CLOUDFLARE | PARTNER NETWORK

ASA
CLOUDFLARE | PARTNER NETWORK
Accredited Services Architect

Managed Performance

Management and Optimization of Cloudflare's core Performance Solutions

Agenda

Performance Services Review

Advanced Caching Behavior

Cache Tuning

Argo Smart Routing & Tiered Caching

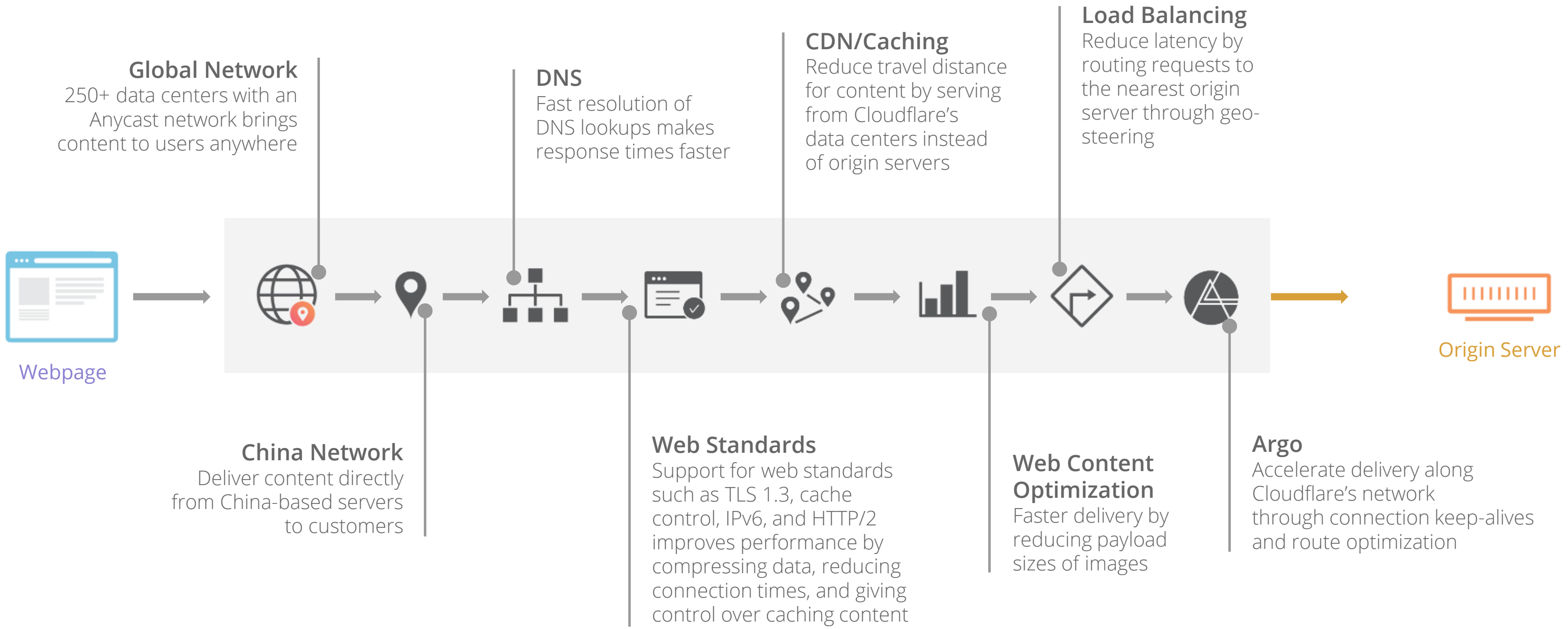
Cloudflare Browser Insight & Analytics

Appendix:
- Stream and Stream Delivery
- Cloudflare Image & Image Resizing
- Updates on Cloudflare Speed Week 2021

Chrisanthy Carlane
Partner Tech Enablement
ccarlane@cloudflare.com

Declan Carlin
Partner Solution Engineer
EMEA
declan@cloudflare.com

Cloudflare Performance Services Review



Global Network
250+ data centers with an Anycast network brings content to users anywhere

China Network
Deliver content directly from China-based servers to customers

DNS
Fast resolution of DNS lookups makes response times faster

Web Standards
Support for web standards such as TLS 1.3, cache control, IPv6, and HTTP/2 improves performance by compressing data, reducing connection times, and giving control over caching content

CDN/Caching
Reduce travel distance for content by serving from Cloudflare's data centers instead of origin servers

Web Content Optimization
Faster delivery by reducing payload sizes of images

Load Balancing
Reduce latency by routing requests to the nearest origin server through geo-steering

Argo
Accelerate delivery along Cloudflare's network through connection keep-alives and route optimization

CLOUDFLARE | PARTNER NETWORK

What’s not included

- Image and Image Resizing
- Stream and Stream Delivery

CLOUDFLARE

// Advanced Caching

Reference:
https://developers.cloudflare.com/cache/

Cloudflare Cache Architecture Review

- Cloudflare hard drives are all encrypted

- Cache has a local eviction manager that tries to maintain SSD drive usage at ~80%. It evicts the least recently used (**LRU**) assets first. Internal alerts occur if capacity exceeds 80%

- We store files on disk at the MD5 hash of their cache key.
- Each cache-key file contains the NGINX metadata (~500 bytes) and the full response header and body.

- The metadata contains information such as: age, expiry date, size, cache tags, etc. When the cached file is fetched, we always check the cache key in case of MD5 collision.

- Whitepaper: How the Cloudflare network maintains data privacy
- Read: https://developers.cloudflare.com/cache/about/cache-keys

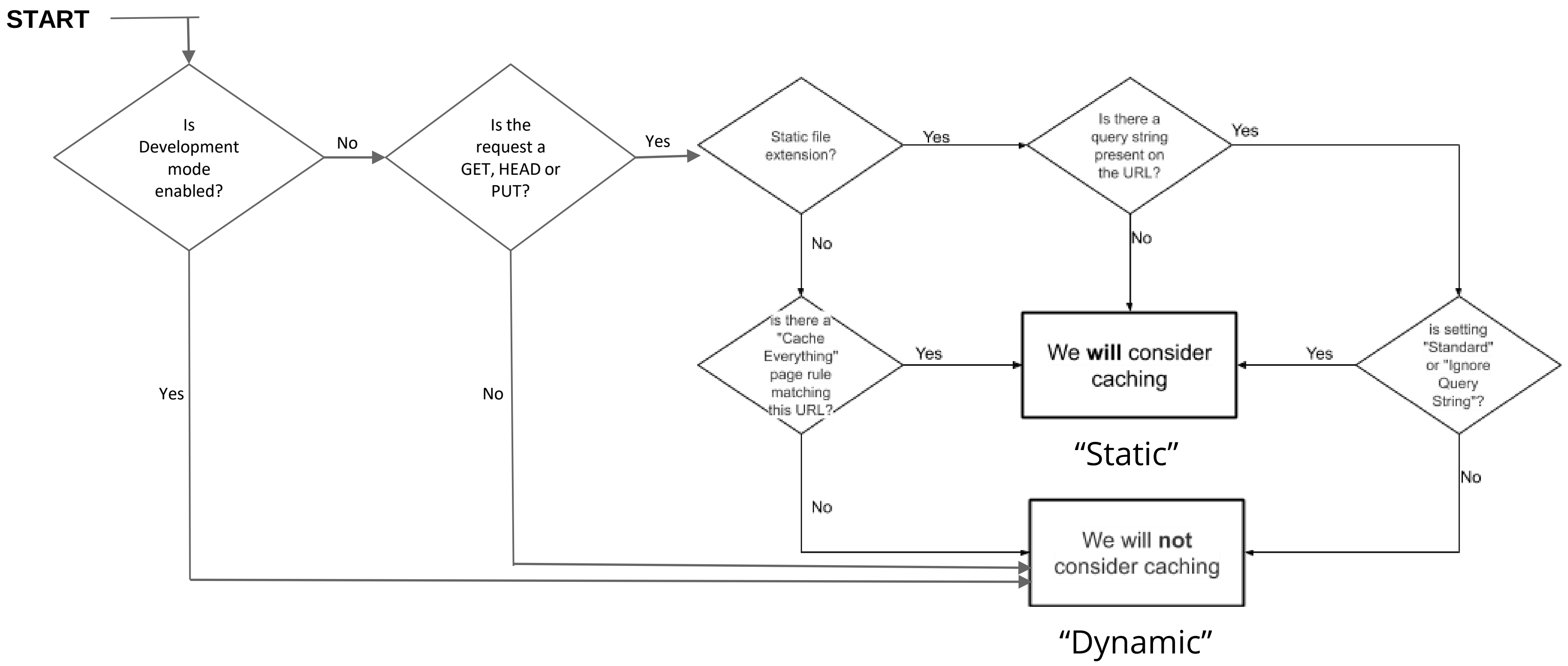
Cacheability

For every request received, Cloudflare will resolve zone ownership, and use the features that are enabled for that zone or for that URL to decide whether to cache:

- Development mode - Temporarily disables Cloudflare's cache for up to 3 hours
- Static File Extension Check - Is this a default caching extension?
- HTTP Method - Only GET, HEAD, and PUT requests are cached by default
    - POST method can be cached using Cloudflare Workers
- Cache Levels:
    - **No Query String:** Only delivers resources from cache when there is no query string. (i.e.com/pic.jpg)
    - **Ignore Query String:** Delivers the same resource to everyone independent of the query string. (ie.com/pic.jpg?~~ignore=this)~~
    - **Standard**: Delivers a different resource each time the query string changes to prevent cache collisions. (i.e.com/pic.jpg?with=query) Recommended

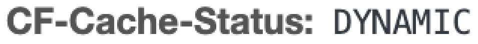
https://developers.cloudflare.com/cache/about/default-cache-behavior

Cacheability Diagram


START
Is Development mode enabled?
No
Is the request a GET, HEAD or PUT?
Yes
Static file extension?
Yes
Is there a query string present on the URL?
Yes
No
No
is there a "Cache Everything" page rule matching this URL?
Yes
We **will** consider caching
Yes
is setting "Standard" or "Ignore Query String"?
"Static"
Yes
No
No
No
We will **not** consider caching
"Dynamic"

1. Definitely non-cacheable or “Dynamic” Requests

- If a request is determined to be dynamic, Cloudflare will contact the origin directly and bypass nginx-cache.



```
CF-Cache-Status: DYNAMIC
```

2. Possibly cacheable or “Static” Requests

- The cache key of the requested asset is created and served if previously cached.

- Whether we actually store an object is determined during the **response phase**.

3. Customizing Cloudflare Cache

https://support.cloudflare.com/hc/en-us/articles/202775670

Default Cache Behavior

By default, Cloudflare respects the origin web server's cache headers in the following order unless an Edge Cache TTL page rule overrides the headers.

1. Cloudflare does not cache the resource if the Cache-Control header is set to private, no-store, no-cache, or max-age=0 or if there is a cookie in the response.
2. Cloudflare caches the resource in the following scenarios:
    - The Cache-Control header is set to public and the max-age is greater than 0.
    - The Expires header is set to a future date.
3. If both the max-age and an Expires header are set, max-age is used.

Default cached file extensions

- Cloudflare only caches based on file extension and not by MIME type.
- The Cloudflare CDN does not cache HTML by default.
- Additionally, Cloudflare caches a website's robot.txt.

AVIF	BMP	EJS	JPEG	PDF	PS	TTF
CLASS	EOT	JPG	PICT	SVG	WEBP	
CSS	EPS	JS	PLS	SVGZ	WOFF	
CSV	GIF	MID	PNG	SWF	WOFF2	
DOC	ICO	MIDI	PPT	TIF	XLS	
DOCX	JAR	OTF	PPTX	TIFF	XLSX	

Cache Customization Options

Cloudflare’s CDN provides 3 cache customization options:

1. Caching behavior for individual URLs via Cloudflare Page Rules
2. Customize caching with Cloudflare Workers
3. Adjust caching level, cache TTL, and more via the Cloudflare Caching app

Cloudflare limits the upload size (HTTP POST request size) per plan type:

- 100MB Free and Pro
- 200MB Business
- 500MB Enterprise by default. Contact Customer Support to request a limit increase.

CF-Cache-Status Header Values

HIT: Resource was served from the edge cache.

MISS: The resource was not in cache, this request was served by the origin.

EXPIRED: The resource was in cache but was considered stale. This request was served by the origin. This can either be because the resource has been modified or purged.

STALE: Expired resource is in cache, served from cache because an origin connection errored out.

BYPASS: Same as a MISS. Only seen when Origin Cache Control is enabled in Page Rule (for Enterprise only).

REVALIDATED: A stale representation of the object is in our cache, but was revalidated by using either an If-Modified-Since header or an If-None-Match header

UPDATING: The resource was is in the process of being revalidated so the stale resource was served to save time.

Cache Determination

The following will influence whether Cloudflare can cache an asset:

- **Cache-Control** header
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control
- **Expires header** : If there is a Cache-Control header with the max-age or s-maxage directive in the response, the Expires header is ignored
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expires
- **Set-Cookie header**
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie
- **HTTP Status code**
  https://developer.mozilla.org/en-US/docs/Web/HTTP/Status
- **File Size per Cloudflare** maximums : The maximum file size Cloudflare's CDN caches is 512MB for Free, Pro, and Business customers and 5GB for Enterprise customers. Enterprise customers can contact their Cloudflare Account Team to request caching of larger files.
- **Query string** per Cache Level setting

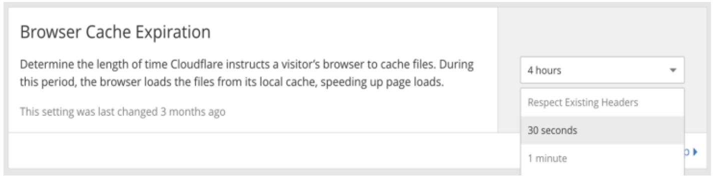
Cache Determination - Cache-Control

- CDN-Cache-Control is a response header field set on the origin to separately control the behavior of CDN caches from other intermediaries that might handle a response. You can set the CDN-Cache-Control or Cloudflare-CDN-Cache-Control response header using the same directives used with the Cache-Control
- Origin Cache-Control headers to tell Cloudflare how to handle content from the origin
- Review this article:

https://developers.cloudflare.com/cache/about/cache-control

Cacheability - Browser Cache TTL Settings

Zone wide setting:



Browser Cache Expiration

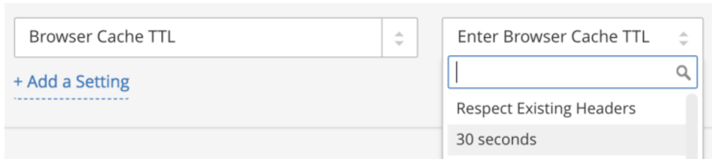
Determine the length of time Cloudflare instructs a visitor's browser to cache files. During this period, the browser loads the files from its local cache, speeding up page loads.

