarts == 1`

Click **Create** to assign the condition. Then click the **Activate** button, near the top right corner of the window, to deploy the new version of the service you just edited.

· How can I follow redirects to S3 objects and cache S3 responses? (/guides/backend-servers/how-can-i-follow-redirects-to-s3-objects-and-cache-s3-responses)

With custom VCL, Fastly can follow redirects to S3 objects and cache the s3 response as well as the 301 or 302 response separately. The ability to upload custom VCL is disabled by default when you first sign up for Fastly services (contact support@fastly.com (mailto:support@fastly.com) to have VCL enabled for your account). Once the ability to upload custom VCL has been enabled, be sure to read our "How do I mix and match Fastly VCL with custom VCL? (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl)" instructions. 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 S3 objects, insert the following VCL snippets in your custom VCL:

Within vcl_recv

```
sub vcl_recv {

  if (req.http.redir != "true") {
    set req.backend = Main_Origin;
  }

  else {
    set req.backend = s3_backend;
    set req.http.host = "s3.amazonaws.com";
  }

#FASTLY recv

  if (req.request != "HEAD" && req.request != "GET" && req.request != "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://s3.amazonaws.com/(.*)$", "/\1");
    set req.http.Fastly-Force-Shield = "yes";
    restart;
}

#FASTLY deliver

  return(deliver);
}
```

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. 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.

· How often will Fastly perform health checks? (/guides/backend-servers/how-often-will-fastly-perform-health-checks)

Fastly performs health checks (/guides/backend-servers/health-checks-tutorial) on your origin server based on the Check Frequency setting you select in the New Health Check window. The Check Frequency setting you select will specify approximately how many requests per minute are checked to see if they pass. Any checks that pass will be reported as "healthy."

You can select the following frequencies:

- **Low** - about 200 requests per minute, with "healthy" meaning 1 out of 2 must pass
- **Normal** - about 800 requests per minute, with "healthy" meaning 3 ouf of 5 must pass
- **High** - about 6,000 requests per minute, with "healthy" meaning 7 out of 10 must pass

You can also set a custom frequency. Instead of one of the above frequencies, select **Custom** and the following controls will appear:

- **Threshold & Window** - The number of successes per total number health checks. For example, specifying `1/2` means 1 out of 2 checks must pass to be reported as healthy.

- **Initial** - The number of requests to assume as passing on deploy.
- **Interval & Timeout (ms)** - 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.

· How to track your origin's name, IP, and port (/guides/backend-servers/how-to-track-your-origins-name-ip-and-port)

Fastly provides three values captured in vcl_fetch that allow you to see and track origin information:

- `beresp.backend.name`
- `beresp.backend.port`
- `beresp.backend.ip`

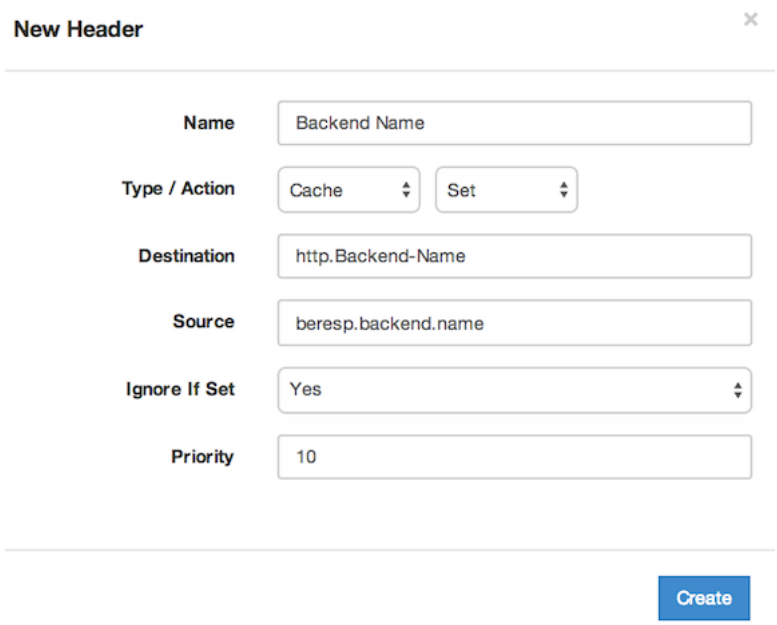
While these three values are immensely useful, you may want to use this information within `vcl_deliver` for things like response information or remote log streaming. You can do this by:

1. Creating cache headers that capture the origin information.
2. Adding a response header to the log format to capture the response output.

Capturing the Origin Information

To track your origin's name, IP, and port, you need to create two separate headers: one that captures the origin name and another that captures the origin's IP and port (e.g., 80, 443).

Create the header that captures the origin name by launching the Fastly application and navigating to **Content** → **Headers**. Then, click the **New** button to display the New Header window.

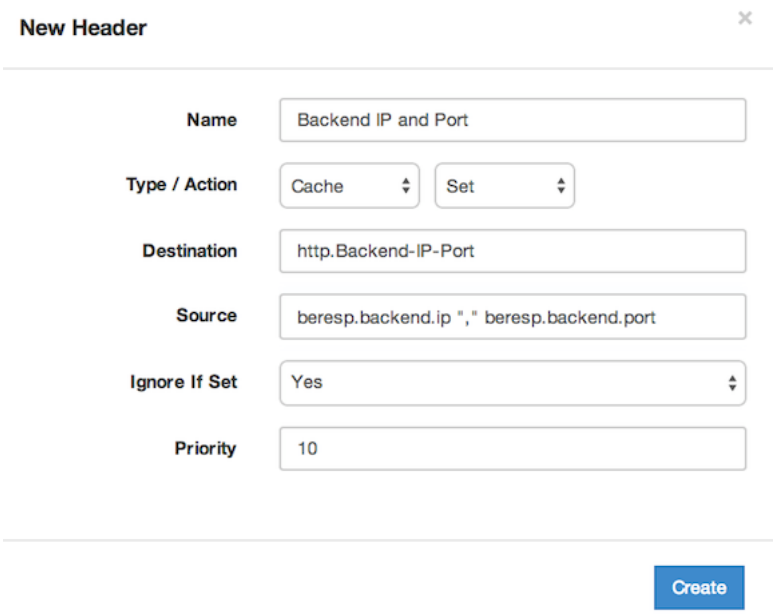


Set this first header's controls as follows:

- **Name:** Backend Name (or preferred name)
- **Type/Action**: Cache/Set
- **Destination:** `http.Backend-Name` (or preferred header variable)
- **Source:** `beresp.backend.name`
- **Ignore If Set:** Yes
- **Priority:** `10`

Once you've set the controls, click **Create** to add the first new header.

Create the second header to capture the IP and port for your origin by navigating again to **Content** → **Headers.** Then, click the **New** button to display the New Header window a second time.



Set the second header's controls as follows:

- **Name:** Backend IP and Port (or preferred name)
- **Type/Action**: Cache/Set
- **Destination:** `http.Backend-Name` (or preferred header variable)
- **Source:** `beresp.backend.ip "," beresp.backend.port`
- **Ignore If Set:** Yes
- **Priority:** 10

Once you've set the controls, click **Create** to add the second new header.

Adding a Response Header to the Log Format

The values captured in a header within `vcl_fetch` will flow to `vcl_deliver`. For example, there will exist a resp.http.Backend-Name header in `vcl_deliver` that corresponds to beresp.http.Backend-Name in `vcl_fetch`. By default, the response header will be included in the response output.

Unfortunately, with remote log streaming, you cannot add the `vcl_fetch` header, beresp.http.Header-Name, to the log format. However, you can add its cousin in `vcl_deliver`, resp.http.Header-Name.

Add resp.http.Header-Name to the log format that you configured by following the instructions in the Remote Log Streaming guide (/guides/log-streaming/setting-up-remote-log-streaming). Using the example above, you would add resp.http.Backend-Name and resp.http.Backend-IP-Port.

Important Notes

Regarding Shielding

Notice within the example the field **Ignore If Set** is set to **Yes**. With shielding, the VCL is executed twice, once on the shield and again on the edge node. This setting will display the original information from the origin without overriding it on the edge node.

You can also set the field **Ignore If Set** to **No** with shielding enabled. In this scenario, the edge node captures the shield's information within `beresp.backend.name`, `beresp.backend.ip`, and `beresp.backend.port`.

If remote log streaming is configured, remember it is executed twice. Thus the first log (from the shield node) will have the origin's information and the second log (from the edge node) will have the shield's information.

Regarding Security

For security purposes, you may want to track the information in logging but not display all or some of it in the response. This is possible but requires custom VCL to strip the information after sending the log line from the edge node. You'll need to first contact support@fastly.com (mailto:support@fastly.com) to have VCL enabled for you account. Don't forget to read our guide to using custom VCL (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl) before you begin. Remember to include the entire boilerplate if you do not intend to override the Fastly default settings.

Then add the following snippet within `vcl_deliver` with the headers you want to strip. Using our example above, we continue to send the value of the origin's name within the response. We strip the origin's IP and port from the response information.

```
sub vcl_deliver {
#FASTLY deliver

if (!req.http.Fastly-FF){
  unset resp.http.Backend-IP-Port;
}

return(deliver);
}
```

· Is it possible to use a domain name as a backend? (/guides/backend-servers/is-it-possible-to-use-a-domain-name-as-a-backend)

Yes, we allow the use of domain names as backends. Obviously there is a performance impact, but it allows things like EC2 machines and App Engine instances to be used. In fact, when we detect certain conditions, such as the host being part of EC2, we optimize our routing to be more efficient.

Note: There are several other ways to set up origins detailed in our Backend Servers Help Guides (/guides/backend-servers/).

· Maintaining Separate HTTP and HTTPS Requests to Backend Servers (/guides/backend-servers/maintaining-separate-http-and-https-requests-to-backend-servers)

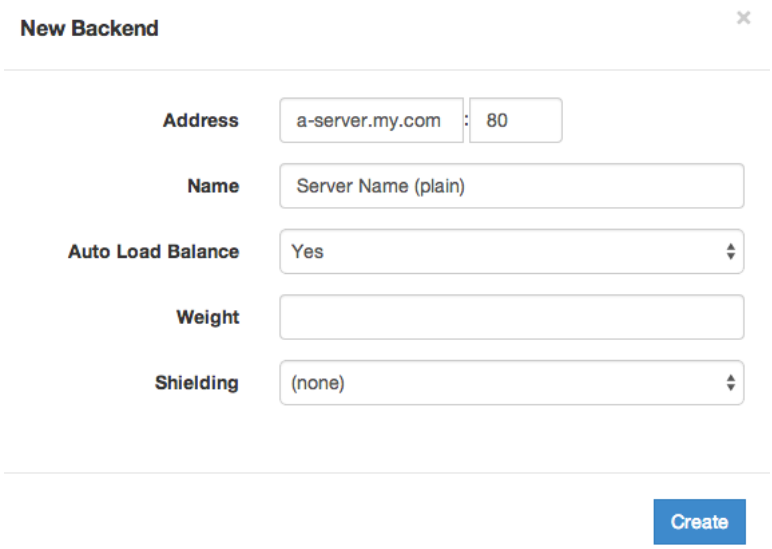
It is common to use the same backend web application to serve both HTTP and HTTPS requests and let the application determine which actions to take depending on the incoming protocol. Fastly allows users to set this up to preserve this functionality within their servers. To set Fastly up to send HTTP requests to the non-secure service and HTTPS requests to the secure service, configure two backends, one each for the secure and non-secure ports, then set up the conditions under which requests will be sent there.

Create multiple backends

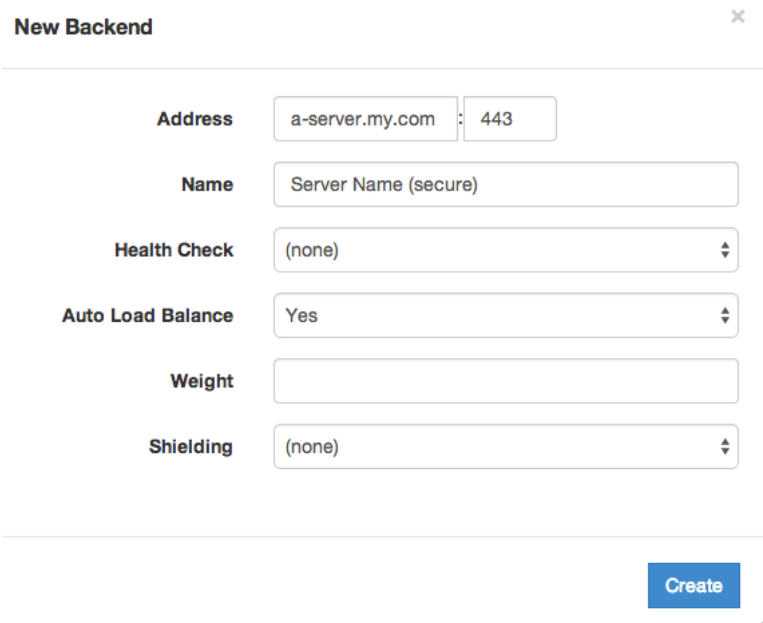
Begin by configuring the same backend address with a different port as a separate backend by following the steps below.

1. Log in to the Fastly application.
2. Click the **configure** tab to access the control panel.

3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **Hosts** section.
6. In the **Backends** area click the **New** button to create a new backend for your non-secure server. The New Backend window appears.



7. Fill out the **New Backend** window as follows:
   - In the **Address** field, type the address of your non-secure server (for example, `a-server.my.com`).
   - In the **Port** field type `80`.
   - In the **Name** field, type the name of your non-secure server (for example, `Server Name (plain)`).
   - Leave the **Health Check**, **Auto Load Balance**, **Weight**, and **Shielding**. controls set to their default values.
8. Click the **Create** button. The non-secure HTTP server appears in the Backends area.
9. In the **Backends** area click the **New** button a second time to create a second, new backend, this time for your secure server.



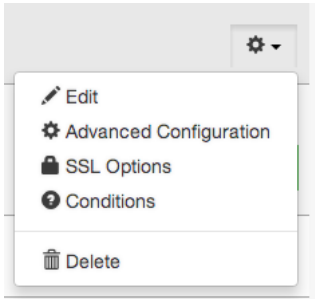
10. Fill out the **New Backend** window as follows:
    - In the **Address** field, type the address of your secure server (for example, `a-server.my.com`).
    - In the **Port** field type `443`.
    - In the **Name** field, type the name of your secure server (for example, `Server Name (secure)`).
    - Leave the **Health Check**, **Auto Load Balance**, **Weight**, and **Shielding**. controls set to their default values.
11. Click the **Create** button. The secure HTTP server appears in the Backends area.

Conditionally send traffic to backends

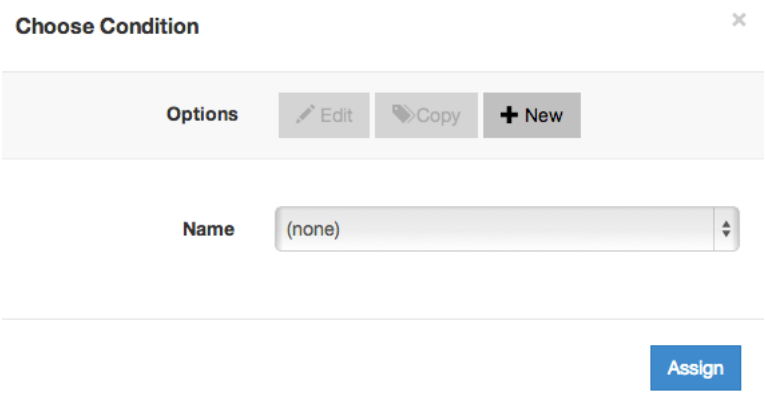
To conditionally determine which server receives secure and non-secure requests, Fastly relies on the presence or absence of a specific header when the: backend is selected. When an incoming connection is received over SSL, Fastly sets the `'req.http.fastly-ssl'` header to determine which server to use.

Set a condition for this header on each backend by following the steps below.

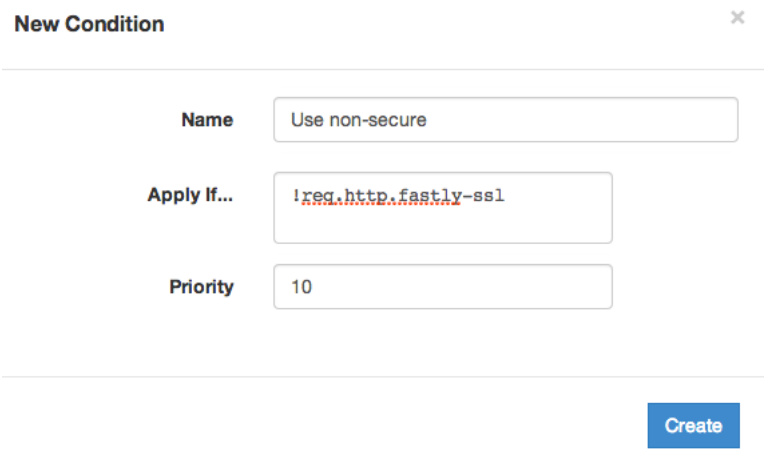
1. In the **Backends** area of the **Hosts** section, click the gear icon next to the name of the non-secure server.



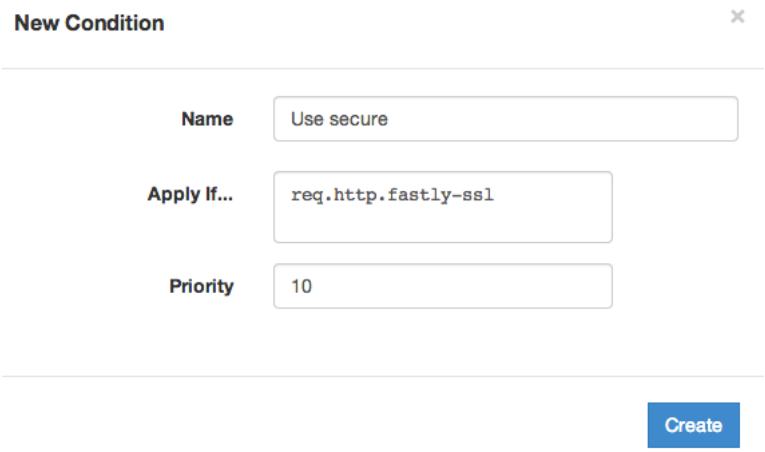
2. From the drop down menu that appears, select **Conditions**. The Choose Condition window appears.



3. Click the **New** button. The New Condition window appears.

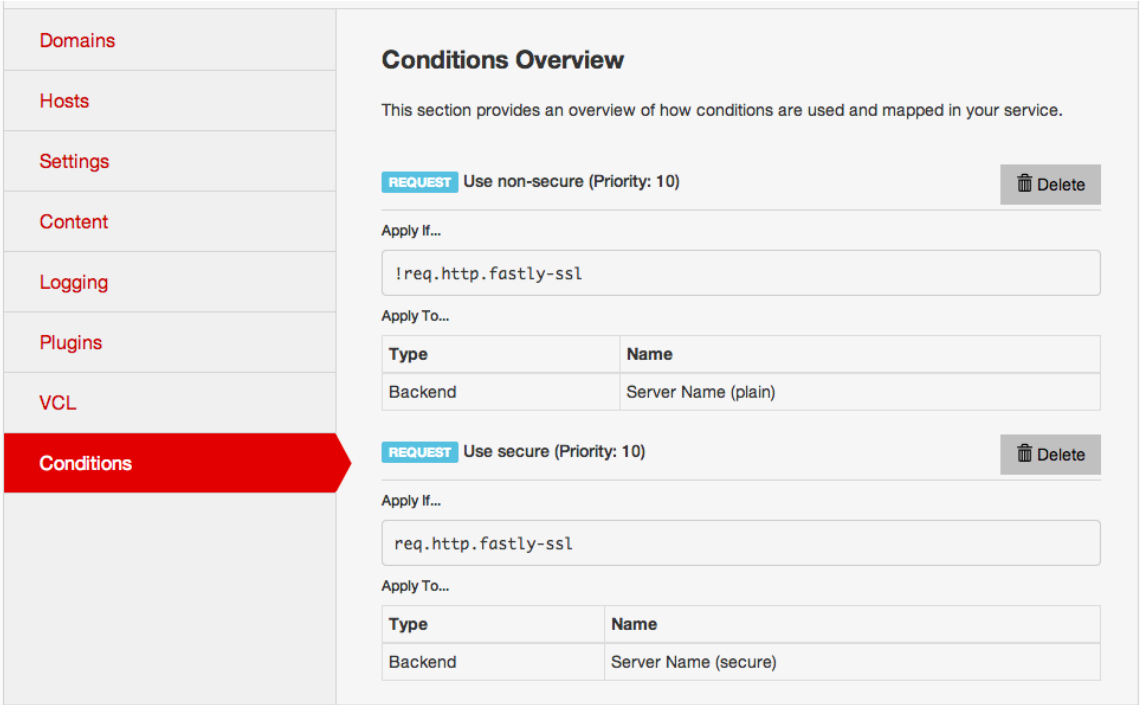


4. Fill out the **New Condition** window as follows:
   - In the **Name** field, type the name of the condition specifying use of the non-secure server (for example, `Use non-secure`).
   - In the **Apply If** field, type `!req.http.fastly-ssl`.
   - Leave the **Priority** field set to its default value.
5. Click the **Create** button to create the new condition.
6. In the **Backends** area of the **Hosts** section, click the gear icon next to the name of the secure server.
7. From the drop down menu that appears, select **Conditions**. The Choose Condition window appears.
8. Click the **New** button. The New Condition window appears.



9. Fill out the **New 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** field set to its default value.
10. Click the **Create** button to create the new condition.

    TIP: You can view all conditions and the criteria under which they're applied, including the ones you just created, by examining the Conditions Overview for your service.



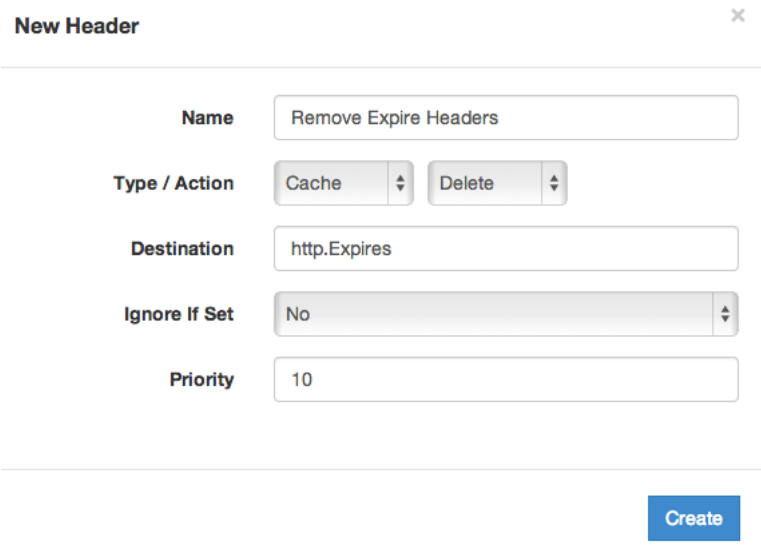
11. Deploy the new backends and their conditions by activating the new version of your service.

· Removing headers from backend response (/guides/backend-servers/removing-headers-from-backend-response)

On occasion you may want to remove headers from backend response. For example, a Drupal install was setting Expires and Cache-Control headers not to cache anything:

```
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
```

In the above example, I did not have a ready access to the Apache configuration and decided to simply drop those from the backend response. To do that I accessed my configuration settings for my service and clicked on the New button under the Headers area of the Content settings. When the New Header window appeared:



I filled out the fields as follows:

Name: a descriptive name I'll remember. **Type**: Cache as the type of header. **Action**: Delete as the action. **Destination**: The name of header (e.g., `http.Expires`)

I left the rest of the controls with their default values and then repeated the same for Cache-Control headers. It worked like a charm.

You may also be interested in our guide to Setting Content Type Based on File Extension (/guides/backend-servers/setting-content-type-based-on-file-extension).

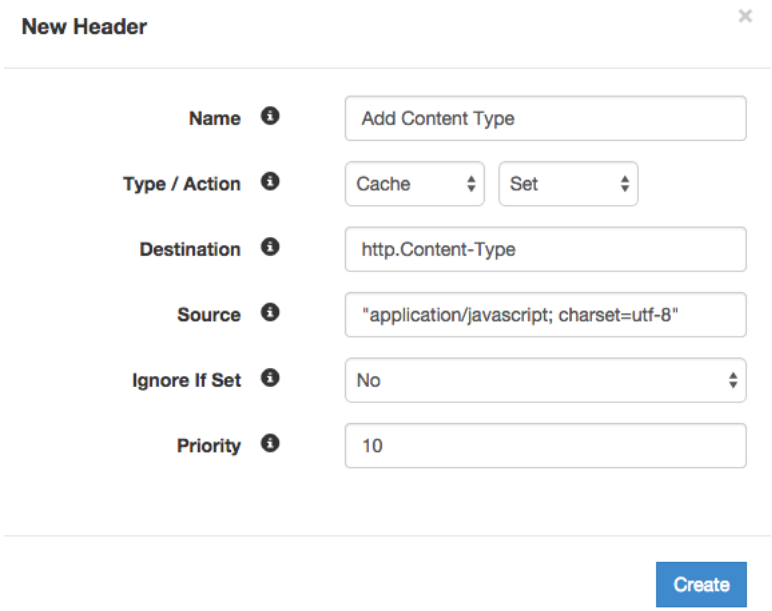
· Setting Content Type based on file extension (/guides/backend-servers/setting-content-type-based-on-file-extension)

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 application.
2. Click the **configure** tab to access the control panel.

3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **Content** section.

6. In the **Headers** area, click the **New** button to create a new header. The New Header window appears.



7. Fill out the **New Header** window as follows:
   - In the **Name** field, type an appropriate name, such as `Add Content Type`.
   - From the **Type/Action** drop down menus, select **Cache** and **Set**.
   - In the **Destination** field type `http.Content-Type`.
   - In the **Source** field type the content type you want match, such as `"application/javascript; charset=utf-8"`. IMPORTANT: Quotes around the content type are required.
   - From the **Ignore if Set** menu, select **No**.
   - In the **Priority** field, type `10`.
8. Click the **Create** button. A new header appears in the Headers area of the Content section.

Once you have created the header object you will need to apply a condition (/guides/tutorials/conditions-tutorial). Otherwise, that particular object will apply to all requests.

1. Click the gear icon to the right of the new header name and select **Cache Conditions** from the drop down menu. The Choose Condition window appears.
2. In the **Options** section, click the **New** button to add a new condition. The New Condition window appears.3. Fill out the next **New Condition** window 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 ~ "\.js$"` (to match our request for files ending in .js).
3. Click **Create** to create the new condition.
4. Deploy your configuration changes.

You may also be interested in our guide to Removing Headers from Backend Response (/guides/backend-servers/removing-headers-from-backend-response).

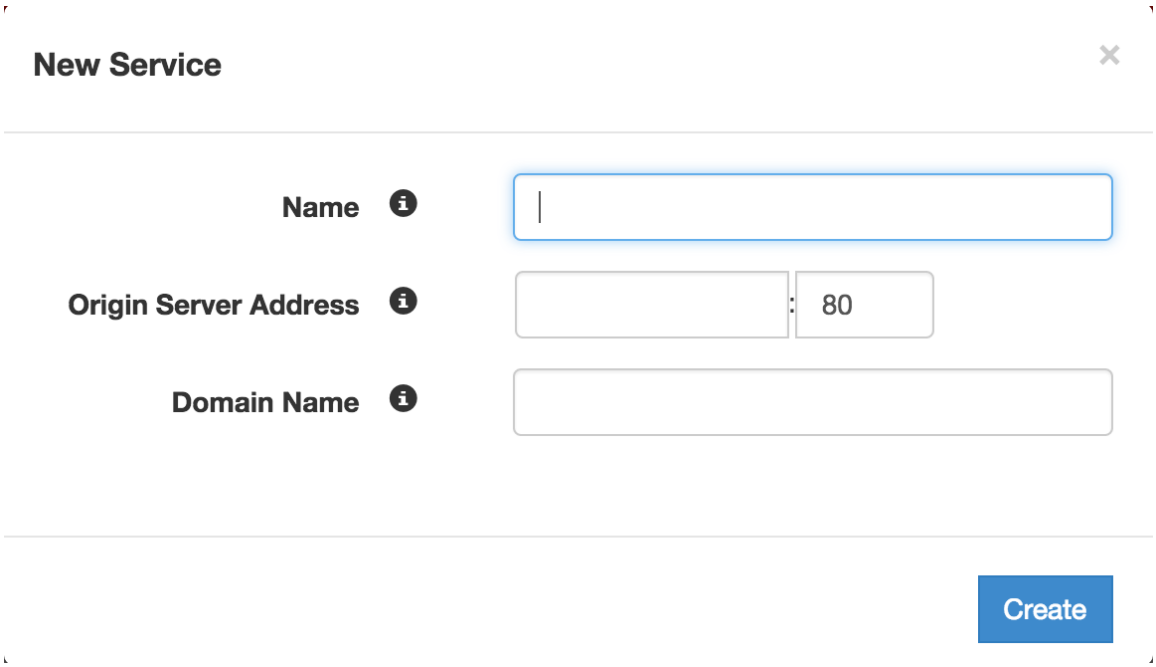
· Using Amazon S3 as a backend (/guides/backend-servers/using-amazon-s3-as-a-backend)

To make your S3 data buckets available through Fastly follow the steps below.

