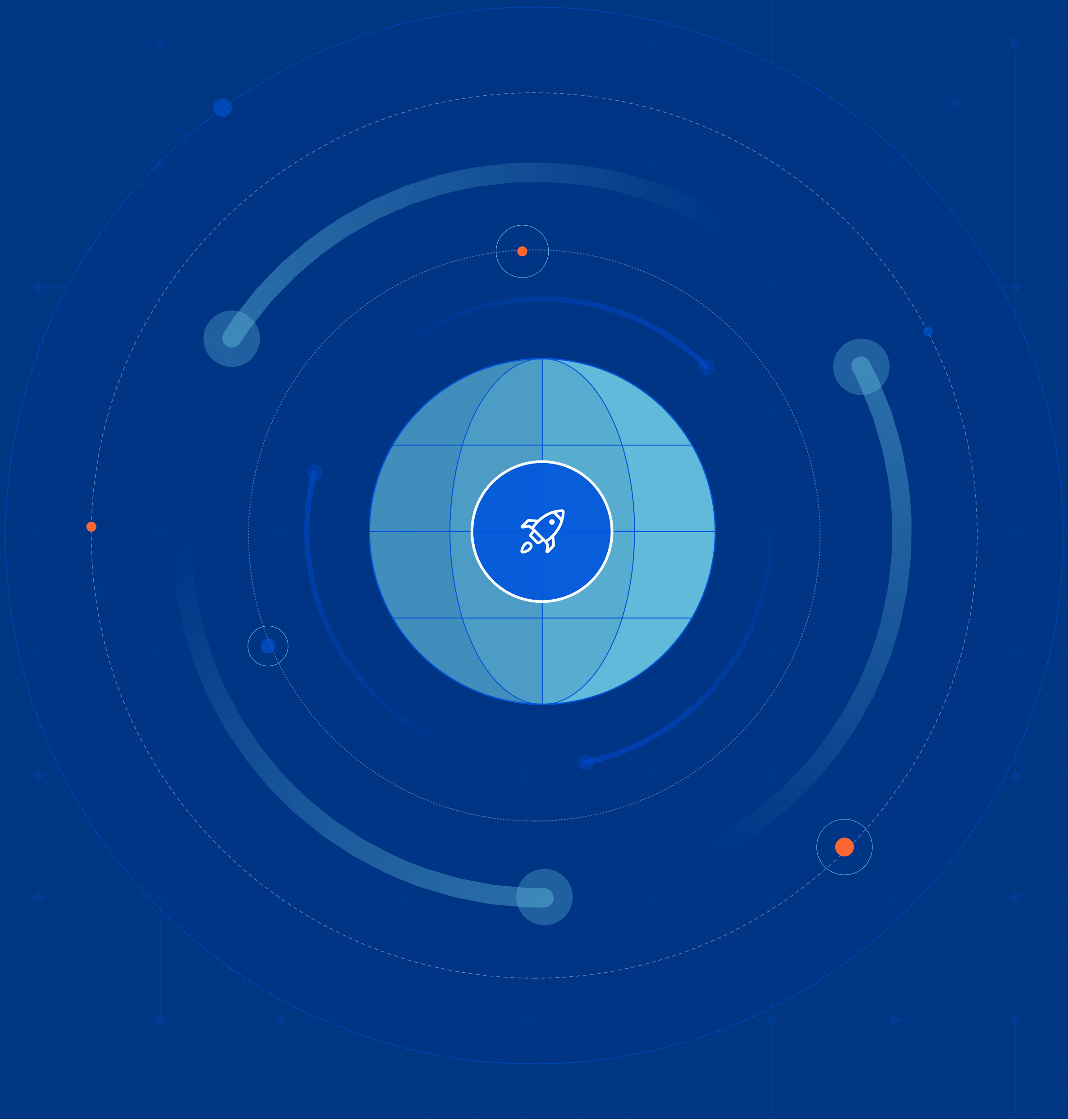
WHITEPAPER

Cloudflare CDN Reference Architecture



1 888 99 FLARE | enterprise@cloudflare.com | www.cloudflare.com

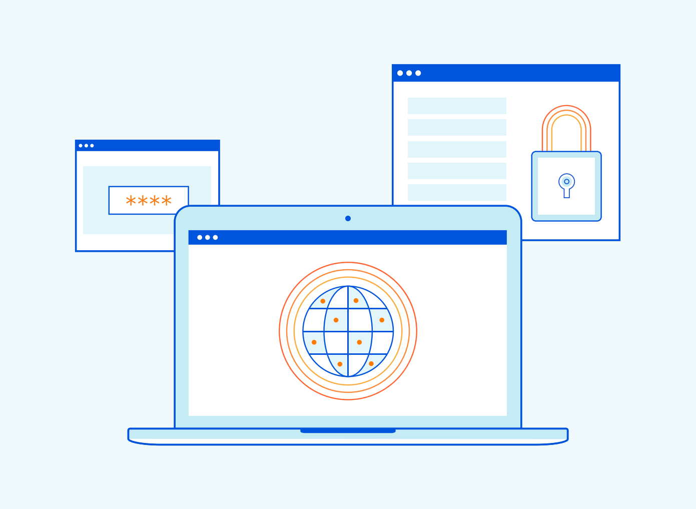
INDEX

Click to skip to each section

Overview

Every day, users of the Internet enjoy the benefits of performance and reliability provided by content delivery networks (CDNs). CDNs have become a must-have to combat latency and a requirement for any major company delivering content to users on the Internet. While providing performance and reliability for customers, CDNs also enable companies to further secure their applications and cut costs. This document discusses the traditional challenges customers face with web applications, how the Cloudflare CDN resolves these challenges, and CDN architecture and design.