This setting was last changed 3 months ago

4 hours

Respect Existing Headers
30 seconds
1 minute

Page Rules setting:



Browser Cache TTL

+ Add a Setting

Enter Browser Cache TTL

Respect Existing Headers
30 seconds

The setting applies to only **'cacheable requests'.**

Browser Cache Expire TTL parameter is the time that Cloudflare instructs a visitor's browser to cache a resource by setting a max-age value. Until this time expires, the browser will load the resource from its local cache thus speeding up the request significantly. *(TTL settings <30 minutes are ENT only features.)*

NOTE:

Default : Cloudflare will add a Cache-Control header to the resources matched by the Page Rule if there is no Cache-Control header added to the resource by the origin server or Cloudflare will replace the Cache-Control header added by the origin server only if the Browser Cache TTL is longer than the TTL in the Cache-Control header set by the origin.

Override behavior using Respect Existing headers.

Cacheability - Set-Cookie

- If an origin returns a ‘set-cookie’ response header, we will default to not cache the response - MISS.
- HOWEVER - If the response also includes a public and/or max-age directive, Cloudflare will cache the response and strip the set-cookie header...
- OR - If a Cache Everything + Edge Cache TTL rule is set, this will “Force Cache” all responses, Cloudflare strips the cookie before serving to other clients.

Cacheability - HTTP Status Codes

- These are the default values we use if a valid TTL/don’t cache directive isn’t sent by the origin.

```
proxy_cache_valid          200 206 301          120m;
proxy_cache_valid          302 303              20m;
proxy_cache_valid          403                  1m;
proxy_cache_valid          404                  3m;
proxy_cache_valid          410                  3m;
proxy_cache_valid          any                  0s;
```

- **Note, we don’t store 5xx errors.**

- We do store 404s which is *useful for primitive DDoS protection* but not useful when combined with poorly considered custom cache keys.

List of Caching Related Page Rules

Page Rules can customize caching actions on a per path basis, adjusting for things like:

- Cache Level: Bypass - Skip the cache for this path.
- Cache Everything - Treats all content as static and caches all file types
- Custom Cache Keys - Control specific variables to include when deciding which resources to cache.
- Cache by Device Type -Separate cached content based on the visitor's device
- Bypass Cache on Cookie - Skip the cache when a cookie matches a regular expression
- Cache on Cookie - Apply the Cache Everything option when a cookie is present based on a regular expression
- Edge Cache TTL - Set a custom Edge Cache TTL
- Origin Cache Control - Respect origin cache headers as edge
- Respect Strong ETags - Turn on or off byte-for-byte equivalency checks

Summary of all Page Rule Settings

Cacheability - Workers

Cloudflare Workers Cache API can be used to provide fine grained control of reading and writing to the cache.

Common Actions:

`cacheEverything`:This option forces Cloudflare to cache the response for this request, regardless of what headers are seen on the response. This is equivalent to setting the page rule "Cache Level" (to "Cache Everything").

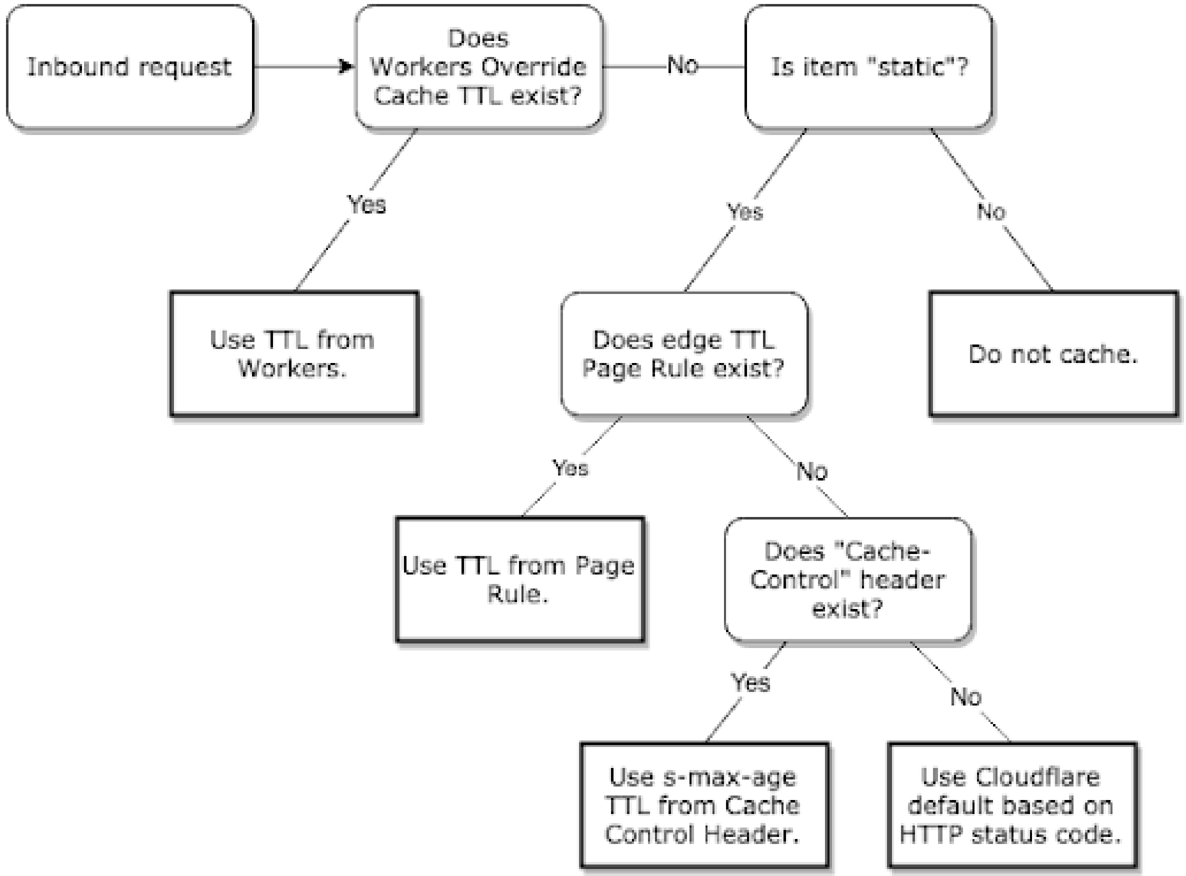
`cacheTtl`: This option forces Cloudflare to cache the response for this request, regardless of what headers are seen on the response. This is equivalent to setting two page rules: "Edge Cache TTL" and "Cache Level" (to "Cache Everything").

`cacheKey`: A request's cache key is what determines if two requests are "the same" for caching purposes. If a request has the same cache key as some previous request, then we can serve the same cached response for both. (Enterprise only)

`cacheTtlByStatus`: This option is a version of the `cacheTtl` feature which chooses a TTL based on the response's status code. If the response to this request has a status code that matches, Cloudflare will cache for the instructed time, and override cache instructives sent by the origin. (e.g. `{ "200-299": 86400, 404: 1, "500-599": 0 }`) (Enterprise only)

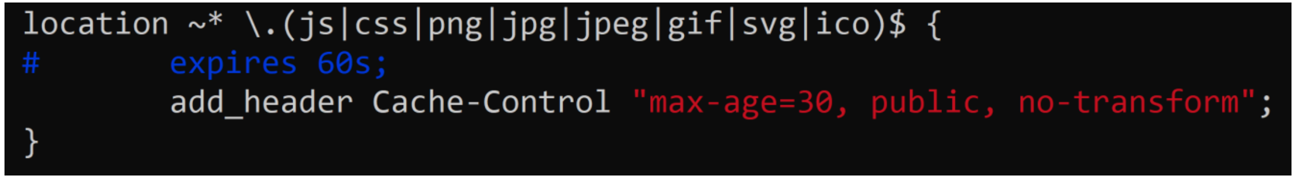
The Cache API is useful if you need to cache POST requests. It is also useful for overriding Cloudflare default cache behaviours for advanced configurations.
Note - In the event conflicting settings are applied by Workers and Page Rules, the Worker takes precendence

Cache Flow Summary Diagram


Inbound request
Does Workers Override Cache TTL exist?
No
Is item "static"?
Yes
Use TTL from Workers.
Yes
No
Does edge TTL Page Rule exist?
Do not cache.
Yes
No
Use TTL from Page Rule.
Does "Cache-Control" header exist?
Yes
No
Use s-max-age TTL from Cache Control Header.
Use Cloudflare default based on HTTP status code.

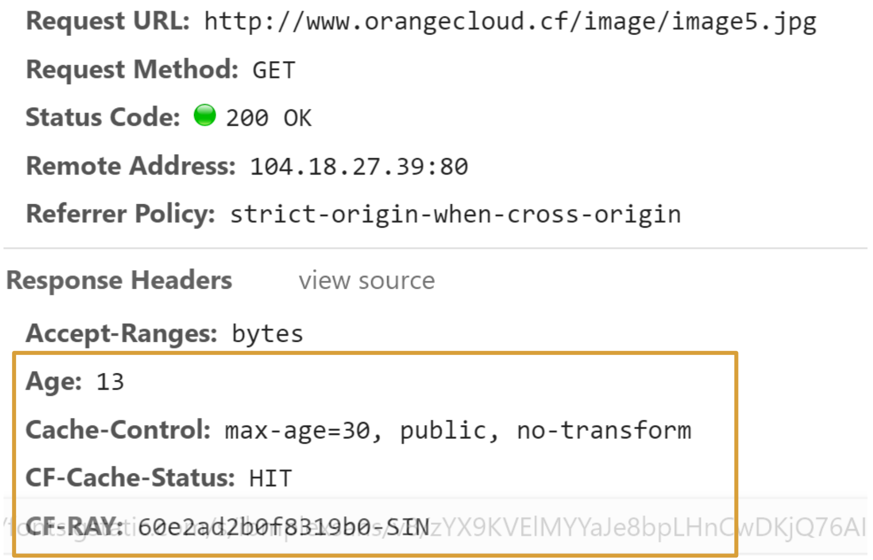
Cache-Control at the Origin Server

This is the Cache-Control configuration in web server: max-age=30, public, no-transform



```
location ~* \.(js|css|png|jpg|jpeg|gif|svg|ico)$ {
#        expires 60s;
         add_header Cache-Control "max-age=30, public, no-transform";
}
```

How the Response Headers looks like
When the image is accessed

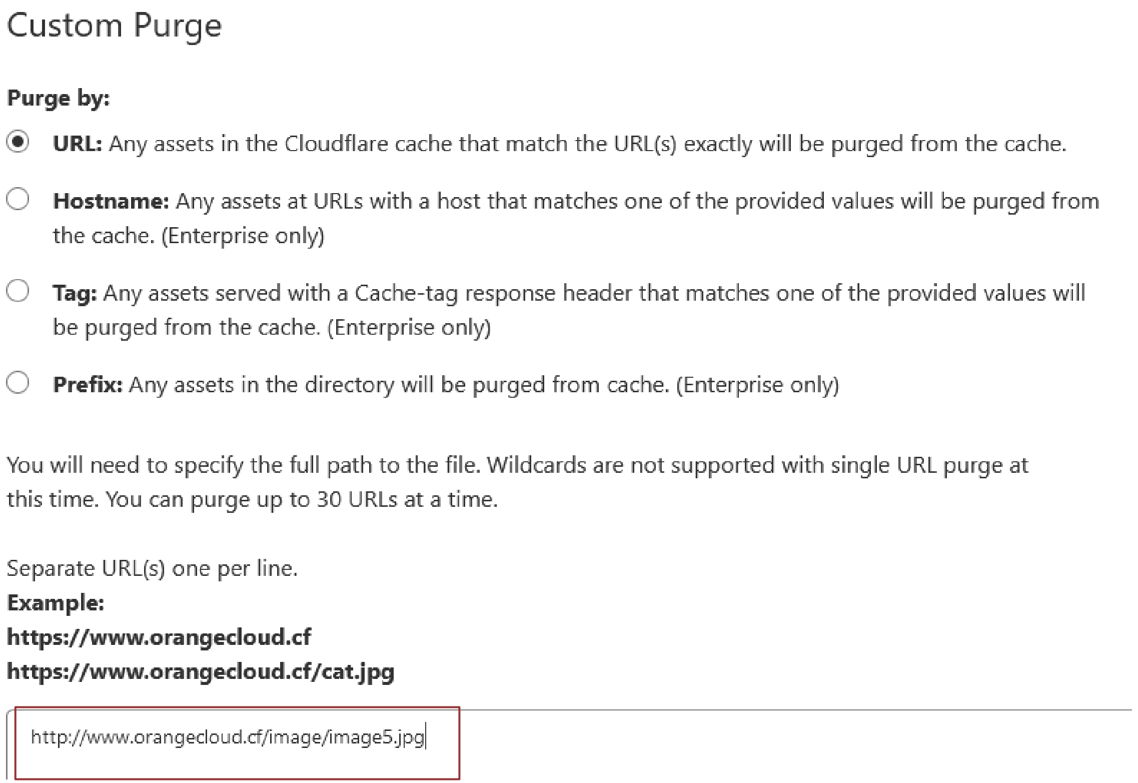


```
Request URL: http://www.orangecloud.cf/image/image5.jpg
Request Method: GET
Status Code: ● 200 OK
Remote Address: 104.18.27.39:80
Referrer Policy: strict-origin-when-cross-origin

Response Headers        view source

Accept-Ranges: bytes
Age: 13
Cache-Control: max-age=30, public, no-transform
CF-Cache-Status: HIT
CF-RAY: 60e2ad2b0f8319b0-SIN
```

Let's Start by Purging the Cache



Custom Purge

Purge by:

- (●) **URL:** Any assets in the Cloudflare cache that match the URL(s) exactly will be purged from the cache.
- ( ) **Hostname:** Any assets at URLs with a host that matches one of the provided values will be purged from the cache. (Enterprise only)
- ( ) **Tag:** Any assets served with a Cache-tag response header that matches one of the provided values will be purged from the cache. (Enterprise only)
- ( ) **Prefix:** Any assets in the directory will be purged from cache. (Enterprise only)

You will need to specify the full path to the file. Wildcards are not supported with single URL purge at this time. You can purge up to 30 URLs at a time.