How to Use Amazon S3 as a Backend

1. Log in to the Fastly application.
2. Click the **configure** tab to access the control panel.

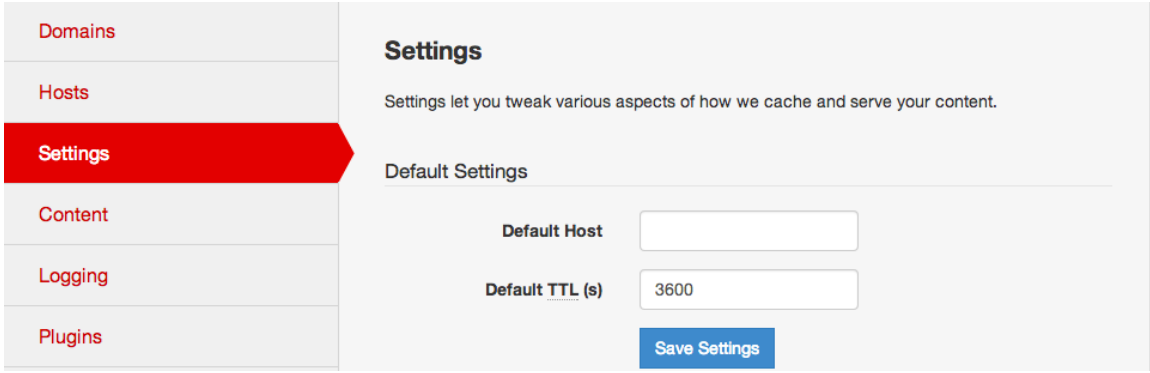
3. Click the green New Service button at the top right of the window. The New Service window appears.



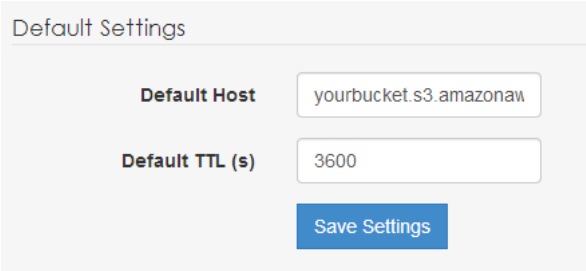
4. Fill out the **New Service** window as follows:
   - In the **Name** field, type any descriptive name for your service.
   - In the **Origin Server Address** field, type the full URL of your S3 web bucket (e.g., `<yourbucket>.s3.amazonaws.com`) and leave the default port as `80`.
   - In the **Domain Name** field, type the host name you want to use as the URL (e.g., `cdn.domain.com`).
5. Click the **Create** button. A new service appears in the list of services available.

Now that the service is created you will need to set the Default Host to the name of **yourbucket**.s3.amazonaws.com following the steps below:

1. Click the **configure** tab to access the control panel.
2. Select the new service you just created from the **Service** drop down.
3. Click the blue **Configure** button to the right of the service name.
4. Click the **Settings** section to access the **Default Settings** area.



5. In the **Default Host** field, type the hostname for your S3 bucket again. For example, `<yourbucket>.s3.amazonaws.com`.



6. Click **Save Settings** and then remember to deploy your changes. Your service should be active within few seconds.

Testing Your Results

By default, we create DNS mapping called **yourdomain.global.prod.fastly.net**. In the example above, it would be `cdn.domain.com.global.prod.fastly.net`. Please test, and if you are satisfied with the results, create a DNS alias for the domain name you specified (e.g., CNAME `cdn.domain.com` to `global.prod.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 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) media check out our Amazon S3 configuration (/guides/tutorials/cache-control-tutorial) in our Cache Control tutorial.

· Using an Amazon S3 private bucket (/guides/backend-servers/using-an-amazon-s3-private-bucket)

To use an Amazon S3 private bucket with Fastly, follow the instructions below.

Before You Begin

Be sure you've already made your S3 data buckets available to Fastly by pointing to the right S3 bucket (/guides/backend-servers/using-amazon-s3-as-a-backend). This needs to be done before authenticating.

Be sure you've got the AWS access key ID, AWS secret key ID, and AWS Bucket name on hand. The Amazon S3 Authorization header takes the following form:

Authorization: AWS `_AWSAccessKeyId_`:`_Signature_`

From your developer Amazon account you will need the following information:

1. The **AWS access key ID** and **AWS secret access key**. The AWS secret access key is issued when you register. If you do not have or remember your AWS secret access key, create a new AWS access key ID. The AWS secret access key will be displayed before disappearing again.
2. Your **AWS Bucket** name.

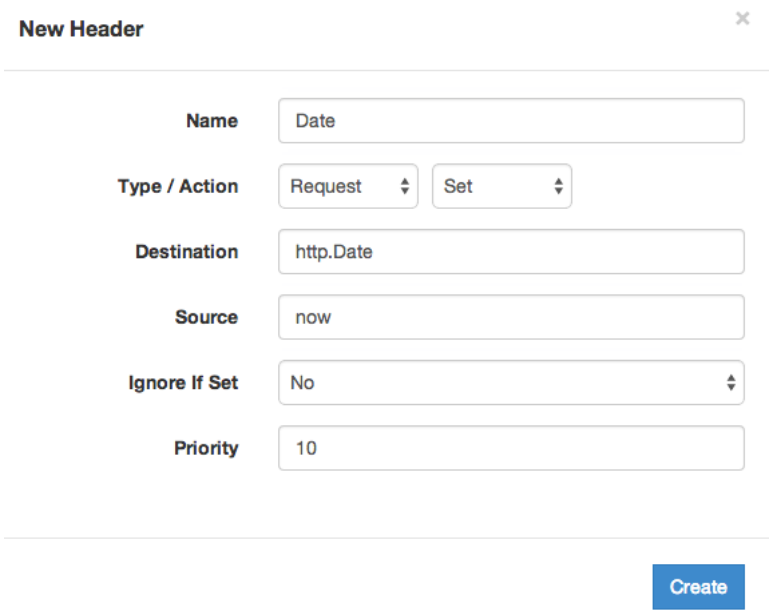
Setting Up Fastly to Use an Amazon S3 Private Bucket

In order to use an Amazon S3 private bucket with Fastly, create two headers (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses), a Date header (for use with the authorization Signature) and an Authorization header.

Create a Date Header

1. Log in to the Fastly application.
2. Click the **configure** tab to access the control panel.

3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **Content** section.
6. In the **Headers** area, click the **New** button to create a new header. The New Header window appears.

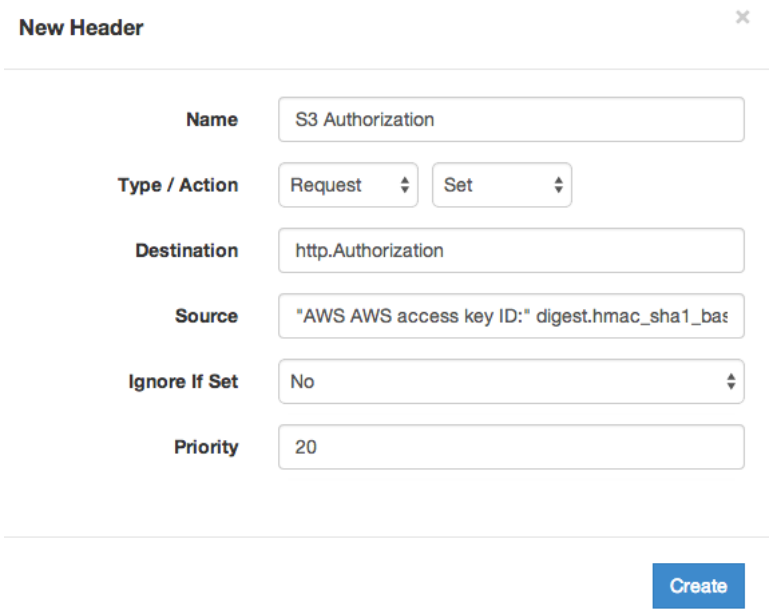


7. Fill out the **New Header** window as follows:
    - In the **Name** field, type `Date`.
    - From the **Type/Action** drop down menus, select **Request** and **Set**.
    - In the **Destination** field type `http.Date`.
    - In the **Source** field type `now`.
    - From the **Ignore if Set** menu, select **No**.
    - In the **Priority** field, type `10`.
8. Click the **Create** button. A new Date header appears in the Headers area of the Content section. 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 **New** button again to create another new header.



2. Fill out the **New Header** window as follows:
   - In the **Name** field, type `S3 Authorization`.
   - From the **Type/Action** drop down menus, select **Request** and **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 <AWS access key ID>:" digest.hmac_sha1_base64("<AWS secret key ID>", req.request LF LF LF req.http.Date LF "/<AW
   S Bucket name>" req.url)
   ```

   replacing `<AWS access key ID>`, `<AWS secret key ID>`, and `<AWS Bucket name>` with the information you gathered before you began. For example:

   ```
   "AWS JKCAUEFV2ONFFOFMSSLA:" digest.hmac_sha1_base64("P2WPSu68Bfl89j72vT+bXYZB7SjlOwhT4whqt27", req.request LF LF LF r
   eq.http.Date LF "/test123" req.url)
   ```

4. Click the **Create** button. A new Authorization header appears in the Headers area of the Content section.

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 Amazon developer's 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 ID from your Amazon developer's 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:

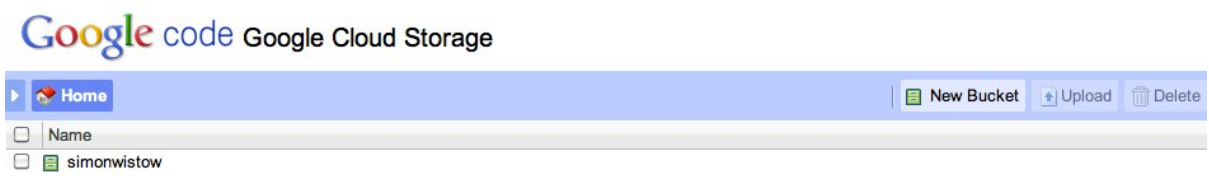
```
<HTTP-verb></n><Content-MD5>/n<Content-Type></n><Date></n><CanonicalizedAmzHeader></n><CanonicalizedResource>
```

It tells us the following:

Element	Description
`HTTP-verb`	The REST verb. We use `req.request` 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).
`CanonicalizedAmzHeader`	The x-amz headers, which customize your S3 implementation. It's often left blank. We use `LF` in this example.
`CanonicalizedResource`	Your Amazon private bucket name. We use `"/test123"` in this example.

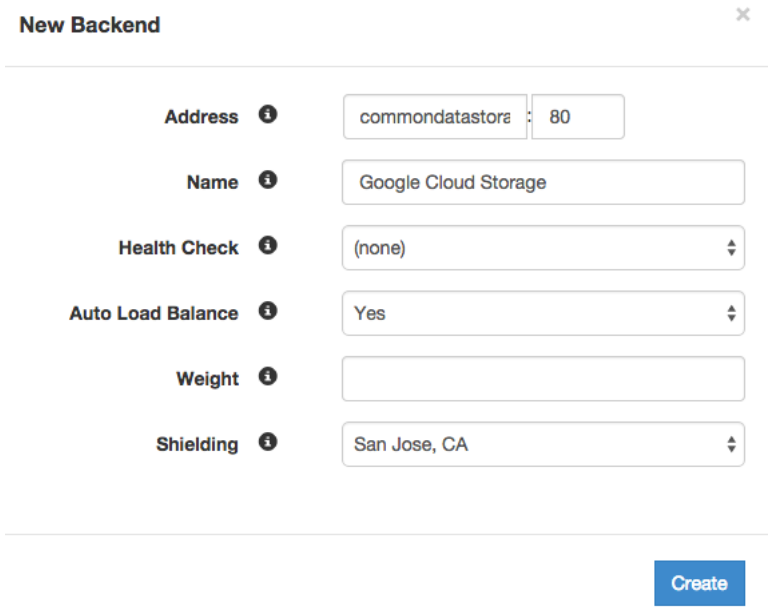
· Using Google Cloud Storage as an origin (/guides/backend-servers/using-google-cloud-storage-as-an-origin)

Google Cloud Storage (GCS) can be used as an origin server with your Fastly services. Start by setting up a Google Cloud Storage account (https://cloud.google.com/products/cloud-storage) and create a bucket.



Next, add a file to the bucket you just created and make the bucket public.

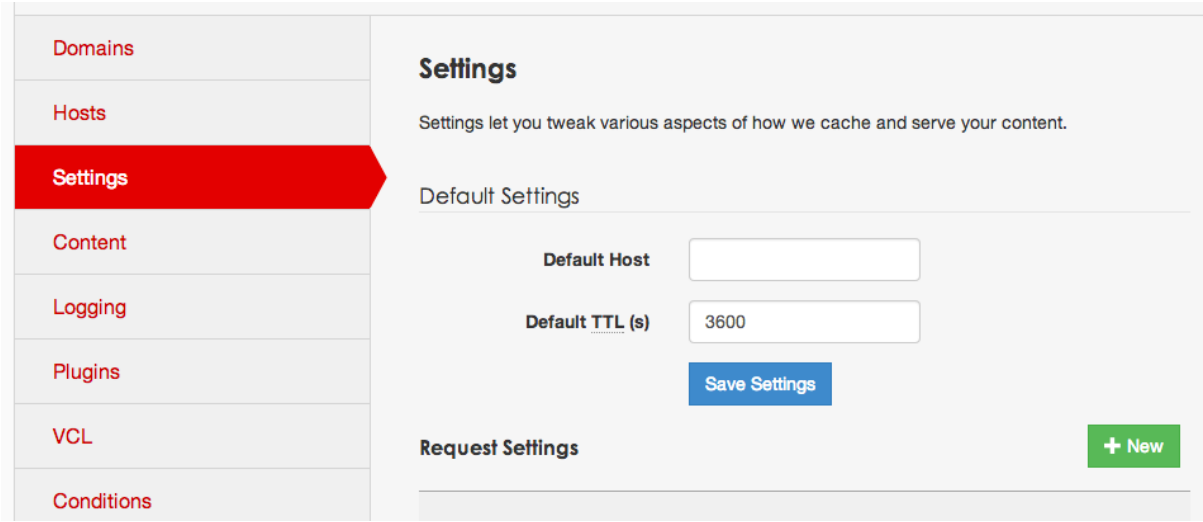
Then, log in to the Fastly application to create a service and add a New Backend.



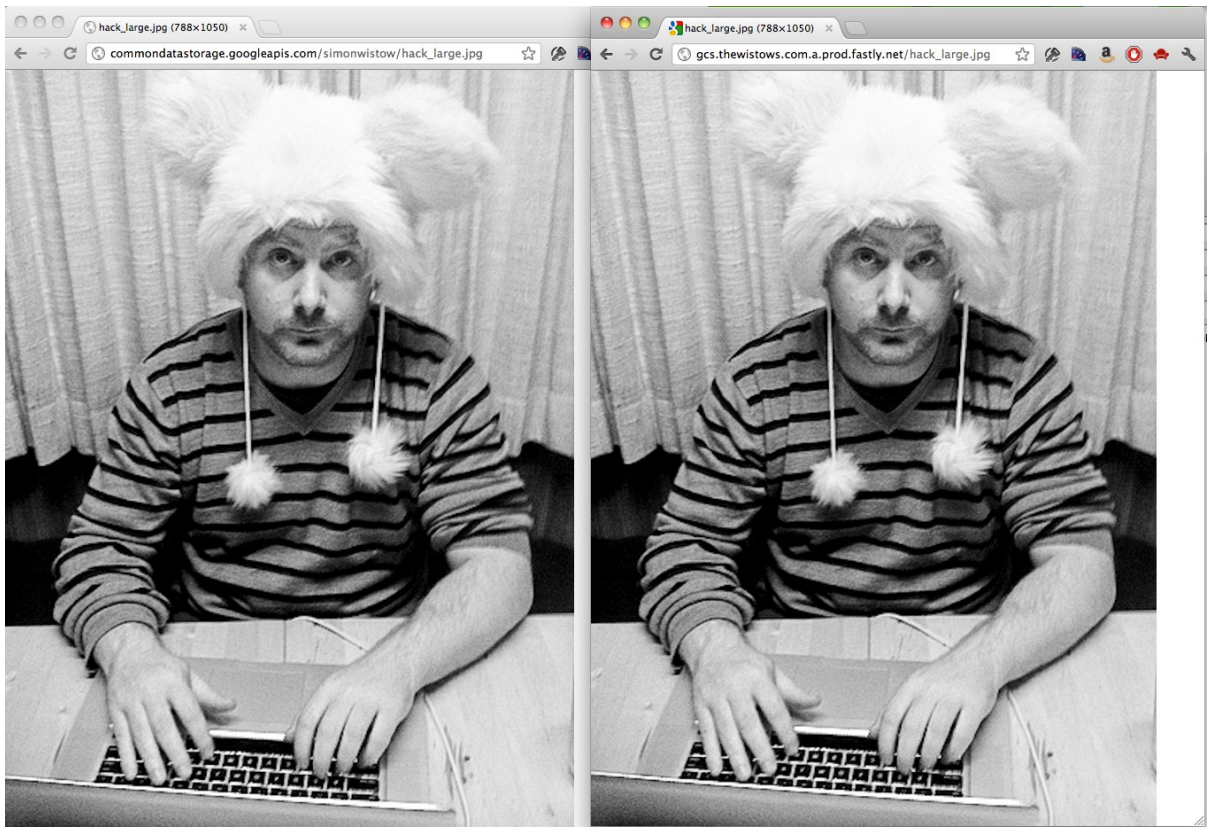
Be sure to set the address of the backend to `commondatastorage.googleapis.com`. Then select either `Ashburn, VA` or `San Jose, CA` from the shielding (http://www.fastly.com/products/origin-shield/) locations.

Then create a new domain (you can call it whatever you want).

Next, set a default host for this service. The name of the default host should be `<your bucket name>.commondatastorage.googleapis.com`.



Deploy the changes to your service.



Once you have completed these steps, email salesgcp@fastly.com (mailto:salesgcp@fastly.com) to ensure you receive Fastly's Cloud Accelerator discount. If you already have GCS configured as your origin and would like to take advantage of this integration, feel free to contact us.

Note: Fastly describes several other ways to set up origins in our Backend Servers Help Guides (/guides/backend-servers/).

· Why am I getting a 503 "backend read error"? (/guides/backend-servers/why-am-i-getting-a-503-backend-read-error)

Note: For more information about the most common 503 errors read our list of descriptions (/guides/backend-servers/503-error-explanations).

One of the most common 503 errors, a "backend read error" appears when a backend response to Fastly takes too long and the process is halted. Server optimization issues, extremely large files, or intermittent network issues represent the biggest causes of backend read errors. If you're seeing 503 backend read errors consistently with your origin servers or with certain files, changing a few quick settings can prevent these errors from happening.

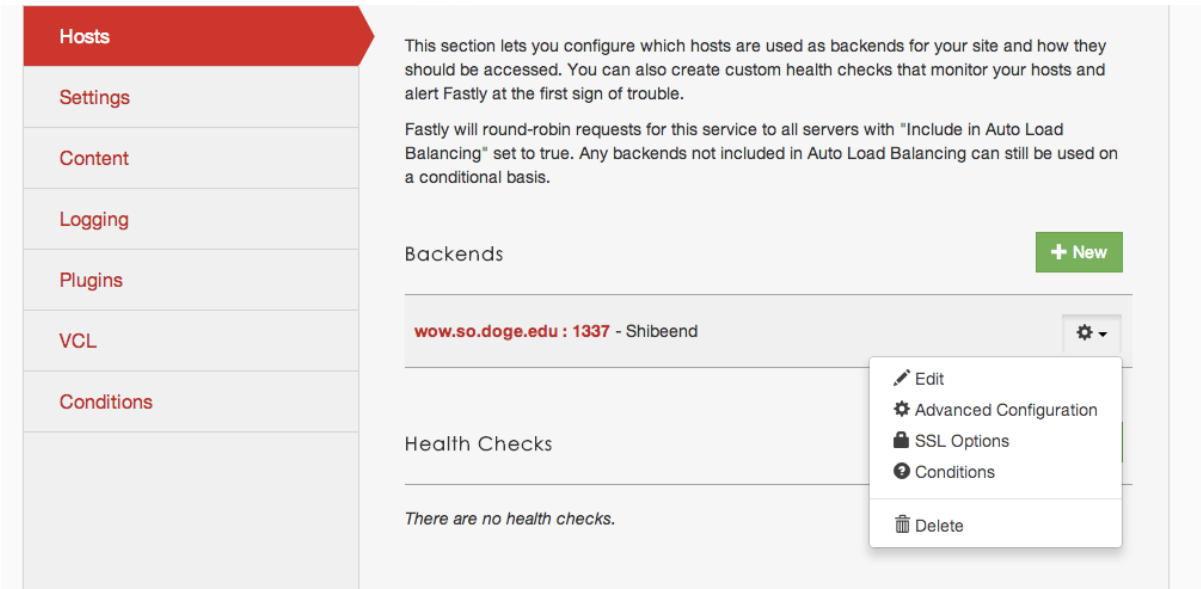
Benchmarking your backend response times

Many outside factors cause backends response times to vary, but running the following command should give you a reasonable estimate of that response time for benchmarking purposes:

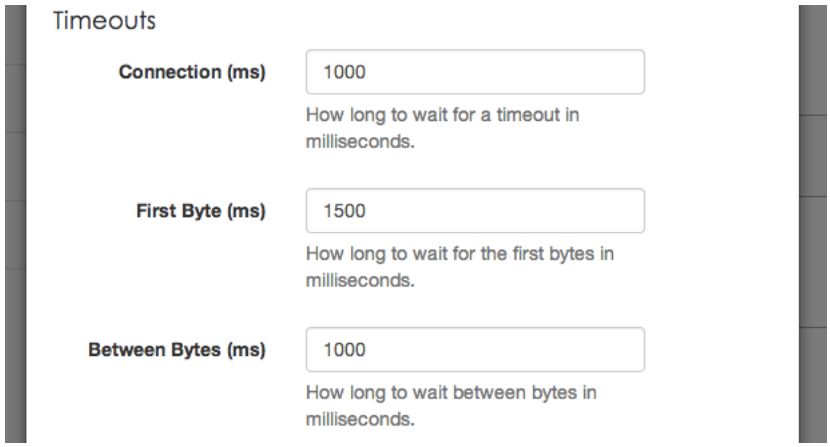
```
curl -s -w "%{time_total}\n" -o /dev/null http://your_backend.com/path/to/file
```

Increasing your backend response time

After benchmarking some of the slower paths in your application, you should have an idea of your ideal backend response time. You can adjust your backend settings on the Hosts pane using the Advanced Configuration settings for your backend:

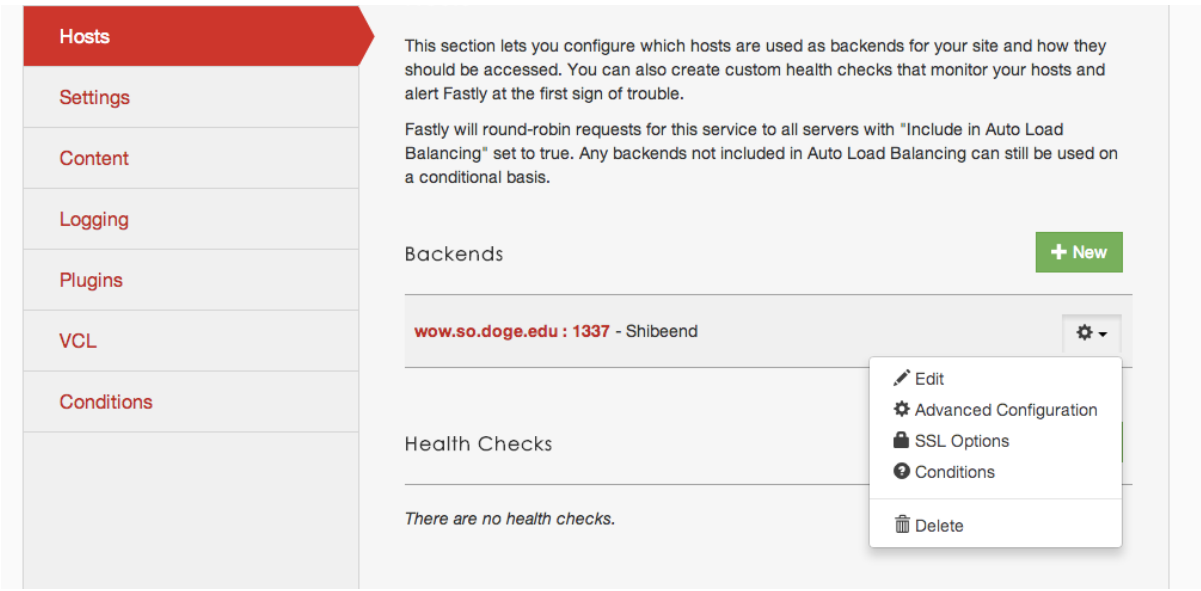


On the Advanced Configuration settings window for your backend, simply increase the value in the First Byte field based on your benchmark. This field represents how long in milliseconds to wait for the first bytes of a response before timing out.

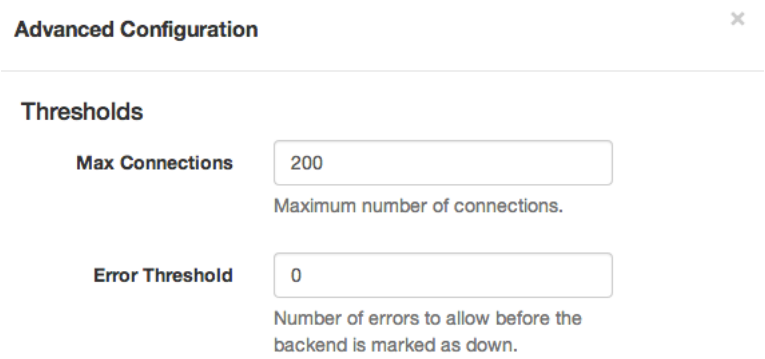


· Why am I getting "Error 503 Backend.max_conn reached" ? (/guides/backend-servers/why-am-i-getting-error-503-backend.max_conn-reached)

By default we limit you to 200 origin connections from a single cache node to protect origins from overload. If you want to change the number of connections, access the Hosts area of the configurations settings and select Advanced Configuration settings on an origin or backend object (via the gear icon next to the name):



Then, modify the Max Connection field to a higher number:



Note: For more information about the most common 503 errors read our list of descriptions (/guides/backend-servers/503-error-explanations).

• Shielding (/guides/shielding/)

· Caveats of Shielding (/guides/shielding/caveats-of-shielding)

With shielding enabled (/guides/shielding/) there are several caveats you will encounter in performance and billing:

Inbound traffic to the shield will be billed as regular traffic, including requests to populate remote POPs.

This 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, but will also go through the shield.

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 (/guides/shielding/understanding-cache-hit-and-miss-headers-with-shielded-services).

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 user interface or API. Other custom VCL will work just fine.

You can only set one shield total if automatic load balancing is selected via the user interface.

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.

To set multiple shields and auto load balancing at the same time, contact support@fastly.com (mailto:support@fastly.com) for assistance with the custom VCL.

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 404 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 is executed twice: once on the shield node and again on the edge node.

Possible changes to `beresp` can affect the caching of a URL on the shield and edge.

For instance, say your goal is to have Fastly cache an object for one hour and then ten seconds on the browser. The origin sends `Cache-control: s-maxage=3600`. You unset `beresp.http.Cache-Control` and then reset to `s-maxage=10`. With shielding, the result will not be what you expect. The object will have `s-maxage=3600` on the shield and then it reaches the edge with `s-maxage=10`.

In this scenario, a better option would be to use `Surrogate-Control` and `Cache-control`. `Surrogate-Control` overrides `Cache-control` and is stripped after the edge node. The `s-maxage` from the `Cache-control` will then communicate with the browser.

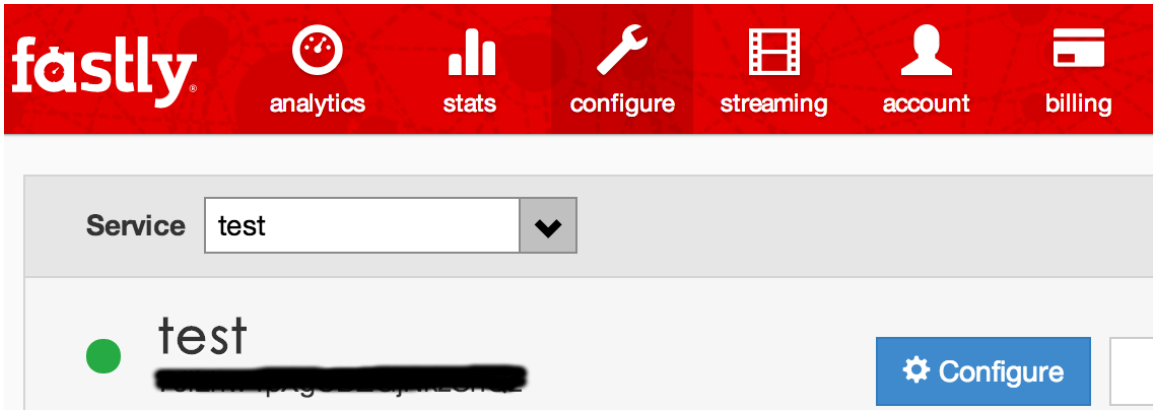
· How do I enable Shielding? (/guides/shielding/how-do-i-enable-shielding)

Fastly's shielding service feature (http://www.fastly.com/products/origin-shield) allows you to designate a specific POP as a shield node to your origins. Once designated, all requests to your origin will go through that datacenter, increasing the chances of getting a HIT (/guides/shielding/understanding-cache-hit-and-miss-headers-with-shielded-services) for a given resource. If a

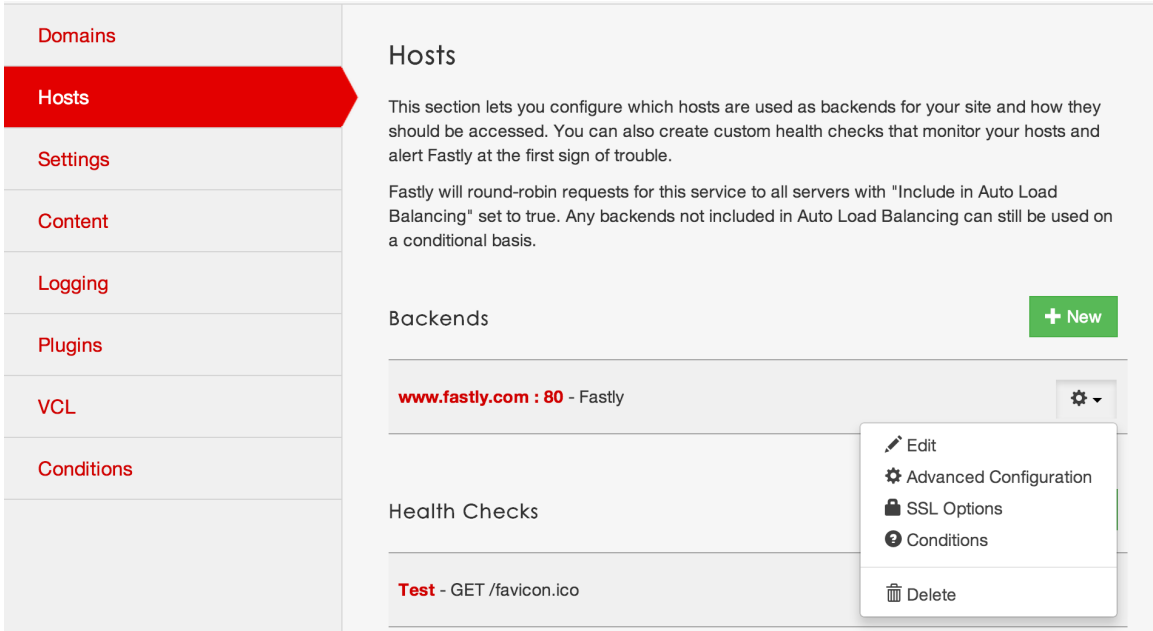
different POP doesn't have a specific object, it will query the shield (provided it's not down for maintenance) instead of your origins.

Enable shielding with these quick steps:

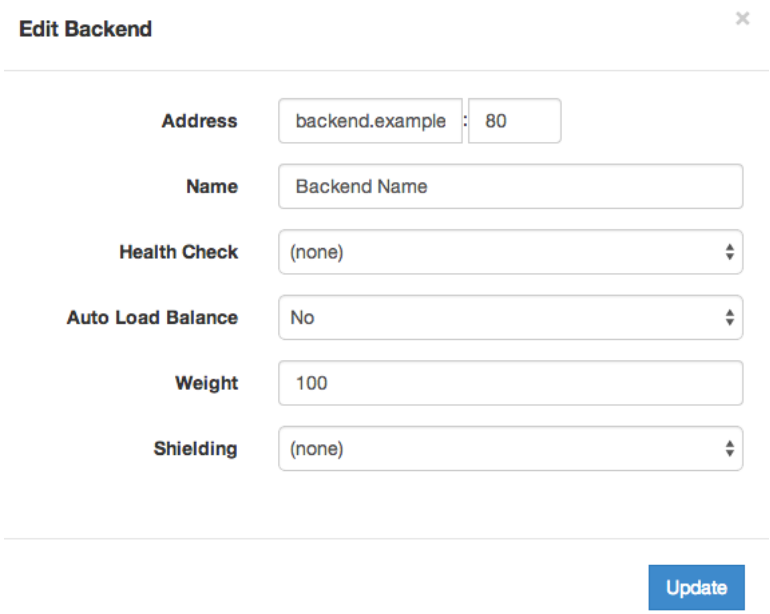
1. Log in to the Fastly application and click the **configure** button (wrench icon).
2. From the **Service** drop down menu, select the service for which shielding will be enabled and then click the blue **Configure** button (with the gear icon) to the right of the service name.



3. Click **Hosts** from the section list on the left.
4. In the **Backends** area, click the gear icon to the right of the appropriate backend and then select **Edit** from the drop down menu.



The Edit Backend window appears.

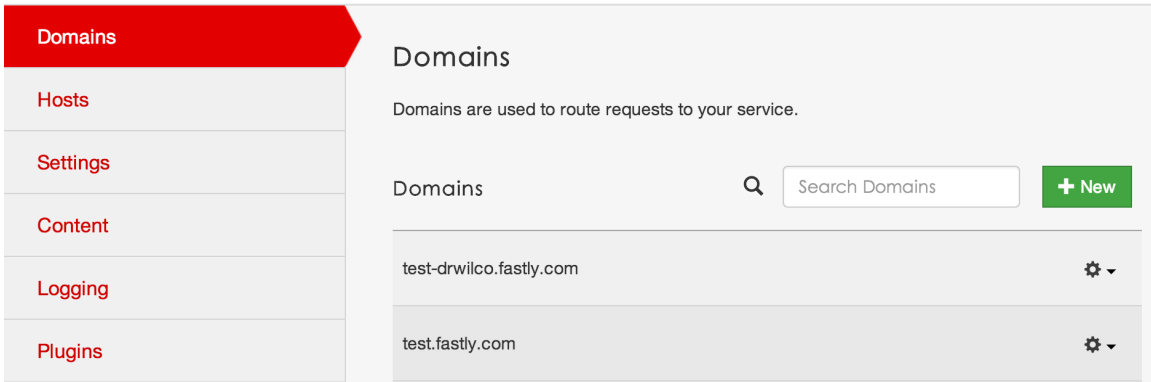


5. From the **Shielding** drop down menu select the datacenter to use as your shield and then click the blue **Update** button to save your changes.

   Note: 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.

6. Click **Domains** from the section list to the left and in the **Domains** area, check to make sure the hostname of the origin server is included in the service's domain list. If it isn't included, add it via the New button.

   With shielding enabled, queries from other POPs appear as incoming requests to the shield. If the shield doesn't know about the origin hostname, it doesn't know the service to which the request should be routed. Including the origin's hostname in the domain list helps eliminate this concern.



7. Deploy a new version to activate shielding.

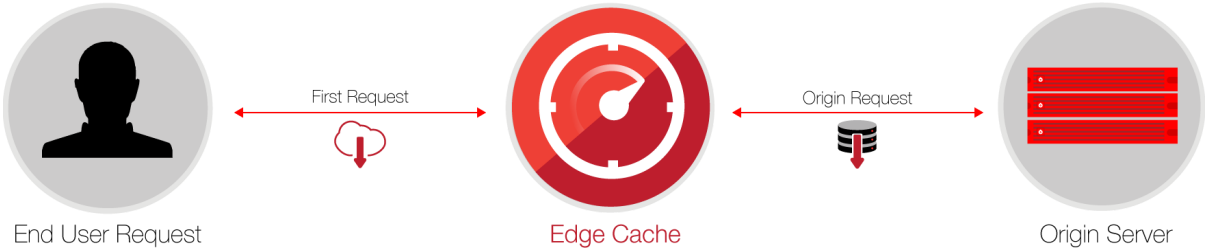
Fast and done! Don't forget to read our Caveats of Shielding (/guides/shielding/caveats-of-shielding) information.

· How POPs handle requests with and without shielding enabled (/guides/shielding/how-pops-handle-requests-with-and-without-shielding-enabled)

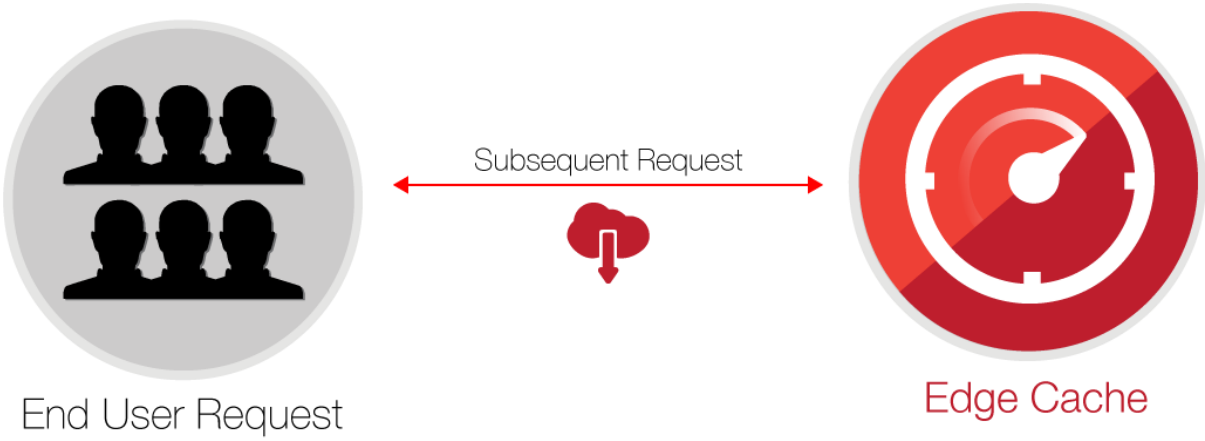
When a user requests brand new content from a customer's server and that content has never been cached by any Fastly POP (/guides/faqs/where-are-fastlys-pops-located), this is what happens with that request when shielding (/guides/shielding/) is and is not enabled.

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.



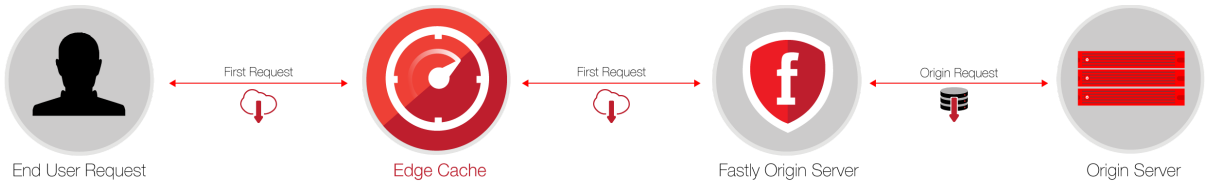
When a second request for that same content arrives at POP A, the content is already cached. No request goes to the customer's origin server. It's merely sent from the cached copy.



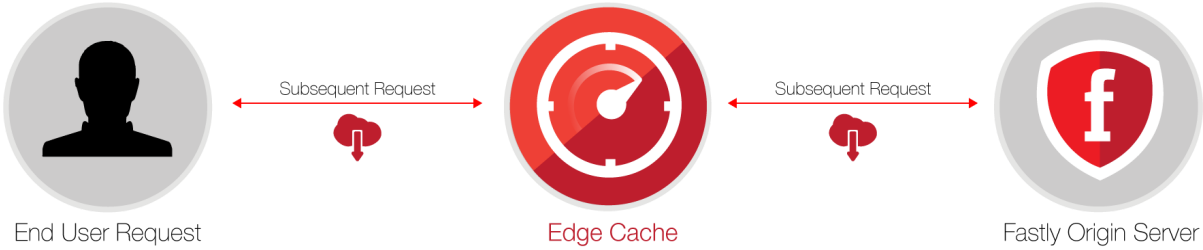
If the second request were to arrive at POP B instead of POP A, however, the request would once again be passed along to the customer's origin server. It would then be cached and passed back to the end user, just like POP A did when that info was first requested.

With shielding enabled

With shielding enabled (/guides/shielding/how-do-i-enable-shielding), 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.

· Understanding Cache HIT and MISS headers with Shielded Services (/guides/shielding/understanding-cache-hit-and-miss-headers-with-shielded-services)

Here's some help deciphering cache hit and miss headers when you have shielding enabled (/guides/shielding/). Let's look at the following requests for the same object.

The first request:

```
X-Served-By: cache-v43-ASH, cache-sv62-SJC3
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
```

Listing two cache-nodes (X-Served-By) shows that shielding is turned on, with cache-43-ASH serving as the cache node and cache-sv62-SJC3 serving as the local delivering node. The MISS,MISS (X-Cache) indicates that the requested object was neither in the shield cache nor the local delivering node. X-Cache-Hits reflects that same information.

The second request:

```
X-Served-By: cache-v43-ASH, cache-sv62-SJC3
X-Cache: MISS, HIT
X-Cache-Hits: 0, 1
```

This time, we actually hit the same local cache-node (cache-sv62-SJC3) and got a hit. The MISS listed for cache-43-ASH reflects the state of that node the last time it queried that node for that object (not its current state), which at that time was a MISS, but in fact, the object resides in that cache now because it was requested from origin.

After waiting a minute or two and requesting the same object again:

```
X-Served-By: cache-v37-ASH, cache-sv60-SJC3
X-Cache: HIT, MISS
X-Cache-Hits: 1, 0
```

This third request shows a new cache (cache-sv60-SJC3) being selected, MISSing, and retrieving the object from the shield node (cache-v37-ASH), which had the object (a HIT). In total, there should have only been a single request for this object to the origin.

NOTE: If the closest delivering cache node is in the shield datacenter, you will only see a single server and hit data such as:

```
X-Served-By: cache-v37-ASH
X-Cache: HIT
X-Cache-Hits: 1
```

After a purge of the object, requesting the object again will be similar the first request scenario.

```
X-Served-By: cache-v37-ASH, cache-sv61-SJC3
X-Cache: MISS, MISS
X-Cache-Hits: 0, 0
```

Advanced Configuration

• VCL FAQs (/guides/vcl-faqs/)

· Can I embed C code in VCL? (/guides/vcl-faqs/can-i-embed-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, please contact support@fastly.com (mailto:support@fastly.com). Our engineering team looks forward to these kinds of challenges.

· Can I override which IP address the GeoIP features use? (/guides/vcl-faqs/can-i-override-which-ip-address-the-geoip-features-use)

By default the GeoIP 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 (http://dev.opera.com/articles/opera-mini-request-headers/#x-forwarded-for). Similarly, the Amazon Silk browser (https://developer.amazon.com/appsandservices/community/post/TxOMW3RNF3FYRK/Amazon-Silk-Tips-for-Site-Owners) can optionally come through a proxy, indicated via the User Agent string.

To work around this, you can set this variable:

`set geoip.use_x_forwarded_for = true;`

To account for both the Opera Mini and Amazon Silk browsers, you would use code like this:

```
if (req.http.X-OperaMini-Features || req.http.User-Agent ~ " Silk-Accelerated=true$") {
    set geoip.use_x_forwarded_for = true;
}
```

which tells Fastly to use the X-Forwarded-For header 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 geoip.ip_override = req.http.Custom-IP-Override;`

Setting either of these variables:

`set geoip.use_x_forwarded_for = true;`

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

will force the GeoIP information to be reloaded.

REMEMBER: The ability to upload custom VCL (/guides/vcl-faqs/where-can-i-upload-custom-vcl)code is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

· Can I perform authentication before returning a request? (/guides/vcl-faqs/can-i-perform-authentication-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 (/guides/vcl-faqs/where-can-i-upload-custom-vcl):

```
sub vcl_recv {
  if (req.url ~ "^/protected/") {
    if (req.restarts == 0) {
      set req.http.X-Orig-Url = req.url;
      set req.url = "/checkauth";
      return(pass);
    } else {
      set req.url = req.http.X-Orig-Url;
      return(lookup);
    }
  }
#FASTLY recv
}

sub vcl_fetch {
  if (req.url == "/checkauth") {
    if (beresp.status == 200) {
      restart;
    } else {
      return(deliver);
    }
  }
#FASTLY fetch
}
```

REMEMBER: Change `"/checkauth"` to whatever your authentication end point is.

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)`.

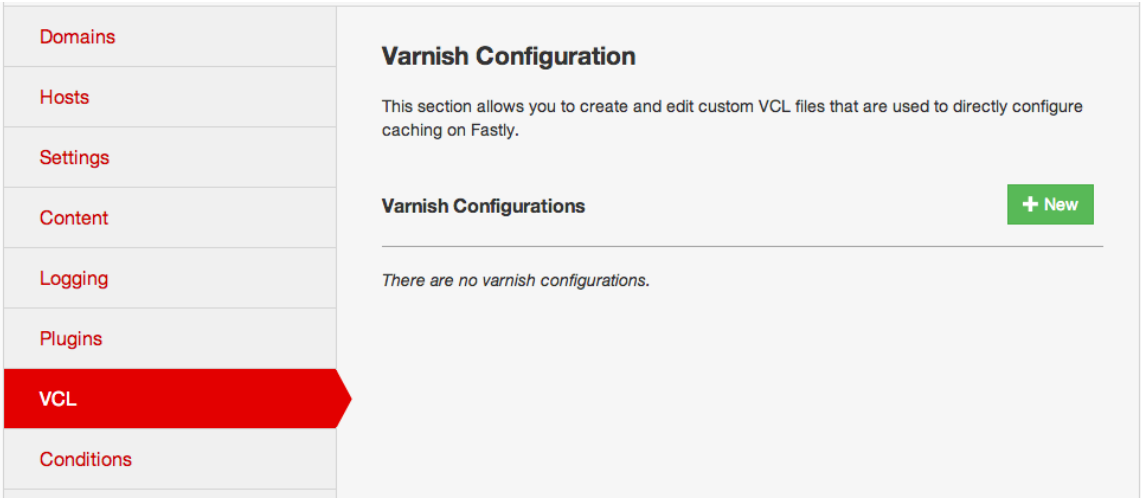
IMPORTANT: The ability to upload custom VCL code (/guides/vcl-faqs/where-can-i-upload-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

· Can I preview VCL before activating it? (/guides/vcl-faqs/can-i-preview-vcl-before-activating-it)

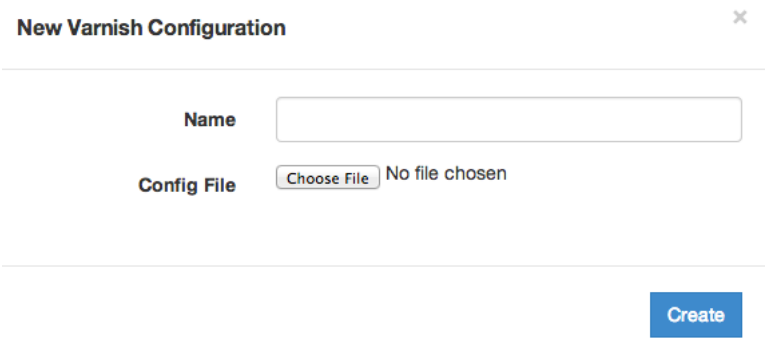
You can always preview VCL prior to activating a service version.

1. Log in to the Fastly application.
2. Click the **configure** tab to access the control panel.

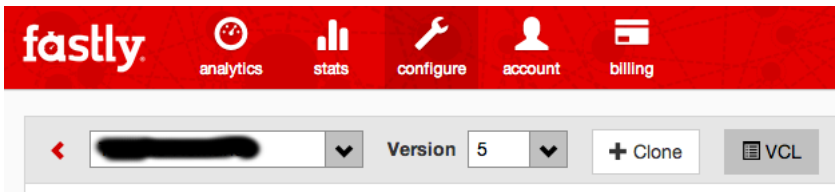
3. Select the appropriate service from the **Service** drop down.
4. Click the blue **Configure** button to the right of the service name.
5. Click the **VCL** section to view the Varnish Configuration information, and then click the **New** button to upload your configuration file.



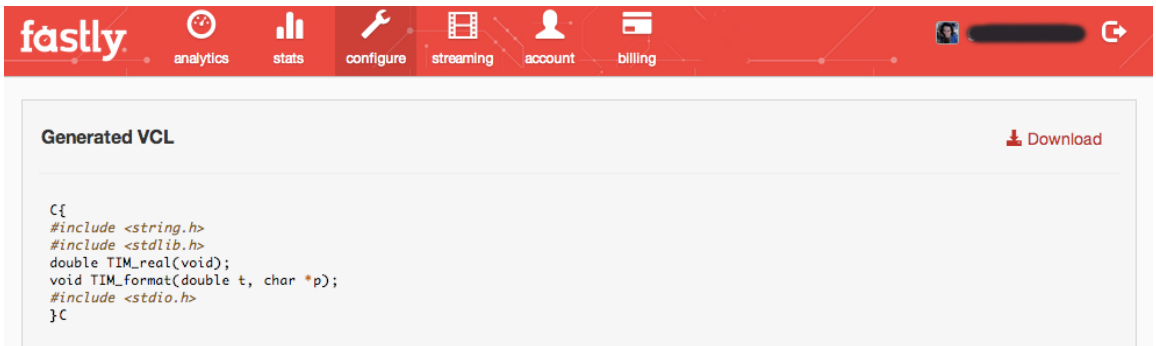
The New Varnish Configuration window appears:



6. In the **Name** field, type the name of the varnish configuration information you're uploading.
7. Click the **Choose File** button and navigate to the text file you plan to upload.
8. Click the **Create** button to create the new varnish configuration.
9. At the top of the application window, click the grey **VCL** button to the right of the service name and version fields.

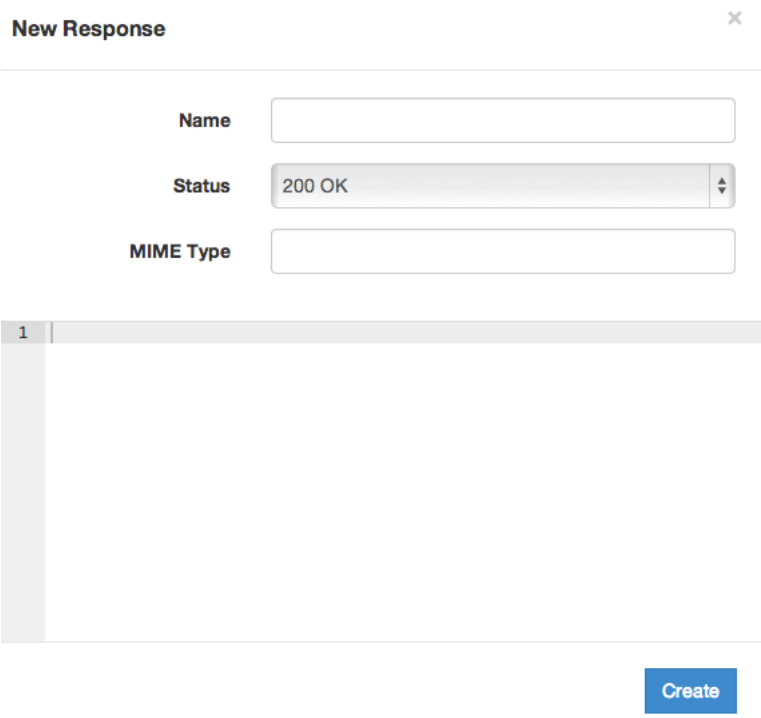


The Generated VCL preview appears.



· Custom responses that don't hit origin servers (/guides/vcl-faqs/custom-responses-that-dont-hit-origin-servers)

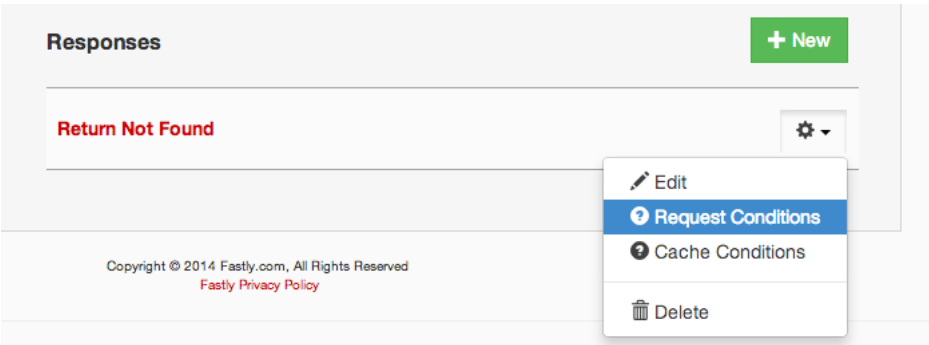
In certain situations you may want to send custom responses to clients that are handled on Fastly edge servers and do not hit origin servers (/guides/backend-servers/). In this example a user wanted to restrict caching to a URL subtree that contained images and scripts and did not want anything else cached. Requests to anything other than `/Content/` *or* `/Scripts/` were supposed to return HTTP 404 (Document not found). To implement this you will need to go to the Content Configuration section of your service then click on New next to Responses. A dialog will appear:



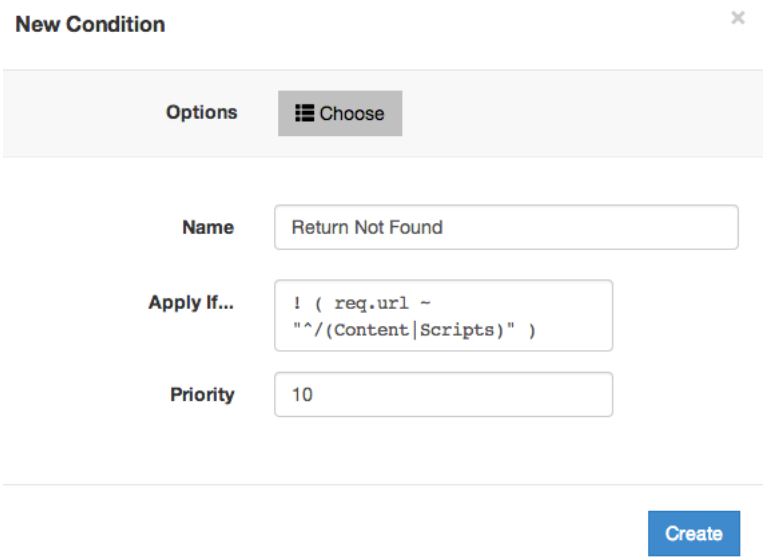
Fill out the fields as follows:

- **Name**: A descriptive name to show in the Service Configuration.
- **Status**: This is the HTTP code to return to the client (e.g., 200, 304, 404).
- **MIME Type**: The MIME type of the response.
- **Content**: Any type of HTML content. It doesn't need to be plain text.

Once the response object is created. You will need to attach a Request condition to it. Click on the gear icon and then select Request Conditions.



Enter your request condition in the New Condition window that appears. For example:



Be sure to fill out the fields as follows:

- **Name**: Descriptive name of a request condition visible in the App Configuration interface
- **Apply If:** This is the actual condition you want inserted into a VCL if statement
- **Priority:** Priority for the request condition. Leave as default if this the only request condition

When done click **Create**, deploy the service, and your settings should be deployed within short period of time.

Examples of request conditions

- True only if URLs don't match a certain mask: /Content/* or /Scripts/*

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

- True only if URLs match /secret/* or are Microsoft Word or Excel documents: *.doc and *.xls

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

- Ignore POST and PUT HTTP requests

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

- Deny spider/crawler with user-agent "annoying_robot"

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

- Deny specific IP or set of IPs from connecting: 225.0.0.1 and 240.24