Traditional challenges deploying web applications



Over the last several years, especially with the advent of the COVID-19 pandemic and the focus on remote work, there has been a significant growth in Internet traffic, further growing the need to efficiently manage network traffic, cut latency, and increase performance.

Companies running their applications in the cloud or on-premise are faced with the challenges of:

1. Implementing solutions to increase performance
2. As demand grows, scaling out their architecture to meet availability and redundancy concerns
3. Securing their environments and applications from growing Internet threats
4. Reining in growing costs related to doing all of the above

With companies serving customers across the globe, the above challenges require a significant undertaking. Traditionally, a website/application is deployed centrally and replicated to another region for availability, or the website/application is deployed across a handful of servers, sometimes across multiple data centers for resiliency.

The servers hosting the websites are called origin servers. When clients access a website, they make a request for resources from the server. Navigating to one website can generate hundreds of requests from the browser for HTML, CSS, images, videos, etc. With versions of HTTP prior to HTTP/2, each of these HTTP requests would also require a new TCP connection.

Enhancements in HTTP/2 allow for multiplexing multiple requests to the same server over a single TCP connection, thus saving server resources. However, compute and network resources are still consumed as servers respond to these requests. As more clients access the website, the following can result:

- The origin server starts to become overloaded with requests, impacting availability; companies start looking at scaling out to handle the additional load
- As each request has to make its way to the origin server, performance and user experience is impacted due to latency
- The latency for end users becomes proportional to the distance between the client and origin server, thus resulting in varying experiences based on client location
- As origin servers respond to the increasing requests, bandwidth, egress, and compute costs increase drastically
- Even as customers scale out to handle the increased demand in traffic, they are left exposed to both infrastructure-level and application-level distributed denial-of-service (DDoS) attacks

Traditional challenges deploying web applications (continued)

In Figure 1 below, there is no CDN present and there is an origin server sitting in the US. As clients access the website, the first step is DNS resolution, typically done by the user's ISP. The next step is the HTTP request sent directly to the origin server. The user experience will vary depending on their location. For example, you can see the latency is much lower for users in the US, where the origin server is located. For users outside the US, the latency increases, thus resulting in a higher round-trip time (RTT).

As more clients make requests to the origin server, the load on the network and server increases, resulting in higher latency and higher costs for resource and bandwidth use.

From a security perspective, the origin server is also vulnerable to DDoS attacks at both the infrastructure and application layer. A DDoS attack could be initiated from a botnet sending millions of requests to the origin server, consuming resources and preventing it from serving legitimate clients.

Further, in terms of resiliency, if the origin server temporarily goes offline, all content is inaccessible to users.