Separate URL(s) one per line.
Example:
https://www.orangecloud.cf
https://www.orangecloud.cf/cat.jpg

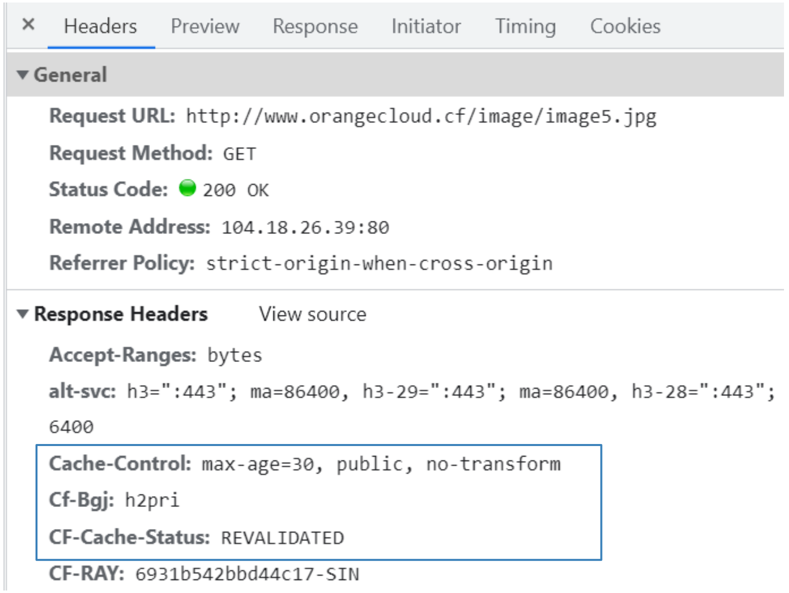
```
http://www.orangecloud.cf/image/image5.jpg
```

Enable Origin Cache Control: On

Page Rules Configuration:

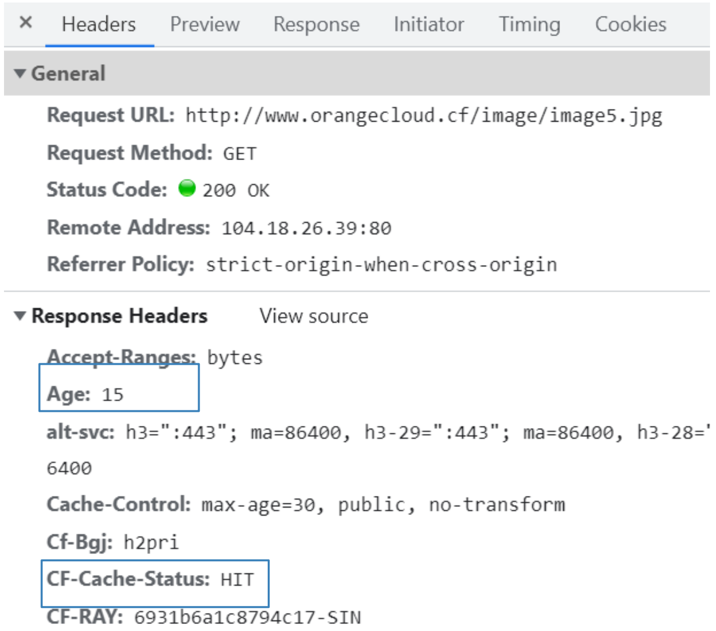
1 *.orangecloud.cf/image/*
Origin Cache Control: On

Starting Point: Cache Status = REVALIDATED



```
×  Headers  Preview  Response  Initiator  Timing  Cookies
▼ General
  Request URL: http://www.orangecloud.cf/image/image5.jpg
  Request Method: GET
  Status Code: ● 200 OK
  Remote Address: 104.18.26.39:80
  Referrer Policy: strict-origin-when-cross-origin
▼ Response Headers     View source
  Accept-Ranges: bytes
  alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3-28=":443";
  6400
  Cache-Control: max-age=30, public, no-transform
  Cf-Bgj: h2pri
  CF-Cache-Status: REVALIDATED
  CF-RAY: 6931b542bbd44c17-SIN
```

Then, Image is cached, Age: 15s

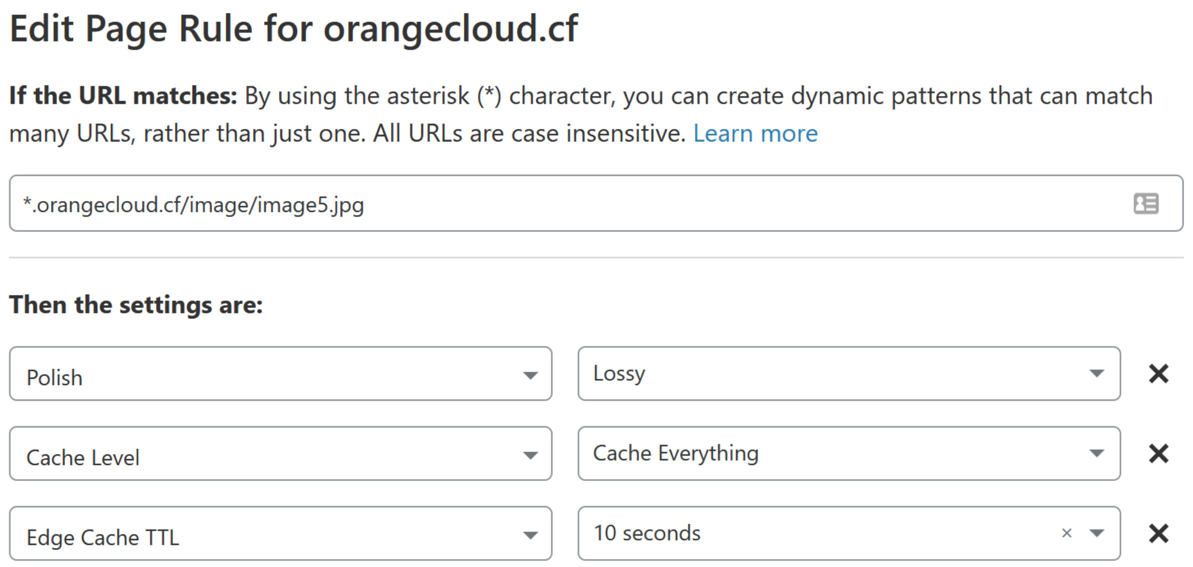


```
×  Headers  Preview  Response  Initiator  Timing  Cookies
▼ General
  Request URL: http://www.orangecloud.cf/image/image5.jpg
  Request Method: GET
  Status Code: ● 200 OK
  Remote Address: 104.18.26.39:80
  Referrer Policy: strict-origin-when-cross-origin
▼ Response Headers     View source
  Accept-Ranges: bytes
  Age: 15
  alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3-28='
  6400
  Cache-Control: max-age=30, public, no-transform
  Cf-Bgj: h2pri
  CF-Cache-Status: HIT
  CF-RAY: 6931b6a1c8794c17-SIN
```

Curl Output

```
* Connected to www.orangecloud.cf (104.18.26.39) port 80 (#0)
> GET /image/image5.jpg HTTP/1.1
> Host: www.orangecloud.cf
> User-Agent: curl/7.58.0
< HTTP/1.1 200 OK
< Date: Thu, 23 Sep 2021 06:35:22 GMT
< Content-Type: image/jpeg
< Cache-Control: max-age=30, public, no-transform
< Cf-Bgj: h2pri
< Last-Modified: Fri, 31 Jan 2020 08:45:42 GMT
< CF-Cache-Status: HIT
< Age: 26
```

Override Origin Cache Control to 10s



Edit Page Rule for orangecloud.cf

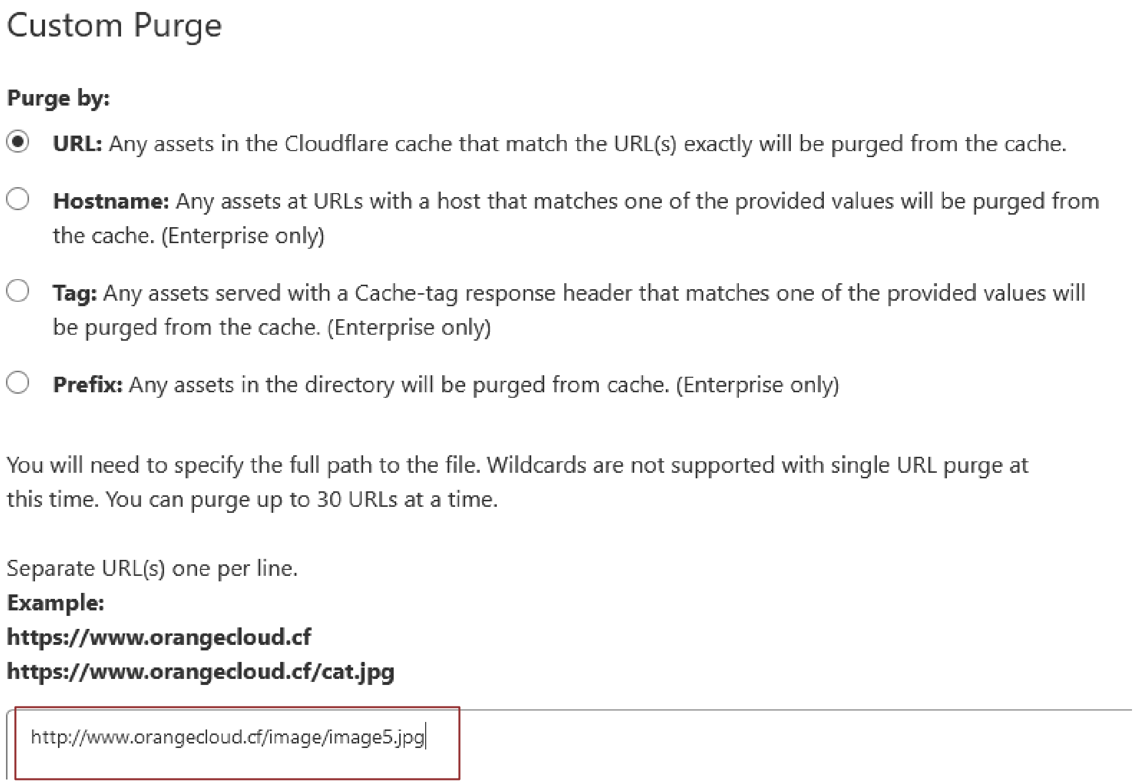
If the URL matches: By using the asterisk (*) character, you can create dynamic patterns that can match many URLs, rather than just one. All URLs are case insensitive. Learn more

*.orangecloud.cf/image/image5.jpg

Then the settings are:

Polish	Lossy
Cache Level	Cache Everything
Edge Cache TTL	10 seconds

Reset Cache by Purging the Cache



Custom Purge

Purge by:

● **URL:** Any assets in the Cloudflare cache that match the URL(s) exactly will be purged from the cache.

○ **Hostname:** Any assets at URLs with a host that matches one of the provided values will be purged from the cache. (Enterprise only)

○ **Tag:** Any assets served with a Cache-tag response header that matches one of the provided values will be purged from the cache. (Enterprise only)

○ **Prefix:** Any assets in the directory will be purged from cache. (Enterprise only)

You will need to specify the full path to the file. Wildcards are not supported with single URL purge at this time. You can purge up to 30 URLs at a time.

Separate URL(s) one per line.
Example:
https://www.orangecloud.cf
https://www.orangecloud.cf/cat.jpg

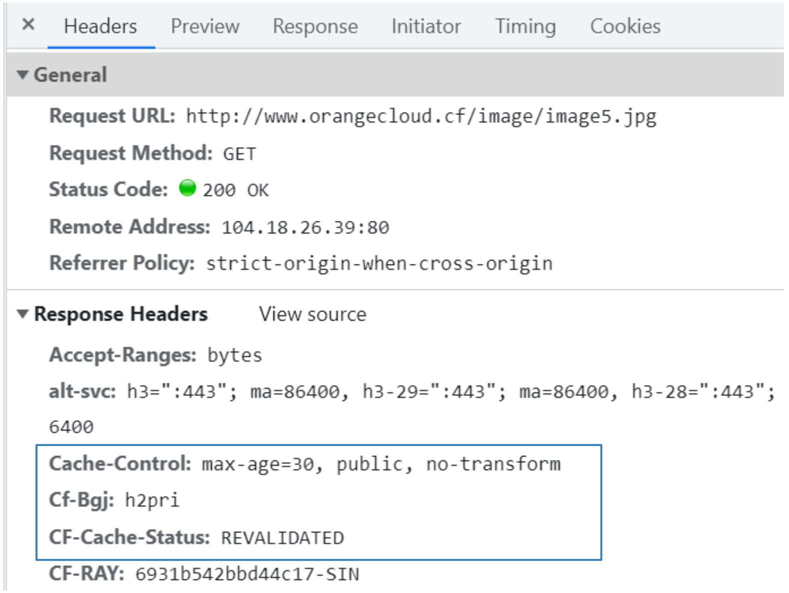
http://www.orangecloud.cf/image/image5.jpg

Origin Cache Control: Off

Page Rules Configuration:

*.orangecloud.cf/image/image5.jpg
Polish: Lossy, Cache Level: Cache Everything, Edge Cache TTL: 10 seconds

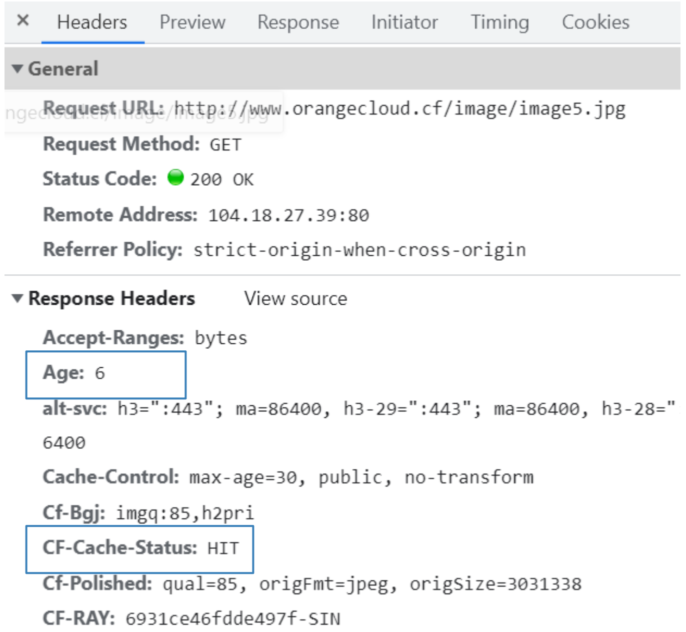
Starting Point: Cache Status = REVALIDATED



```
▼ General
Request URL: http://www.orangecloud.cf/image/image5.jpg
Request Method: GET
Status Code: ● 200 OK
Remote Address: 104.18.26.39:80
Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers    View source
Accept-Ranges: bytes
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3-28=":443";
6400
Cache-Control: max-age=30, public, no-transform
Cf-Bgj: h2pri
CF-Cache-Status: REVALIDATED
CF-RAY: 6931b542bbd44c17-SIN
```

When Page Rules take precedence, Age should be <10s



```
▼ General
Request URL: http://www.orangecloud.cf/image/image5.jpg
Request Method: GET
Status Code: ● 200 OK
Remote Address: 104.18.27.39:80
Referrer Policy: strict-origin-when-cross-origin

▼ Response Headers    View source
Accept-Ranges: bytes
Age: 6
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3-28="
6400
Cache-Control: max-age=30, public, no-transform
Cf-Bgj: imgq:85,h2pri
CF-Cache-Status: HIT
Cf-Polished: qual=85, origFmt=jpeg, origSize=3031338
CF-RAY: 6931ce46fdde497f-SIN
```

Curl Output

```
* Connected to www.orangecloud.cf (104.18.26.39) port 80 (#0)
> GET /image/image5.jpg HTTP/1.1
> Host: www.orangecloud.cf
> User-Agent: curl/7.58.0
< HTTP/1.1 200 OK
< Date: Thu, 23 Sep 2021 06:46:45 GMT

< Cache-Control: max-age=30, public, no-transform
< Cf-Bgj: imgq:85,h2pri
< Cf-Polished: degrade=85, origSize=3031338

< CF-Cache-Status: HIT
< Age: 8
< CF-RAY: 6931d156e8012ed4-SIN
```

Expected behavior:
After 10s, the CF-cache-status changed to REVALIDATED, following Page Rules even though
Max-age = 30

Quiz

Webserver setting for a JSON file: max-age=30, public

What is Cloudflare expected behavior if a JSON file when Origin Cache-Control: ON and no Page Rules configured for the file/asset ?

a. Cloudflare will cache JSON file for 30s
b. Cloudflare will not cache JSON file and present cf-cache-status=DYNAMIC

Answer

Webserver setting for a JSON file: max-age=30, public

What is Cloudflare expected behavior if a JSON file when Origin Cache-Control: ON and no Page Rules configured for the file/asset ?

