Google Cloud

# The global front end solution of Google Cloud

### How to use this document?

This document covers high level details about the global front end solution of Google Cloud. The document first delves into different architecture options. It then discusses some of the use cases in the context of architecture options explained before. The document also summarizes some of the best practices/recommendations from Google Cloud.

### Audience

Network architects and network or infrastructure administrators

### Objective

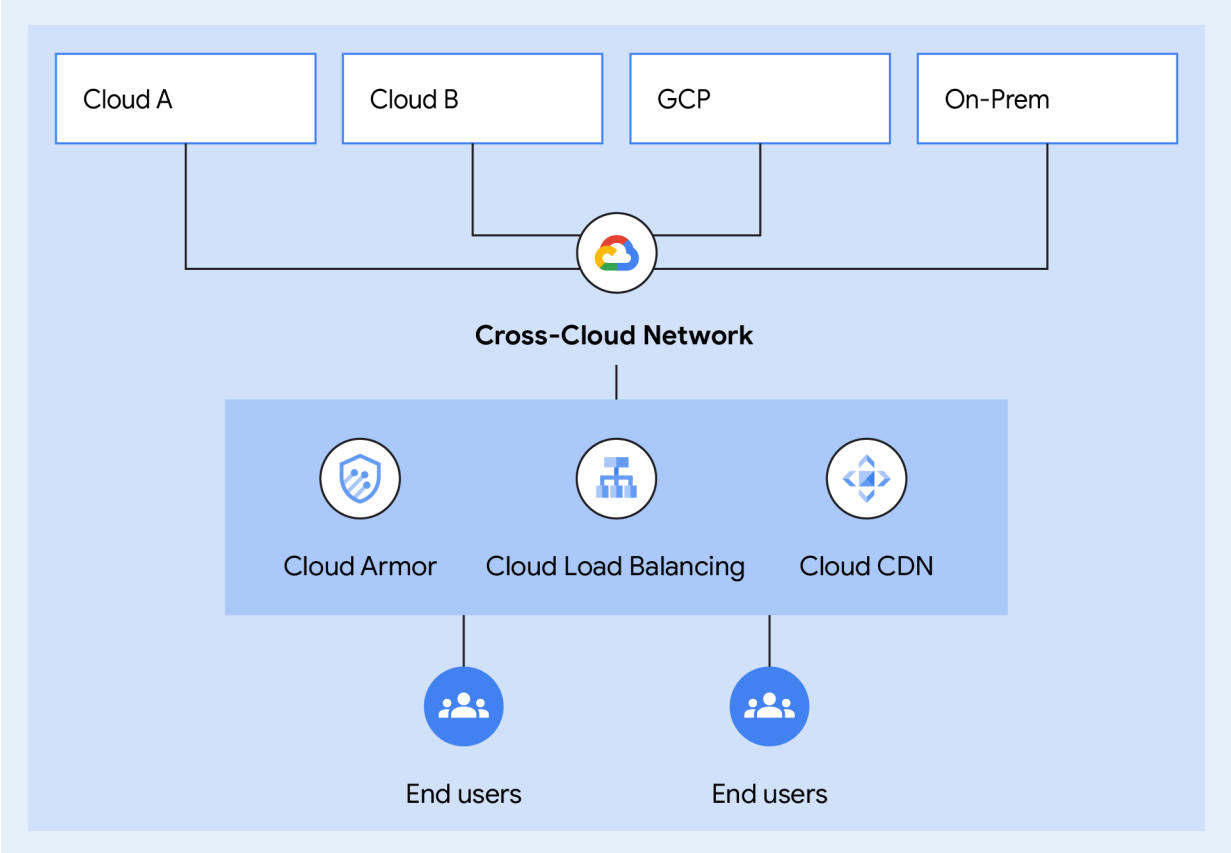
- Gain a high level understanding of the global front end architecture
- Learn about the architecture option for deploying the global front end
- Learn about some of the best practices for deploying the global front end

# 1. Overview

Managing a globally scaled application is incredibly challenging and complex especially when you add the complexity of multiple origins addressing parts of the same workload. Customers are challenged to deliver high performance at scale and at the same time protect their origins from web attacks and intrusions. Moreover, the future deployment of web workload might span multiple clouds and on-prem. Navigating the complexities of hybrid and multicloud web application management is a challenge for many organizations. To address these challenges, Google Cloud has developed the global front end solution.

**What is the global front end?**

The global front end is a solution to help organizations deliver, scale, and protect their internet facing applications using Google Cloud's global infrastructure. This is done by providing a solution that can deliver not just from Google Cloud but from infrastructure hosted in any cloud, colocation, or datacenter. The primary products in this solution are Cloud Load Balancing (providing scalability, availability, and reliability), Cloud Armor (providing distributed denial of service [DDoS] and web application protection), and Cloud CDN (improving latency, reducing TCO, and enhancing customer experience).