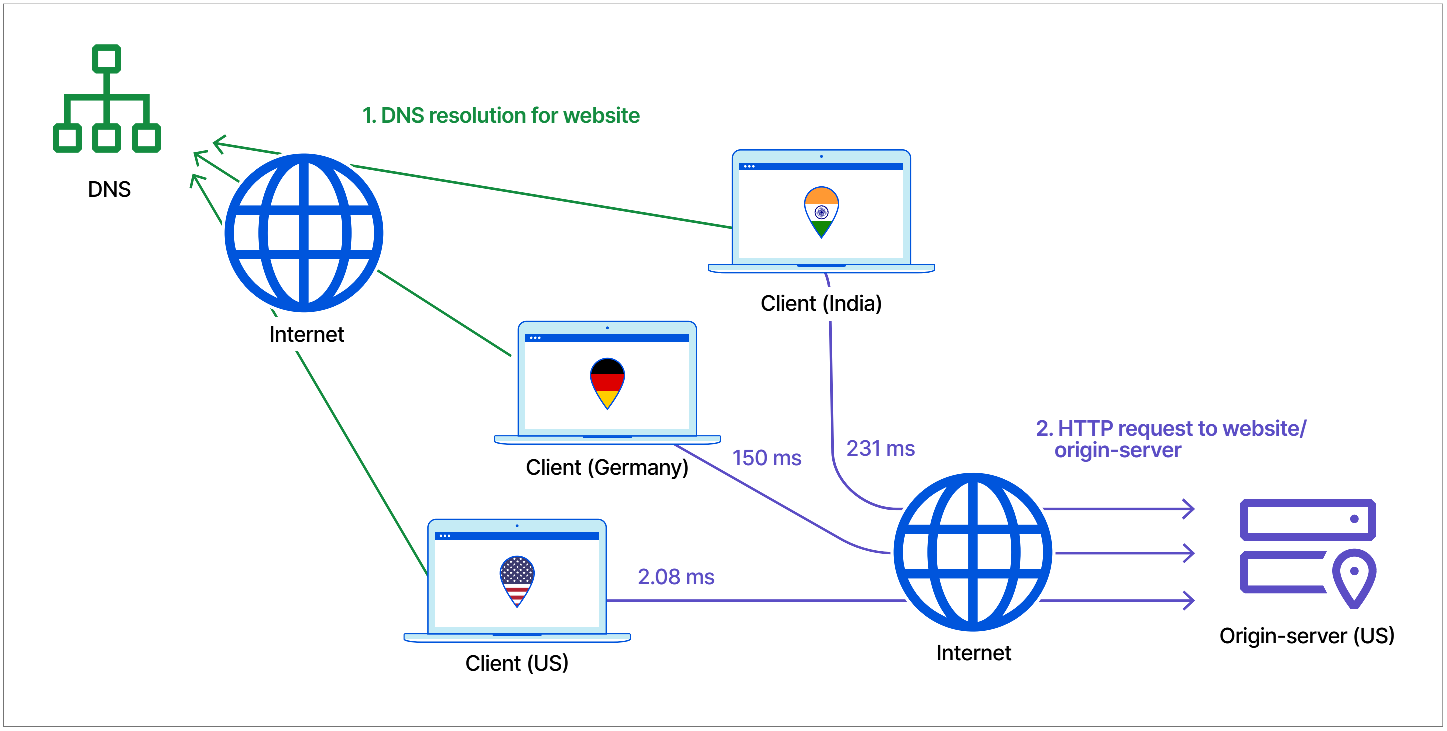


Figure 1: HTTP Request with no CDN

How a CDN tackles web application challenges

A CDN helps address the challenges customers face around latency, performance, availability, redundancy, security, and costs. A CDN's core goal is to decrease latency and increase performance for websites and applications by caching content as close as possible to end users or those accessing the content.

CDNs decrease latency and increase performance by having many data center locations across the globe that cache the content from the origin. The goal is to have content cached as close as possible to users, so content is cached at the edge of the CDN provider's network.

The impact this has:

- **Improved website load time**
  Instead of every client making a request to the origin server, which could be located a considerable distance away, the request is routed to a local server that responds with cached content, thus decreasing latency and increasing overall performance. Regardless of where the origin server and clients are located, performance will be more consistent for all users, as the CDN will serve locally cached content when possible.
- **Increased content availability and redundancy**
  Because every client request no longer needs to be sent to the origin server, CDNs provide not only performance benefits, but also availability and redundancy. Requests are load balanced over local servers with cached content; these servers respond to local requests, significantly decreasing overall load on the origin server. The origin server only is contacted when needed (when content is not cached or for dynamic non-cacheable content).
- **Improved website security**
  A CDN acts as a reverse proxy and sits in front of origin servers. Thus it can provide enhanced security such as DDoS mitigation, improvements to security certificates, and other optimizations.
- **Reduced bandwidth costs**
  Because CDNs use cached content to respond to requests, the number of requests sent to the origin server is reduced, thus also reducing associated bandwidth costs.

An important difference in some CDN implementations is how they route traffic to the respective local CDN nodes.

Routing requests to CDN nodes can be done via two different methods:

1. **DNS unicast routing**
   In this method, recursive DNS queries redirect requests to CDN nodes; the client's DNS resolver forwards requests to the CDN's authoritative nameserver. CDNs based on DNS unicast routing are not ideal in that clients may be geographically dispersed from the DNS resolver. Decisions on closest-proximity CDN nodes are based on the client's DNS server instead of client's IP address.

   Also, if any changes are needed for the DNS response, there is a dependency on DNS time to live (TTL) expiration.

   Further, since DNS routing uses unicast addresses, traffic is routed directly to a specific node, creating possible concerns when there are traffic spikes, as in a DDoS attack.

   Another challenge with DNS-based CDNs is that DNS is not very graceful upon failover. Typically a new session or application must be started for the DNS resolver with a different IP address to take over.
2. **Anycast routing**
   The Cloudflare CDN, which is discussed in more detail in the next section, uses Anycast routing. Anycast allows for nodes on a network to have the same IP address. The same IP address is announced from multiple nodes in different locations, and client redirection is handled via the Internet's routing protocol, BGP.

Using an Anycast-based CDN has several advantages:

- Incoming traffic is routed to the nearest data center with the capacity to process the requests efficiently.
- Availability and redundancy is inherently provided. Since multiple nodes have the same IP address, if one node were to fail, requests are simply routed to another node in close proximity.
- Because Anycast distributes traffic across multiple data centers, it increases the overall surface area, thus preventing any one location from becoming overwhelmed with requests. For this reason, Anycast networks are very resilient to DDoS attacks.

Introducing the Cloudflare CDN

Cloudflare provides a Software as a Service (SaaS) model for CDN. With Cloudflare's SaaS model, customers benefit from the Cloudflare CDN without having to manage or maintain any infrastructure or software.

The benefits of the Cloudflare CDN can be attributed to the below two points, discussed in more detail in this section.