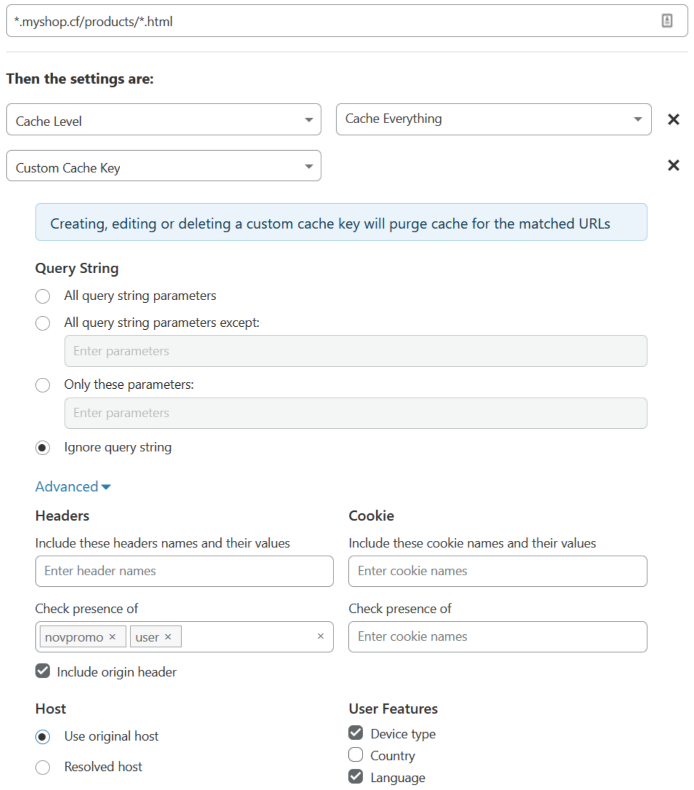
a. Cloudflare will cache JSON file for 30s
b. Cloudflare will not cache JSON file and present cf-cache-status=DYNAMIC

JSON file extension is not cached by default
With no Page Rules configuration, Cloudflare will treat the file as DYNAMIC asset

// Cache Tuning:
Custom Cache Key
Cache Purge

Using Custom Cache Keys

https://support.cloudflare.com/hc/en-us/articles/115004290387



.myshop.cf/products/.html

Then the settings are:

Cache Level: Cache Everything

Custom Cache Key

Creating, editing or deleting a custom cache key will purge cache for the matched URLs

Query String

- ○ All query string parameters
- ○ All query string parameters except: Enter parameters
- ○ Only these parameters: Enter parameters
- ● Ignore query string

Advanced

Headers

Include these headers names and their values: Enter header names

Check presence of: novpromo, user

✓ Include origin header

Cookie

Include these cookie names and their values: Enter cookie names

Check presence of: Enter cookie names

Host

- ● Use original host
- ○ Resolved host

User Features

- ✓ Device type
- ☐ Country
- ✓ Language

Custom Cache Keys control specifically what variables to include when deciding which resources to cache. This allows customers to determine what to cache based on something other than just the URL.

- In the example:
- When the path matches: `http/https://*.myshop.com/product/*.html`
- Cache Everything, if the page variables matches:
- Ignore query string
- These headers exist: novpromo, user
- Original host is myshop.com
- Device type and language header exists

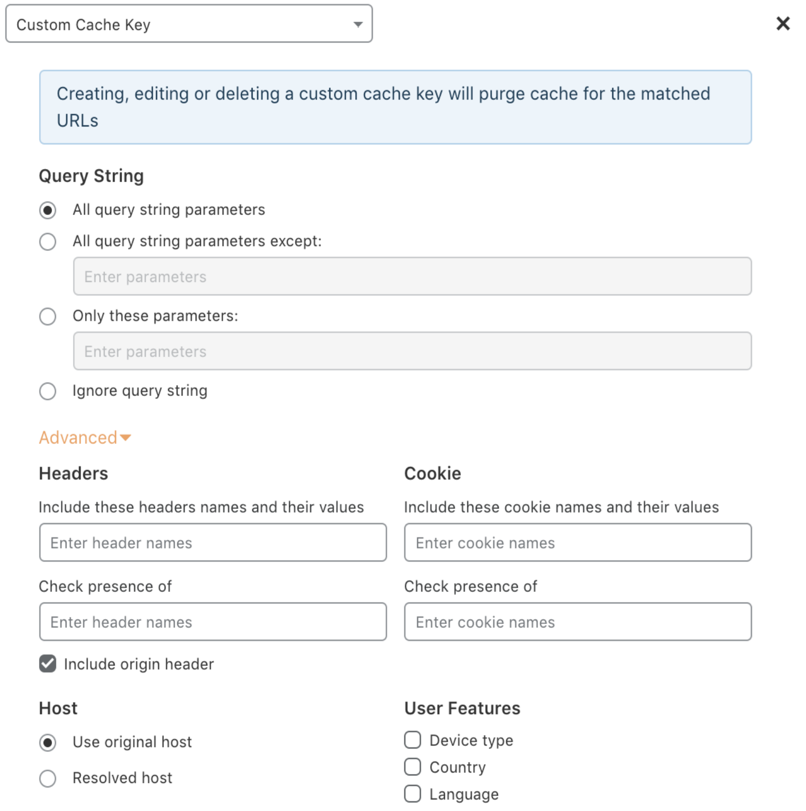
Cache Key Detail - Custom

- Custom cache keys help us to maximize cache hit ratios for customers.

- Configurable in Page Rules and Workers for Enterprise customers
- Can configure:
  - Standard Cache Level
  - Ignore Query String Cache Level
  - Include Query String args
  - Exclude Query String args

  - Include Headers
  - Include Cookies
  - Is Header present
  - Is Cookie present

  - Original host
  - Resolved host


Custom Cache Key

Creating, editing or deleting a custom cache key will purge cache for the matched URLs

Query String

[✓ All query string parameters]

All query string parameters except:

Enter parameters

Only these parameters:

Enter parameters

Ignore query string

Headers	Cookie
Include these headers names and their values	Include these cookie names and their values
Enter header names	Enter cookie names
Check presence of	Check presence of
Enter header names	Enter cookie names

[✓ Include origin header]

Host	User Features
[✓ Use original host]	Device type
Resolved host	Country
	Language

Purging the Cache

Purging Options:

1. **Purge Everything** - Purge Everything may be a dangerous option. For high traffic sites this will likely DDoS the origin.
2. **Custom Purge**
   a. URL
   b. Hostname
   c. Tag
   d. Prefix

a. Purge by URL

Deletes files from Cloudflare cache.
Up to 30 URLs per API call, no wildcard
Globally cleared in ~30 seconds.
Cleared in dashboard or via API.
When custom cache keys are used – cache purge must be done through the API. (Except custom cache keys that use {geo} and {devicetype} params).

Reference: https://developers.cloudflare.com/cache/how-to/purge-cache

Remember… You can’t remotely purge browser caches, so set TTL’s wisely.

Purging the Cache (cont)

b. Purge by Host (Enterprise only)
Invalidates all files for a specific hostname/subdomain, No need to individually specify all files,Useful for SSL for SaaS customers.
ENT only feature.

c. Purge by Tag (Enterprise only)
Gives customer granular purge control, 30 tags at a time,Cache-Tag header must be added to resource response headers. Up to 2000 purges per 24 hour period....
Maximum tag size: 1K per tag, 16K for all tags, and 1000 tags total.

d. Purge by Prefix (Enterprise Only)

Enterprise customers can purge their cache by URL prefix or path separators in their URL. Purging by prefix is useful for many occasions, such as: purge everything within a directory; increase control over cached objects in a path; simplify the number of purge calls sent

Purging by Prefix - Detail

Purge by Prefix

Purge by prefix will allow customers to purge based on path separators in their URL.

Given a URL like https://www.example.com/foo/bar/baz/qux.jpg

Customers are able to purge any of the following:

- www.example.com/*
- www.example.com/foo/*
- www.example.com/foo/bar/*
- www.example.com/foo/bar/baz/*
- www.example.com/foo/bar/baz/qux.jpg

- This is ENT only (for now)
- URI query strings & fragments do not purge by prefix currently
  - www.example.com/foo?a=b (query string)
  - www.exmaple.com/foo#bar (fragment)
- Limit 31 path separators deep for a prefix (example.com/a/b/c/d/e/f/g/h/i/j/k/l/m…)
- Limited to 30 prefixes per call, traditional purge rate-limits apply

Why might I want to Purge by Prefix:

- You want to purge everything within a directory
- You have more control over cached objects in a path
- You can simplify the number of purge calls sent
- You have more control over what to purge and what to keep in cache

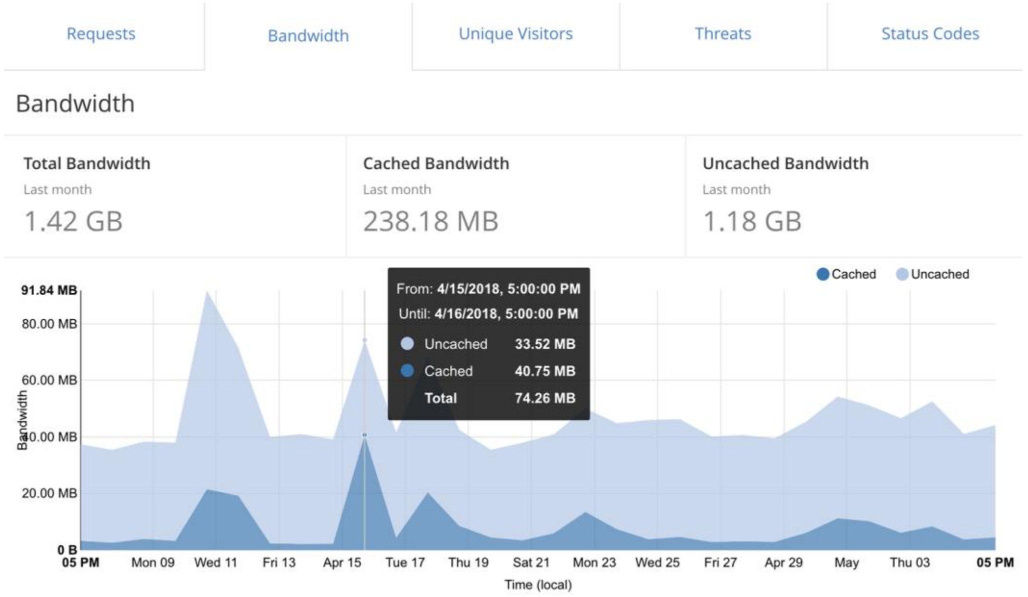
How do I do this:

Currently this is available via the Purge API.

```
 curl -X POST
"https://api.cloudflare.com/client/v4/zones/XXXXXXX
XXXXXXXXXXXXXXXXXXXXXXX/purge_cache" \
     -H "X-Auth-Email: user@example.com" \
     -H "X-Auth-Key:
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX" \
     -H "Content-Type: application/json" \
     --data
'{"prefixes":["www.example.com/foo","www.example.co
m/bar/baz"] }
```

Cloudflare Analytics

- Cloudflare ‘Analytics’ shows the amount of bandwidth used serving requests from origin vs served from cache.
- Total cache HIT’s and REVALIDATED responses (as well as security block and Cloudflare error pages) are recorded as cached bandwidth.
- More detailed analysis of traffic is available using Cloudflare Logs



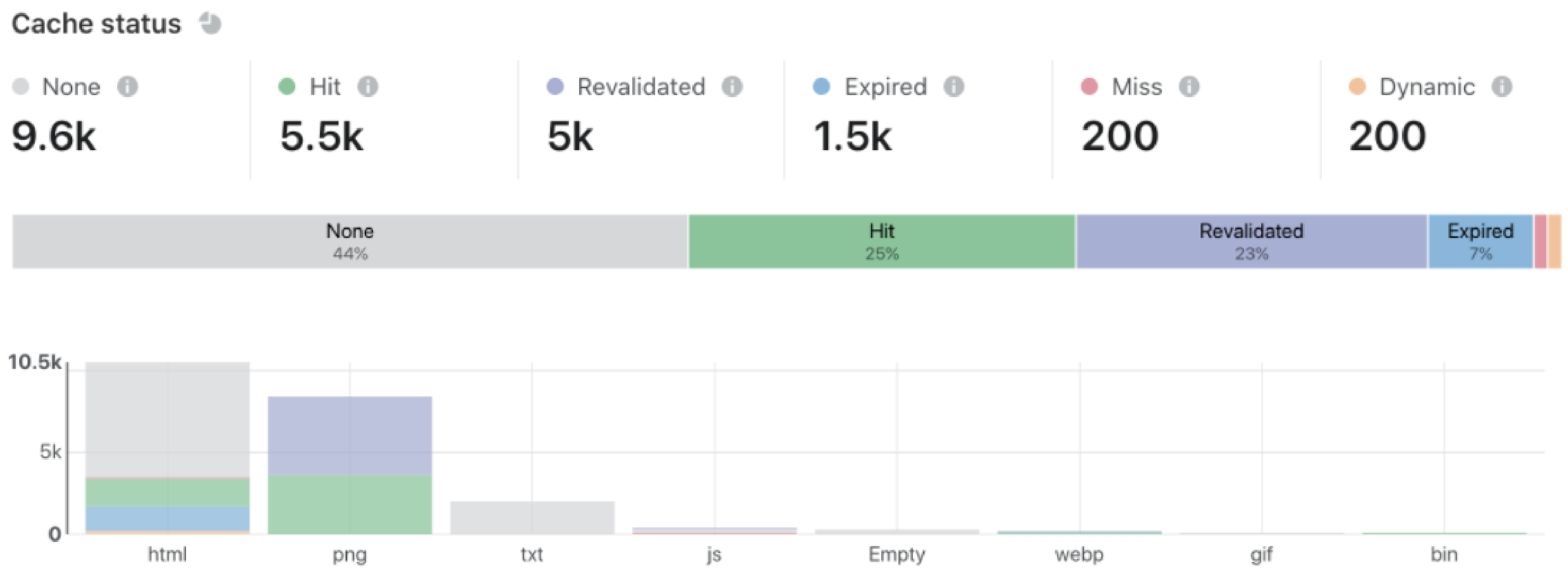
Bandwidth

Total Bandwidth	Cached Bandwidth	Uncached Bandwidth
Last month	Last month	Last month
1.42 GB	238.18 MB	1.18 GB