```
client.ip == "225.0.0.1" || client.ip ~ "240.24"
```

· Does Fastly have a list of VCL extensions? (/guides/vcl-faqs/does-fastly-have-a-list-of-vcl-extensions)

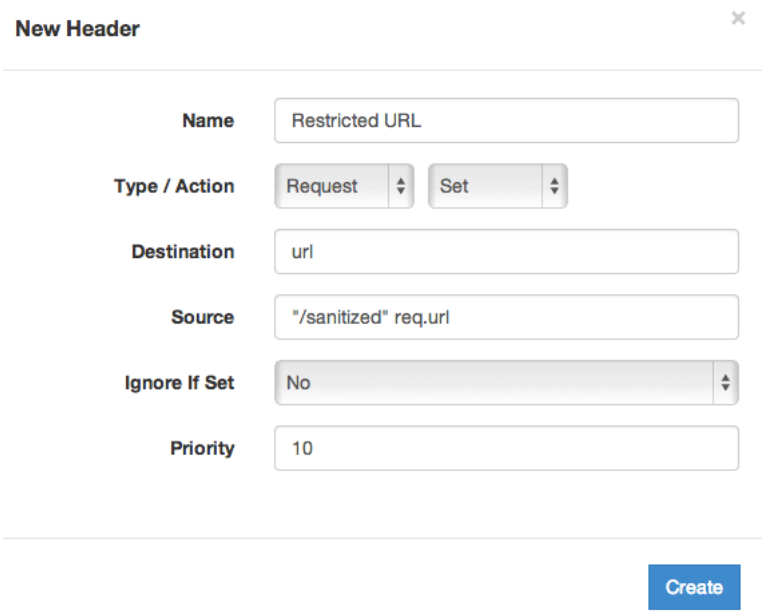
We've included a number of extensions to VCL. You can find them here (/guides/vcl-extensions/).

· How can I change the URL based on the domain? (/guides/vcl-faqs/how-can-i-change-the-url-based-on-the-domain)

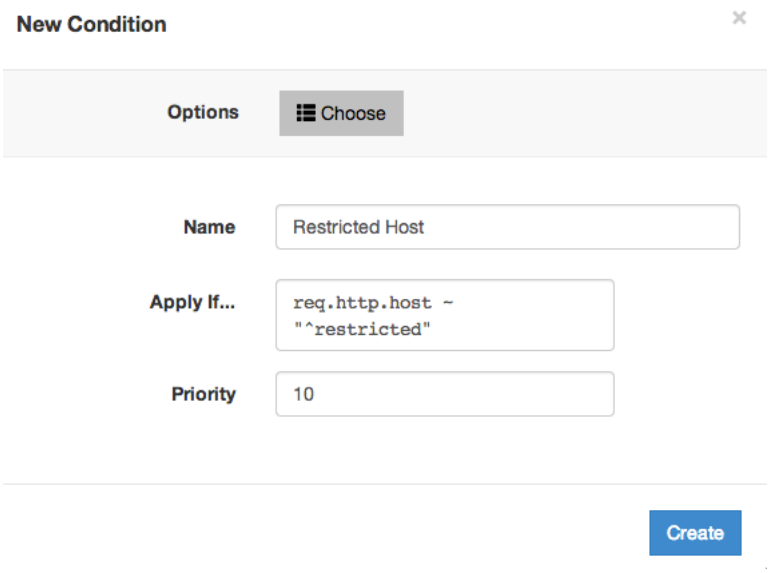
The VCL looks something like this

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

Which translates, in Fastly's UI, to creating a new Header (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses) in the Content pane of your control panel:



and then creating a request condition that restricts connections to that host:

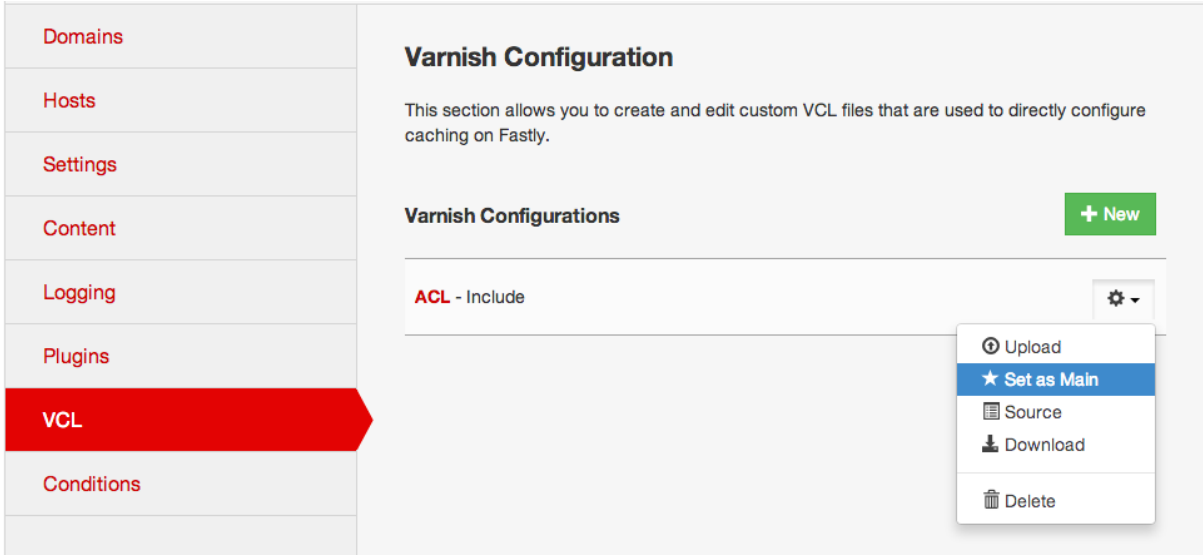


· How can I control access to certain resources? (/guides/vcl-faqs/how-can-i-control-access-to-certain-resources)

Varnish allows you to use Access Control Lists (ACLs), a feature that enables fast matching of the client's IP address against a list of defined IP addresses. An ACL looks like this:

```
# 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 dialin router */
}
```

To define an ACL in Fastly, first ask support (mailto:support@fastly.com) to turn on VCL uploading (/guides/vcl-faqs/where-can-i-upload-custom-vcl) for your account. Once activated, create a file with your ACL definitions and upload it in the VCL configuration section. Make sure the uploaded file is marked as a 'Main' type using the menu to the right of the name (not 'Include'):



Once uploaded, you can reference it in your configuration with a match operation:

```
# block any requests to Admin pages not from local IPs
sub vcl_recv {
  if (req.url ~ "^/admin" && ! (client.ip ~ local)) {
    error 403 "Forbidden";
  }
}
```

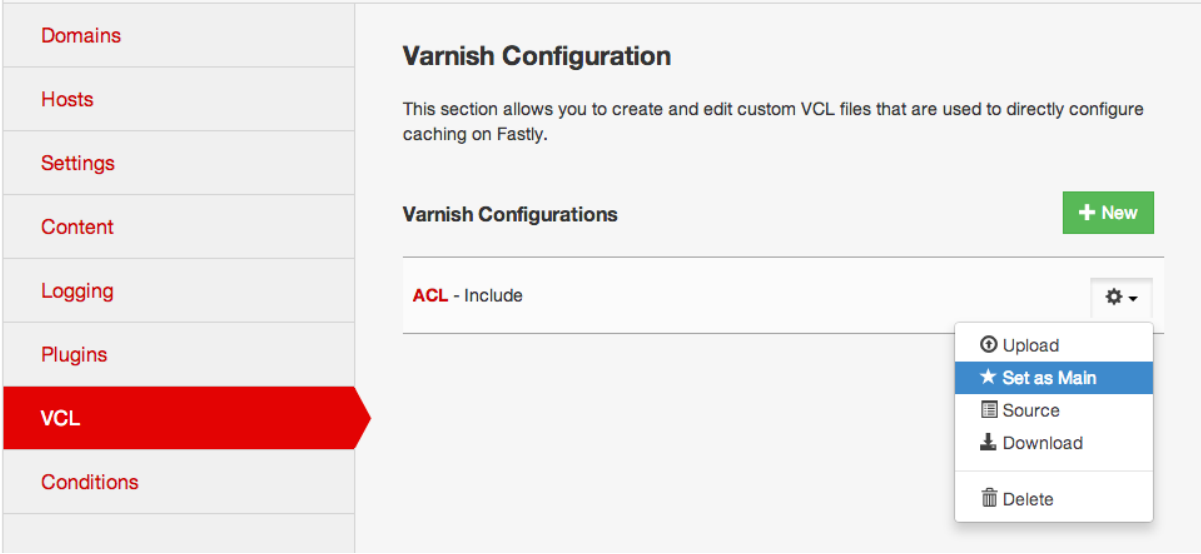
You can, of course, create different behavior based on any other attributes from a request as well, such as location and cookie presence.

· How do I create location-based tagging? (/guides/vcl-faqs/how-do-i-create-locationbased-tagging)

Varnish allows you to use Access Control Lists (ACLs), a feature that enables fast matching of the client's IP address against a list of defined IP addresses. An ACL looks like this:

```
# 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 dialin router */
}
```

To define an ACL in Fastly, first ask support (mailto:support@fastly.com) to turn on VCL uploading (/guides/vcl-faqs/where-can-i-upload-custom-vcl) for your account. Once activated, create a file with your ACL definitions and upload it in the VCL configuration section. Make sure the uploaded file is marked as a 'Main' type using the menu to the right of the name (not 'Include'):



Once uploaded, you can reference it in your configuration with a match operation:

```
# block any requests to Admin pages not from local IPs
sub vcl_recv {
  if (req.url ~ "^/admin" && client.ip !~ local) {
    error 403, "Forbidden";
  }
}
```

You can, of course, create different behavior based on any other attributes from a request as well, such as location and cookie presence.

· How do I escape characters in logs? (/guides/vcl-faqs/how-do-i-escape-characters-in-logs)

Logs respond to VCL (/guides/vcl-faqs/) like any other object in Fastly. For example, the following code can escape quotes from User-Agent your log stream:

```
log {"syslog serviceid endpointname :: "} {"""} regsuball(req.http.user-agent, {"""}, {"\"}) {"""};
```

· How do I mix and match Fastly VCL with custom VCL? (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl)

IMPORTANT: The ability to upload custom VCL code (/guides/vcl-faqs/where-can-i-upload-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Fastly's VCL parser has its own pre-processor. To use the auto-generated VCL code Fastly supplies you can put

```
sub vcl_recv {
  # my custom code
  if (req.http.User-Agent ~ "Googlebot") {
    set req.backend = F_special_google_backend;
  }

#FASTLY recv
}
```

or the equivalent for other functions.

To mix and match Fastly VCL with your custom VCL successfully, remember the following:

- **Fastly's VCL is based on Varnish 2.1.** Our Varnish syntax is specifically compatible with Varnish 2.1.5. We run a custom version with added functionality.
- **You can only restart Varnish three times.** This limit exists to prevent infinite loops.
- **VCL doesn't take kindly to Windows newlines (line breaks).** It's best to avoid them entirely.
- **It's best to use `curl -X PURGE` to initiate purges via API (/api/purge).** To restrict access to purging, check for the `FASTLYPURGE` method not the `PURGE` method. When you send a request to Varnish to initiate a purge, the HTTP method that you use is "PURGE", but it has already been changed to "FASTLYPURGE" by the time your VCL runs that request.
- **If you override TTLs with custom VCL, your default TTL set in the configuration (/guides/caching/how-can-i-serve-stale-content-on-errors) will not be honored** and the expected behavior may change.
- **Don't override the default VCL if that's not your intent.** To avoid overriding the default VCL accidentally, include all of the boilerplate shown below as a template in your custom VCL file. Add your custom code *in between* the different sections.

Fastly's VCL Boilerplate

IMPORTANT: Include all of the boilerplate shown below as a template in your custom VCL file. Add your custom code *in between* the different sections.

```
sub vcl_recv {
#FASTLY recv

    if (req.request != "HEAD" && req.request != "GET" && req.request != "FASTLYPURGE") {
      return(pass);
    }

    return(lookup);
}

sub vcl_fetch {
#FASTLY fetch

  if ((beresp.status == 500 || beresp.status == 503) && req.restarts < 1 && (req.request == "GET" || req.request == "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);
  }

  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
}

sub vcl_pass {
#FASTLY pass
}
```

· How does Fastly support the Edge-Control header? (/guides/vcl-faqs/how-does-fastly-support-the-edgecontrol-header)

VCL now gives you the building blocks to access information inside the Edge-Control response header field from the origin. Fastly supports 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, you would include the following snippet in your `vcl_fetch` function via custom VCL:

```
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"));
}
```

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. In summary, the above snippet honors `cache-maxage` and `downstream-ttl` from Edge-Control if present and usable.

REMEMBER: The ability to upload custom VCL code (/guides/vcl-faqs/where-can-i-upload-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

· Setting Up Redundant Origin Servers using VCL (/guides/vcl-faqs/setting-up-redundant-origin-servers-using-vcl)

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 with custom VCL (/guides/vcl-faqs/where-can-i-upload-custom-vcl). The ability to upload custom VCL code (/guides/vcl-faqs/where-can-i-upload-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

Once you can upload custom VCL configurations, create a custom configuration that looks something like this:

```
sub vcl_recv {
#FASTLY recv
    set req.backend = F_<origin_server_1>;
    if (!req.backend.healthy) {
      set req.backend = F_<origin_server_2>;
    }
    if (req.request != "HEAD" && req.request != "GET" && req.request != "PURGE") {
      return(pass);
    }
    return(lookup);
}
```

REMEMBER: Health checks (/guides/backend-servers/health-checks-tutorial) should in be set in the app in conjunction with this as a best practice.

· VCL Regular Expression Cheat Sheet (/guides/vcl-faqs/vcl-regular-expression-cheat-sheet)

Remember that Varnish (/guides/vcl-faqs/) regular expressions are case sensitive.

Basic matching

```
req.url ~ "phrase"
```

Matches "phrase" anywhere.

```
req.url == "phrase"
```

Matches only if req.url is exactly "phrase".

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 "jpg", "jpeg", "css" or "js".

```
req.url ~ "\.php(\?.*)?$"
```

Matches if req.url ends with ".php" ".php?foo=bar" or ".php?", but not ".phpa".

```
req.url ~ "\.[abc]server$"
```

Matches if req.url ends with either ".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".

```
req.url ~ "jp.*g$"
```

Matches jp followed by 0 or more random characters ending with the letter 'g' ("jpeg", "jpg", and "jpeeeeg" all match).

Replacing content

```
set req.http.host = regsub(req.http.host, "^www\.","");
```

Replaces a leading "www." in the Host-header with a blank, if present.

```
set req.http.x-API-Test = regsub(req.http.host, "^www.","api.");
```

Sets the x-API-Test header to contain the host-header, but replaces a leading "www." with "api" example:

```
Host: www.example.com ->
Host: www.example.com
X-API-Test: api.example.com
Host: example.com ->
Host: example.com
X-API-Test: example.com
```

· What is Varnish and why does Fastly use it? (/guides/vcl-faqs/what-is-varnish-and-why-does-fastly-use-it)

Varnish is the open source software (http://www.fastly.com/about/open-source/) Fastly commercialized with performance and capacity (among other) enhancements. The principal configuration mechanism of Varnish software is the Varnish Configuration Language (VCL), the scripting language used to configure and add logic to Varnish caches. Varnish allows Fastly to apply changes to the cache software as it is executing. Specifically, VCL is generated, compiled, transmitted to all Fastly caches, loaded into the operating software, and activated immediately, with no waiting for maintenance windows and no server downtime.

The Fastly application generates VCL automatically per a customer's specifications via the web interface (/guides/fastlys-web-interface/). Fastly also allows customers to create their own VCL files with their specialized configurations, or to mix and match (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl) Fastly's VCL with custom VCL. Customer-written VCL files can be uploaded (/guides/vcl-faqs/where-can-i-upload-custom-vcl) into Fastly caches and activated. The ability to upload custom (/guides/vcl-faqs/where-can-i-upload-custom-vcl) VCL code is disabled by default when customers sign up for Fastly's CDN Service. To enable this ability for an account, customers must contact support@fastly.com (mailto:support@fastly.com) and specifically request it.

· What's the best way to QA test Fastly configurations? Should I create a second account? (/guides/vcl-faqs/whats-the-best-way-to-qa-test-fastly-configurations-should-i-create-a-second-account)

You don't need a second account to test your VCL configurations with the Fastly application. We recommend adding a new service within your existing account that's specifically designed for testing. A name like "QA" or "testing" or "staging" makes distinguishing between services much easier.

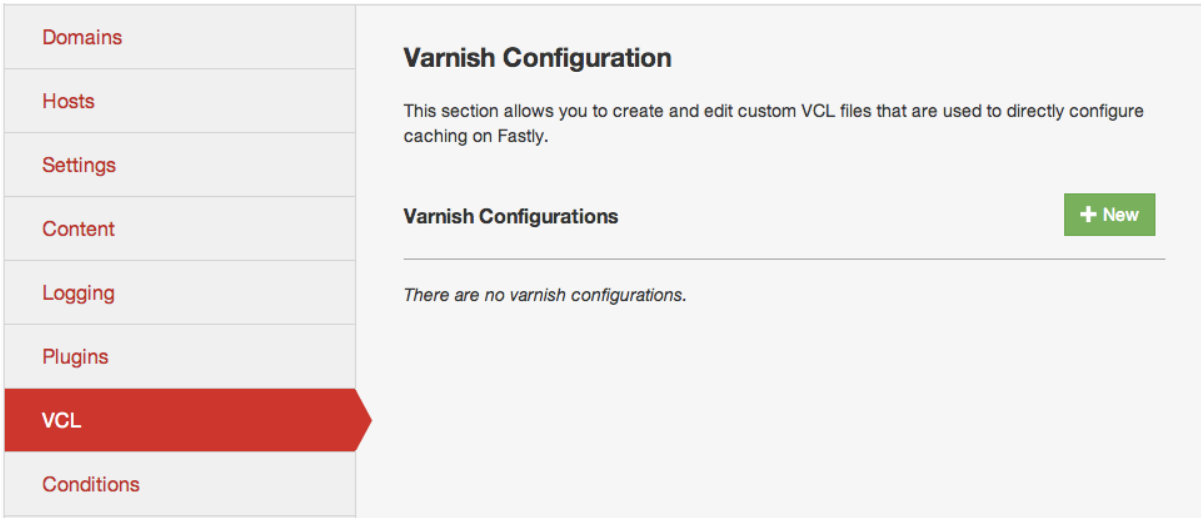
Once created, simply point your testing service to your testing or QA environment. Edit your Fastly configurations for the testing service as if you were creating them for production. Test things out and tweak them to get them perfect. You can also preview your generated VCL (/guides/vcl-faqs/can-i-preview-vcl-before-activating-it).

When your testing is complete, make the same changes in your production service that you made to your testing service. If you are using custom VCL, upload the VCL file (/guides/vcl-faqs/where-can-i-upload-custom-vcl) to the production service you'll be using.

REMEMBER: The ability to upload custom VCL code (/guides/vcl-faqs/where-can-i-upload-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

· Where can I upload custom VCL? (/guides/vcl-faqs/where-can-i-upload-custom-vcl)

The ability to upload custom VCL code is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it. Once enabled, you'll find the controls to upload custom VCL in the configuration controls, under a new VCL area as shown below.



The first VCL file that you upload will be marked as **Main**. Any other VCL files will be set as **Include**. They must be included from the main VCL using the syntax

```
include "VCL Name"
```

Where *VCL Name* is the name of an "Include" VCL object you've created. So if you've created an "Include" VCL object called "Extra Functions" using the file extra_functions.vcl then your "Main" VCL will have

```
include "Extra Functions"
```

When uploading custom VCL code you should read our guide to mixing-and-matching Fastly VCL with custom VCL (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl).

· Will I lose the ability to configure things via the UI if I activate custom VCL uploading? (/guides/vcl-faqs/will-i-lose-the-ability-to-configure-things-via-the-ui-if-i-activate-custom-vcl-uploading)

Custom VCL and Fastly VCL can be used 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. Our article on mixing and matching custom VCL with Fastly VCL (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl) talks about this in more detail.

REMEMBER: The ability to upload custom VCL code (/guides/vcl-faqs/where-can-i-upload-custom-vcl) is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

• VCL Extensions (/guides/vcl-extensions/)

· Cryptographic- and hashing-related VCL functions (/guides/vcl-extensions/cryptographic-and-hashingrelated-vcl-functions)

Fastly provides several functions in VCL (/guides/vcl-faqs/) for cryptographic- and hashing-related purposes. It is based very heavily on Kristian Lyngstøl's digest vmod (https://github.com/varnish/libvmod-digest) for Varnish 3 (which means you can also refer to that documentation for more detail).

Functions

Function	Description
`digest.hmac_md5(<key>,<message>)`	Hash-based message authentication code (http://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using MD5. Returns a Hex-encoded string prepended with 0x.
`digest.hmac_sha1(<key>, <message>)`	Hash-based message authentication code (http://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using SHA1 (http://en.wikipedia.org/wiki/Secure_Hash_Algorithm). Returns a Hex-encoded string prepended with 0x.
`digest.hmac_sha256(<key>, <message>))`	Hash-based message authentication code (http://en.wikipedia.org/wiki/Hash-based_message_authentication_code) using SHA256 (http://en.wikipedia.org/wiki/Secure_Hash_Algorithm). Returns a Hex-encoded string prepended with 0x.
`digest.base64(<string>)`	Base64 (http://en.wikipedia.org/wiki/Base64) encoding. Returns the base64-encoded version of the input-string.
`digest.base64url(<string>)`	Base64 (http://en.wikipedia.org/wiki/Base64#URL_applications) encoding. Returns the base64-encoded version of the input-string. Replaces +/ with -_ for url safety.
`digest.base64url_nopad(<string>)`	Base64 (http://en.wikipedia.org/wiki/Base64#URL_applications) encoding. Returns the base64-encoded version of the input-string. Replaces +/ with -_ for url safety. Has no length padding (http://en.wikipedia.org/wiki/Base64#Padding).
`digest.base64_decode(<string>)`	Decode Base64. Returns a string.
`digest.base64url_decode(<string>)`	Decode Base64 with url safe characters in. Returns a string.
`digest.base64url_nopad_decode(<string>)`	Decode Base64 with url safe characters. Returns a string. Identical to base64_url_decode.
	Use the SHA1

`digest.hash_sha1(<string>)`	(http://en.wikipedia.org/wiki/Secure_Hash_Algorithm) hash. Returns a string.
`digest.hash_sha224(<string>)`	Use the SHA224 (http://en.wikipedia.org/wiki/Secure_Hash_Algorithm) hash. Returns a string.
`digest.hash_sha256(<string>)`	Use the SHA256 (http://en.wikipedia.org/wiki/Secure_Hash_Algorithm) hash. Returns a string.
`digest.hash_sha384(<string>)`	Use the SHA384 (http://en.wikipedia.org/wiki/Secure_Hash_Algorithm) hash. Returns a string.
`digest.hash_sha512(<string>)`	Use the SHA512 (http://en.wikipedia.org/wiki/Secure_Hash_Algorithm) hash. Returns a string.
`digest.hash_gost(<string>)`	Use the Gost (http://en.wikipedia.org/wiki/GOST_(hash_function)) hash function. Returns a string.
`digest.hash_md2(<string>)`	Use the MD2 (http://en.wikipedia.org/wiki/MD2_(cryptography)) hash. Returns a string.
`digest.hash_md4(<string>)`	Use the MD4 (http://en.wikipedia.org/wiki/MD4) hash. Returns a string.
`digest.hash_md5(<string>)`	Use the MD5 (http://en.wikipedia.org/wiki/MD5) hash. Returns a string.
`digest.hash_crc32(<string>)`	Use a 32 bit Cyclic Redundancy Checksum (http://en.wikipedia.org/wiki/CRC32). Returns a string.
`digest.hash_crc32b(<string>)`	A reversed CRC32 (for compatibility with some PHP applications (http://www.php.net/manual/en/function.hash-file.php#104836)). Returns a string.
`digest.hash_adler32(<string>)`	Use the Adler32 (http://en.wikipedia.org/wiki/Adler32) checksum algorithm. Returns a string.
`digest.hash_haval128(<string>)`	Use the 128 bit Haval (http://en.wikipedia.org/wiki/Haval) hash function. Returns a string.
`digest.hash_haval160(<string>)`	Use the 160 bit Haval (http://en.wikipedia.org/wiki/Haval) hash function. Returns a string.
`digest.hash_haval192(<string>)`	Use the 192 bit Haval (http://en.wikipedia.org/wiki/Haval) hash function. Returns a string.
`digest.hash_haval224(<string>)`	Use the 224 bit Haval (http://en.wikipedia.org/wiki/Haval) hash function. Returns a string.
`digest.hash_haval256(<string>)`	Use the 256 bit Haval (http://en.wikipedia.org/wiki/Haval) hash function. Returns a string.
	Use the 128 bit RIPMD

`digest.hash_ripemd128(<string>)`	(http://en.wikipedia.org/wiki/RIPEMD) hash function. Returns a string.
`digest.hash_ripemd160(<string>)`	Use the 160 bit RIPMD (http://en.wikipedia.org/wiki/RIPEMD) hash function. Returns a string.
`digest.hash_ripemd256(<string>)`	Use the 256 bit RIPMD (http://en.wikipedia.org/wiki/RIPEMD) hash function. Returns a string.
`digest.hash_ripemd320(<string>)`	Use the 320 bit RIPMD (http://en.wikipedia.org/wiki/RIPEMD) hash function. Returns a string.
`digest.hash_tiger(<string>)`	Use the standard, 192 bits Tiger (http://en.wikipedia.org/wiki/Tiger_(cryptography)) hash function. Returns a string.
`digest.hash_tiger128(<string>)`	Use the truncated 128 bit Tiger (http://en.wikipedia.org/wiki/Tiger_(cryptography)) hash function. Returns a string.
`digest.hash_tiger160(<string>)`	Use the truncated 160 bit Tiger (http://en.wikipedia.org/wiki/Tiger_(cryptography)) hash function. Returns a string.
`digest.hash_snefru128(<string>)`	Use the 256 bit Snefru (http://en.wikipedia.org/wiki/Snefru) hash function. Returns a string.
`digest.hash_snefru256(<string>)`	Use the 128 bit Snefru (http://en.wikipedia.org/wiki/Snefru) hash function. Returns a string.
`digest.hash_whirlpool(<string>)`	Use the Whirlpool (http://en.wikipedia.org/wiki/Whirlpool_(cryptography)) hash function. Returns a string.
`digest.time_hmac_md5(<base64 encoded key>, <interval>, <offset>)`	Time based One Time Password using MD5. Returns base64 encoded output.
`digest.time_hmac_sha1(<base64 encoded key>, <interval>, <offset>)`	Time based One Time Password using SHA1. Returns base64 encoded output.
`digest.time_hmac_sha256(<base64 encoded key>, <interval>, <offset>)`	Time based One Time Password using SHA256. Returns base64 encoded output.

Notes

A note about base64 decoding - theoretically the output could be in binary but is interpreted as a string. So if the binary output contains '\0' then it could be truncated.

The time based One-Time Password algorithm initializes the HMAC using the key and appropriate hash type. Then it hashes the message

`(<time now in seconds since UNIX epoch> / <interval>) + <offset>`

as a 64bit unsigned integer (little endian) and base64 encodes the result.

Examples

One-Time Password Validation (Token Authentication)

Use this to validate tokens with a URL format like the following:

`http://cname-to-fastly/video.mp4?6h2YUl1CB4C50SbkZ0E6U3dZGjh+84dz3+Zope2Uhik=`

Example implementations for token generation in various languages can be found in Github (https://github.com/fastly/token-functions).

Example VCL

```
sub vcl_recv {
#FASTLY recv

 set req.http.token = regsub(req.url, ".*\?(.*)$", "\1");
 if (req.http.token != digest.time_hmac_sha256("RmFzdGx5IFRva2VuIFRlc3Q=", 60, 0) &&
 req.http.token != digest.time_hmac_sha256("RmFzdGx5IFRva2VuIFRlc3Q=", 60, -1)) {
 error 403;
 }
}
```

Signature

`set resp.http.x-data-sig = digest.hmac_sha256("secretkey",resp.http.x-data);`

Base64 decoding

`synthetic { digest.base64_decode(req.http.x-parrot) };`

· Date- and time-related VCL features (/guides/vcl-extensions/date-and-timerelated-vcl-features)

By default VCL provides the `now` variable, which provides the current time in [(for example, `Wed, 17 Sep 2014 23:19:06 GMT`).

Fastly also adds several new functions to Varnish and variables that allow you to have more flexibility when dealing with dates and times.

The new variables are:

Name	Location	Description
`now.sec`	all	Like the `now` variable but in seconds.
`time.start`	all	The time the request started in seconds since the Epoch.
`time.start.sec`	all	The time the request started in seconds since the Epoch.
`time.start.msec`	all	The time the request started in milliseconds since the Epoch.
`time.start.msec_frac`	all	The time the request started in fractional milliseconds.
`time.start.usec`	all	The time the request started in microseconds since the Epoch.
`time.start.usec_frac`	all	The time the request started in fractional microseconds since the Epoch.
`time.end`	deliver log	The time the request ended in RFC 850 format (https://www.ietf.org/rfc/rfc1123.txt) format. Also useful for strftime.
`time.end.sec`	deliver log	The time the request ended in seconds since the Epoch.
`time.end.msec`	deliver log	The time the request ended in milliseconds since the Epoch.
`time.end.msec_frac`	deliver	The time the request ended in fractional milliseconds since the Epoch.

	log	
`time.end.usec`	deliver log	The time the request ended in microseconds since the Epoch.
`time.end.usec_frac`	deliver log	The time the request ended in fractional microseconds since the Epoch.
`time.elapsed`	deliver log	The time since the request start in RFC 850 format (https://www.ietf.org/rfc/rfc1123.txt). Also useful for strftime.
`time.elapsed.sec`	deliver log	The time since the request start in seconds since the Epoch.
`time.elapsed.msec`	deliver log	The time since the request start in milliseconds since the Epoch.
`time.elapsed.msec_frac`	deliver log	The time the since request start in fractional milliseconds since the Epoch.
`time.elapsed.usec`	deliver log	The time since the request start in microseconds since the Epoch.
`time.elapsed.usec_frac`	deliver log	The time since the request start in fractional microseconds since the Epoch.

There are also a few functions:

Name	Description
`time.hex_to_time(<divider>, <hextime>)`	Takes a hexadecimal string value, divides by `divider` and interprets the result as seconds since UNIX epoch.
`time.add(<time>, <offset>)`	Adds `offset` to `time`.
`time.sub(<time>, <offset>)`	Subtracts `offset` from `time`.
`time.is_after(<time1>, <time2>)`	Returns `TRUE` if `time1` is after `time2`. (Normal timeflow and causality required.)
`strftime`	Format a time to a string. This uses the standard POSIX strftime formats (http://www.unix.com/man-page/FreeBSD/3/strftime/)).

Examples:

```
set resp.http.Now = strftime({"%Y-%m-%d %H:%M"}, now)
set resp.http.Start = strftime({"%a, %d %b %Y %T %z"}, time.start)
if (time.is_after(time.add(now, 10m), time.hex_to_time(1, "d0542d8")) {
  ...
}
```

> Note the way the format string is written `{"..."}`, which is a VCL "long string". Regular strings in VCL use `%xx` escapes (percent encoding) for special characters, which would conflict with the `%` used in the strftime format. Alternatively, one could use `%25` for each `%`.

· GeoIP-related VCL features (/guides/vcl-extensions/geoiprelated-vcl-features)

Fastly exposes a number of geographic variables for you to take advantage of inside VCL (/guides/vcl-extensions/guide-to-vcl). These are based on the MaxMind IP Geolocation database and appear as follows:

Variable	Description
`geoip.latitude`	The latitude associated with the IP address.
`geoip.longitude`	The longitude associated with the IP address.

`geoip.city`	The city or town name associated with the IP address. See the MaxMind list of cities (http://www.maxmind.com/GeoIPCity-534-Location.csv) to see all the possible return values. Their list is updated on a regular basis.
`geoip.continent_code`	A two-character code representing the continent associated with the IP address. Possible codes are: **AF** - Africa, **AS** - Asia, **EU** - Europe, **NA** - North America, **OC** - Oceania, **SA** - South America.
`geoip.country_code`	A two-character ISO 3166-1 (http://en.wikipedia.org/wiki/ISO_3166-1)country code for the country associated with the IP address. In addition to the standard codes, we may also return one of the following: **A1** - an anonymous proxy (http://dev.maxmind.com/faq/geoip#anonproxy), **A2** - a satellite provider (http://dev.maxmind.com/faq/geoip#satellite), **EU** - an IP in a block used by multiple European (http://dev.maxmind.com/faq/geoip#euapcodes) countries, **AP** - an IP in a block used by multiple Asia/Pacific region (http://dev.maxmind.com/faq/geoip#euapcodes) countries. Note: The **US** country code is returned for IP addresses associated with overseas US military bases.
`geoip.country_code3`	The same as country_code; however, it returns a ISO 3166-1 alpha-3 (http://en.wikipedia.org/wiki/ISO_3166-1_alpha-3) three-character code.
`geoip.country_name`	The country name associated with the IP address.
`geoip.postal_code`	The postal code associated with the IP address. These are available for some IP addresses in Australia, Canada, France, Germany, Italy, Spain, Switzerland, United Kingdom, and the US. We return the first 3 characters for Canadian postal codes. We return the first 2-4 characters (outward code) for postal codes in the United Kingdom.
`geoip.region`	The region name associated with the IP address.
`geoip.area_code`	The telephone area code associated with the IP address. These are only available for IP addresses in the US.
`geoip.metro_code`	The metro code associated with the IP address. These are only available for IP addresses in the US. MaxMind returns the same metro codes as the Google AdWords API (https://developers.google.com/adwords/api/docs/appendix/metrocodes).

Fastly also exposes codes that describe the location of the datacenter the request came through as follows:

Variable	Description
`server.region`	A code representing the general region of the world in which the POP location resides. One of: Asia, EU-Central, EU-East, EU-West, US-Central, US-East, US-West.
`server.datacenter`	A code representing one of Fastly's POP locations (/guides/faqs/where-are-fastlys-pops-located).

REMEMBER: All strings are returned as ISO-8859-1 encoding (http://en.wikipedia.org/wiki/ISO/IEC_8859-1) (also known as latin1).

· Guide to VCL (/guides/vcl-extensions/guide-to-vcl)

Fastly's Varnish is based on Varnish 2.1 and our Varnish syntax is specifically compatible with Varnish 2.1.5. We run a custom version with added functionality.

Online Varnish Software and VCL Documentation

The official Varnish documentation is a good place to start when looking for online information:

https://www.varnish-cache.org/docs/2.1/tutorial/vcl.html (https://www.varnish-cache.org/docs/2.1/tutorial/vcl.html)

In addition, Varnish Software, who provides commercial support for Varnish, has written a free book:

https://www.varnish-software.com/static/book/index.html (https://www.varnish-software.com/static/book/index.html)

Specifically, the VCL references are located here:

https://www.varnish-software.com/static/book/VCL_Basics.html (https://www.varnish-software.com/static/book/VCL_Basics.html)

https://www.varnish-software.com/static/book/VCL_functions.html (https://www.varnish-software.com/static/book/VCL_functions.html)

Books on Varnish and VCL

Roberto Moutinho has written a book on using Varnish that may help:

http://www.amazon.com/Instant-Varnish-Cache-Roberto-Moutinho/dp/178216040X/ (http://www.amazon.com/Instant-Varnish-Cache-Roberto-Moutinho/dp/178216040X/)

Fastly's 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
cryptographic and hashing functions (/guides/vcl-extensions/cryptographic-and-hashingrelated-vcl-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 (/guides/vcl-extensions/date-and-timerelated-vcl-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.
GeoIP features (/guides/vcl-extensions/geoiprelated-vcl-features)	Provides the ability to search a database from MaxMind.com for a given host or IP address, and return information about the country, city or Internet Service Provider (ISP) for that IP address.
randomness features (/guides/vcl-extensions/randomnessrelated-vcl-features)	Supports the insertion of random strings, content cookies, and decisions into requests.
size-related variables (/guides/vcl-extensions/sizerelated-vcl-variables)	Supports reporting variables that offer insight into what happened in a request.
miscellaneous features and variables (/guides/vcl-extensions/miscellaneous-VCL-extensions)	Provides miscellaneous VCL extensions not easily grouped into other categories.

REMEMBER: The ability to upload custom VCL (/guides/vcl-faqs/where-can-i-upload-custom-vcl)code is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

· Miscellaneous VCL features (/guides/vcl-extensions/miscellaneous-VCL-extensions)

Fastly has added several miscellaneous features (/guides/vcl-extensions/guide-to-vcl) to Varnish that don't easily fit into specific categories (/guides/vcl-extensions/).

Feature	Description
`req.restarts`	Counts the number of times the VCL has been restarted.
`req.grace`	Defines how long an object can remain overdue and still have Varnish consider it for grace mode.

`std.toupper`	Changes the case of a string to upper case. For example, `std.toupper("hello");` will return `"HELLO"`.
`std.tolower`	Changes the case of a string to lower case. For example, `std.tolower("HELLO");` will return `"hello"`.
`urlencode`	Encodes a string for use in a URL. This is also known as percent-encoding (http://en.wikipedia.org/wiki/Percent-encoding). For example, `urlencode("hello world");` will return `"hello%20world"`.
`urldecode`	Decodes a percent-encoded string. This is the counterpart to `urldecode`. For example, `urldecode("hello%20world+!");` will return `"hello world !"`.
`http_status_matches`	Determines whether or not an HTTP status code matches a pattern. The arguments are an integer (usually `beresp.status` or `resp.status`) and a comma-separated list of status codes, optionally prefixed by a `!` to negate the match. It returns `TRUE` or `FALSE`. For example, `if(http_status_matches(beresp.status, "!200,404")) {`.
`boltsort.sort`	Sorts URL parameters. For example, `boltsort.sort("/foo?b=1&a=2&c=3");` returns `"/foo?a=2&b=1&c=3"`.

· Randomness-related VCL features (/guides/vcl-extensions/randomnessrelated-vcl-features)

Fastly provides several functions in VCL (/guides/vcl-faqs/) that control randomness-related activities. These functions do not, however, use true randomizing sources and should not be used for cryptographic (/guides/vcl-extensions/cryptographic-and-hashingrelated-vcl-functions) or security purposes.

Random strings:

Use the function `randomstr(length [, characters])`. When characters aren't provided, the default will be the 64 characters of `A-Za-z0-9_-`.

```
sub vcl_deliver {
   set resp.http.Foo = "randomstuff=" randomstr(10);
   set resp.http.Bar = "morsecode=" randomstr(50, ".-"); # 50 dots and dashes
}
```

Random content cookies in pure VCL:

```
sub vcl_deliver {
    add resp.http.Set-Cookie = "somerandomstuff=" randomstr(10) "; expires=" now + 180d "; path=/;";
}
```

This adds a cookie named "somerandomstuff" with 10 random characters as value, expiring 180 days from now.

Random decisions:

Use the function `randombool( _numerator_, _denominator_ )`, which has a numerator/denominator chance of returning true.

```
sub vcl_recv {
    if (randombool(1, 4)) {
        set req.http.X-AB = "A";
    } else {
        set req.http.X-AB = "B";
    }
}
```

This will add a X-AB header to the request, with a 25% (1 out of 4) chance of having the value "A", and 75% chance of having the value "B".

REMEMBER: The ability to upload custom VCL (/guides/vcl-faqs/where-can-i-upload-custom-vcl) code is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

· Size-related VCL variables (/guides/vcl-extensions/sizerelated-vcl-variables)

In order to allow better reporting Fastly has added several variables to VCL to give more insight into what happened in a request.

Name	Where	Description
`req.bytes_read`	deliver log	How big a request was in total bytes.
`req.header_bytes_read`	all	How big the header of a request was in total bytes.
`req.body_bytes_read`	deliver log	How big the body of a request was in total bytes.
`resp.bytes_written`	log	How many bytes in total were sent as a response.
`resp.header_bytes_written`	log	How many bytes were written for the header of a response.
`resp.body_bytes_written`	log	How many bytes were written for body of a response.
`resp.completed`	log	Whether the response completed successfully or not.

· Accept-Language Header VCL features (/guides/vcl-extensions/vmod-accept)

Fastly provides some functions in VCL to parse and normalize the `Accept-Language` header.

Functions

Language Lookup

An implementation of the Lookup functionality as defined by RFC 4647, section 3.4 (http://tools.ietf.org/html/rfc4647#section-3.4).

Syntax

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

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

Return Values

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

Examples

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

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

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

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

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

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

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

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

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

Language Filter (Basic)

An implementation of the Basic Filtering functionality as defined by RFC 4647, section 3.3.1 (http://tools.ietf.org/html/rfc4647#section-3.3.1).

The implementation is not exact when the wildcard tag (`*`) is used. If a wildcard is encountered and no matches have been found yet, the default is returned. If there are matches, those are returned and the remainder of the priority list is ignored.

(There is no implementation of Extended Filtering, but if you are in need you could always file a feature request with Support. :))

Syntax

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

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

Return Values

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

Examples

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

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

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

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

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

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

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

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

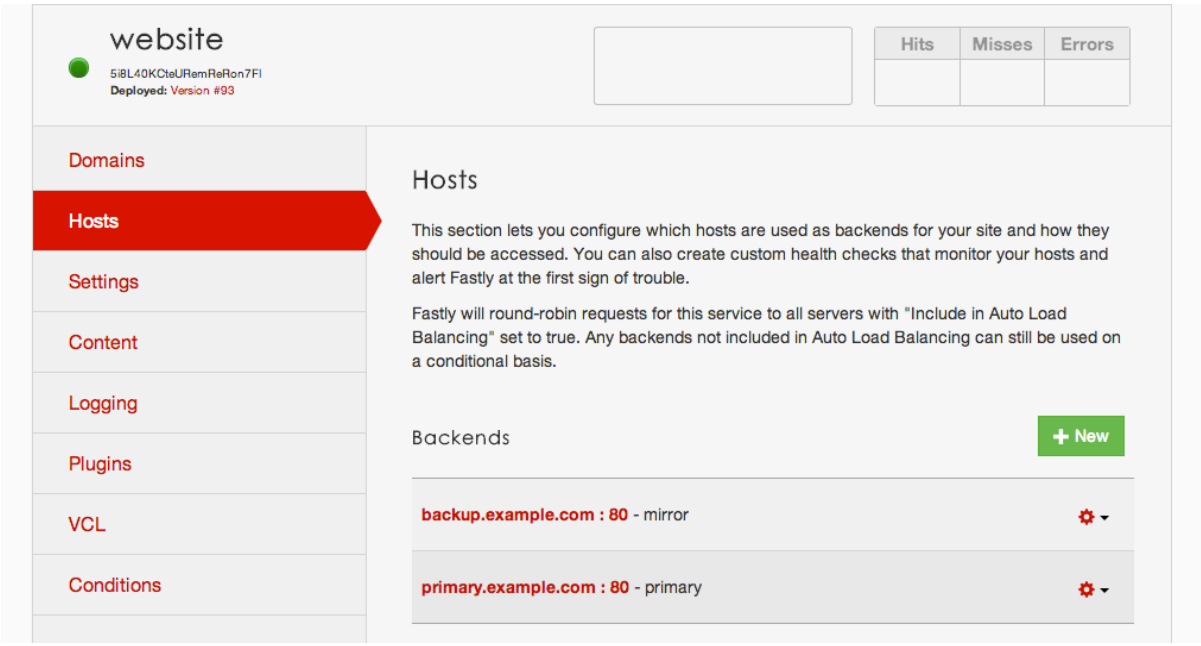
```
set req.http.Filtered-Language =
    accept.language_filter_basic("en-US:nl", "nl", req.http.Accept-Language, 2);
if (!req.http.Filtered-Language) {
  # User will get Dutch even if he doesn't want it!
  # (Because none of our languages were acceptable)
  set req.http.Filtered-Language = "nl";
}
```

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

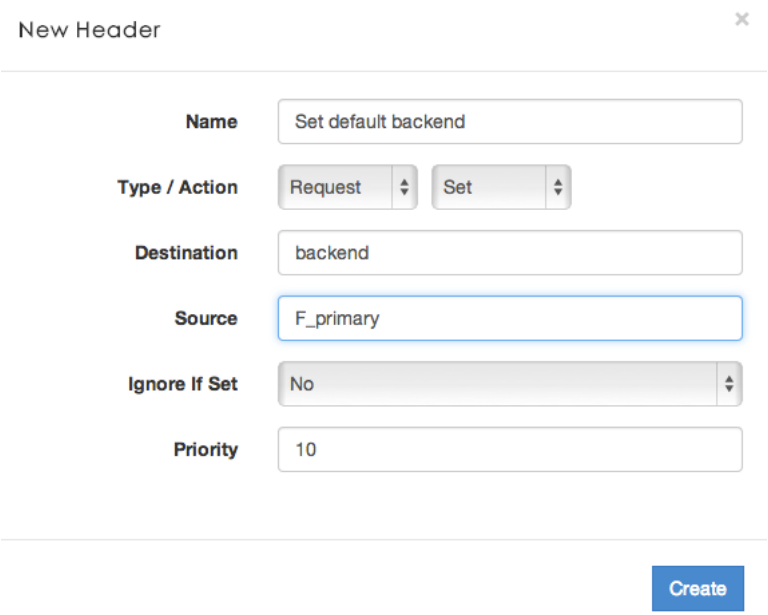
• Load Balancing (/guides/load-balancing/)

· How do I configure a failover backend server? (/guides/load-balancing/how-do-i-configure-a-failover-backend-server)

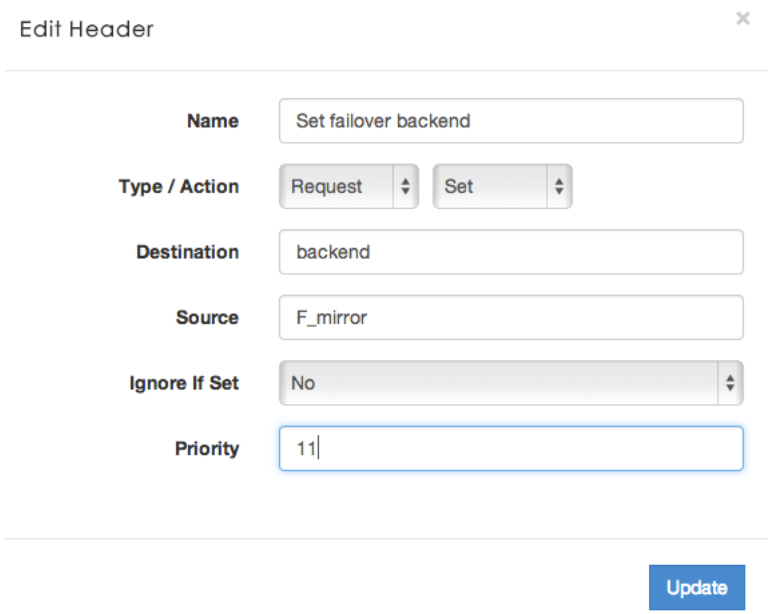
The easiest way to set up a failover is to set up two backend servers (/guides/backend-servers/): one as your primary and one your mirror. Make sure that "auto load balance" is turned off for each server listed among your hosts:



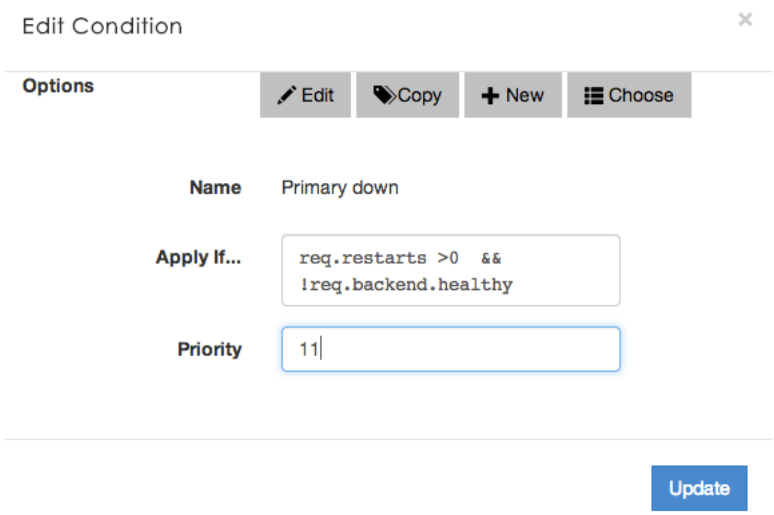
Next, set a request header (in the Content configuration settings) that sets the primary as the default backend:



and another one that sets the backend to be the failover. Note how the priority is a higher number:



Then attach a request condition (right click the gear icon next to the header name and choose Cache Conditions) to the last request header that says only to do it if the number of restarts is more than 0 or the backend is unhealthy.



Finally, save your conditions and deploy the new configurgation.

· How do I implement sticky load balancing? (/guides/load-balancing/how-do-i-implement-sticky-load-balancing)

If you're load balancing between backends, sometimes it's desirable to have a given user (or group of users) always go to the same server. The easiest way to do this is using the `client.identity` variable and a client_director in the Fastly API.

First, set up a director via the API and the Director (/api/config#director) and Director Backend (/api/config#director_backend) endpoints:

1. Set up a new Director using the API with the type 'client'.
2. Look at your generated VCL and find the name of the backends you want to load balance between.
3. Add mappings for each Backend you want to load balance between using the name of the Director and the names of the Backends.

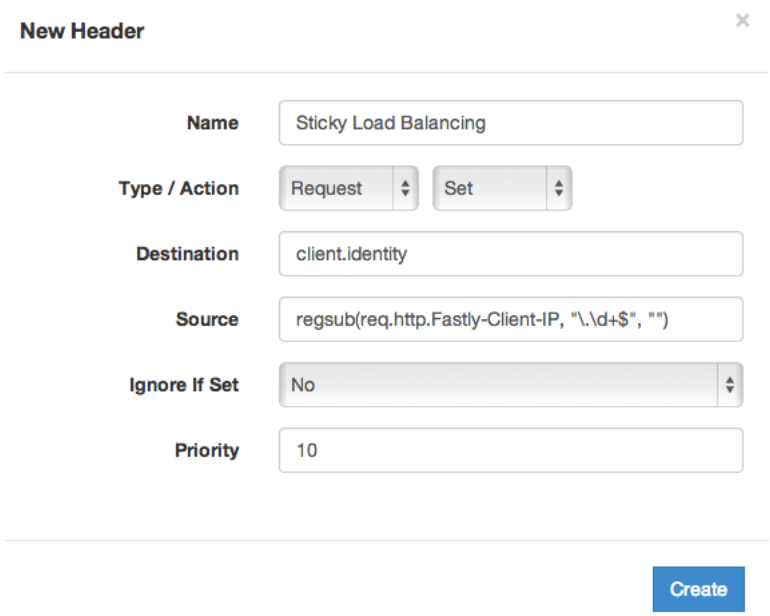
If you look at your generated VCL again you should see something like:

```
backend foo {
    ...
}

backend bar {
    ...
}

director baz client {
        {
                .backend = foo;
                .weight = 1;
        }
        {
                .backend = bar;
                .weight = 1;
        }
}
```

Next, set the default `req.backend` to be the name of your director using a new Header from the Content pane. Under the Content pane create a new Header (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses) that looks something like this:



They key entries here are the **Destination** and **Source** fields. This particular example sets the `client.identify` (in the **Destination** field) to `regsub(req.http.Fastly-Client-IP, "\.\d+$", "")` (in the **Source** field), which is setting the stickyness to be based on the first three octets of an IP address (e.g., 192.168.0.1 becomes 192.168.0). In practice, this roughly maps everyone from a particular netblock to the same server. The assumption here is that everyone from a particular company will be in the same netblock and this will likely put everyone from the same company on the same server. If your service is very company oriented (e.g., HR software or a ticketing system) then this technique may help because the company's data will already be loaded.

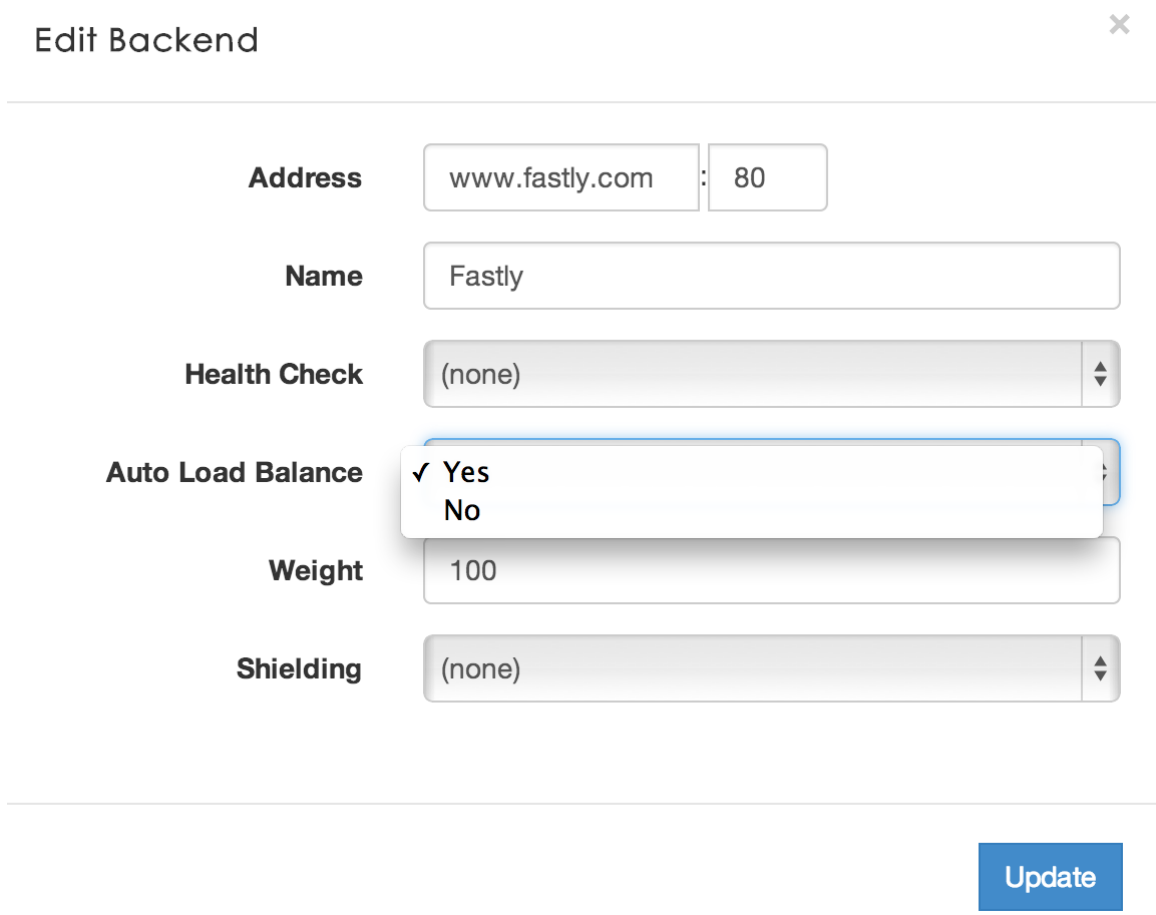
However, you can use any variable available. For example you might want to use a cookie (or part of a cookie) or the location of the customer (/guides/vcl-extensions/geoiprelated-vcl-features) (maybe using `geoip.country`).

· How do I set up load balancing between my backends? (/guides/load-balancing/how-do-i-set-up-load-balancing-between-my-backends)

Enable the Fastly autodirector feature to randomly load balance between two or more backends.

Enabling Autodirector

1. Log in to the Fastly application.
2. Click on the **Configure** tab (the wrench icon at the top of the application window).
3. Select the appropriate service from the **Service** drop down menu.
4. Click the **Configure** button to the right of the displayed service name.
5. Click **Hosts** from the section list on the left.
6. In the list of backends, click the gear icon to the right of the named backend you wish to edit, then select **Edit** from the drop down menu. The Edit Backend window appears.



7. Change the setting for **Auto Load Balance** to **Yes** and then click **Update**.

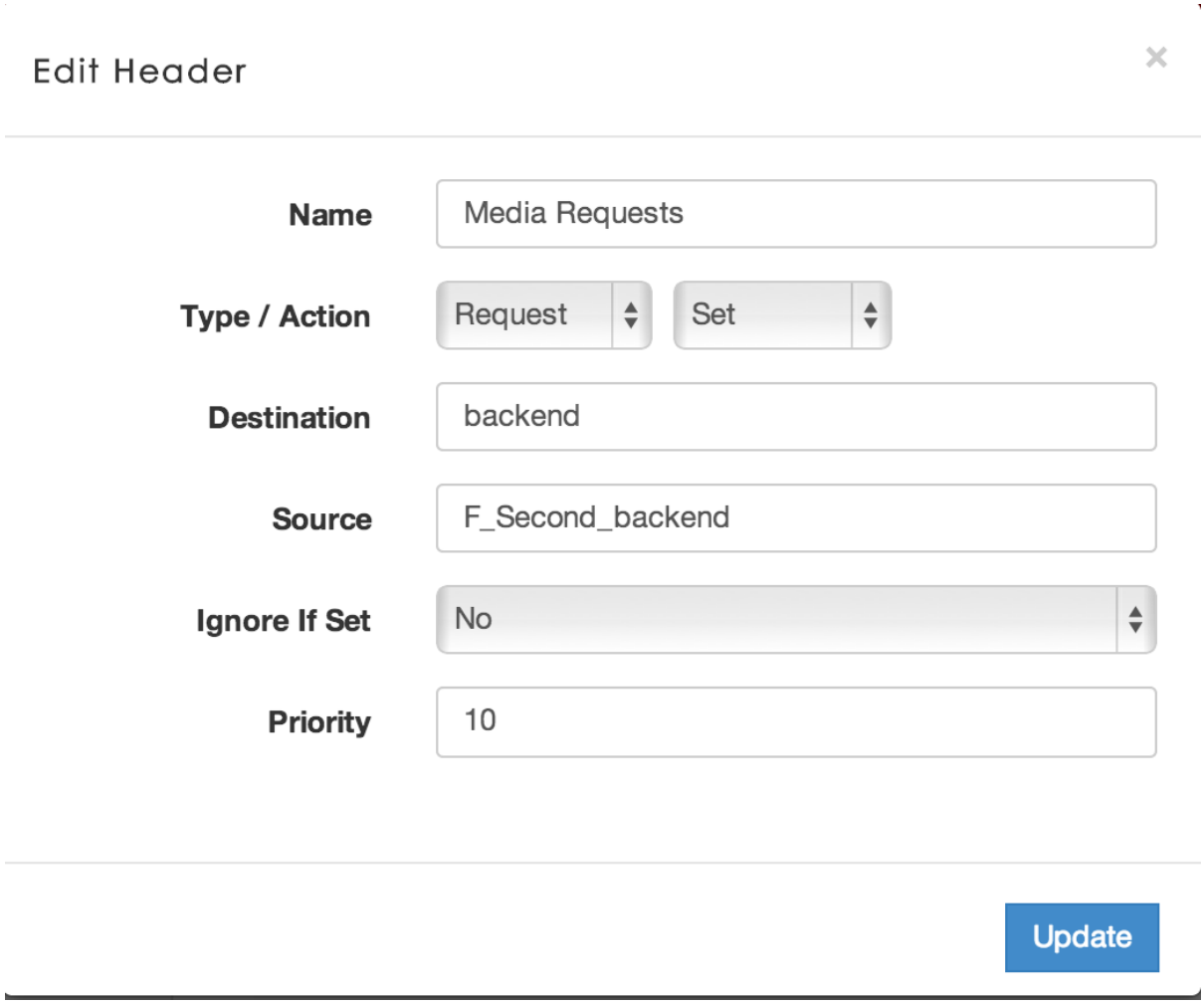
You can repeat steps 6 and 7 for each backend you want to include in the load balancing group.

Setting Conditions

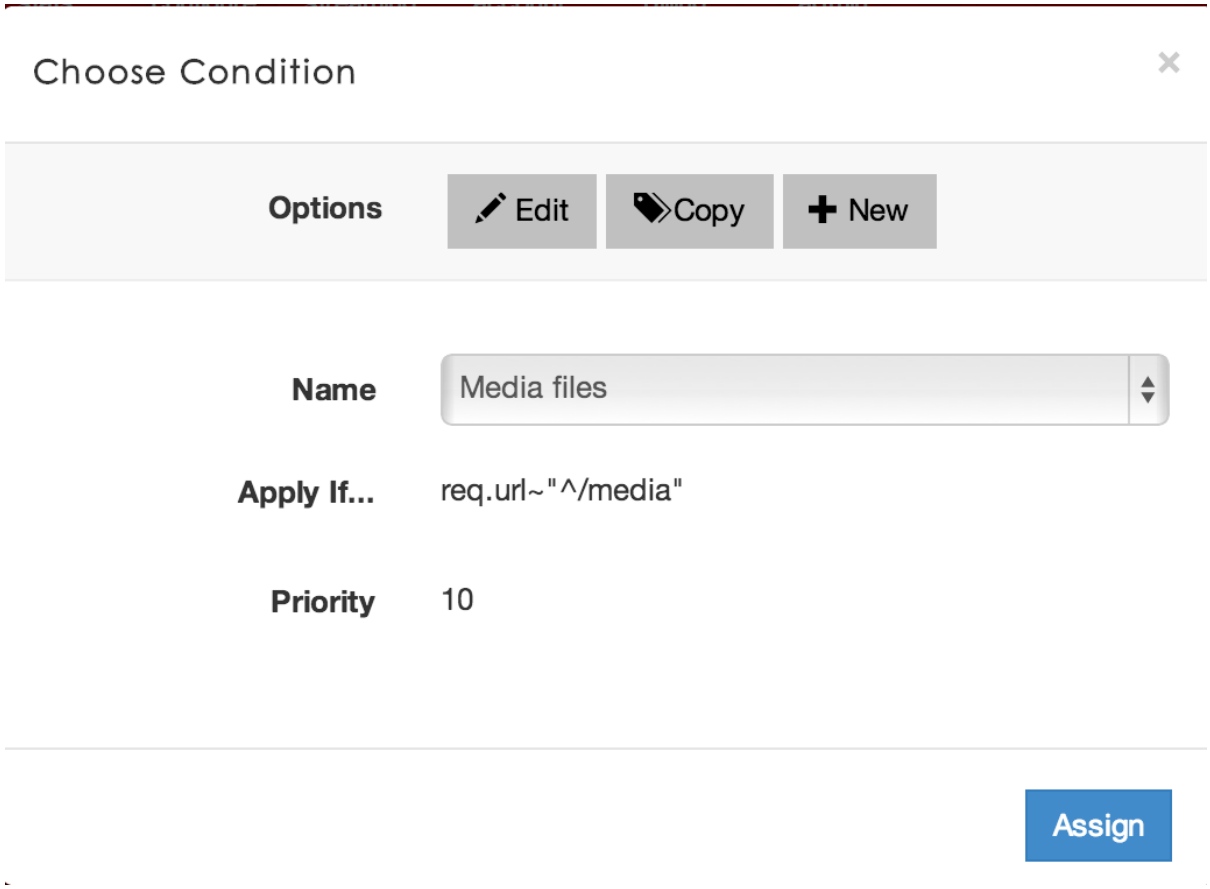
With **Auto Load Balance** set to **Yes** on a backend, be aware when setting conditions (/guides/tutorials/conditions-tutorial) on the actual backend under Hosts->Backends->Edit->Conditions.

The autodirector groups servers together based on like conditions, giving you the flexibility to create sub autodirectors by assigning a condition to one group of backends and another condition to another set of backends. If each backend has a different condition, the auto load function will not randomly load balance between the different servers.

Conditions can also be assigned to a server through a header. For example, say 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. To do this, you could create a header (under Content->Headers->New) to set requests to the second backend.



After the header is created, you would click the gear icon to the right of the header name and select **Request Conditions**, then add a new condition to apply if the URL matches the desired prefix.



The generated VCL below illustrates the autodirector set for all three servers. Within the section **sub vcl_recv**, the default backend is set to the autodirector and, if the media condition is met, requests are then forwarded to the second backend.

