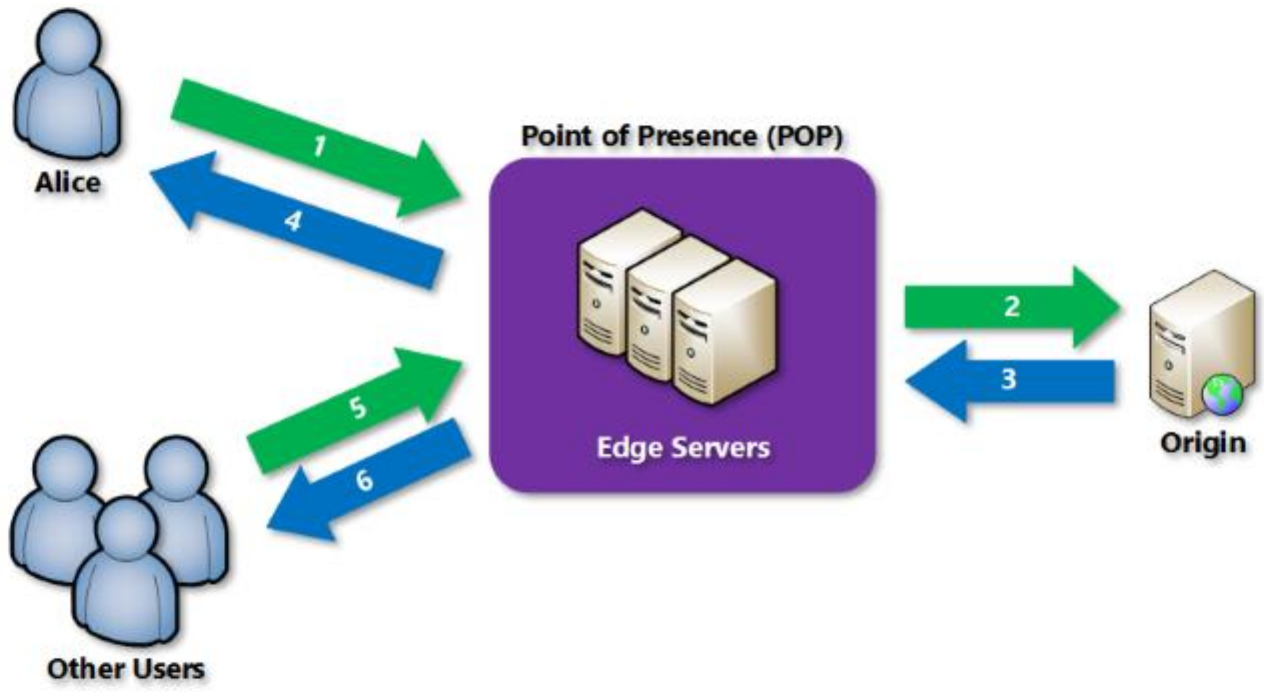
Azure Content Delivery Network (CDN)

The Azure Content Delivery Network (CDN) caches static web content at strategically placed locations to provide maximum throughput for delivering content to users. The CDN offers developers a global solution for delivering high-bandwidth content by caching the content at physical nodes across the world.

The benefits of using the CDN to cache web site assets include:

- Better performance and user experience for end users, especially when using applications where multiple round-trips are required to load content.
- Large scaling to better handle instantaneous high load, like at the start of a product launch event.
- By distributing user requests and serving content from edge servers, less traffic is sent to the origin.

How it works



1. A user (Alice) requests a file (also called an asset) using a URL with a special domain name, such as `<endpointname>.azureedge.net`. DNS routes the request to the best performing

Point-of-Presence (POP) location. Usually this is the POP that is geographically closest to the user.

2. If the edge servers in the POP do not have the file in their cache, the edge server requests the file from the origin. The origin can be an Azure Web App, Azure Cloud Service, Azure Storage account, or any publicly accessible web server.
3. The origin returns the file to the edge server, including optional HTTP headers describing the file's Time-to-Live (TTL).
4. The edge server caches the file and returns the file to the original requestor (Alice). The file remains cached on the edge server until the TTL expires. If the origin didn't specify a TTL, the default TTL is seven days.
5. Additional users may then request the same file using that same URL, and may also be directed to that same POP.
6. If the TTL for the file hasn't expired, the edge server returns the file from the cache. This results in a faster, more responsive user experience.