1. **CDNs inherently increase performance by caching content on servers close to the user**
2. **The unique Cloudflare architecture and integrated ecosystem**

Figure 2 shows a simplified view of the Cloudflare CDN. Clients are receiving their response back from a server on Cloudflare's global Anycast edge network closest to where the clients are located, thus drastically reducing the latency and RTT. The diagram depicts a consistent end-user experience regardless of the physical location of the clients and origin.

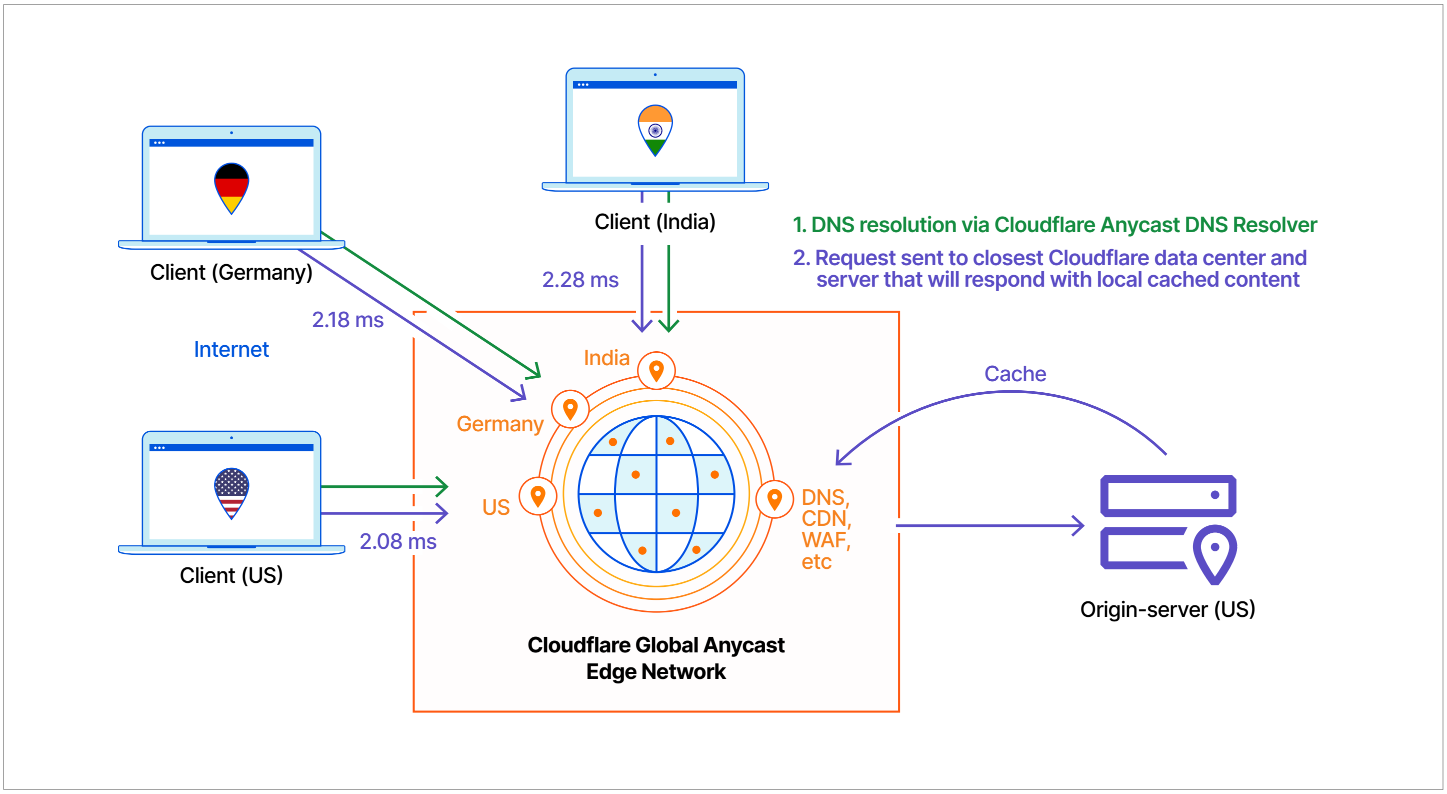

Client (India)
Client (Germany)
Client (US)
1. DNS resolution via Cloudflare Anycast DNS Resolver
2. Request sent to closest Cloudflare data center and server that will respond with local cached content
2.28 ms
2.18 ms
2.08 ms
Internet
India
Germany
US
DNS, CDN, WAF, etc
Cache
Origin-server (US)
Cloudflare Global Anycast Edge Network

Figure 2: HTTP request to Cloudflare CDN with Anycast

Cloudflare CDN architecture and design

Figure 3 is a view of the Cloudflare CDN on the global Anycast edge network. In addition to using Anycast for network performance and resiliency, the Cloudflare CDN leverages Argo Tiered Cache to deliver optimized results while saving costs for customers. Customers can also enable Argo Smart Routing to find the fastest network path to route requests to the origin server. These capabilities are discussed in detail in the remainder of this document.