```
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) {
#--FASTLY TIMER CODE
    if(req.http.X-Timer) {
      C{
        char *timer = VRT_GetHdr(sp, HDR_REQ, "\010X-Timer:");
        double start = 0;
        char time[255];
        sscanf(timer, "S%lf", &start);

        if(start) {
          snprintf(time, 255, "%s,VS%.f", timer, (TIM_real() - start) * 1000);
          VRT_SetHdr(sp, HDR_REQ, "\010X-Timer:", time, vrt_magic_string_end);
        } else {
          snprintf(time, 255, "S%.9f,VS0", TIM_real());
          VRT_SetHdr(sp, HDR_REQ, "\010X-Timer:", time, vrt_magic_string_end);
        }
      }C #;

    } else {
      C{
        char time[255];
        snprintf(time, 255, "S%.9f,VS0", TIM_real());
        VRT_SetHdr(sp, HDR_REQ, "\010X-Timer:", time, vrt_magic_string_end);
      }C #;
    }
  }
            set req.backend = autodirector_;

  # 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
}
```

· How do I set up load balancing between my directors via the API? (/guides/load-balancing/how-do-i-set-up-load-balancing-between-my-directors-via-the-api)

To set up load balancing between directors configure the Fastly API by first creating an origin and then adding your directors to the origin.

For example, here's how to do it using the Fastly admin shell:

```
fastly> create service my_service version 5 origin my_origin
fastly> add service my_service version 1 director director_1 to origin my_origin
fastly> add service my_service version 1 director director_2 to origin my_origin
```

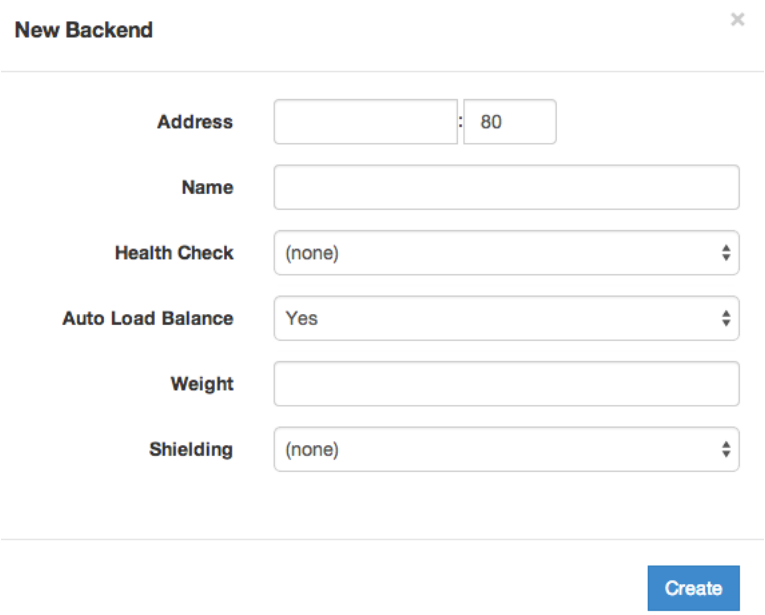
Here's how to do it using the Perl client library:

```
my $origin = $fastly->create_origin(service_id => $service->id, version => $service->version->number, name => "My Origin")
;
$origin->add_director($director1);
$origin->add_director($director2);
```

And here's how to do it using the Ruby client library:

```
origin = fastly.create_origin(:service_id => service.id, :version => service.version.number, :name => "My Origin")>
origin.add_director(director1)
origin.add_director(director2)
```

· How does Weight affect the load on my origin servers? (/guides/load-balancing/how-does-weight-affect-the-load-on-my-origin-servers)



The Weight setting on the Backend window (accessed when you edit Backend on the Hosts pane of the Configuration Control Panel (/guides/fastlys-web-interface/about-the-configuration-control-panel)) appears when Auto Load Balance is set to Yes.

When you specify weight, you specify the percentage of traffic to send to a specific origin server. Each origin receives / of the traffic.

For example, if you have origin server A with Weight set to 0.5 and origin server B with a Weight of 0.5, traffic is split equally between those origins at a ratio of 50:50. If you increase the Weight setting on origin server A to .55 and decrease the Weight setting on origin server B to .45, the traffic ratio changes accordingly to 55:45.

· What does '503 Quorum weight not reached' error mean? (/guides/load-balancing/what-does-503-quorum-weight-not-reached-error-mean)

The "503 Quorum weight not reached" error means that too many of your backends are being marked as sick. This error is only an issue if you've set the 'quorum' field of a Director (/api/config#director).

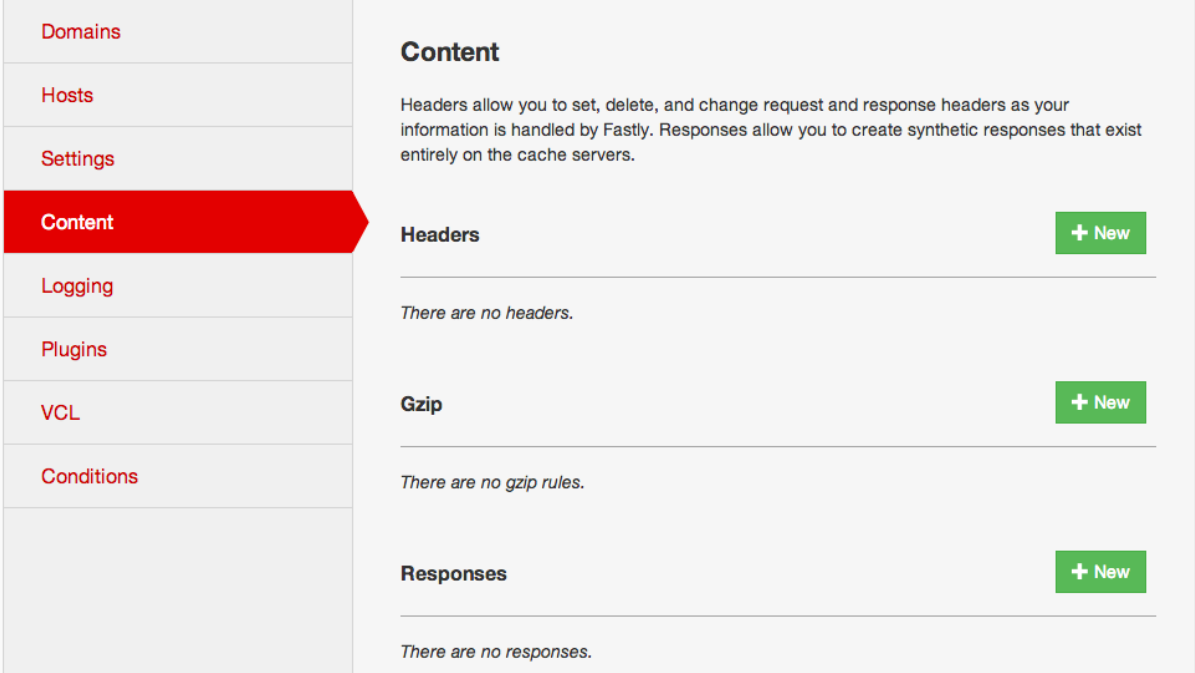
• Tutorials (/guides/tutorials/)

· Adding or Modifying Headers on HTTP Requests and Responses (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses)

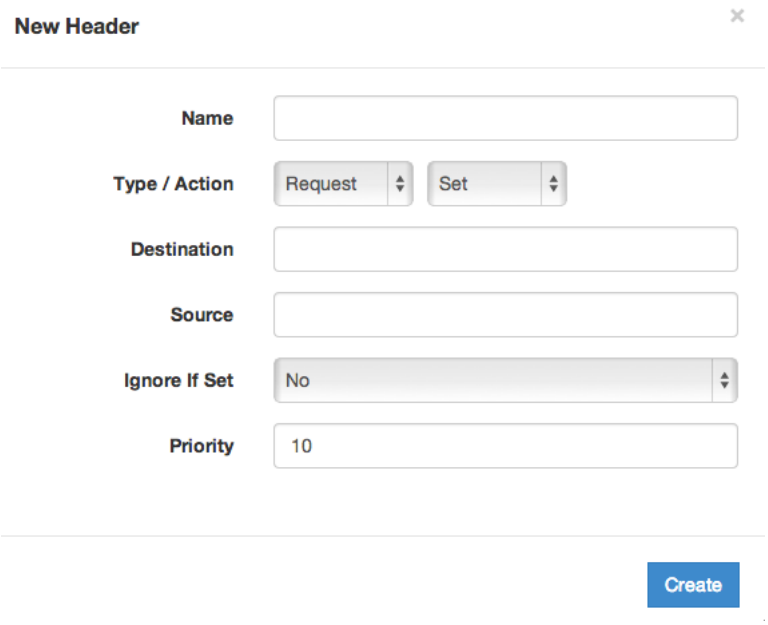
Welcome to the Headers Tutorial. The steps below show you how to use the Fastly application to add, delete, and modify headers on HTTP requests and responses.

We'll start with the simplest case: adding a static header to all of the responses that Fastly returns to your users.

To start with, log in (https://app.fastly.com/) to your Fastly account, select the service you'd like to work on, and click **Configure**. Next, choose the **Content** pane in the left sidebar. This will display the list of headers associated with your service. If it's empty, it should look like this:



Click **New** and you'll be presented with a form to create a New Header object.



Let's go through what each of these fields mean:

Field	Description
Name	The Name field specifies a handle for you to recognize and remember a particular Header rule. Type whatever memorable word or phrase you would like in this field.
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** 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. **Regex** and **Regex All** also exist but are not covered in this article.
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 and Append actions. 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 `geoip.city`. See the list of Common Sources below for more common sources of new content.
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.
`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`
`geoip.*`	Request, Cache, Response	GeoIP values for the client's IP (see our GeoIP article (/guides/vcl-extensions/geoiprelated-vcl-features) for more information).

· Cache Control Tutorial (/guides/tutorials/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 Varnish Configuration Language (VCL). If you'd like to use VCL, you will need to contact our support team (mailto:support@fastly.com) so that it can be enabled on your account.

Please remember that you can use longer expirations coupled with our purging API to make your site faster and your backend less loaded. Expirations in the following examples are provided as guidance only.

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, as well as any caches between Fastly and the browser.

Cache-Control: max-age

Format:

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

Will be respected by Fastly caches, any caches between Fastly and the browser.

Expires

Caches content until it expires as specified. Please read section 14.21 of RFC2616 (http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.21) for explanation of the format. This header will be respected by Fastly caches, any caches between Fastly and the browser and the browser.

Do not cache

If you don't want certain resources cached, set the header as follows:

```
Cache-Control: private
```

If you just set max-age=0 or an Expires in the past, there is still no guarantee that the content won't be used to satisfy multiple outstanding requests at the same time. Also, an expired (stale) object might be used in case of errors or while a request to your backend for the same object is already in progress.

More Complicated Examples

Say that you want Fastly to cache your resources forever but send headers to browsers so that they don't cache it at all (so that every browser miss hits Fastly but isn't a cache miss from your service).

The best way to do this would be to send us 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. So for instance:

```
Cache-Control: max-age=3600
Surrogate-Control: max-age=2592000
```

The Surrogate-Control header takes priority over Cache-Control, and unlike Cache-Control it is stripped so the browsers don't see it either.

Example backend configs

Apache Config

If you are using Apache, the easiest way to add headers is to use the mod_expires module (http://httpd.apache.org/docs/current/mod/mod_expires.html). 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:

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

You can also protect whole URL subtrees. For example:

```
<Location "/css">
  ExpiresActive On
  ExpiresDefault "access plus 1 year"
</Location>

<Location "/static/">
  ExpiresActive On
  ExpiresDefault "access plus 1 day"
</Location>
```

NGINX Configuration

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

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

Alternatively, if you need more flexibility in modifying headers you can try the HttpHeadersMore Module (http://wiki.nginx.org/HttpHeadersMoreModule).

Amazon S3 configuration

By default, S3 doesn't set any Cache Control headers, 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. For example, the following Blackpawn blog post on "Setting cache Metadata on Files in S3 (http://blackpawn.com/blog/setting-cache-metadata-on-files-in-s3/)" does a good job of describing how to set Cache Control headers on all your JPG and PNG documents in all your buckets. In their example, they set data cache control headers in order to cache data for 10 days. Please remember that you can combine long cache time with instant purges to enhance your performance.

Also note that the example below is an example for customers with small to medium size buckets. It iterates over every object in your buckets, so if you have millions of objects this may not be the right approach. In that case we would recommend using Varnish Configuration Language (/guides/vcl-extensions/guide-to-vcl) (VCL). Please contact us (mailto:support@fastly.com) for assistance.

```
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
        )
```

Custom Headers in Programming Languages and Frameworks

PHP

More information: http://php.net/manual/en/function.header.php (http://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 (https://docs.djangoproject.com/en/dev/ref/request-response/#setting-headers)

Example:

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

Sinatra

More information: http://sinatra.rubyforge.org/doc/ (http://sinatra.rubyforge.org/doc/)

Example:

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

· Conditions Tutorial (/guides/tutorials/conditions-tutorial)

Welcome to the Conditions Tutorial! Conditions are one of the powerful (and complicated) features Fastly has to offer. Because they are so complex we will only touch on the basics and give ideas for how to use conditions in a more advanced setting.

In this tutorial, we assume that you already know how to edit and deploy configurations using the web-based configuration application (https://app.fastly.com). If you are not familiar with basic editing using the app please see our Help Guides (/guides/) to learn more before moving forward.

We also assume you are familiar with basic C-style logical expression syntax (e.g., basic logic, operators such as &&, precedence). If you do not have experience with these types of expressions, please refer to a basic programming guide (such as the Tizag Perl tutorial (http://www.tizag.com/perlT/perlif.php)) that deals with "IF" style expressions in either C or Perl. (Note these will not be directly applicable to the tutorial as VCL is neither C nor Perl. The syntax, however, is quite similar.)

Overview

In a nutshell, conditions are used to add logic to nearly any basic configuration object in a service. When you assign a condition to an object in your service you are essentially wrapping the object in a VCL if-statement. If the condition is met, then the object is used when processing an end-user request; otherwise, it is not. The configuration parameters for a condition are actually rather simple:

- Name - A human readable identifier for the condition
- Apply If - The logical expression to execute in VCL to determine the condition resolves as True or False
- Priority - A number that is used to determine the order in which multiple conditions execute. Lower numbers execute first.

From this perspective conditions seem easy enough to grasp. But because the "Apply If" parameter is defined in terms of actual VCL code it is easy to see why it can be hard to know exactly how to create and leverage conditions in your configuration. (We will cover this in more detail in the next section.)

All conditions are saved in your service like any other configuration object but can only be accessed, assigned, and mutated through a special menu in the "Gear Dropdown" to the right of an existing configuration object (e.g., Backends, Headers). This means that when you create a condition you can apply it in multiple places across your configuration. Now that we've covered some of the basics, let's dive into some of the details on how one actually writes conditions.

Writing Basic Conditions with VCL

As mentioned above, conditions use actual VCL to define when the object should be applied when processing requests to the cache server. Lets look at a very basic example:

`req.url ~ "^/api/"`

This example says that the condition passes if the URL of the request begins with the string `"/api/"`. The `"~"` operator performs a Perl style regular expression match with the `"^/api/"` regex and returns true if `req.url` matches. This particular example is useful when you want to perform special routing in your service. For instance, if your API exists on a different backend than your regular website you can use a condition like this to use that API backend whenever a request comes in that begins with `"/api/"`.

Let's take a look at some more examples:

1. `client.ip ~ "127.0.0.1"`
2. `req.http.host ~ "example.com"`
3. `req.request ~ "POST|PUT" && req.url ~ "^/api/articles/"`

Here's what they mean:

1. The client requesting a resource on your service has the IP "127.0.0.1"
2. The host header of the incoming request is "example.com"
3. The request is a POST or a PUT and the url begins with "/api/articles/"

As you can see, conditions can perform matches or more complicated multi-predicate logical expressions to zero in on specific scenarios. In the examples above "client.ip", "req.http.header.host", "req.url", and "req.request" are all "Varnish Configuration Variables". There are many variables that you can use to craft your own conditions. The Varnish Cache documentation on variables (https://www.varnish-cache.org/docs/2.1/reference/vcl.html#variables) provides a complete list (note that some of the variables Varnish allows may not be available or may have no meaning in the context of Fastly).

All of the examples we have covered so far are conditions that deal with handling requests to the Fastly cache servers. There are actually three types of requests that can be applied depending on what type of object they are being assigned to. The next section will cover the three types of conditions in detail.

Request, Response, and Cache Conditions

Because conditions can occur in various stages of the caching pipeline, we have categorized them into three types: Request, Response, and Cache conditions. Configuration objects also only apply in various stages of the pipeline and thus can have different types of conditions associated with them. Each stage of the pipeline also has access to a different set of VCL variables (as described in the Varnish-Cache.org documentation (https://www.varnish-cache.org/docs/2.1/reference/vcl.html#variables)) that a condition can use to determine if an object applies.

Here is a basic list of each type of condition, the objects to which they apply, and the VCL variables that are available:

- **Request**: Conditions that deal with objects while processing a request.
  - Objects: Request Settings, Backends, Headers, and Responses
  - VCL Variables: client*., server.*, req.*
  - Example: `req.url ~ "^/foo/bar$"`
- **Response**: Conditions that deal with object while processing a response
  - Objects: Headers, Syslog
  - VCL Variables: client*., server.*, req*., resp.*
  - Example: `resp.status ~ "404"`
- **Cache**: Conditions that are applied when we receive a response from your backend, just before it is (potentially) cached
  - Objects: Cache Conditions, Headers, Gzip, Responses
  - VCL Variables: client*., server.*, req*., beresp.*
  - Example: `!beresp.cachable && req.url ~ "^/articles/"`

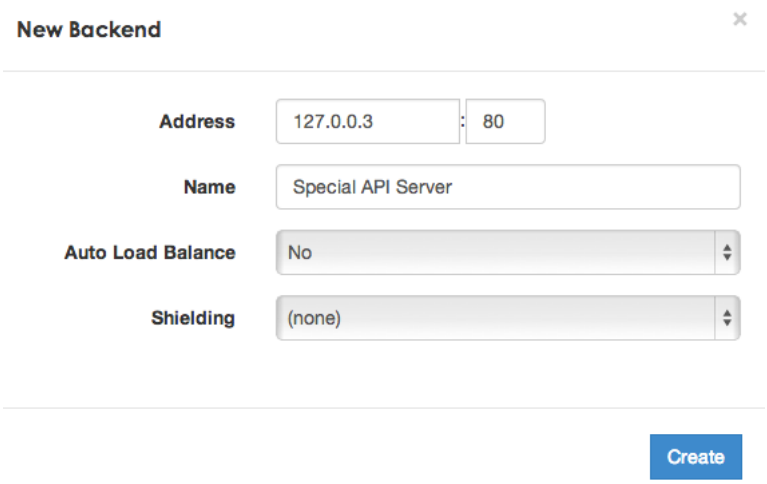
These three types of conditions give you nearly full control over your configuration with minimal actual programming. The next couple of sections will show you how to add, manipulate, and assign conditions using the Fastly configuration application.

Adding Conditions in the Fastly Application

This portion of the tutorial is a little more involved than most Fastly interface tutorials. We will work from a scenario (inspired by real life!) and show you how to add each of the three different types of conditions in conjunction to solve the problem.

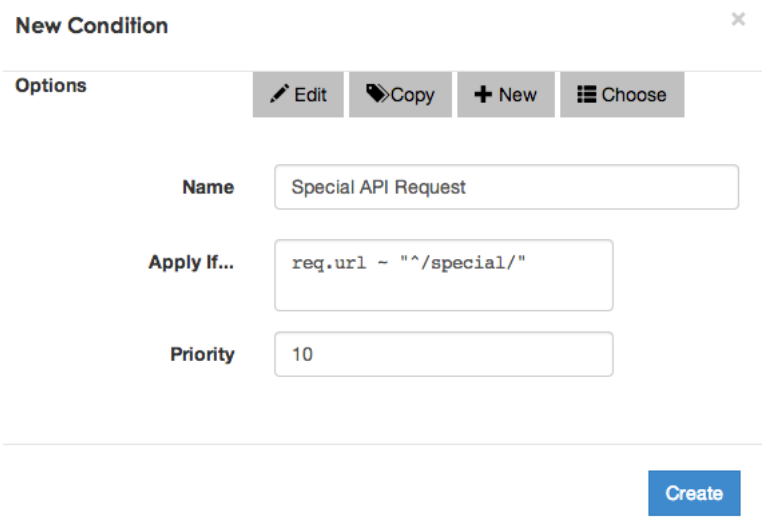
The Scenario: You want to add a new backend that handles a specific portion of your API requests. Some requests to this API must cached in a different way than normal requests to your API and you must also set special headers for specific types of requests. Let's begin...

First, navigate to the Fastly application via your browser and enter the configuration editor for your service. Add the special backend with the following parameters:

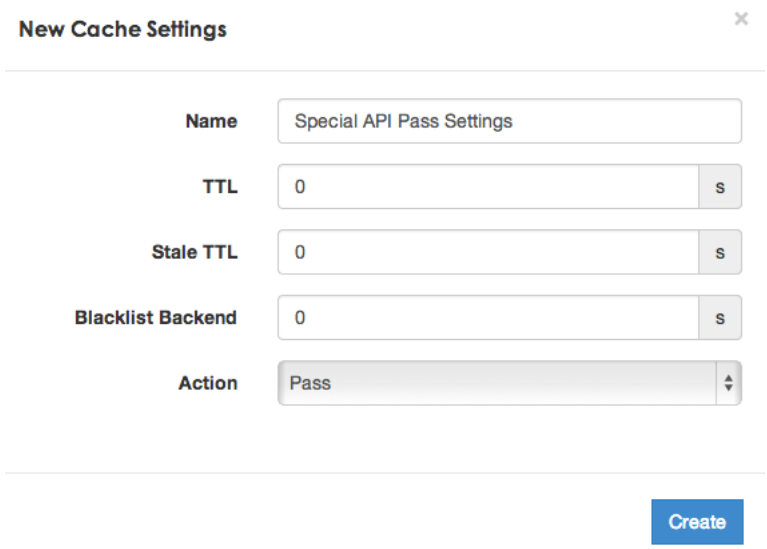


Note how we chose "No" for the "Auto Load Balance" parameter. This tells us that you do not want this backend to be included in the normal automatic load balancing pool. Next we need to tell the cache how to route requests to this backend. Let's assume that all of the requests coming to the new special API are prefixed with "/special". This sounds like a perfect job for our handy-dandy Request condition.

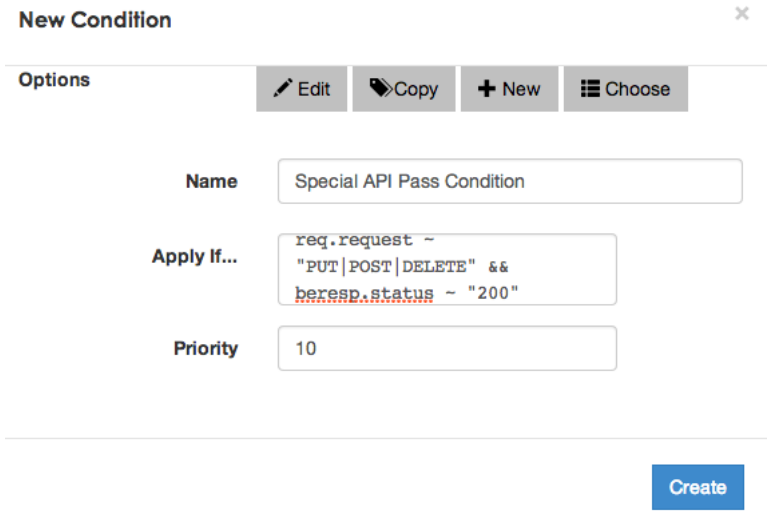
To create one, click the gear box next to your newly created backend and select "Conditions". Then enter the following information in a new condition window and click the Create button:



Okay, now that we have requests being properly routed to our new backend we need to make sure it does doesn't go caching any responses from PUT, POST, or DELETE requests (because they're special for this particular API and send back extra, time dependent, meta-information in each response). In order to do this we first need to create a cache settings object in our configuration:

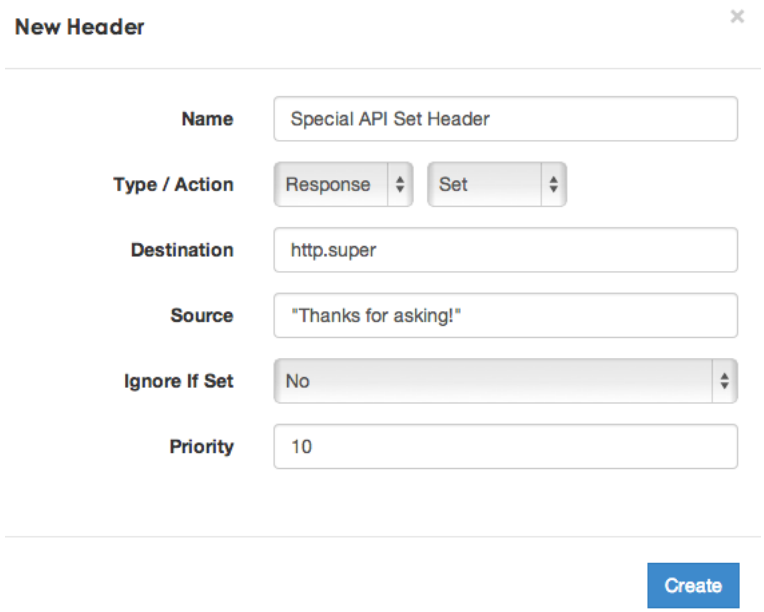


With this guy in place let's make sure it is only applied when we are caching responses from the special API. Click the gear box, click conditions, and add the following as a new Cache condition:

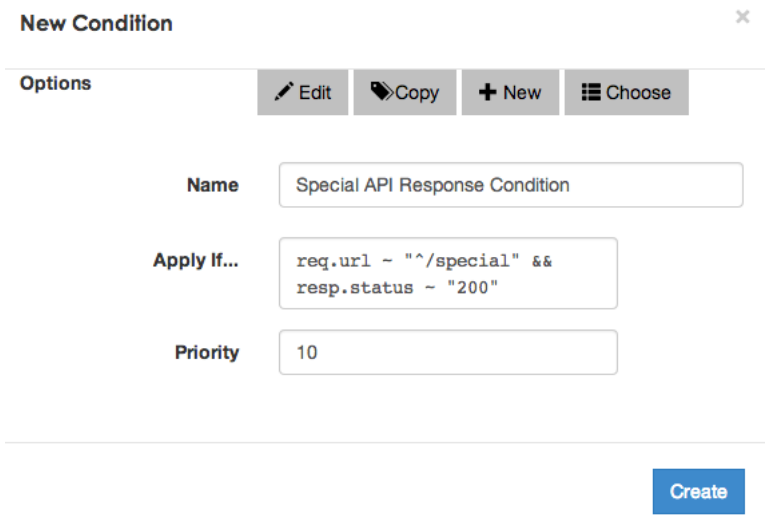


A quick note: the interface is smart enough to know you want a cache condition so it will always pick the right type if there is only one type of condition available to an object.

Finally, we need to make sure that some requests set a special header because someone in business metrics department wants to track the effectiveness of our new API and is using that header to tally and gather information about those kinds of requests. For this we simply need to add a new request header rule like so:



You might be noticing a pattern. Here's the condition we need to use to make sure that this header is only set on that special type of request:



Phew, that was a lot of conditions work. But in the end it seems pretty straight forward, right? We had a basic problem definition and conditions allowed us to tackle the problem without writing a single line of custom VCL. Before we go ahead and deploy the app, let's take a look at the generated VCL and see how Fastly converted our objects and conditions into actual VCL, here:

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

and here:

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

and finally, here:

```
# 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 it's always a good idea to 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).