****

Content Delivery Networks (CDNs) accelerate web traffic across the internet through servers residing in strategic locations (known as points of presence or PoPs) across the globe. Each PoP has a number of caching servers, each of which contains a cached version of your website or application. By bringing content closer to each visitor, CDNs improve performance and reduce load on the origin server.

Caching is the *raison d’etre* for CDNs.

The reason for this is a CDN’s effectiveness can be measured by the cache hit ratio, which is the percentage of requests the CDN was able to serve out of its own cache. The higher the caching ratio, the better your website or application performance.

CDNs typically cache static files, such as images, JavaScript files and CSS. This type of content rarely changes and can be heavily cached.

Keeping your content fresh

While response and fast load times are crucial for users, the freshness (or accuracy) of your content is no less important for your business reputation.

Caches are refreshed based on caching policies each website defines which determine the amount of time a given resource remains in the cache. When that duration expires, the content must be retrieved from the origin server. If the content in the origin server has changed, the new version will be uploaded to the cache.

In certain situations, however, you may not want to wait for the caching period to expire. An example of this would be an update on your website's pricing page or a bug on an online game. That's where cache purging comes in. Purging refers to the active removal of a resource from the cache without waiting for the predetermined cache expiry time. As soon as a user requests the purged resource, the CDN will cache a copy of the updated content from the origin server.

When to use rapid cache purge

If you're the owner of a blog or an online academic journal whose content doesn't change once it's uploaded to the site, you most likely can do without purging since your content is not time-sensitive.

Other types of websites, such as news and sports sites with live updates, or e-commerce sites with frequent pricing changes, would benefit from caching policies that reflect the almost continuous information updates. This ensures that content is always fresh without requiring frequent cache purges.

However, there are situations in which rapid purging can be critical for your business and your users. Consider, for example, the following "crisis" scenarios:

- You run an e-commerce site specializing in jewelry and you've run out of inventory for a popular ring. You need to temporarily remove the item from your online catalog until your supplier delivers new stock. Waiting for the cache to refresh is going to result in disgruntled customers and an increase in customer service calls.
- You've uploaded new content to your website around an important product launch and discovered a bug in one of the pages. A major PR campaign is underway driving thousands of daily visitors to your site and you can't afford a mishap. In such a case, the content must be corrected and replaced immediately with the updated version. Failure to do so could result in substantial reputation loss.
- You've just updated or patched your online game and want to be sure that the latest version is available to your users. If the old version contained a critical error, your users are likely to go elsewhere for their gaming entertainment.

In addition to the above, there are applications and websites with cacheable content where certain types of business activities will trigger the need for an immediate purge. For example:

- The removal of an account from a photo sharing application should result in the deletion of all photos related to that account. As soon the photos are deleted from the application database, you need to be sure that there are no traces of those photos on caching servers located around the world.
- You own a blog and decide to delete an outdated blog post from your online platform. Similar to the example above, you don’t want copies of the deleted post to be accessed from your CDN’s caching servers.

If you need to respond quickly to business triggers or handle an emergency fix, and your changes need to be propagated across the CDN immediately, rapid purging is the answer.

How fast is a cache purge?