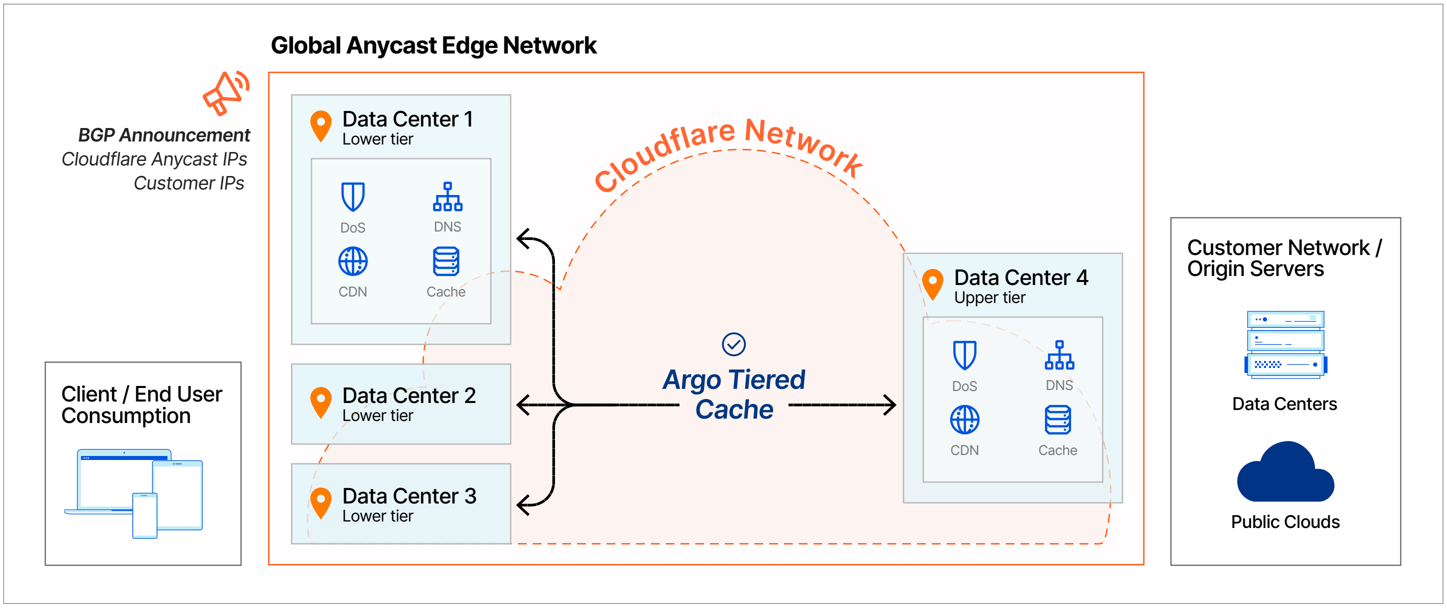


Figure 3: Cloudflare CDN with Argo Tiered Cache on global Anycast edge network

In the above diagram, there are a few important key points to understand about the Cloudflare CDN and the global Anycast edge network it resides on:

- An important differentiator is that Cloudflare utilizes one global network and runs every service on every server in every Cloudflare data center, thus providing end users the closest proximity to Cloudflare's services, with the highest scale, resiliency, and performance.
- Cloudflare is a reverse proxy, meaning it receives requests from clients and proxies the requests back to the customer's origin servers.

  Thus, every request traverses through Cloudflare's network before reaching the customer's network.

  Since Cloudflare has hardened and protected its infrastructure at the edge (ingress), all customers are consequently also protected from infrastructure-level and volumetric DDoS attacks. Requests and traffic must go through the protected Cloudflare network before reaching the customer's origin server.
- The Cloudflare CDN leverages the Cloudflare global Anycast edge network. Thus the incoming request is routed to and answered by the node closest to the user (eyeball).
- The inherent benefits of Anycast are decreased latency, network resiliency, higher availability, and increased security due to larger surface area for absorbing both legitimate traffic loads and DDoS attacks.

Cloudflare CDN architecture and design (continued)

Cloudflare's global Anycast edge network spans more than 250 cities across 100+ countries, reaching 95% of the world's Internet-connected population within 50 milliseconds while providing 100 Tbps of network capacity and DDoS protection capability.

- Edge nodes within the Cloudflare network cache content from the origin server and are able to respond to requests via a cached copy. Cloudflare also provides DNS, DDoS protection, WAF, and other performance, reliability, and security services using the same edge architecture.
- Argo uses optimized routing and caching technology across the Cloudflare network to deliver responses to users more quickly, reliably, and securely. Argo includes Smart Routing and Tiered Cache. Cloudflare leverages Argo to provide an enhanced CDN solution.

Argo Tiered Cache

Once a site is onboarded, standard caching is configured by default. With standard caching, each data center acts as a direct reverse proxy for the origin servers. A cache miss in any data center results in a request being sent to the origin server from the ingress data center.

Although standard caching works, it is not the most optimal design — cached content closer to the client may already exist in other Cloudflare data centers, and origin servers are sometimes unnecessarily overloaded as a result. Thus, it is best to enable Argo Tiered Cache, which is included with every Cloudflare plan. With Argo Tiered Cache, certain data centers are reverse proxies to the origin for other data centers, resulting in more cache hits and faster response times.