From: 4/15/2018, 5:00:00 PM
Until: 4/16/2018, 5:00:00 PM

● Uncached	33.52 MB
● Cached	40.75 MB
Total	74.26 MB

Lab - Cache and Network Analytics Dashboard



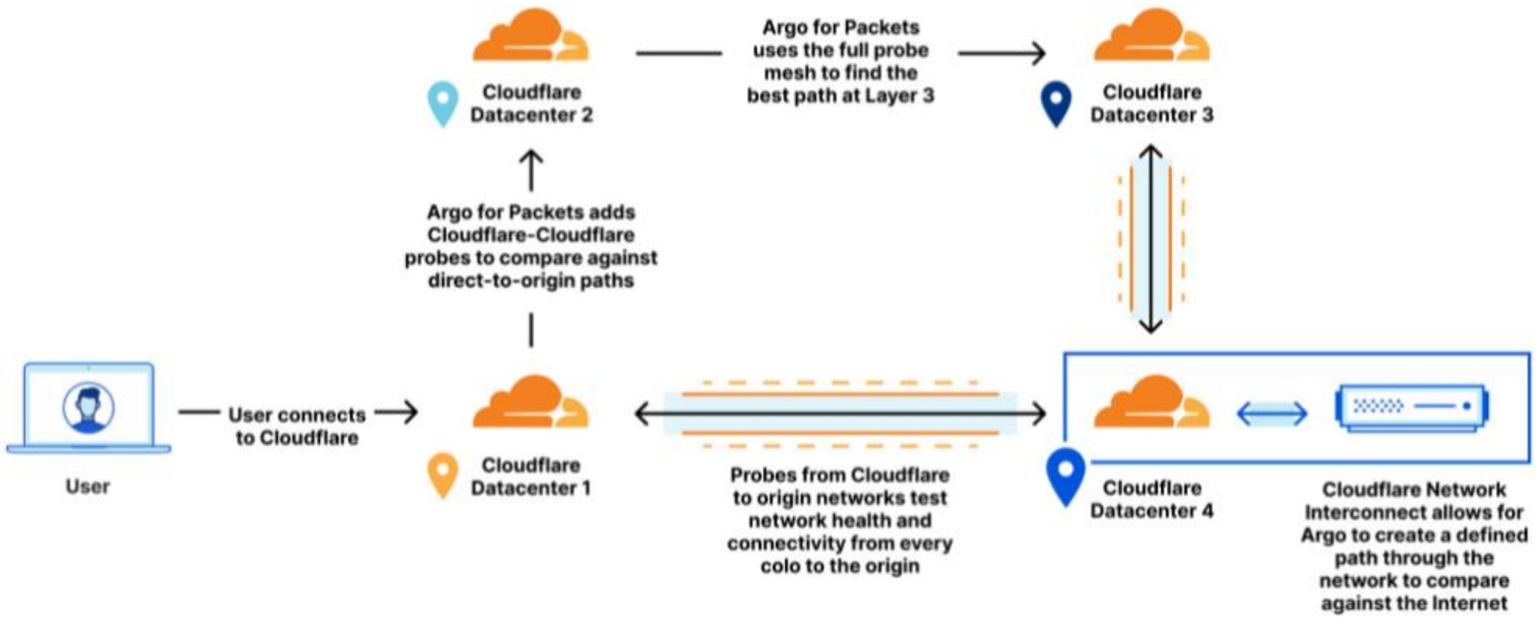
Cache status

None	Hit	Revalidated	Expired	Miss	Dynamic
9.6k	5.5k	5k	1.5k	200	200

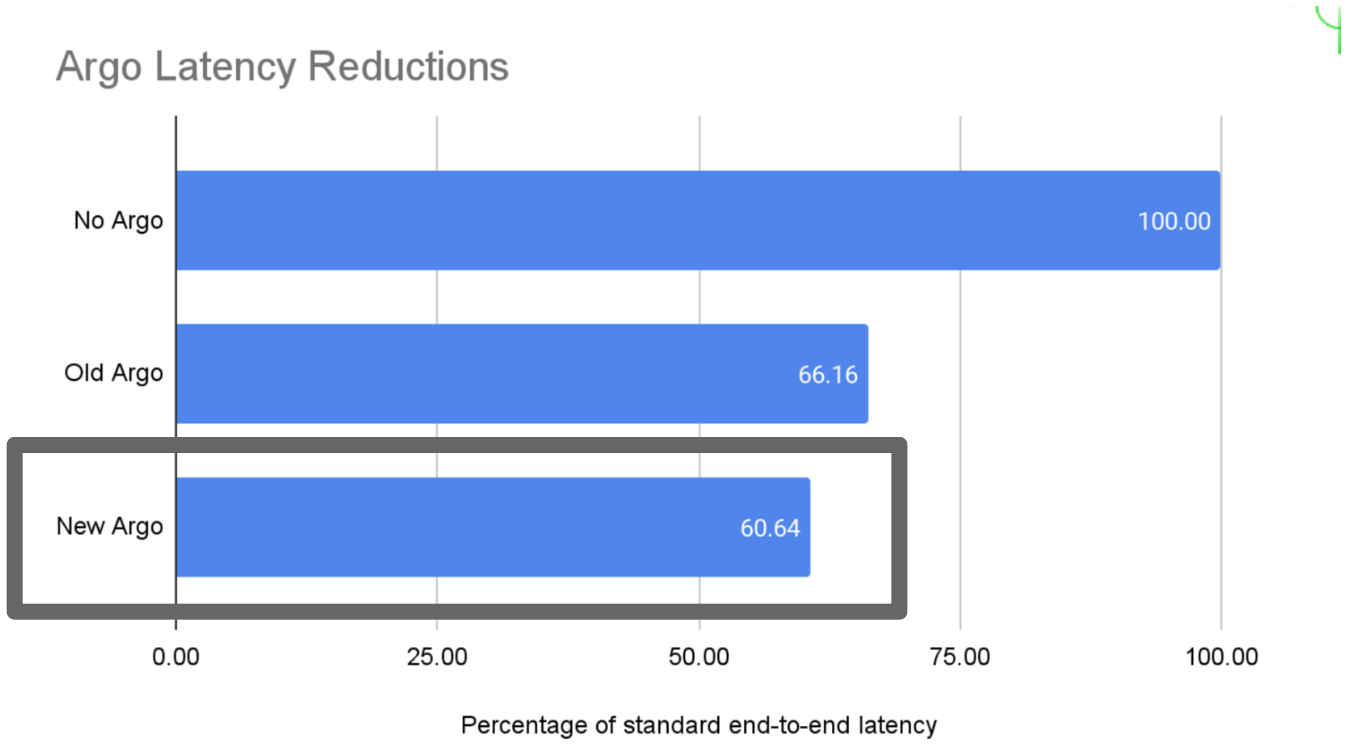
Reference: Understanding Cache Analytics

// Argo Smart Routing

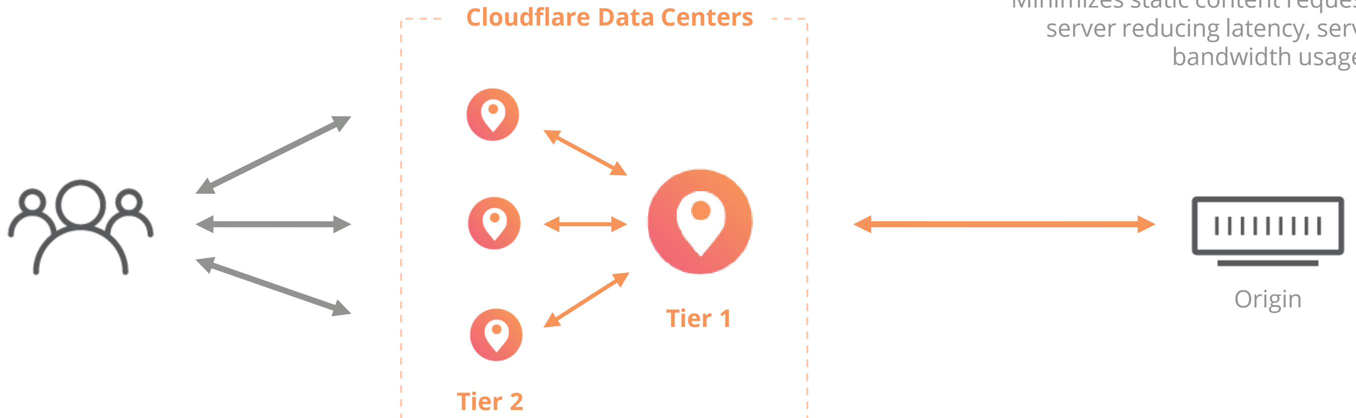
Argo Smart Routing


Argo for Packets uses the full probe mesh to find the best path at Layer 3
Cloudflare Datacenter 2
Cloudflare Datacenter 3
Argo for Packets adds Cloudflare-Cloudflare probes to compare against direct-to-origin paths
User connects to Cloudflare
User
Cloudflare Datacenter 1
Probes from Cloudflare to origin networks test network health and connectivity from every colo to the origin
Cloudflare Datacenter 4
Cloudflare Network Interconnect allows for Argo to create a defined path through the network to compare against the Internet

Argo 2.0 (Smart Routing)



Argo Tiered Caching - Introduction



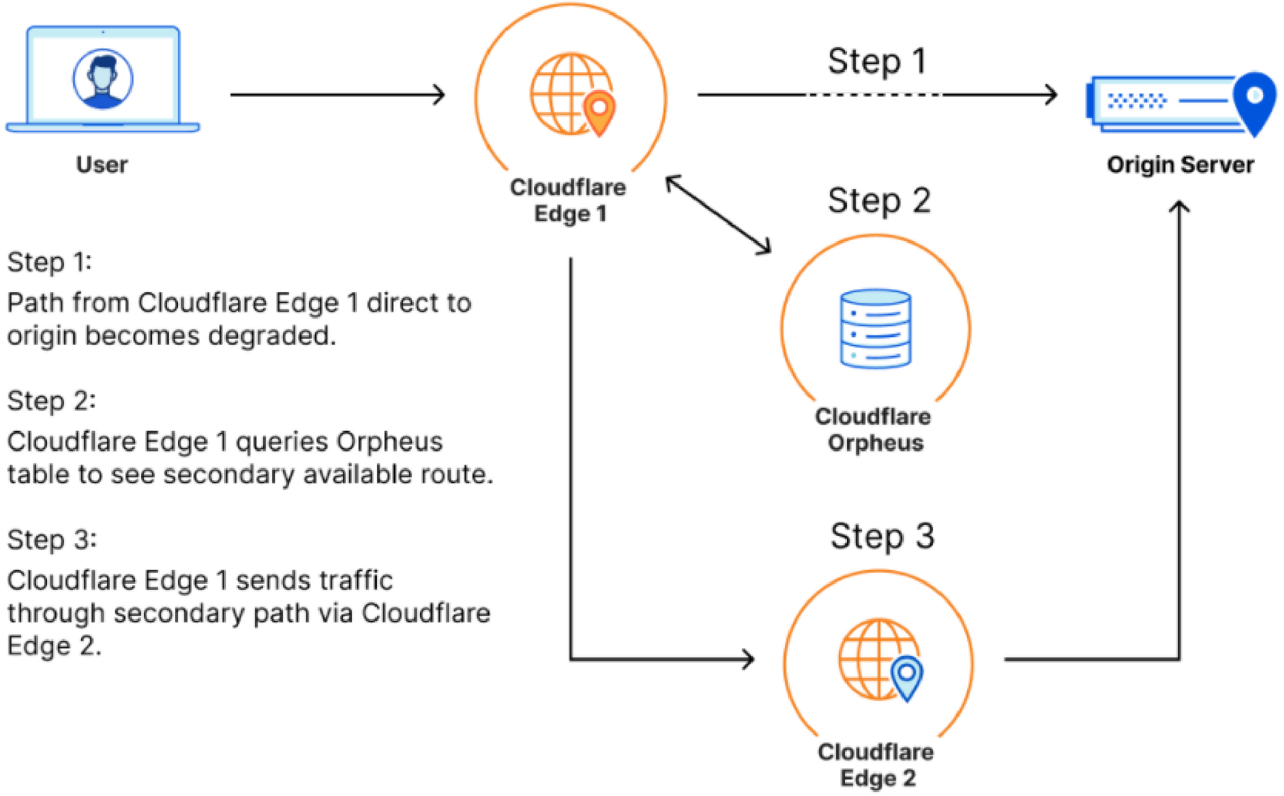
Tier 1 data centers propagate content to a global network of data centers in 200+ cities

Reduced Costs

Minimizes static content requests to the origin server reducing latency, server load and bandwidth usage

Topologies can be customized.

Argo Tiered Caching with Orpheus



Step 1:

Path from Cloudflare Edge 1 direct to origin becomes degraded.

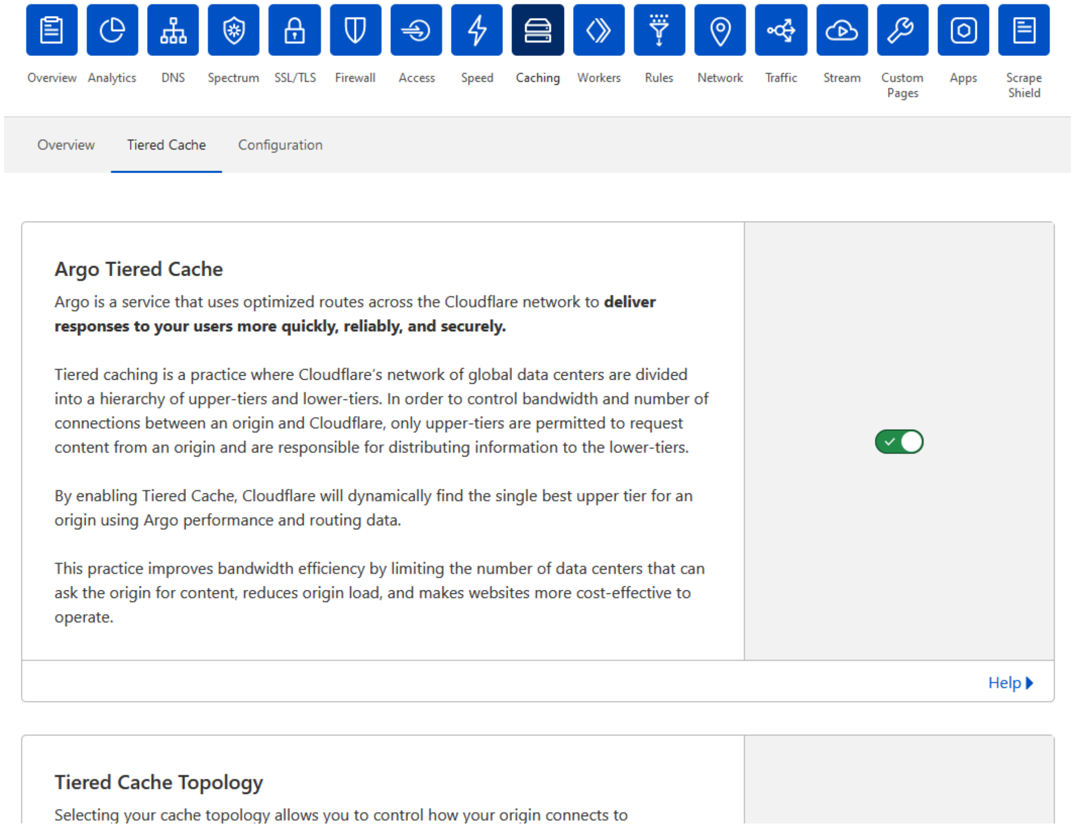
Step 2:

Cloudflare Edge 1 queries Orpheus table to see secondary available route.