REMEMBER: The ability to upload custom VCL code is disabled by default when you sign up. To enable this ability for your account, contact support@fastly.com (mailto:support@fastly.com) and request it.

· Enabling Token Authentication (/guides/tutorials/enabling-token-authentication)

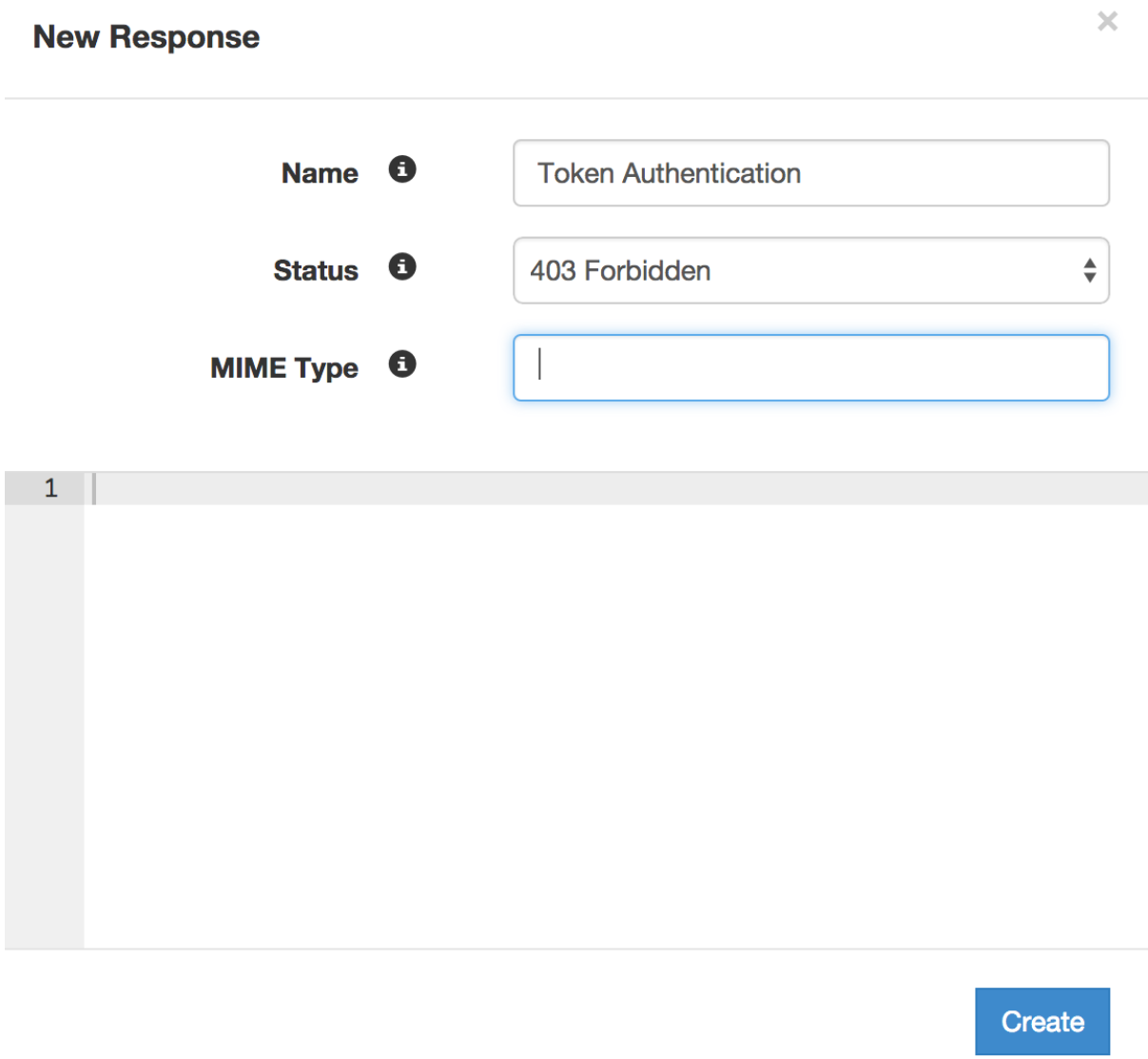
Token Authentication allows requests to be authenticated at Fastly's edge without relying on your origin server for each request. Tokens are generated within your web application and appended to URLs in a query string. When Fastly receives a request for the URL, the token is validated before serving the content. After a configurable period of time, the token expires.

Each token can be specific to a URL or client IP address. We provide examples of how to generate tokens within your application in our Token Functions (https://github.com/fastly/token-functions) Github repository.

Basic Token Authentication

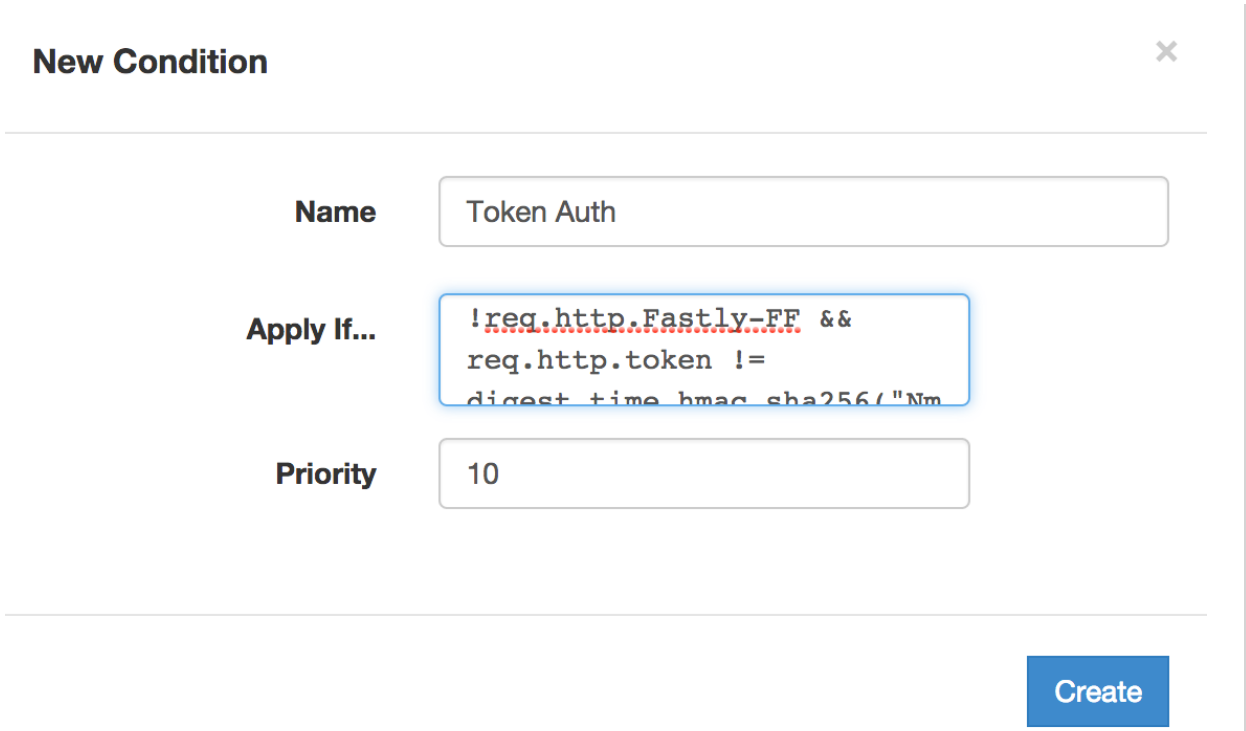
Log in to the Fastly application, click on the configure tab to access the configuration controls, and then click on the Content pane to begin creating two objects, a New Response (/guides/tutorials/responses-tutorial) and a New Condition (/guides/tutorials/conditions-tutorial), that will allow you to enable token authentication.

First, create a New Response object:



Fill in the Name field with your chosen name, select the appropriate code (in our example 403 Forbidden) from the Status menu, and add the response content to the text editor area below.

Once the response is created, add a request condition by clicking the gear icon next to the response and selecting "Request Conditions".



Give your Request Condition a name so it can be identified easily in the future, then set a priority if needed.

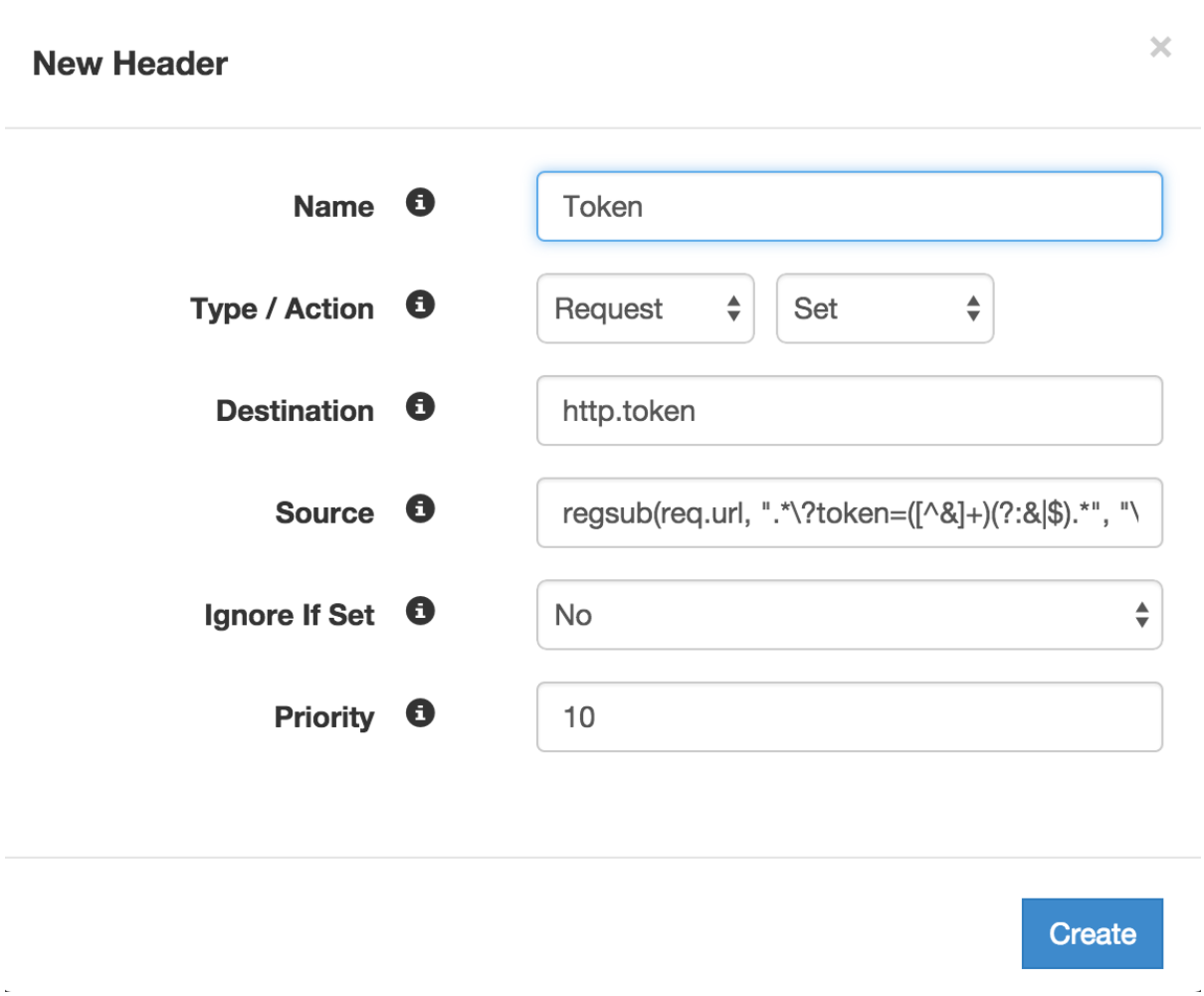
The **Apply If** field requires special attention. The full text of the condition shown in the above example reads:

```
!req.http.Fastly-FF && req.http.token != digest.time_hmac_sha256("<YourSecretKey>", 60, 0) && req.http.token != digest.time_hmac_sha256("<YourSe
```

where `digest.time_hmac_sha256()` takes the format `digest.time_hmac_sha256(<base64 encoded key>, <interval>, <offset>)`.

The first two arguments of `digest.time_hmac_sha256()` should be modified. The first argument represents a secret key. The secret key should be a randomly generated string, and must be base64 encoded. The second argument is the length of time in seconds the token will be valid for. Use the same key and interval values when generating tokens in your application.

Next, parse the token from the query string by adding a new Header (/guides/tutorials/adding-or-modifying-headers-on-http-requests-and-responses) object.



Give your New Header a name so it can be identified easily in the future, then set the **Type** and **Action** to **Request** and **Set** respectively (the defaults).

In the **Destination** field, type `http.token`.

In the **Source** field, type `regsub(req.url, ".*\?token=([^&]+)(?:&|$).*", "\1")`.

You can leave the **Ignore If Set** and **Priority** fields set to their defaults.

Tokens Per URL or IP

Tokens can be specific to each URL, IP address, or any other information available in VCL, specifically in `vcl_recv`. To do this, modify the condition on the token authentication response you create (it's the "Token Auth" in the example above).

For IP-specific tokens, modify the key as follows:

```
digest.base64("Fastly Token Test" client.ip)
```

The entire condition will look like this:

```
!req.http.Fastly-FF && req.http.token != digest.time_hmac_sha256(digest.base64("Fastly Token Test" client.ip), 60, 0) && req.http.token != diges
```

For URL-specific tokens, modify the key as follows:

```
digest.base64("Fastly Token Test" req.url)
```

Note that in both key examples immediately above, the key is *not* base64 encoded. Instead, we use a plain text key and append another string to it prior to base64 encoding in VCL. You can use this technique to vary tokens based on any variable available in `vcl_recv`.

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/index.html?token=pbKSCvYeN1fSdG8jLz6cdCYT4GVcagVtweKkBSG8Je0=`

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

· Responses Tutorial (/guides/tutorials/responses-tutorial)

Welcome to the Fastly Responses tutorial! Responses allow 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.

We assume that you already know how to edit and deploy configurations using the web-based configuration application (https://app.fastly.com/). If you are not familiar with basic editing using the app please see our Help Guides (/guides/) to learn more before moving forward.

Overview

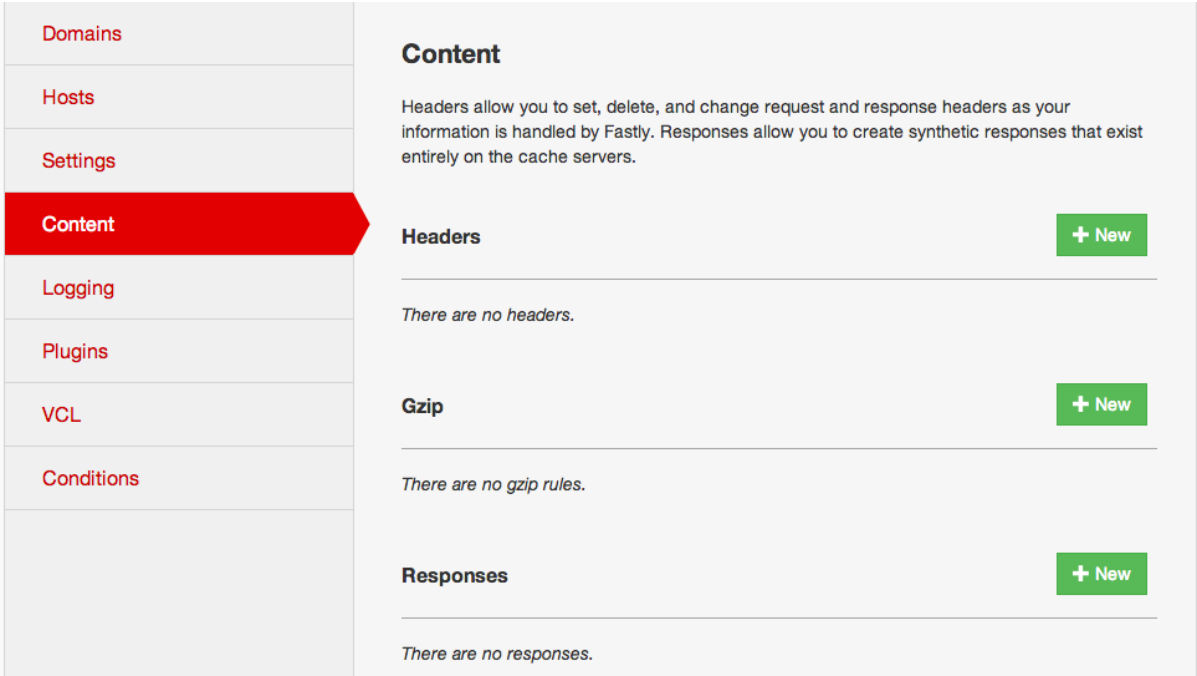
A Response has three basic attributes:

- Name - A human readable identifier for the Response
- Status - An HTTP status code to include in the header of the Response
- Content - The content to be served when delivering the Response

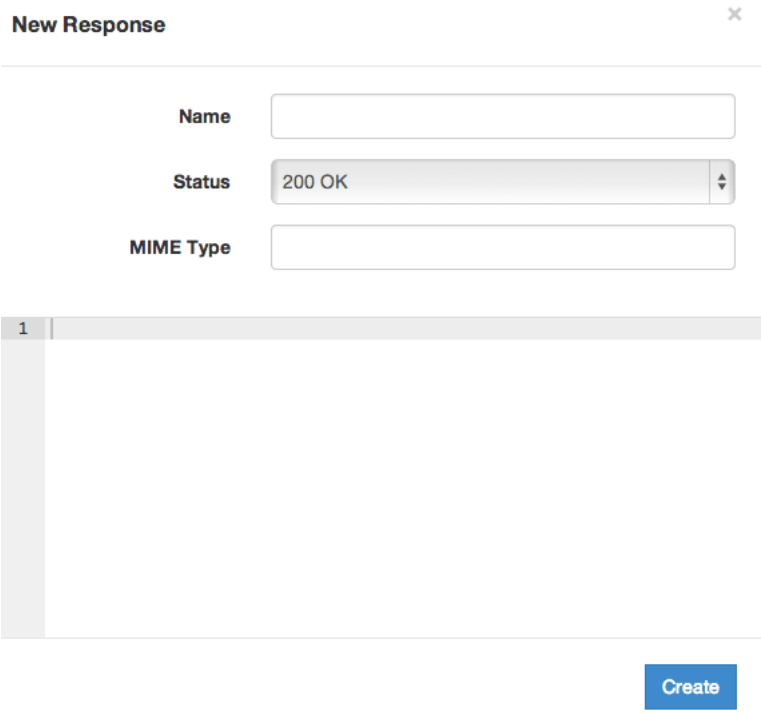
By setting these three attributes and adding a condition to the Response you can very quickly get one up and running on your service. Let's get started!

Creating a Response

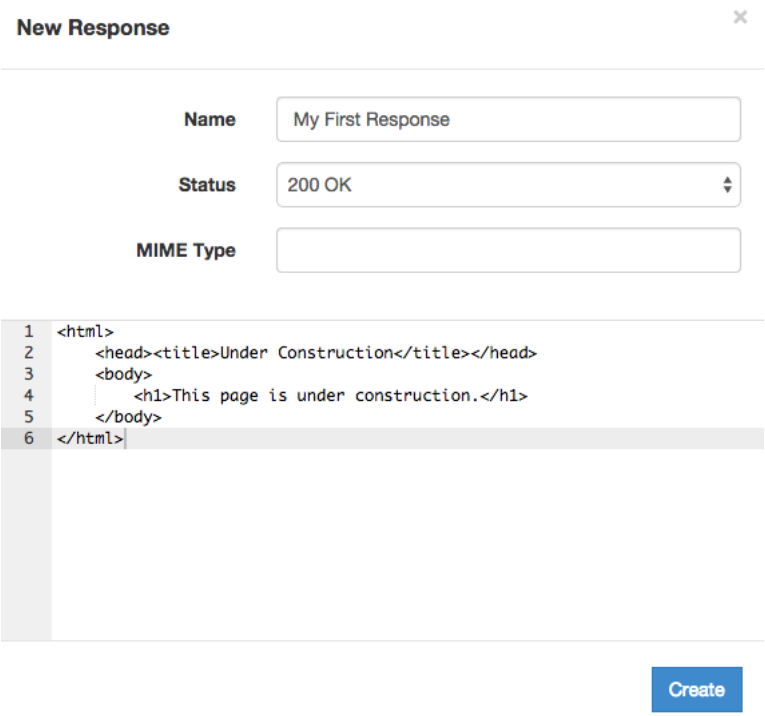
First, open up https://app.fastly.com, navigate to the configuration editor for the service for which you want to make the response, and click the "Content" tab on the left side navigation menu.



Scroll to the bottom of the page and find the Responses section, then click the New button to the right of the title for that section. The "New Response" window will appear.



Fill in the **Name** field with your chosen name, select to the appropriate code (in our case 200 OK, the default) from the **Status** menu, and add the response content to the text editor area below.



Finally click the "Create" button at the bottom of the window. Your newly created response should appear in the list of Responses. Next, click on the gear menu and select **Request Conditions**.

If you have not yet added any conditions to your service then the new condition form will appear automatically. If the new form does not appear, select New near the top of the popup.

Give your Request Condition a name so that it can be identified easily in the future, then add a Request Condition that identifies the path that should be associated with your new Response, and finally set 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).

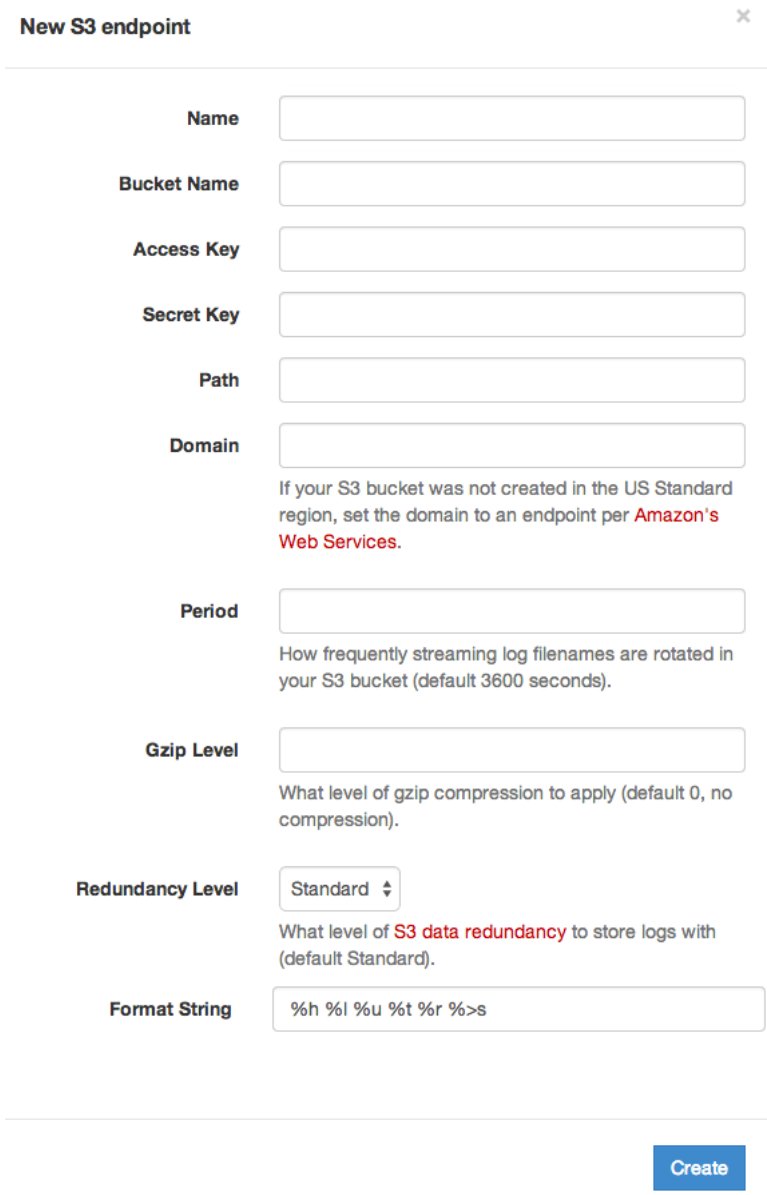
Finally, click Create and your Response should be good to go! Make sure there are no errors with your configuration and then deploy your application. Give Fastly about 15-20 seconds to load the new configuration into each of our POPs and then point your browser to the URL you used in your condition. The response should show up exactly how you wanted it to.

• Log Streaming (/guides/log-streaming/)

· Log Streaming: Amazon S3 (/guides/log-streaming/log-streaming-amazon-s3)

IMPORTANT: Be sure to review our information on Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming) before following the provider-specific information for Amazon S3 shown below.

Fastly can periodically dump logs to Amazon S3 or an S3-compatible service.



The only required parameters are your Amazon Bucket Name, Access Key, and Secret Key. We recommend adding an IAM (Identity and Access Management) user specifically for Fastly with ListBucket, GetObject, and PutObject permissions on the directory you want to put the logs into. You can read more about this at Amazon's "Getting Your Access Key ID and Secret Access Key (http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSGettingStartedGuide/AWSCredentials.html)" page.

If you want to use an S3-compatible storage system (such as Dreamhost's DreamObjects (http://www.dreamhost.com/cloud/dreamobjects/)), then you should set the domain name for that service (for example, in the case of DreamObjects the domain name would be objects.dreamhost.com).

IMPORTANT: If your Amazon Web Services S3 bucket was not created in the US Standard region, you must set the domain to match the appropriate endpoint under the table in the S3 section of the Regions and Endpoints (http://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region) AWS documentation page.

By default logs are dumped in your root directory every hour in a file with the format `YYYY-mm-ddThh:mm:ss-<server id>` but you can change the frequency and path by changing the information in the **Period** and **Path** fields respectively on the **S3 Endpoint** window. S3 logs rotate into new files at a timestamp determined by truncating the time you configure in the **Period** field. For example, a period of 86400 rotates at midnight UTC, 3600 at the top of every hour, and so on. We do this so that even across service versions each set of logs will be contain a well-defined period of time; it also makes the timestamps friendlier.

Keep in mind that although we continuously stream logs into S3, their UI and API do not make files available for access until after the upload is complete.

· Log Streaming: Cloud Files (/guides/log-streaming/log-streaming-cloudfiles)

IMPORTANT: Be sure to review our information on Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming) before following the provider-specific information for Rackspace Cloud Files shown below.

Create a Cloud Files User and Container

Start by creating a Cloud Files user with restricted permissions via Rackspace's cloud console (https://mycloud.rackspace.com/).

1. Navigate to **Account > User Management**.
2. Click **Create User** and fill in all appropriate details.
3. In the **Account Access** section, select **Custom**.
4. Review the **Product Access** list and set Role and Access of all except the **Files** item to No Access
5. Set the **Files** item **Role** to **Admin** and **Capabilities** to **View, Create, Edit, Delete**. This will allow you to create the files to store the logs in, but not access any other services.
6. Show the API key in the **Login details** and make a note of it. You'll use the API key later to set up how logging authenticates with the Cloud Files API.