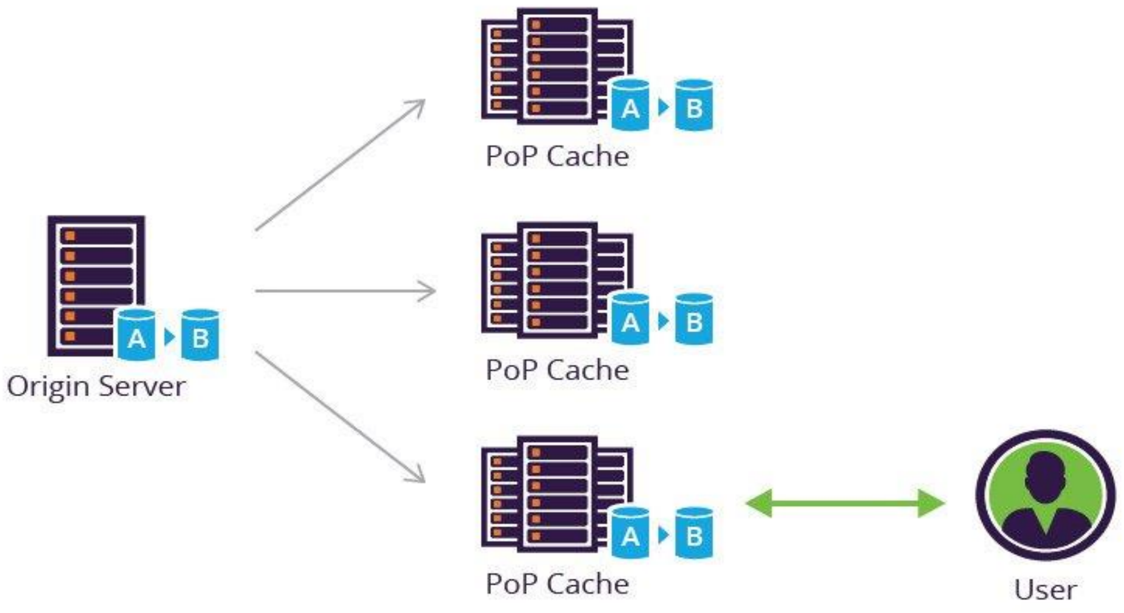
Once you’ve decided to purge the cache, the time it takes to execute depends on your CDN. The effectiveness of a purge request is measured in the time it takes for it to propagate worldwide through the entire network. The speed of the purge depends on several factors including hardware type, network size and the number of CDN customers requesting cache purges at the same time.

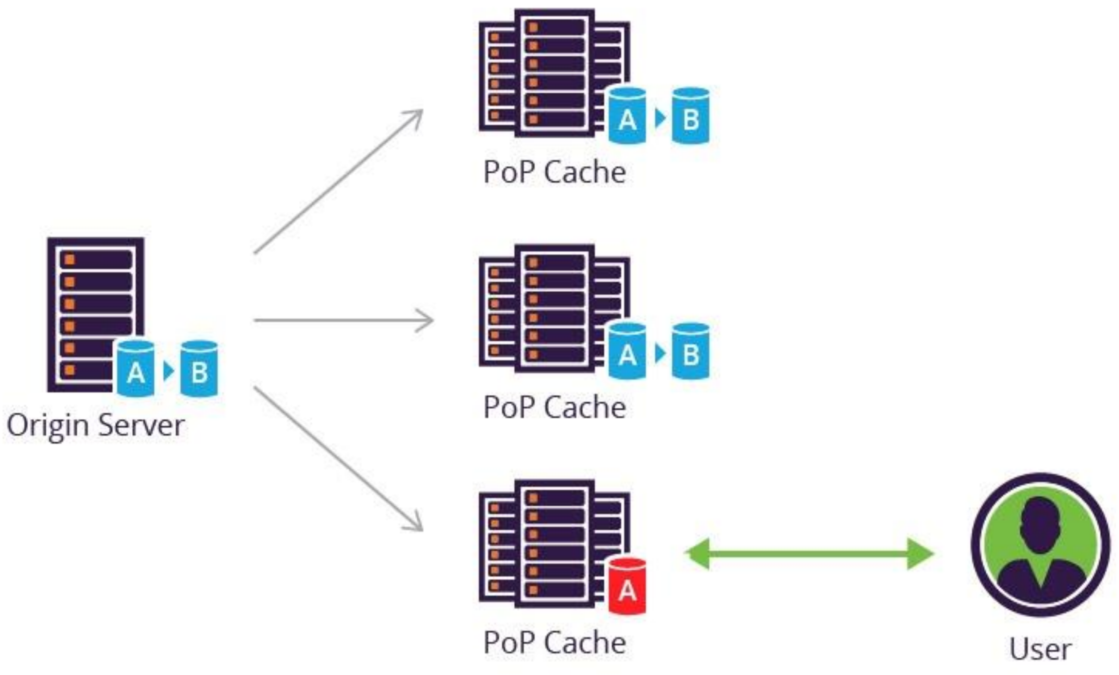
While some CDNs, like Incapsula, can achieve purge times of less than 500 milliseconds, CDNs with larger networks and outdated hardware may take longer to propagate the request and delete the relevant content from all the caching servers. Most CDNs let you purge a specific resource on demand, but some only give you the option of purging the entire site cache.