Step 3:

Cloudflare Edge 1 sends traffic through secondary path via Cloudflare Edge 2.

Argo Tiered Cache Dashboard



Argo Tiered Cache

Argo is a service that uses optimized routes across the Cloudflare network to **deliver responses to your users more quickly, reliably, and securely.**

Tiered caching is a practice where Cloudflare's network of global data centers are divided into a hierarchy of upper-tiers and lower-tiers. In order to control bandwidth and number of connections between an origin and Cloudflare, only upper-tiers are permitted to request content from an origin and are responsible for distributing information to the lower-tiers.

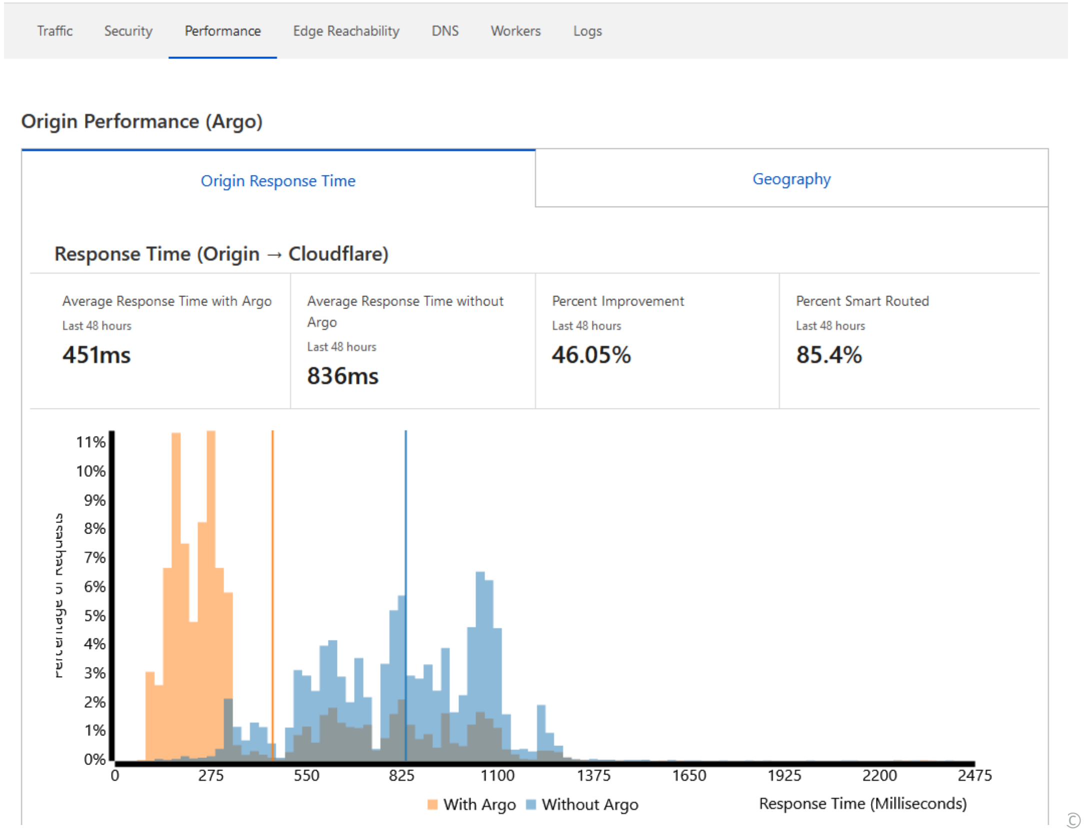
By enabling Tiered Cache, Cloudflare will dynamically find the single best upper tier for an origin using Argo performance and routing data.

This practice improves bandwidth efficiency by limiting the number of data centers that can ask the origin for content, reduces origin load, and makes websites more cost-effective to operate.

Tiered Cache Topology

Selecting your cache topology allows you to control how your origin connects to

Argo | Analytics



Traffic Security Performance Edge Reachability DNS Workers Logs

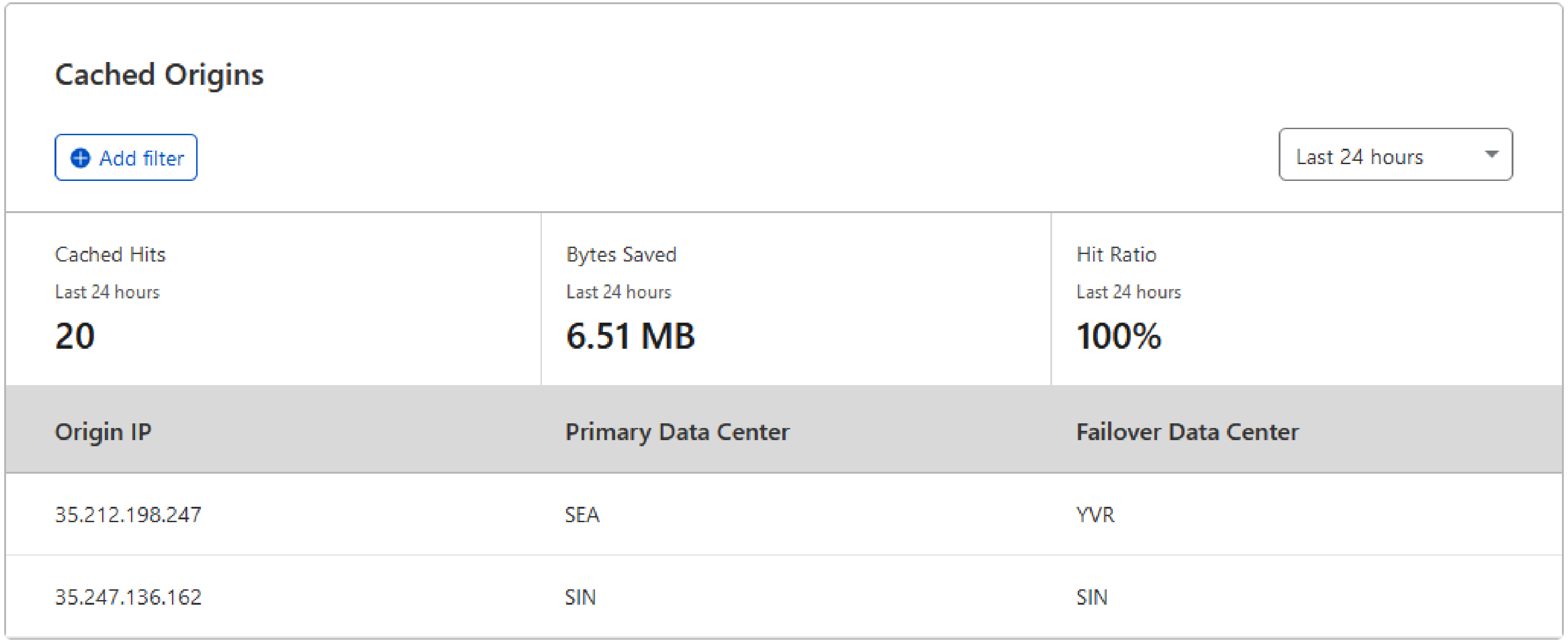
Origin Performance (Argo)

Origin Response Time | Geography

Response Time (Origin → Cloudflare)

Average Response Time with Argo	Average Response Time without Argo	Percent Improvement	Percent Smart Routed
Last 48 hours	Last 48 hours	Last 48 hours	Last 48 hours
451ms	836ms	46.05%	85.4%

Argo | Analytics



Cached Origins

Add filter

Last 24 hours

Cached Hits	Bytes Saved	Hit Ratio
Last 24 hours	Last 24 hours	Last 24 hours
20	6.51 MB	100%

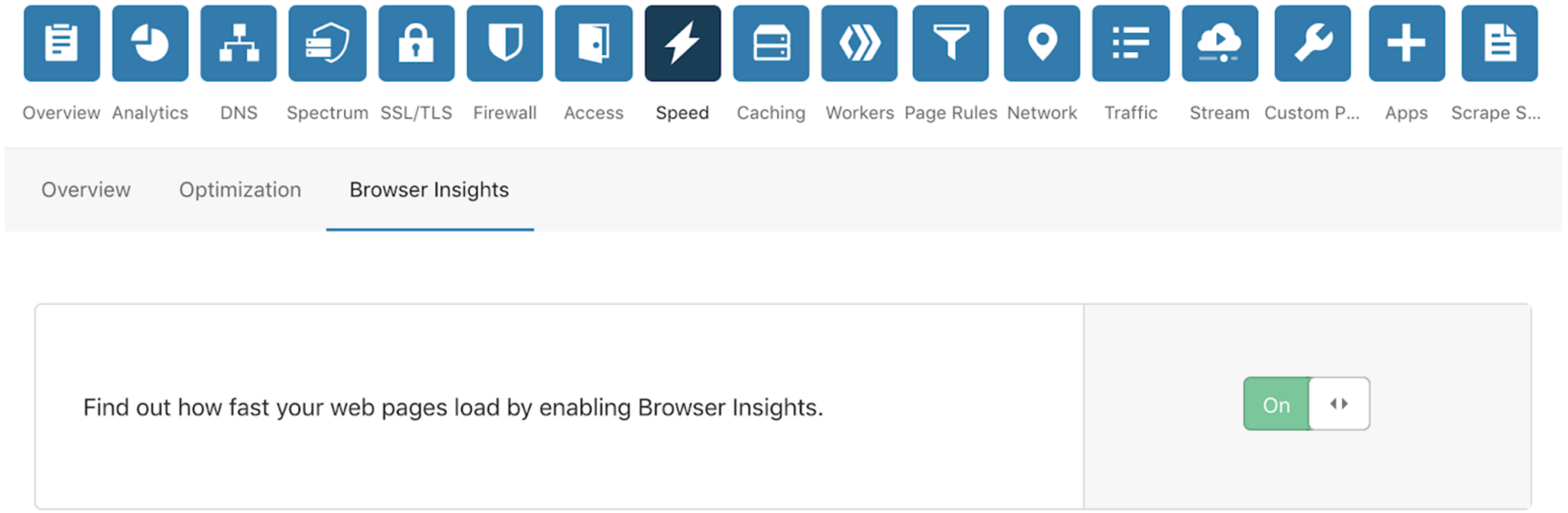
Origin IP	Primary Data Center	Failover Data Center
35.212.198.247	SEA	YVR
35.247.136.162	SIN	SIN

CLOUDFLARE

// Browser Insights + Analytics

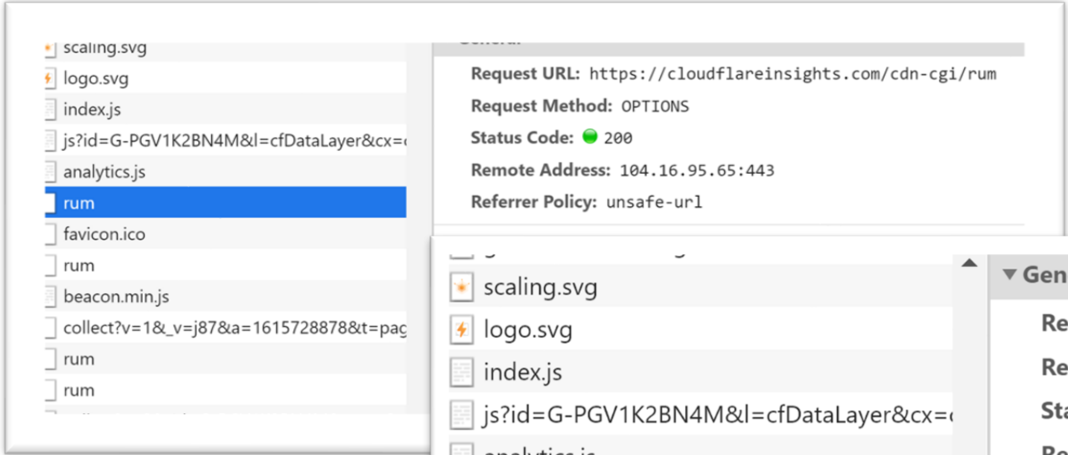
Browser Insights

Browser Insights lets you measure the TCP connection time, DNS response time, Time to First Byte (TTFB), page load time, and more from the perspective of your visitors all over the globe. When it's enabled, we add a small snippet of JavaScript code to each HTML page load that uses the standard Performance API to collect timing info. Then we can start showing you metrics about how your web pages are performing in the real world.



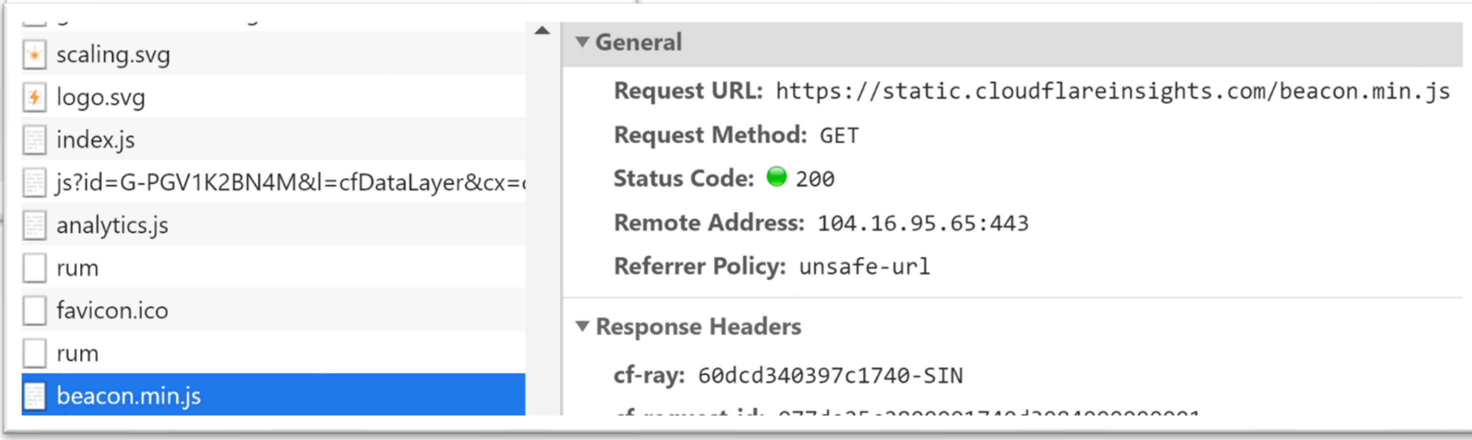
Find out how fast your web pages load by enabling Browser Insights.