### Benefits of using the global front end

There are several advantages of using the global front end. The key benefits include:

Elevate an application's performance, security, and global reach with our comprehensive front end proxy solution. Google Cloud has a large network with 187+ PoP locations and presence in over 200+ countries. It differentiates with its premium network which minimizes the distance and number of network hops resulting in lower latency and enhanced security. Performance is further enhanced by Cloud CDN which provides caching services at the Edge location for optimized web and application performance.

Google Cloud Armor protection against network layer (L3) and transport layer (L4) attacks is set by default and preconfigured for all Google Cloud projects deployed using proxy load balancers. Automatically detect and help mitigate high volume Layer 7 DDoS attacks with an ML system trained locally on your applications. Help defend applications from DDoS or web attacks and enforce Layer 7 security policies whether your application is deployed on Google Cloud or in a hybrid or multi cloud architecture. This includes protection from (not a comprehensive list) direct botnet attacks, UDP floods, SYN floods, TCP floods, DNS floods, ICMP ping floods and UDP amplification/reflection attacks. Recently in late 2023, we blocked the planet's largest ever known DDoS attack at about 398mRPS.

Provide a high level of redundancy and availability for the application while residing on any cloud or on-prem. The Google Cloud external Application Load Balancer provides you the ability to group your backend resource, balance the workload across the group members and provide the elasticity / scale for components of the application.The backend may be distributed across multiple regions providing high resiliency and optimal load distribution. The newly introduced Service Extensions integration lets you use Cloud Load Balancing to make gRPC calls to user-managed services during data processing. You write callout extensions against Envoy's external processing gRPC API. Callout extensions backends run as general purpose gRPC servers on user-managed compute VM instances and Google Kubernetes Engine pods on Google Cloud, multicloud, or on-premises environments.

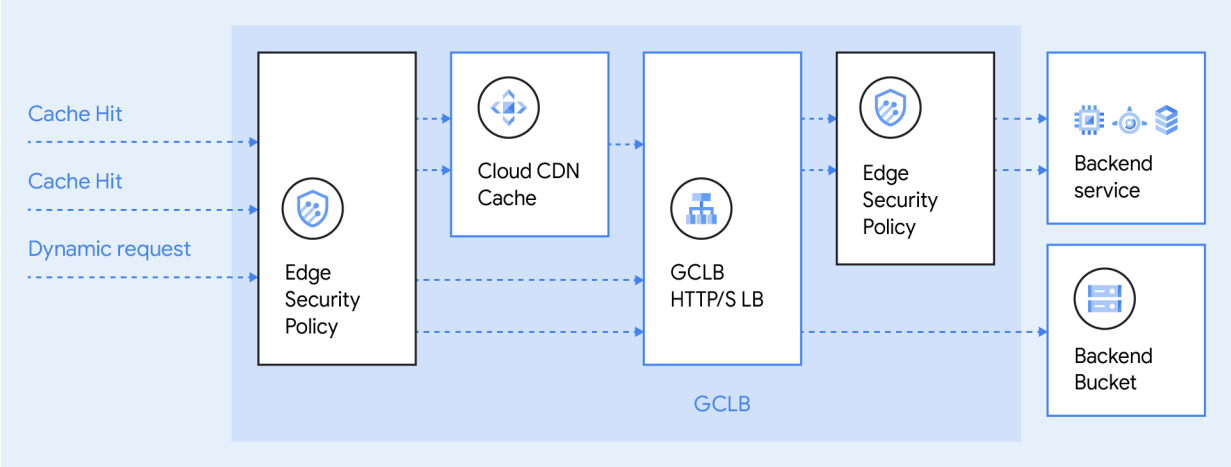
The global front end toolkit provides an out-of-the-box, curated solution to accelerate the delivery of internet-facing applications. It combines Cloud Load Balancing, Cloud Armor, and Cloud CDN into one solution with support for Cloud Build or third-party CI/CD tools like Jerkins and Gitlab. The goal is to provide platform and DevOps engineers the ability to accelerate their web deployments in the cloud.

### Who should consider using the global front end?

The global front end solution is for any deployment which is delivering internet facing applications and content to its users. The use case can be deploying an ecommerce application, or hosting a news/media website or API acceleration or delivering software to end clients. The application and content can exist in any location: i.e. any cloud provider or any data centers. The global front end can connect and monitor any backend location in an efficient, performant, and secure manner.

# 2. Architecture for Securing/Scaling and Delivering Application

This architecture has the following workflow:




A user requests a web page from the global front end solution. The request is evaluated by Cloud Armor edge security policies, If CDN is enabled and CDN has the content in cache (cache hit scenario). If CDN does not have the content in cache (cache miss scenario), the content will need to be retrieved from the backend, then the request is evaluated by Cloud Armor backend security policies before being distributed to one of the backend servers. Once the content has been retrieved, it will be cached at the CDN and a cached response will be served for future requests. Lastly, the global front end returns the web page content to the user.