The goal is real-time syncing

Most web developers and website owners want the ability to control their cached content in a manner similar to how they manage their database. This means that the CDN caching mechanism should be easy to use and allows instant propagation of changes or fast purging.

Fast purging means near real-time syncing between the origin server (the application database) and the CDN. It’s designed to minimize the delay between updates to the application resources, such as database and files, and updates of the cached resource, allowing website owners to be in full control of what their visitors see. The first graphic below shows data on all PoPs are instantly synced . The second graphic shows the PoP serving traffic to the user is not synced in time resulting in a data integrity issue where the user can only view old cached content.





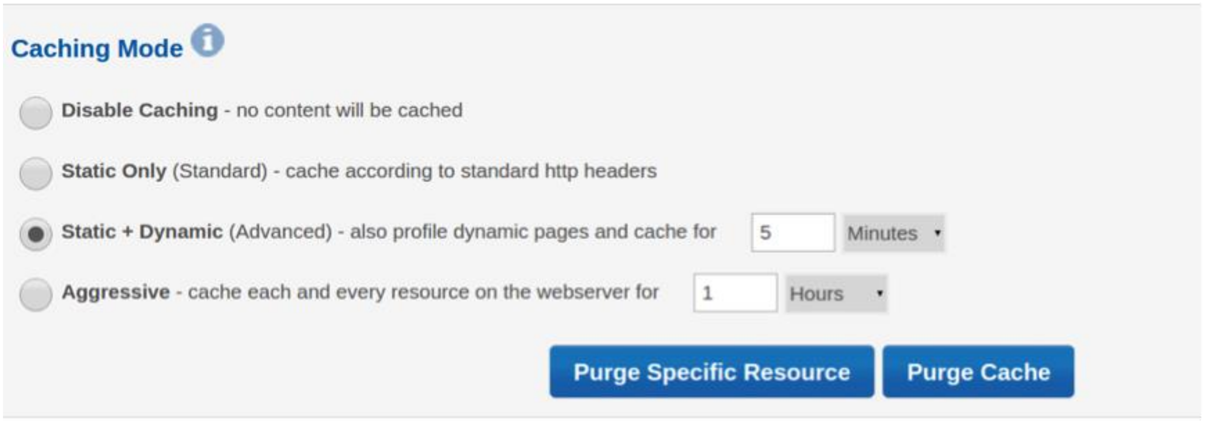
How to implement fast cache purge on Incapsula

In our experience we’ve seen that purging is a last resort that should be reserved for exceptional situations. The best way to avoid cache purging is to implement efficient caching policies that reflect the behavior of your website or application. Incapsula provides intelligent caching functionality that identifies resources that change more frequently and automatically sets the caching rules accordingly. This gives customers an optimal default refresh policy, so content always stays fresh. These settings can be manually adjusted as needed.

That said, the Incapsula CDN is built for rapid purging, comprising a relatively small number of high-capacity PoPs that allow for instant propagation. Incapsula users can clear the cache as needed using the following methods:

1. **Manually via the GUI**

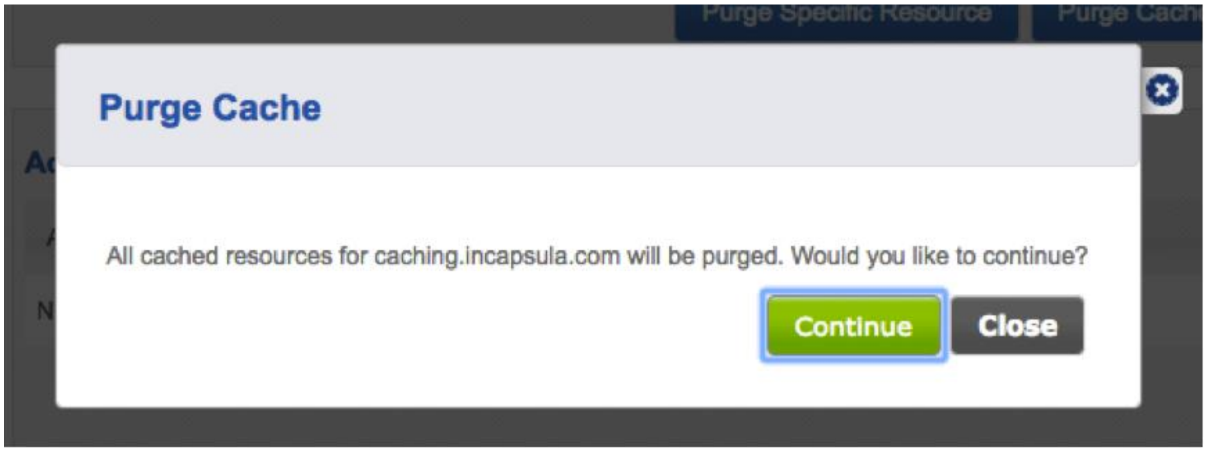
The Incapsula GUI lets you purge the entire cache or a specific resource.



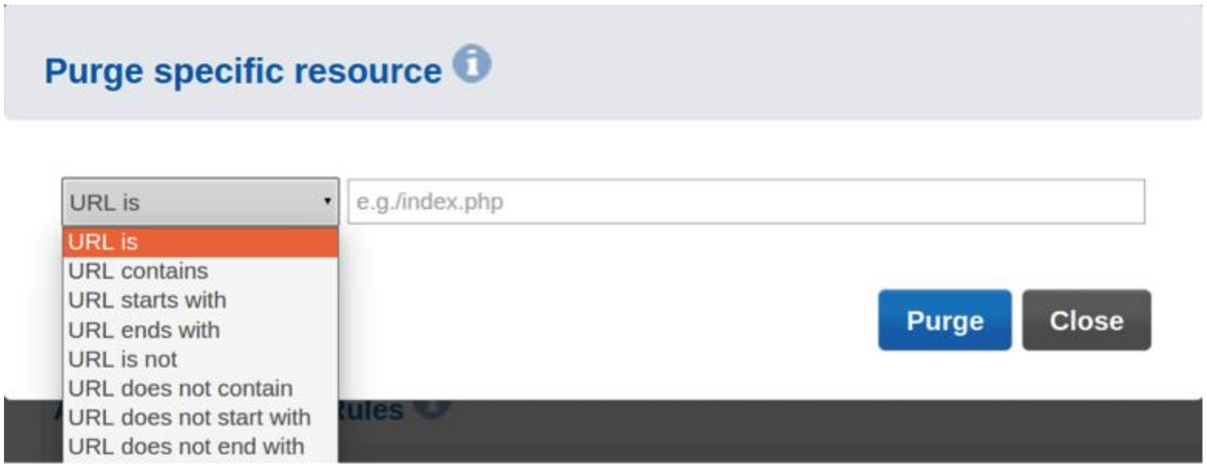
Caching Mode

- **Disable Caching** - no content will be cached
- **Static Only** (Standard) - cache according to standard http headers
- [✓ **Static + Dynamic** (Advanced) - also profile dynamic pages and cache for 5 Minutes]
- **Aggressive** - cache each and every resource on the webserver for 1 Hours

After a major change to your website, such as a version update, you may want to clear all resources in the cache immediately, without waiting for the caching period to expire. Just click “Purge Cache” and “Continue” and you’re done.


Purge Specific Resource
Purge Cache
Purge Cache
All cached resources for caching.incapsula.com will be purged. Would you like to continue?
Continue
Close

Clicking “Purge specific resource” will display the following screen:



Select one of the URL rule options to specify a particular URL (URL is) or a string to be matched (URL is, URL contains, URL starts with or URL ends with). Then click the **Purge** button.