How to check if Browser Insight is Active?



scaling.svg
logo.svg
index.js
js?id=G-PGV1K2BN4M&l=cfDataLayer&cx=
analytics.js
rum
favicon.ico
rum
beacon.min.js
collect?v=1&_v=j87&a=1615728878&t=pag
rum
rum

Request URL: https://cloudflareinsights.com/cdn-cgi/rum
Request Method: OPTIONS
Status Code: ● 200
Remote Address: 104.16.95.65:443
Referrer Policy: unsafe-url



scaling.svg
logo.svg
index.js
js?id=G-PGV1K2BN4M&l=cfDataLayer&cx=
analytics.js
rum
favicon.ico
rum
beacon.min.js

▼ General

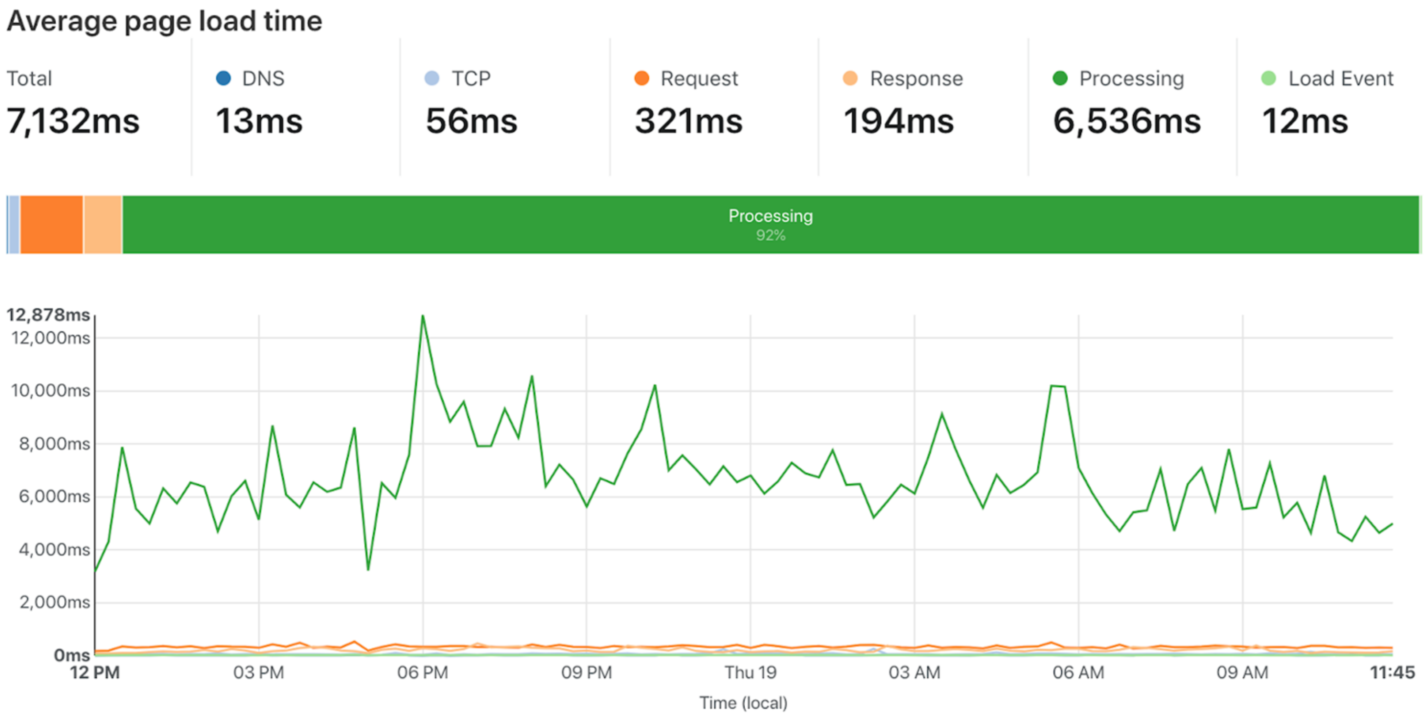
Request URL: https://static.cloudflareinsights.com/beacon.min.js
Request Method: GET
Status Code: ● 200
Remote Address: 104.16.95.65:443
Referrer Policy: unsafe-url

▼ Response Headers

cf-ray: 60dcd340397c1740-SIN

CLOUDFLARE

Lab - Browser Insights Review



Average page load time

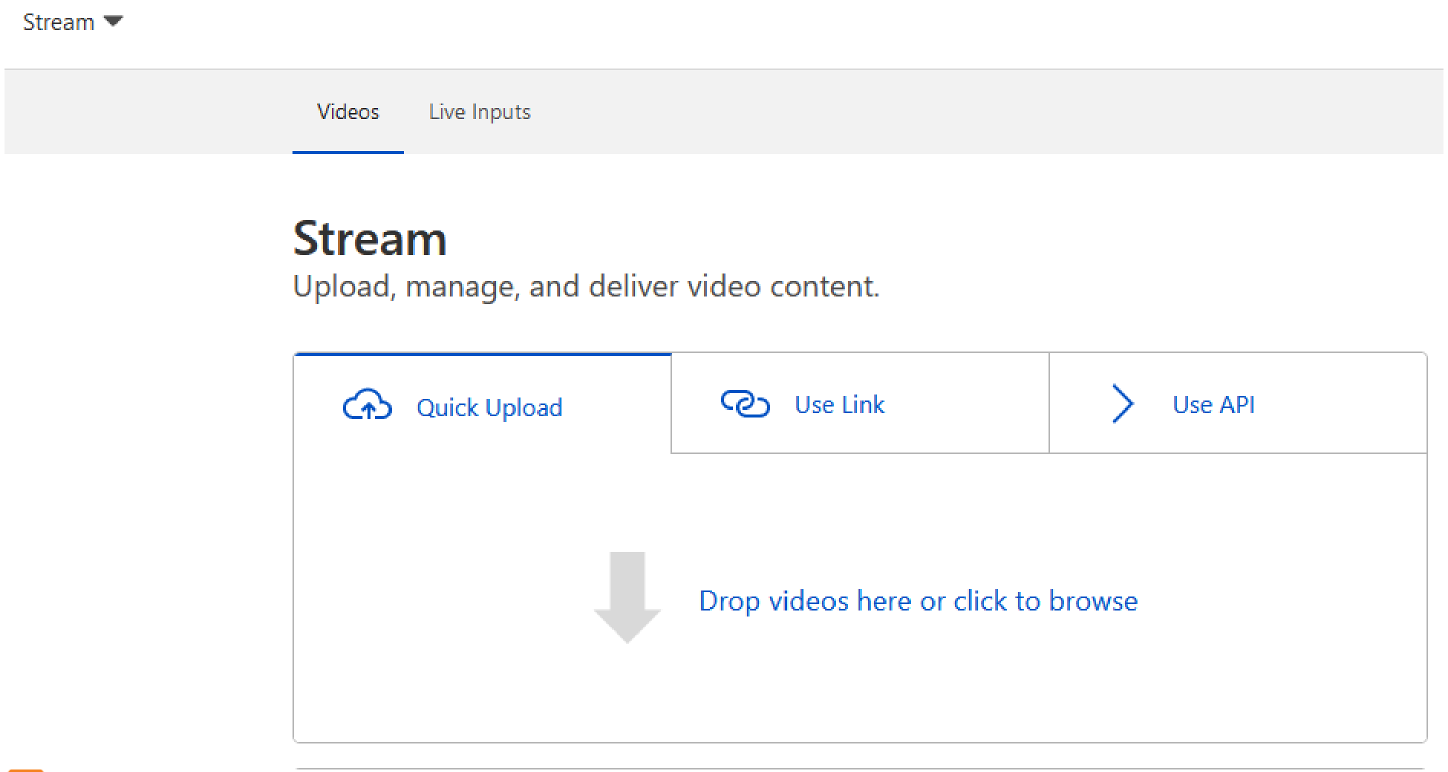
Total	● DNS	● TCP	● Request	● Response	● Processing	● Load Event
7,132ms	13ms	56ms	321ms	194ms	6,536ms	12ms

//Additional Performance Features

Cloudflare Stream and Stream Delivery

	Stream	**Stream Delivery**
Customers	ENT + Self Serve	ENT only
What role does Cloudflare play	End-to-end. All video tasks handled by Cloudflare. Uploading, processing, delivery and analytics.	Cloudflare only does delivery. CDN-only.
Customer Video expertise	Does not require video expertise. Cloudflare chooses correct settings and components required.	Customer does: • video storage • encoding, segmenting • manifest generation • packaging (generation of .mp4 or .ts files) • DRM • analytics • customer brings their own player
VOD or Live video?	VOD only (video-on-demand)	VOD and Live Streaming
Adaptive bitrate streaming	Adaptive bitrate streaming only. Customers don't have control over this.	Adaptive bitrate streaming recommended but not necessary. However, chunked files (cacheable files of size 1-10MB) form the video.

Stream Dashboard



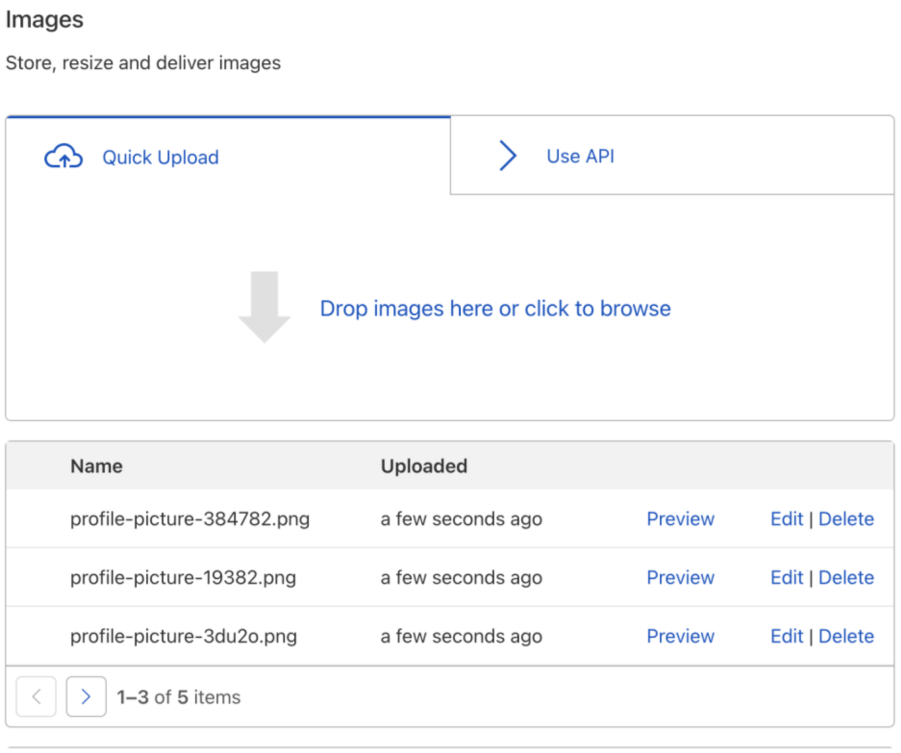
Stream

Upload, manage, and deliver video content.

2021 Cloudflare Speed Week

Monday	Tuesday	Wednesday	Thursday	Friday ~ Weekend
Cloudflare passes 250 cities, 100Tbps capacity	Argo 2.0: Faster Smart Routing	Cloudflare Images	Early Hints	Benchmarking: Cloudflare vs AWS, Akamai, Google, Fastly
Fast serverless platform: Workers	Orpheus: Improved origin connectivity	Web Analytics	Cloudflare One for speed - Zero Trust built for speed - Magic makes network faster	Project Turpentine
Cloudflare support for Vary	Improve SEO with Signed Exchange	Image Optimization Testing Tool	Cloudflare Backbone	Pages: Jamstack
	Live-updating analytics		NEL: Last mile insights	Workers Profiling
	Instant Logs			Cloudflare passes 10,000 connected networks

Cloudflare Images



Images

Store, resize and deliver images

Name	Uploaded		
profile-picture-384782.png	a few seconds ago	Preview	Edit | Delete
profile-picture-19382.png	a few seconds ago	Preview	Edit | Delete
profile-picture-3du2o.png	a few seconds ago	Preview	Edit | Delete

1–3 of 5 items

Cloudflare Images and Image Resizing

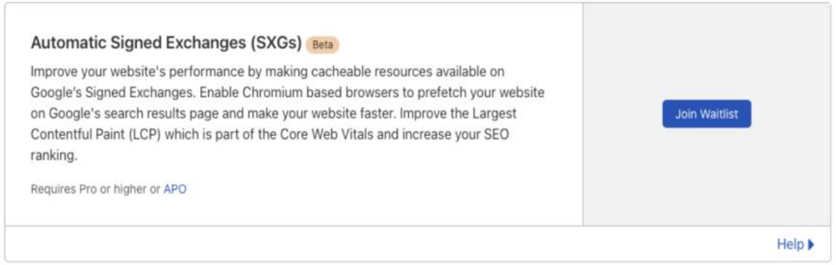
- Cloudflare Images is an end-to-end solution that offers storage, resizing, optimization, and delivery.
- Image Resizing only offers resizing and optimization.
- Reference:

https://images.cloudflare.com/

https://developers.cloudflare.com/images/

https://developers.cloudflare.com/image-resizing/

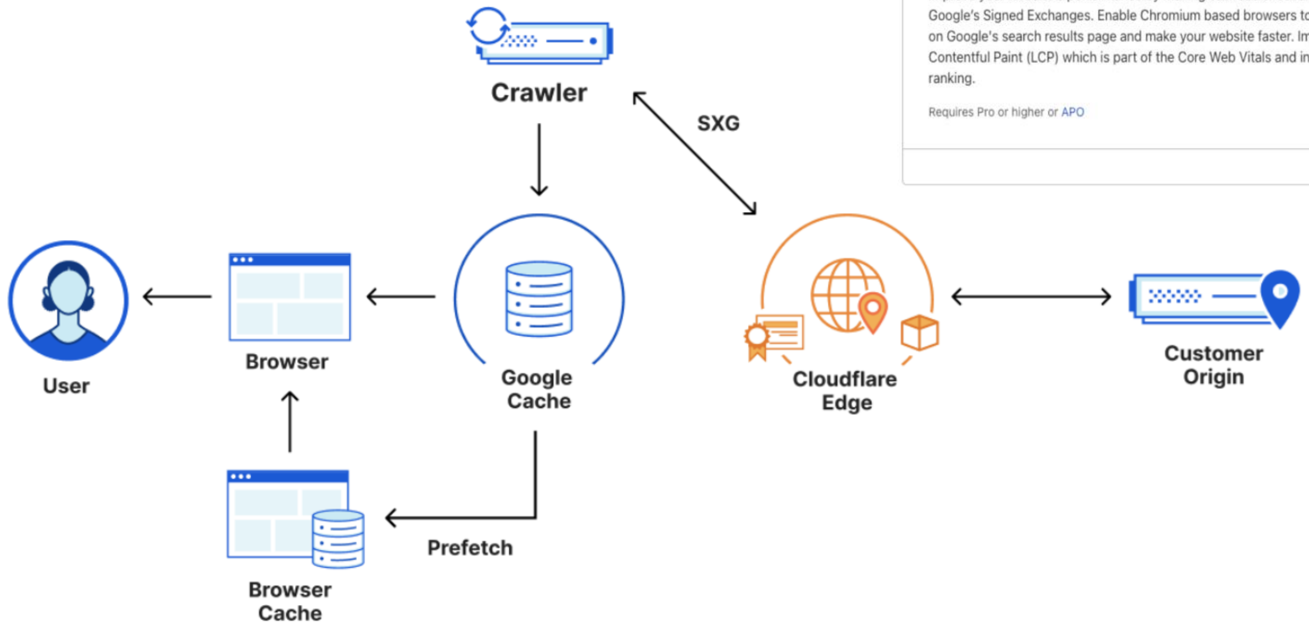
Automatic Signed Exchanges -> helps to improve SEO