Argo Tiered Cache leverages the scale of Cloudflare's network to minimize requests to customer origins. When a request comes into a Cloudflare data center, if the requested content is not locally cached, other Cloudflare data centers are checked for the cached content.

Cloudflare data centers have shorter distances and faster paths between them than the connections between data centers and customer origin servers, optimizing the response to the client with a significant improvement in cache hit ratio. The Cloudflare CDN leverages Argo Smart Routing data to determine the best upper tier data centers to use for Argo Tiered Cache. Argo Smart Routing can also be enabled as an add-on to provide the fastest paths between data centers and origin servers for cache misses and other types of dynamic traffic.

The Cloudflare CDN allows customers to configure tiered caching. Note that depending on the Cloudflare plan, different topologies are available for Argo Tiered Cache. By default, tiered caching is disabled and can be enabled under the caching tab of the main menu.

Argo Tiered Cache Topologies

The different cache topologies allow customers to control how Cloudflare interacts with origin servers to help ensure higher cache hit ratios, fewer origin connections, and reduced latency.

Argo Tiered Cache Topologies		
Smart Tiered Cache Topology (All plans)	**Generic Global Tiered Topology** (Enterprise Only)	**Custom Tiered Cache Topology** (Enterprise Only)
• Recommended for most deployments. It is the default configuration once Tiered Cache is enabled. • Ideal for customers who want to leverage CDN for performance but minimize requests to origin servers and bandwidth utilization between Cloudflare and origin servers. • Cloudflare will dynamically find the single best upper tier for an origin using Argo performance and routing data.	• Recommended for those who have high traffic that is spread across the globe and desire the highest cache usage and best performance possible. • Generic Global Tiered Topology balances between cache efficiency and latency. Instructs Cloudflare to use all Tier 1 data centers as upper tiers.	• Recommended for customers who have additional data on their user base and have specific geographic regions they would like to focus on. • Custom Tiered Cache Topology allows customers to set a custom topology that fits specific needs (ex: upper tiers in specific geographic locations serving more customers). • Engage with a Customer Success Manager (CSM) to build a custom topology.

Traffic flow: Argo Tiered Cache, Smart Tiered Cache Topology

In Figure 4, Argo Tiered Caching is enabled with Smart Tiered Cache Topology. The diagram depicts two separate traffic flows, summarized below. The first traffic flow (Client 1 in green) is a request from a client that comes into Data Center 1. The second traffic flow (Client 2 in purple) is a subsequent request for the same resource into a different data center, Data Center 2.

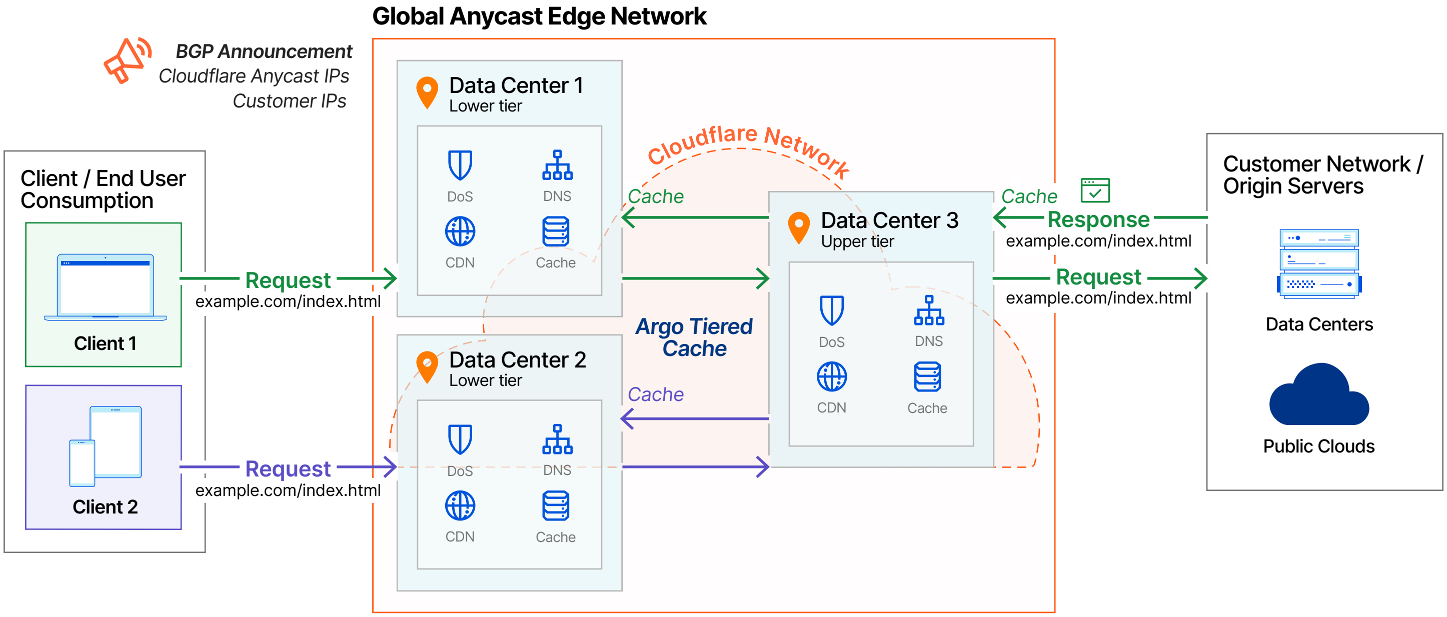


Figure 4: HTTP requests and traffic flow through Cloudflare CDN

Client 1

- First request received in Data Center 1 results in cache miss, as request had not been made previously by any client.
- No cached content found, so Data Center 1 checks with upper tier data center to request a copy of the content.
- Upper tier data center also does not have content cached locally, so it makes a request to the origin server for content. Upon receiving the content, the upper tier data center caches it locally and relays the content to the requesting lower tier data center. The lower tier data center caches the content and responds to the client.