2. **Automated cache purge option using the Incapsula API**

Using the GUI is simple but it can take a couple of minutes to login and execute the purge. It also needs to be repeated each time you update a specific piece of content.

The Incapsula API automates the purge cache action. Here the customer must define in advance the types of updates that will trigger an automatic purge like an update to an HTML page. A piece of code is added to the application that calls the API in the pre-defined conditions. The purge is propagated in less than half a second. The API call can be used to purge the entire cache for a site or a specific resource using the resource name as a parameter.

Purging the Cache

This rule purges all cached content on the Incapsula proxy servers for a specific site. For example, you may want to purge the cache after making adjustments in your site. To purge the entire cached content for the site, just use the API call with no parameters. If you want to purge a specific resource, add the resource name in the purge_pattern parameter.

This script adds this rule with these other parameters of the rule, as defined in the Site Management API.

- site_id
- purge_pattern — The pattern of the resource to be purged from the cache

```
#!/bin/sh

curl -X POST "https://my.incapsula.com/api/prov/v1/sites/cache/purge?api_id=12593&api_key=082ced47-cbf4-4488-a3d3-b852ab899437&site_id=78925715"
```

Sample response to the command, when successful, is

```
{"res":0,"res_message":"OK","debug_info":{"id-info":"9123"}}
```

The API offers a more advanced option because it requires some software development on the application side. The decision to use a manual approach or API will depend on your application's sensitivity to data integrity, as well as the volume and frequency of changes. If it's low, use the GUI; if it's high, use the API.

Conclusion

CDN caching gives websites and web applications an effective way to reduce bandwidth costs, improve user experience and ensure reliable content delivery. At the same time, your visitors also expect continuous access to fresh content.

Fast purging allows real-time syncing between the state of the application and the content being served to users from the CDN. It's particularly useful for reacting quickly to business triggers and crisis situations where you can't afford to wait for the cache to refresh automatically. For this reason, whether you're using a manual or automated purge mechanism, speed is of the essence and propagation times across your global CDN are critical for staying on the same page as your users.

Look for a future post on the underlying technology challenges of cache purge and how to handle it.

https://www.incapsula.com/blog/purge-cache-keeps-content-fresh-responsive.html

****

The Differences Between Push And Pull CDNs

An increasingly popular way to make travel blogs more efficient is through the use of content delivery networks (CDNs). They are an inexpensive and efficient way to improve loading time on several fronts but can be confusing to set up initially. There are two primary types of CDN – "pull" CDNs and "push" CDNs – which have different benefits and costs so today we'll take a look at the which one might be best for you.

What Is A CDN?

A CDN helps to speed up static components of your blog (especially pictures of say, your bed in Rome) by distributing them across a number of servers around the world. This does two things – one is to put much of your travel blog's content closer to the person who wants to view it. So, a person in Japan will download your beautiful photos of Granada from Tokyo, rather than your server sitting in Boston, for example. Less distance traveled means a faster download and secondly, if the content isn't being pulled directly from your server, it saves overall computing power (important to have plenty of for when traffic gets really busy).
You can set up a CDN on your own server (in effect creating two download points for your photos) but most people opt for paid services like Amazon's Cloudfront. (Next time I'll talk about Cloudflare, which is a free alternative, although not a CDN by design.) With paid CDNs, you are generally charged by the amount of data in gigabytes that are downloaded from your site each month. (Here's Amazon's pricing chart to give you an idea.) Most travel blogs that aren't *very* video intensive don't have the kind of traffic or the content (e.g. podcasts) to make CDN usage very expensive.
Many hosting providers like Media Temple are also now including CDNs as extra features to their hosting packages though they tend to be much more expensive that 3rd-party alternatives.

What Is A Pull CDN Versus A Push CDN?

A very watered down explanation of the two is with a pull CDN, the CDN caches parts of your site upon request to their servers. A push CDN is where you upload your entire travel blog to the CDN so it's ready for users at any given time. Let's look a bit further.