Automatic Signed Exchanges (SXGs) Beta

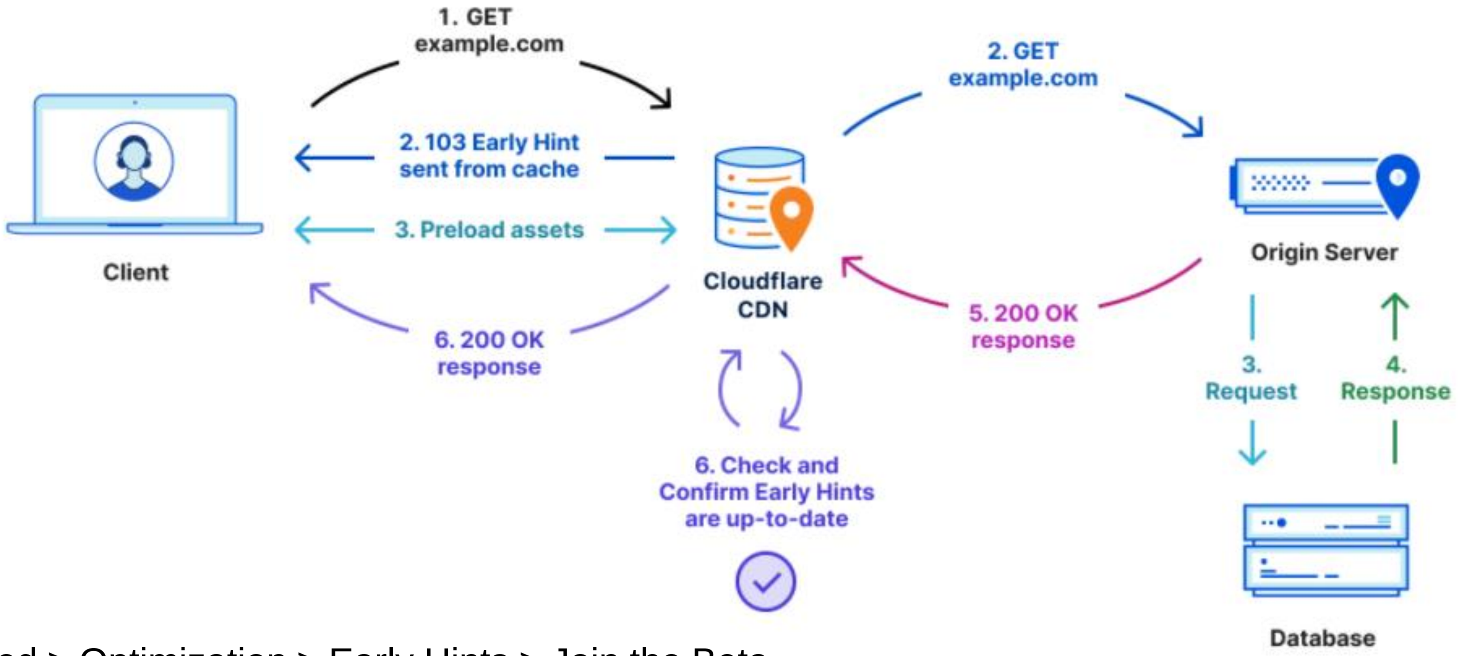
Improve your website's performance by making cacheable resources available on Google's Signed Exchanges. Enable Chromium based browsers to prefetch your website on Google's search results page and make your website faster. Improve the Largest Contentful Paint (LCP) which is part of the Core Web Vitals and increase your SEO ranking.

Requires Pro or higher or APO

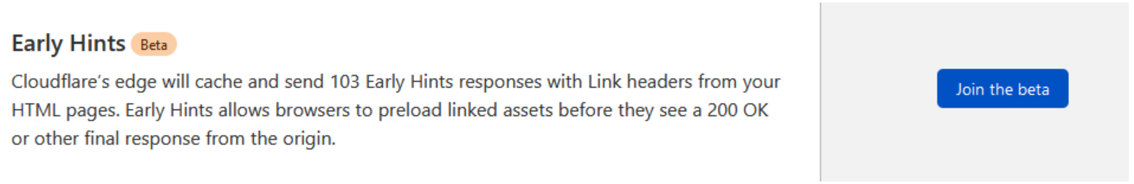


Go to Speed > Optimization > Automatic Signed Exchange > Join Waitlist

Early Hints (Smart Prefetch)

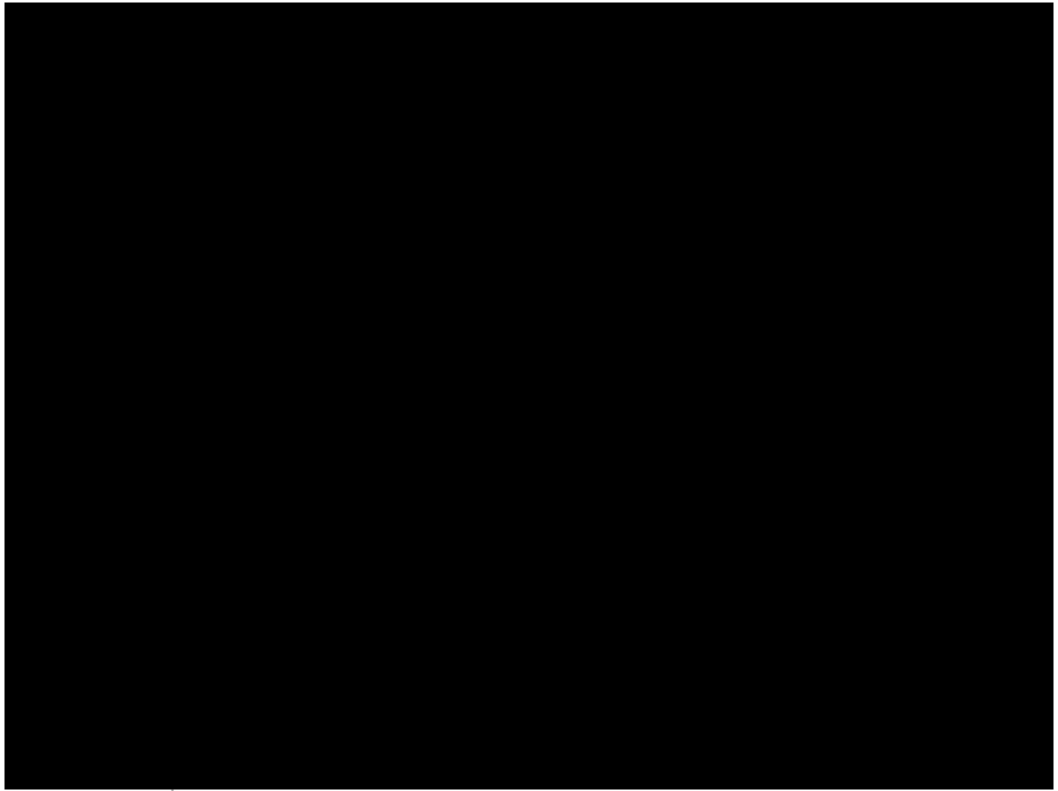


Go to Speed > Optimization > Early Hints > Join the Beta

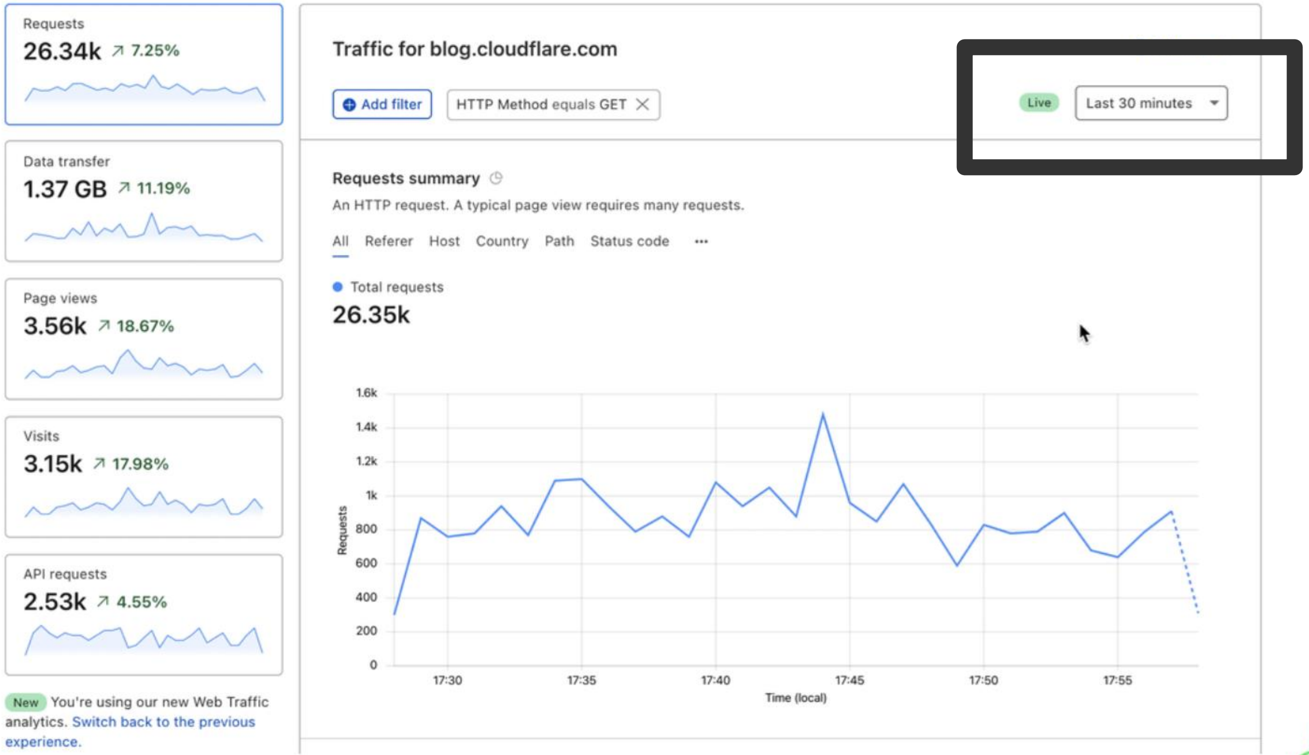

Early Hints Beta

Cloudflare's edge will cache and send 103 Early Hints responses with Link headers from your HTML pages. Early Hints allows browsers to preload linked assets before they see a 200 OK or other final response from the origin.

Real-time Analytics and Instant Logs



Real-time Analytics and Instant Logs



You're using our new Web Traffic analytics. Switch back to the previous experience.

Traffic for blog.cloudflare.com

HTTP Method equals GET

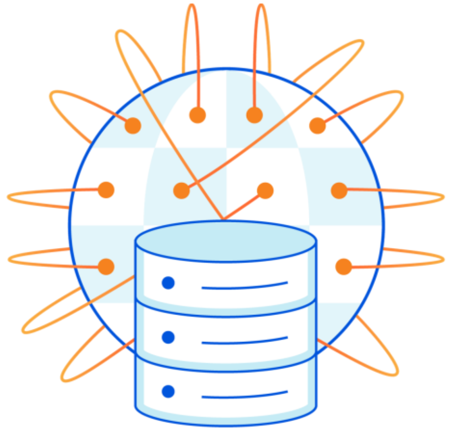
Last 30 minutes

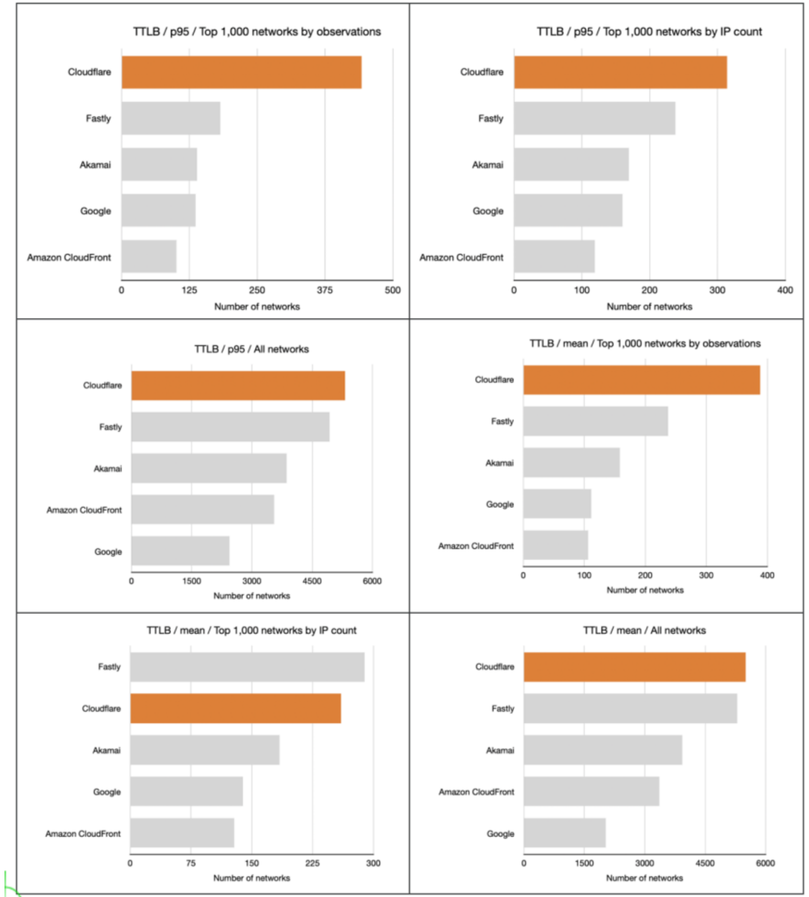
Requests summary

An HTTP request. A typical page view requires many requests.

● Total requests
26.35k

Benchmarking Edge Network Performance: Akamai, Cloudflare, Amazon CloudFront, Fastly, and Google





CLOUDFLARE | PARTNER NETWORK

These are what you learned today:

- Advanced Caching Behavior
- Cache Configuration and Tuning
- Understanding Cache Analytics Dashboard
- Argo Smart Routing and Tiered Caching
- Browser Insights and Analytics
- Cloudflare Image & Image Resizing
- Stream and Stream Delivery
- Updates on Cloudflare Speed Week 2021

End of Day 3 Summary

Further Reading

- https://support.cloudflare.com/hc/en-us/articles/218411427-Understanding-and-Configuring-Cloudflare-Page-Rules-Page-Rules-Tutorial-
- https://support.cloudflare.com/hc/en-us/articles/224509547-Recommended-Page-Rules-to-Consider
- https://support.cloudflare.com/hc/en-us/articles/360023040812-Best-Practice-Caching-Everything-While-Ignoring-Query-Strings
- https://support.cloudflare.com/hc/en-us/articles/229373388-Understand-Cache-by-Device-Type-Enterprise-plans-only-
- https://support.cloudflare.com/hc/en-us/articles/206190798-Using-Resolve-Override-in-Page-Rules
- https://support.cloudflare.com/hc/en-us/articles/206652947-Using-Page-Rules-to-Re-Write-Host-Headers