Client 2

- Second request by a different client received in Data Center 2 results in cache miss, as request had not been made previously by any client served by Data Center 2.
- No cached content found, so Data Center 2 checks with the upper tier data center to request a copy of the content.
- Cached content found at the upper tier data center. Data Center 2 retrieves and caches this content locally and responds to the client.

Traffic flow: Argo Tiered Cache, Smart Tiered Cache Topology (Continued)

In Figure 4, Client 1 traffic flow displays the traffic flow when a client request is received by a data center closest to the client, Data Center 1. Since there is nothing locally cached on the ingress data center and tiered caching is enabled, a request is sent to the upper tier data center to request a copy of the content to cache.

Because the upper tier data center also does not have the content cached, it sends the request to the origin server, caches the received content upon response, and responds to the lower tier data center with the cached content. The lower tier data center caches the content and responds to the client.

Notice that when a new request for the same content is made to another data center (Client 2 traffic flow), Data Center 2, the content is not locally cached; however, the content is retrieved from the upper tier data center, where it was cached from the first request for the same content.

With the upper tier data center returning the cached content for the second request, the trip to the origin server is prevented, resulting in higher cache hit ratios, faster response times, saved bandwidth cost between the Cloudflare network and the origin server, and reduced load on the origin server responding to requests.

Argo Smart Routing

Argo Smart Routing is a service that finds optimized routes across the Cloudflare network to deliver responses to users more quickly. As discussed earlier, Cloudflare CDN leverages Argo Smart Routing to determine the best upper tier data centers for Argo Tiered Cache.

In addition, Argo Smart Routing can be enabled to ensure the fastest paths over the Cloudflare network are taken between upper tier data centers and origin servers at all times. Without Argo Smart Routing, communication between upper tier data centers to origin servers are still intelligently routed around problems on the Internet to ensure origin reachability.

Argo Smart Routing accelerates traffic by taking into account real-time data and network intelligence from routing over 28 million HTTP requests per second; it ensures the fastest and most reliable network paths are traversed over the Cloudflare network to the origin server. On average, Argo Smart Routing accounts for 30% faster performance on web assets.

Traffic Flow: Argo Tiered Cache, Smart Tiered Cache Topology with Argo Smart Routing

Figure 5 details the traffic flow when Argo Tiered Cache and Argo Smart Routing are not enabled. The request comes into the closest data center, and, because content is not locally cached and Argo Tiered Cache is not enabled, the request is sent directly to the origin server for the content Also, since Argo Smart Routing is not enabled, a reliable, but perhaps not the fastest, path is taken when communicating with the origin server.