- **Pull CDN:** Imagine a person loading your latest travel blog post. It probably has pictures in it, as does your site's theme (e.g. icons, background images, etc.) For this example let's have your hosting server be in Boston. You've just published your latest travel blog post and your biggest fan in Japan wants to read it. With a pull CDN, the very first time she does, the content isn't on the CDN. During this first request, the CDN "pulls" the images and so forth to CDN server nearest your Japanese fan. That could be Tokyo or Hong Kong, whichever it is, the very first time the CDN has to pull the post, meaning your server and reader won't see any gain in speed. The second time however (and usually for 1-30 days later) the CDN has the content loaded and it's will be available to everyone who is closest to that Tokyo or Hong Kong CDN server.

- **Push CDN:** Going along with the example above, instead of waiting around for the CDN to pull the content when it's needed, you simply upload the entire content of your travel blog to the CDN beforehand. That way your pictures, theme files, videos, and the rest are always on the CDN servers around the world.

Keep in mind that this isn't the entire story or process of how CDNs work, it's a look at the system from very high up in the sky. That said, it may seem like a push CDN is superior to a pull, but that's not always the case.

The Benefits of Pulling or Pushing Your CDN

In general, a pull CDN is much easier to configure than a push CDN. Once initially configured, a pull CDN rather seamlessly stores and updates content on its servers as its requested. The data usually stays there for 24 hours or longer if the CDN doesn't detect that a file has been modified. For low traffic sites or those that are sufficiently optimized with caching, good code, and more, a pull CDN provides speed without asking much of your server. Once your content is pulled (give it 48 hours to get enough data to make it a noticeable difference) the maintenance required is low.

- So, what makes a pull CDN so easy can also be a pain, especially when you're making changes to your travel blog. Typically, you don't have control over how long the pull CDN cache lasts, so if you update a photo or theme, it might take up to 24 hours for all your readers (and you) to see it. You lose control for ease so when it comes to making widespread changes like updating your theme, you often must shut off the CDN during the process.

Conversely, a push CDN can put added strain on your server if it's underpowered for your traffic, or you have lots of changing content in a given day. The reason being, pushing all of your data and any changes as they happen to the CDN takes work on your server's part. If your server is already struggling under heavy load (here are a few tips to optimize your site) or has new content several times a day, all of them syncing between your server and the CDN might do more harm than good.

Which One Is Best For You?

The decision on which CDN type to go with revolves in large part around traffic and downloads. Travel blogs that are hosting videos and podcasts (aka. large downloads) will find a push CDN cheaper and more efficient in the long run since the CDN won't re-download content until you actively push it to the CDN. A pull CDN can help high-traffic-small-download sites by keeping the most popular content on CDN servers. Subsequent updates (or “pulls”) for content aren’t frequent enough to drive up costs past that of a push CDN.

Whichever CDN type you end up choosing, I would strongly recommend you keeping track of your usage over the first 1-3 months as well as loading times. You may find that your specific travel blog is better configured for a push or a pull with a little experimentation. Experimentation that should take place on the weekend or other dead times because either way you go, DNS changes are going to make the first 48 hours messy.

http://www.travelblogadvice.com/technical/the-differences-between-push-and-pull-cdns/

==

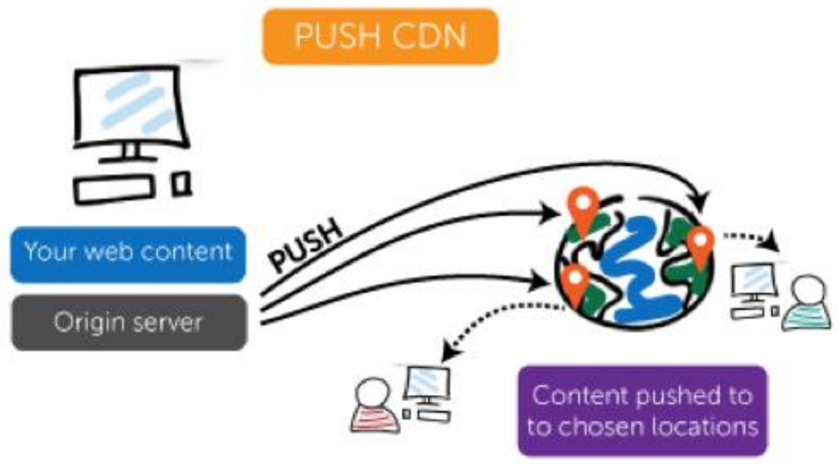
Push CDNs

Generally speaking, a push CDN works very much like secondary server. The user uploads content directly to the CDN (automatically or manually) and links to it.