Once the Cloud Files user is created, set up a Cloud Files container.

1. Navigate to **Files** and click **Create Container**.
2. Assign the container a meaningful name like `Fastly logs - my service`.
3. Choose what region to keep the files in and make sure the container is private.

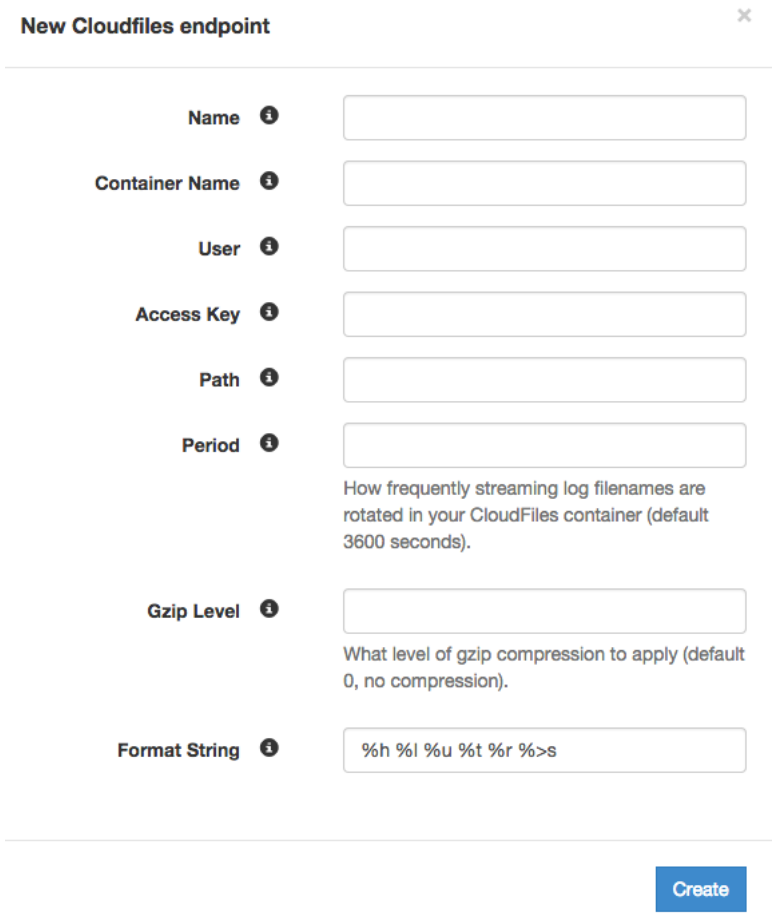
Create a Cloud Files Endpoint

Once you have the Cloud Files user and container created, create a new endpoint in the Fastly application.

1. Log in the Fastly application.
2. Click the **configure** tab (the wrench icon) to access the control panel.

1. Select the appropriate service from the **Service** drop down.
2. Click the blue **Configure** button to the right of the service name.

3. Click the **Logging** section.
4. Find the **Cloud Files** area and click **New** to add a new Cloud Files endpoint. The New Cloud Files Endpoint window appears.

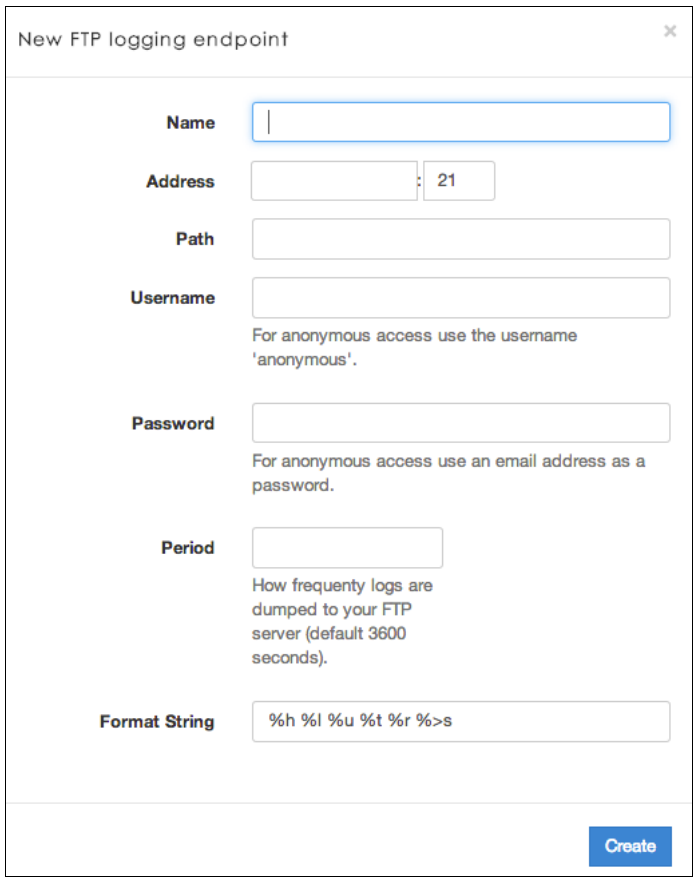


1. Fill out the fields as follows:
   - In the **Name** field, type a meaningful name for the new endpoint. For example, `Rackspace Cloud Files Logging`.
   - In the **Container Name** field, type the name of the container you just created. For example, `Fastly logs - my service`.
   - In the **User** field, type the name of the user created in the Rackspace cloud console above.
   - In the **Access Key** field, type the API key you noted previously.
   - In the **Path** field type the path in which the logs will be stored being sure to end the path with a forward slash (`/`). Remember that these are only pseudo paths in Cloud Files for logical separation.
   - In the **Period** field, type a time in seconds. This determines the time period in seconds over which to buffer logs into a single file. The default period is 3600 seconds (1 hour).
   - In the **Gzip Level** field type whole number between `0` and `9`. This number represents the level of gzipping Fastly will perform on the log files. A gzip level of `0`, the default, represents uncompressed logs that are the fastest to deliver and easiest to read. They also consume the most space in the container. A gzip level of `9` will produce the most compressed log files to save maximum amounts of space, but will be the slowest to be delivered to the container.
   - In the **Format String** field, set the logging format as noted in the log streaming setup (/guides/log-streaming/setting-up-remote-log-streaming).
2. Click **Create** to create the new logging endpoint.

· Log Streaming: FTP (/guides/log-streaming/log-streaming-ftp)

IMPORTANT: Be sure to review our information on Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming) before following the provider-specific information for FTP shown below.

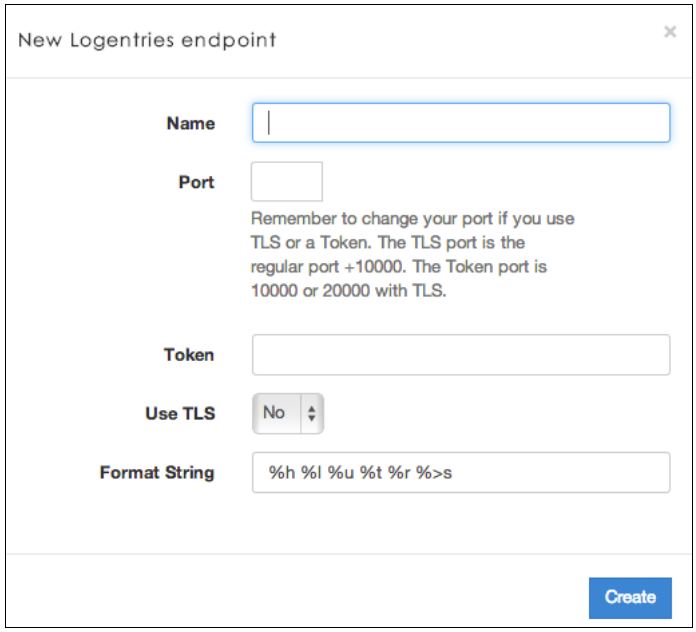
Fastly supports periodic dumping to both password-protected and anonymous FTP.



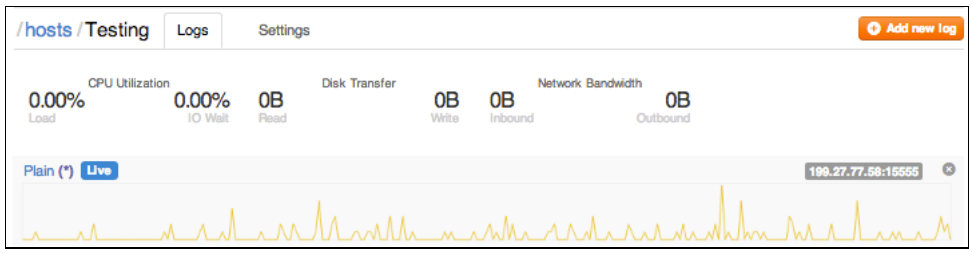
· Log Streaming: Logentries (/guides/log-streaming/log-streaming-logentries)

IMPORTANT: Be sure to review our information on Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming) before following the provider-specific information for Logentries shown below.

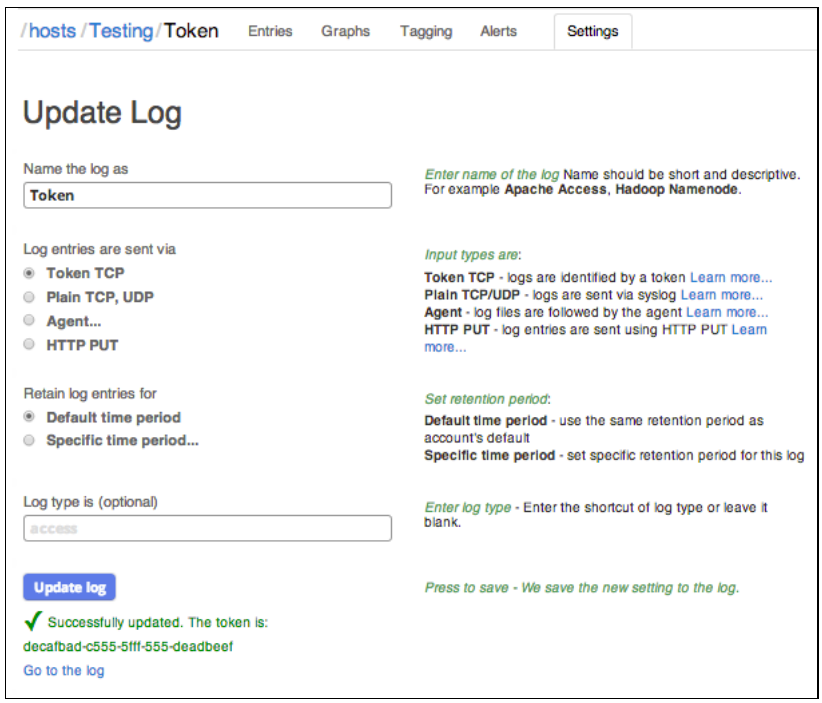
Fastly has the ability to log to Logentries either via plain TCP, TCP+TLS, Token, or Token+TLS.



Remember to change your port number if you switch between TCP or Token and Plain or TLS. You can find your port on your Logentries config panel (in the example below, it's 15555, which appears at the bottom right of the image, just above the graph):



Your token appears at the bottom of the Update Log screen when your create or update a log of the type 'Token':



· Log Streaming: Papertrail (/guides/log-streaming/log-streaming-papertrail)

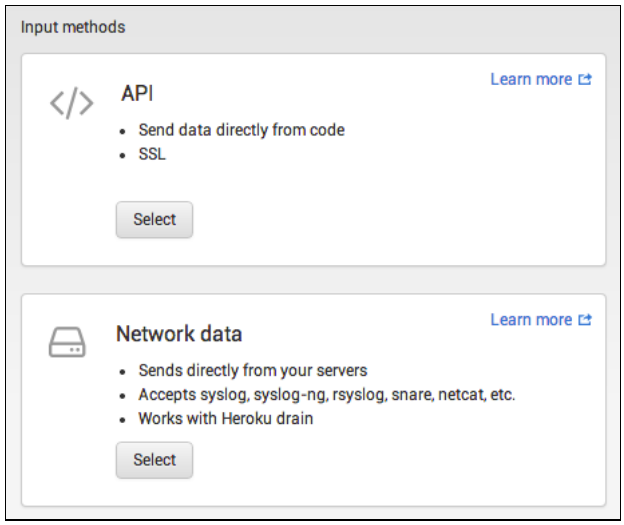
All Papertrail traffic occurs over TCP and TLS (http://en.wikipedia.org/wiki/Transport_Layer_Security). Instructions for setting up remote log streaming via Papertrail are detailed fully in the Papertrail setup and configuration documentation (http://help.papertrailapp.com/kb/hosting-services/fastly/). Be sure to review our information on Setting Up remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming) before following the Papertrail instructions.

DISCLAIMER: This is a service provided by a third party and governed by their terms of service. Fastly accepts no responsibility for third party services or their availability, performance, security or compliance with law.

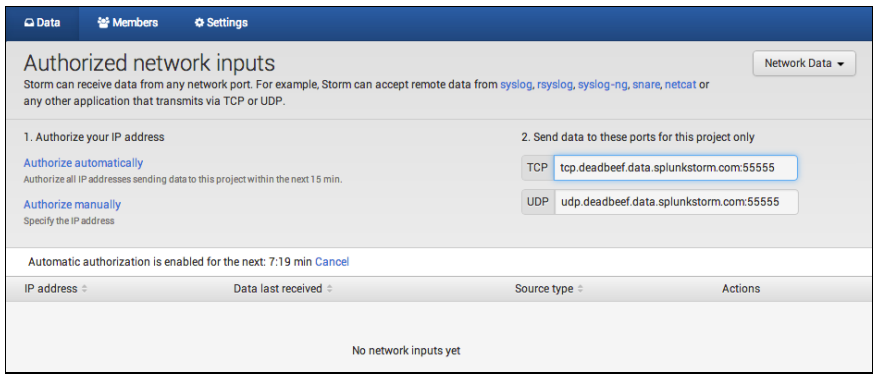
· Log Streaming: Splunk Storm (/guides/log-streaming/log-streaming-splunk-storm)

IMPORTANT: Be sure to review our information on Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming) before following the provider-specific information for Splunk Storm shown below.

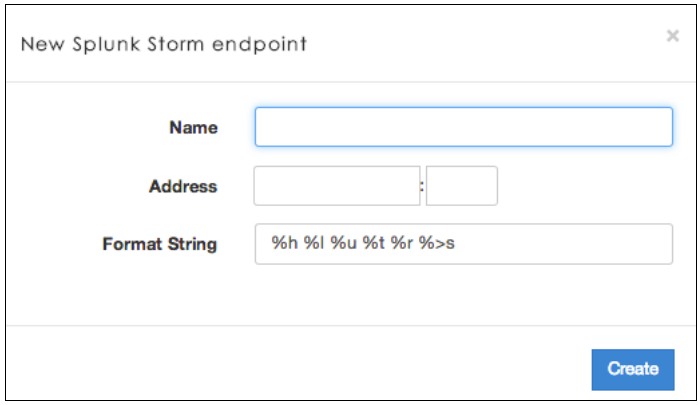
To configure your Splunk Storm endpoint, create a new Network data input method in your Splunk Storm account:



Copy the address and port (in the example below, they are tcp.deadbeef.data.splunkstorm.com and 55555):



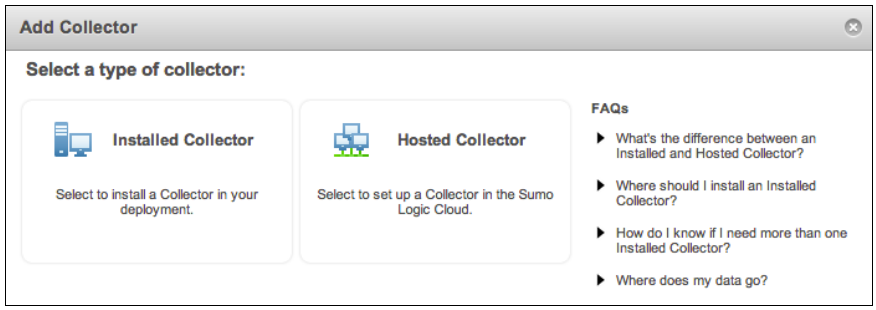
and paste them in the Address and Port fields in the New Splunk Storm endpoint window in the Fastly application:



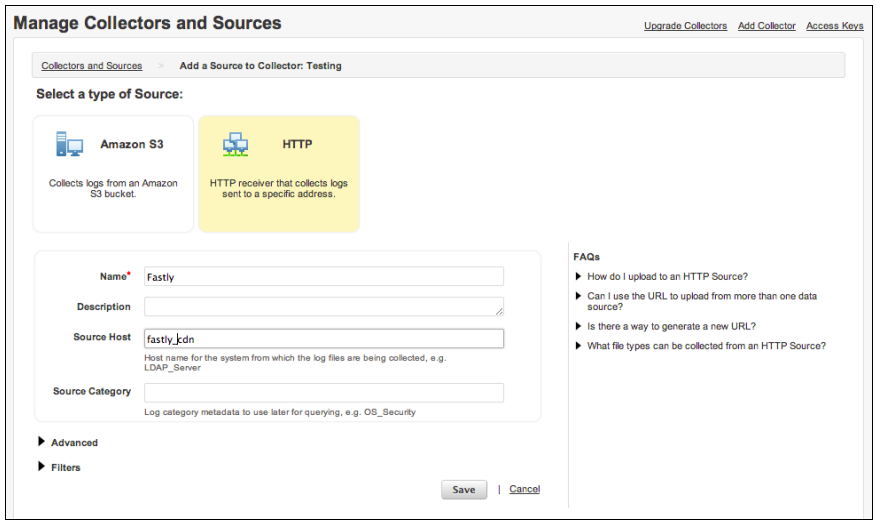
· Log Streaming: Sumologic (/guides/log-streaming/log-streaming-sumologic)

IMPORTANT: Be sure to review our information on Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming) before following the provider-specific information for Sumologic shown below.

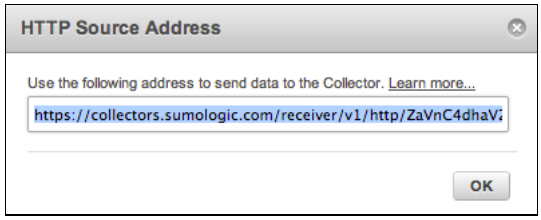
To set up Sumologic, first add a new Hosted Collector via your Sumologic account:



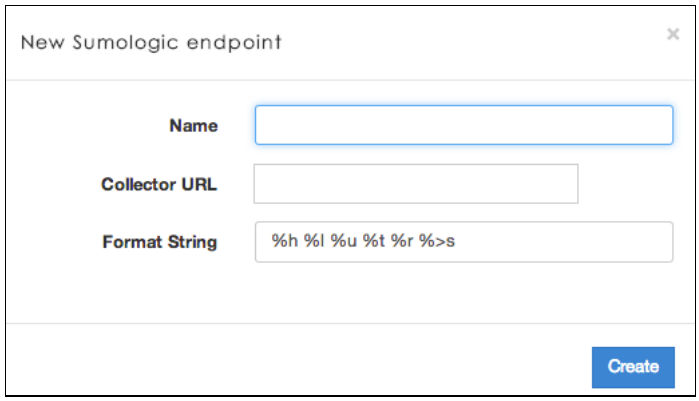
Then add a new source to it when prompted, making sure to select the HTTP as the source type:



Copy the URL for the HTTP Source Address given to you:



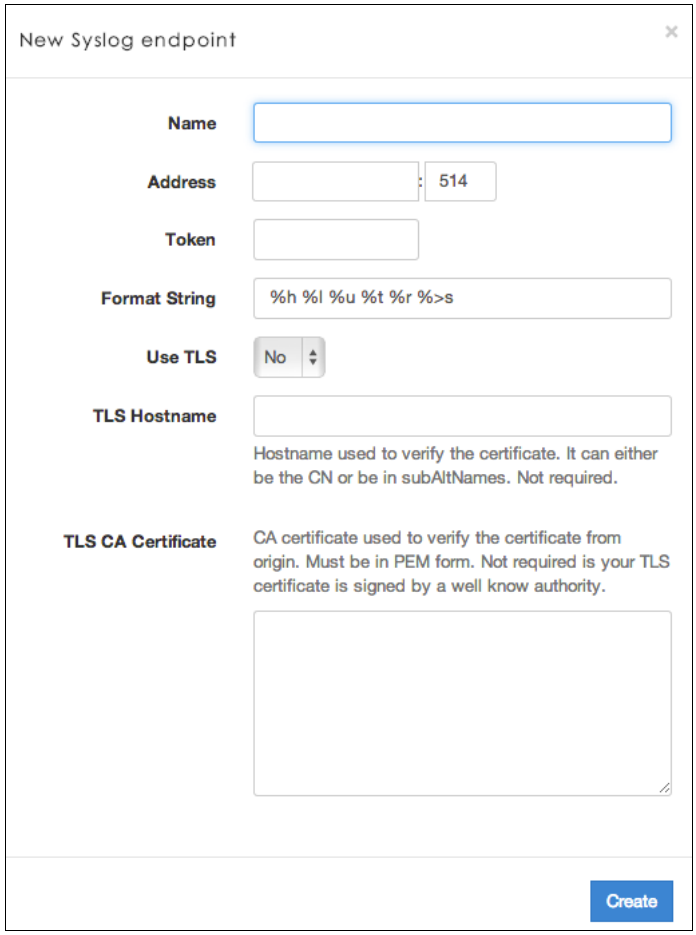
and paste it into the **Collector URL** field in the **New Sumologic endpoint** window in the Fastly application:



· Log Streaming: Syslog (/guides/log-streaming/log-streaming-syslog)

IMPORTANT: Be sure to review our information on Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming) before following the provider-specific information for Syslog shown below.

Fastly allows you encrypt all logging traffic to your server using TLS (Transport Layer Security). In addition, if your TLS certificate is not signed by a well-known certificate authority, you can upload your own certificate.



Keep in mind that logs are streamed over TCP not UDP.

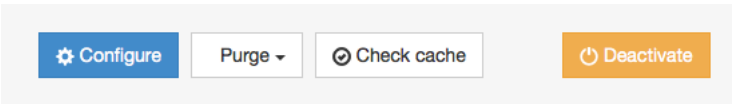
For Advance Configuration in Syslog, you can add a **Token**. The token is a string prefix that will be set before the syslog line on your log server.

· Setting Up Remote Log Streaming (/guides/log-streaming/setting-up-remote-log-streaming)

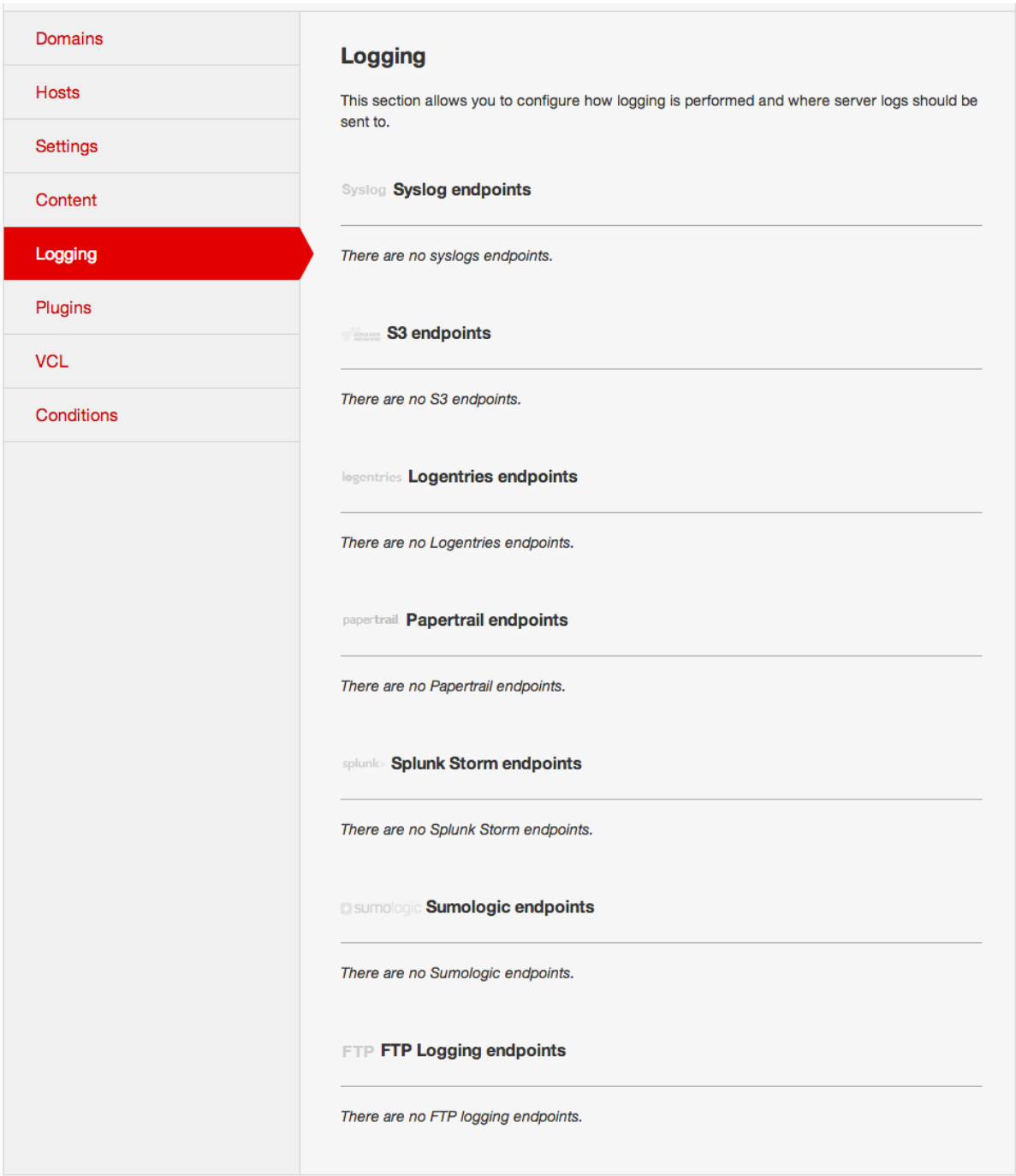
We've made some big improvements to Fastly's logging infrastructure to deal with increased traffic. In addition, we've added support for TLS-encrypted syslogs and have the ability to stream logs to Amazon S3 as well as several other hosted logging providers. To set up remote log streaming, follow the steps below.

Setting Things Up

Logging options appear in the configuration settings. Once you've logged into the Fastly application, click the config tab and then click the blue **Configure** button in the top middle of the screen while viewing the appropriate service:



Once the configuration settings appear, click the **Logging** option on the left:



To create new logging settings, click the **New** button on the right side of a provider name. The new logging window for that specific provider will appear. At the end of this support article, we've included some examples of the specific information you'll need to enter for each of the supported providers.

Format Strings

Regardless of which provider you create logging for, Apache style format strings define what gets logged (e.g., `%h %l %u %t %r %>s %b`). The available formats are:

String	Description
`h`	remote IP

`l`	remote log name
`u`	remote user
`t`	the time
`r`	the first line of the request
`>s`	the status
`b`	the content size of the response

Note: Even though we adopted the default LogFormat in Apache, `l` and `u` don't apply in our context. They always appear as the hardcoded value `"-"`.

Examples of Other Useful Things You Can Log

Other useful things you can log include:

- `resp.http.X-Cache` - add HIT/MISS logging
- `req.http.LiterallyAnyHeaderYouWant` - any request header from the client
- `req.http.Accept-Language` - logs users' Accept Language

or any Varnish variable (https://www.varnish-cache.org/docs/2.1/reference/vcl.html#variables).

Key Things to Remember About Logs

Keep in mind that logs are streamed over TCP not UDP. Additionally, if you are using custom VCL be sure to include the `#FASTLY deliver` (/guides/vcl-faqs/how-do-i-mix-and-match-fastly-vcl-with-custom-vcl) macro in your vcl_deliver handler.

Once you've deployed your changes you should start seeing logs on your log server within a few minutes. There is an approximate two-minute delay in logs actually sending data, as well as a two-minute delay from an event happening to it getting logged.

By default, logs are dumped in your root directory every hour in a file using the naming format `YYYY-mm-ddThh:mm:ss-<server id>` (but you are able to change the frequency and path). There are four log files created each hour. Fastly has four different log-server aggregation points. Each will send logs to you and none of the four files will contain duplicate entries.

Note: If your syslog traffic exceeds your normal service traffic, please contact sales@fastly.com (mailto:sales@fastly.com).

Firewalling

Our logs can be sent from two locations, and since there's no security in plain syslog, it is best to firewall your syslog server and only allow TCP traffic through on your configured port from the following two netblocks:

- 199.27.72.0/24
- 199.27.77.0/24

Additional Information for Specific Logging Provider Methods

- Syslog (/guides/log-streaming/log-streaming-syslog)
- Amazon S3 (/guides/log-streaming/log-streaming-amazon-s3)
- Logentries (/guides/log-streaming/log-streaming-logentries)
- Papertrail (/guides/log-streaming/log-streaming-papertrail)
- Splunk Storm (/guides/log-streaming/log-streaming-splunk-storm)
- Sumologic (/guides/log-streaming/log-streaming-sumologic)
- FTP (/guides/log-streaming/log-streaming-ftp)