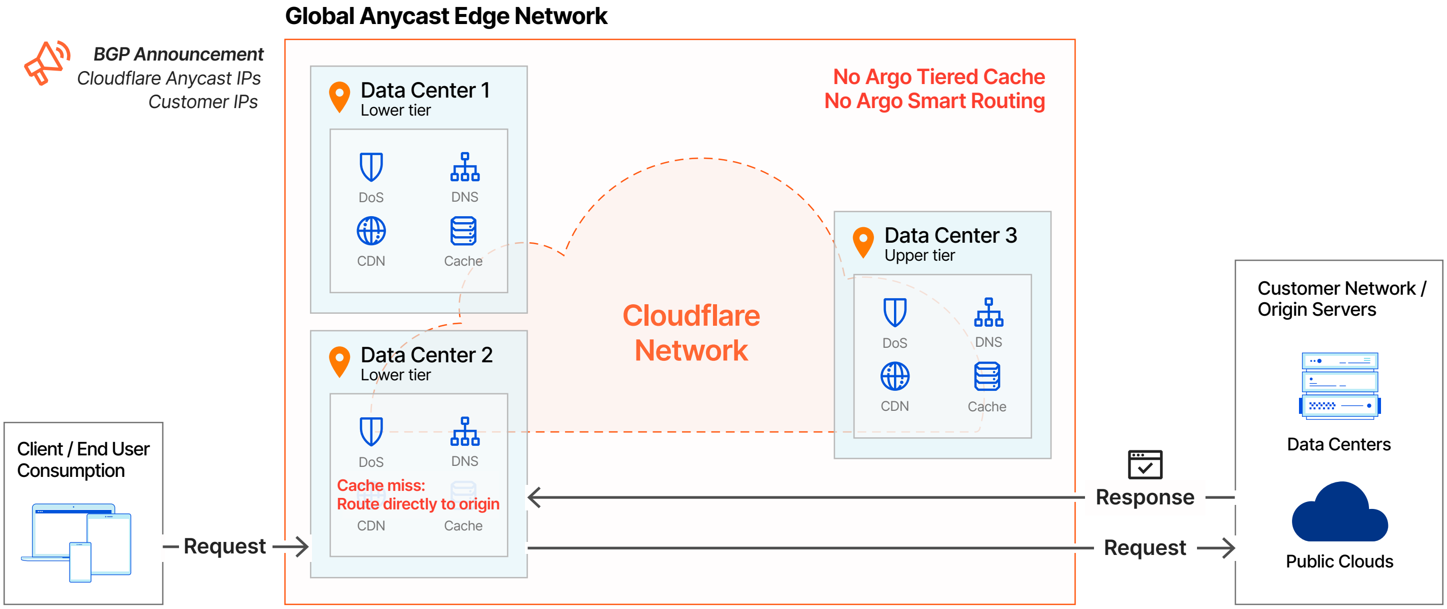


Figure 5: Cloudflare CDN without Argo Tiered Cache and Argo Smart Routing

Traffic Flow: Argo Tiered Cache, Smart Tiered Cache Topology with Argo Smart Routing (continued)

Figure 6 articulates the traffic flow with both Argo Tiered Cache and Argo Smart Routing enabled.

In Figure 6, when a request is received by Data Center 1 and there is a cache miss, the cache of the upper tier data center, Data Center 3, is checked. If the cached content is not found at the upper tier data center, with Argo Smart Routing enabled, the request is sent on the fastest path from the upper tier data center to the origin.

The fastest path is determined by the Argo network intelligence capabilities, which take into account real-time network data such as congestion, latency, and RTT.

With the Cloudflare CDN, Argo Smart Routing is used when:

1. There is a cache miss and the request needs to be sent to the origin server to retrieve the content,
2. There is a request for non-cacheable content, such as dynamic content (ex: APIs), and the request must go to the origin server.

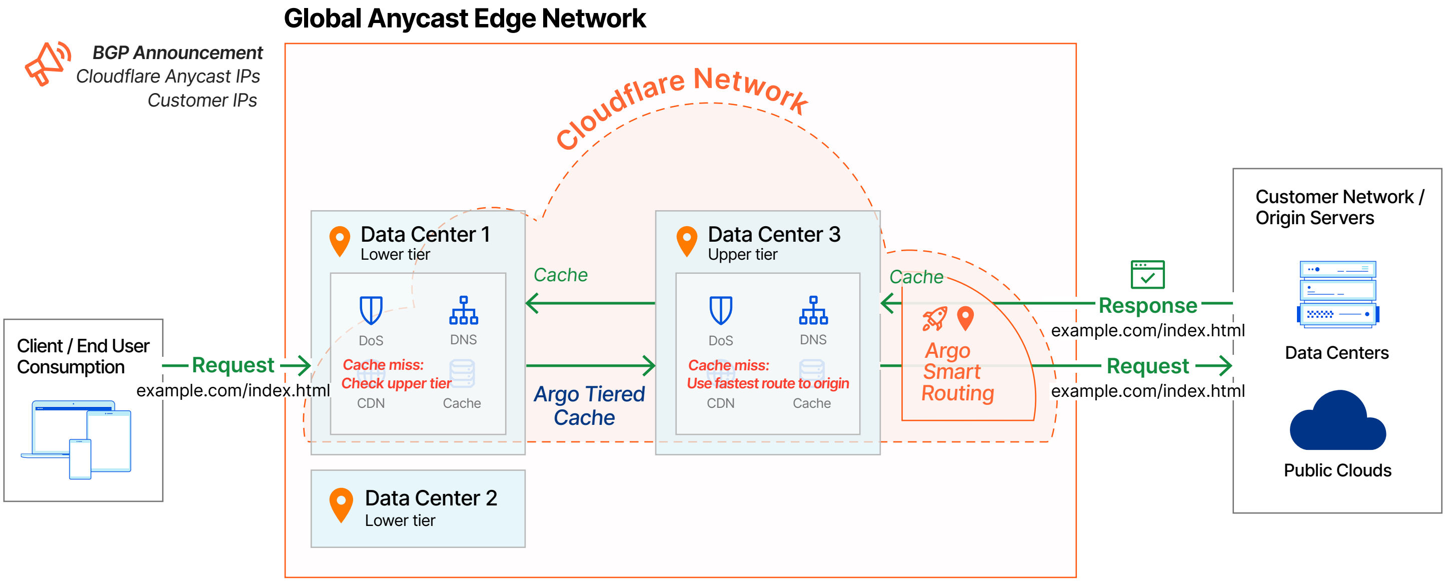


Figure 6: Cloudflare CDN with Argo Tiered Cache and Argo Smart Routing enabled

Summary

To summarize, the Cloudflare CDN is SaaS that helps address the challenges customers face around latency, performance, availability, redundancy, security, and costs. The Cloudflare CDN leverages Cloudflare's global Anycast edge network and Argo Tiered Cache to deliver optimized results while saving costs for customers. Customers can also enable Argo Smart Routing to ensure the fastest network path is used to route requests to the origin server.