Some push CDNs support FTP. Others, like Amazon Cloudfront, support REST, SOAP and other protocols.

The idea is that the user, or the primary server, takes responsibility for providing content to the CDN, *pushing* it to the network. This is flexible: users can specify the content that is uploaded, when it expires and when is updated.

This method is also the most efficient in terms of how the traffic is used. Content is uploaded only when it is new or changed, thus keeping traffic to a minimum.



Origin Pull CDNs

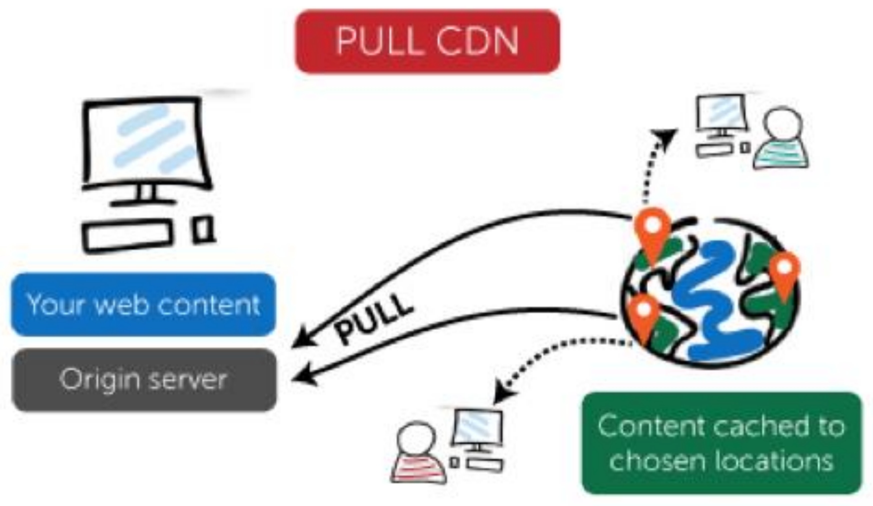
An origin pull CDN works very differently from a push CDN. Instead of the user taking responsibility for putting files onto the CDN, the CDN does it for them.

With a pull CDN, the site owner leaves the content on their server and rewrites their URLs to point to the CDN. When asked for a specific file, the CDN will first go to the original server, *pull* the file and serve it.

The CDN will then cache that file until it expires.

Pull CDNs are easy to set up, and this method also minimizes storage space. However, it’s less flexible, can create redundant traffic as files are re-queried before they have been changed.

This method of content delivery can also be slower. People who are trying to access the file for the first time, or accessing it after it has expired, may notice a small difference in speed. Setting the expiration correctly can help to minimize this problem, but it can involve trial and error.



Choosing the Correct CDN

For most sites, setting up a CDN can be automated. WordPress plugins like W3 Total Cache make the process painless. But the two different CDN types suit different kinds of sites.

- Sites that receive a great deal of traffic work best with pull CDNs. Content remains relatively stable and the traffic is spread out fairly evenly, so it makes sense to limit unneeded content pulls by allowing the webmaster to set a higher expiration. This ensures that the vast majority of visitors get served content cached on the CDN.
- Sites with minimal amounts of traffic will fare better with the push system; the content is put onto the CDN once and left up, rather than re-pulled at regular intervals.

If this is confusing think about a podcast: most of the episodes are older and rarely accessed, but also never updated. The podcast is likely best served over a push CDN.

However, a high-traffic image hosting site would be best with a pull CDN because the traffic will be evenly spread.

https://www.whoishostingthis.com/blog/2010/06/30/cdns-push-vs-pull/

https://cdn.net/push-vs-pull-cdn/