## 2.1 Deployment Options

Before we start detailing the architecture, there are some considerations which would impact the design of the solution:

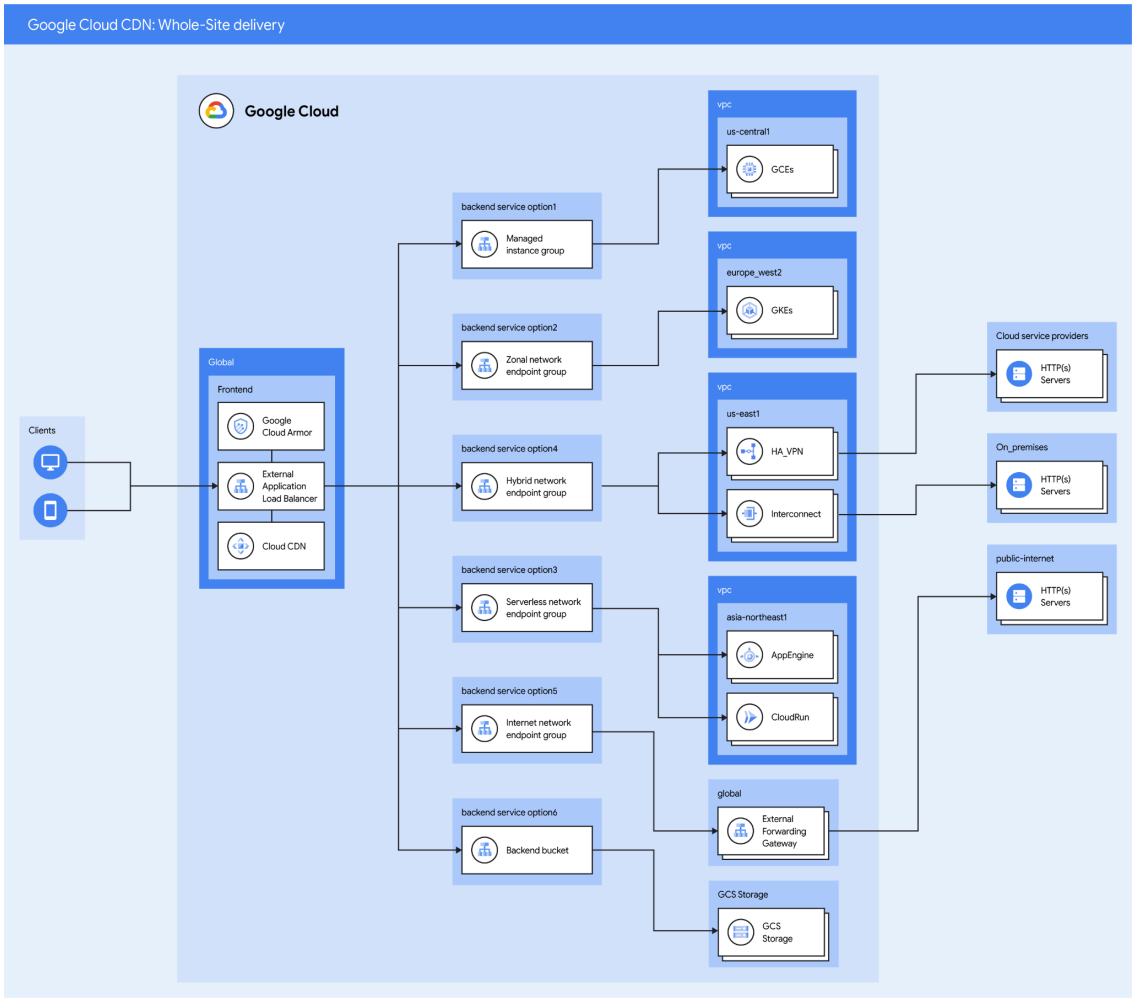
**Users:** It's important to understand if the user, which the application is serving, is local to the region or global. Specifically for global applications, careful considerations should be made for redundancy, capacity, and high availability design. Anycast design greatly simplified the availability of applications to the global users.

**Application:** The application serving the user can reside on on-prem or any other cloud or in a mix of locations. They can also be hosted in specific regions or multiple regions. They can be connected to the front end via private connectivity or over the internet or use VPN. Cloud Load Balancing allows for configuring and deploying applications with high levels of redundancy and availability.

**Database:** The database where all the storage resides and its high availability requirements .

**Security:** What are security requirements for content served from cache ? What are the web application firewall (WAF) or DDoS requirements for the web application?

A typical architecture is depicted below:



Leveraging Premium Tier external Application Load Balancers empowers your global web application with intelligent, multi-regional traffic distribution. Its single, globally accessible IP address simplifies accessibility while its proximity-based routing directs users to the closest available backend server, minimizing latency. This ensures optimal performance and responsiveness for users worldwide. Importantly, the load balancer dynamically adapts to changing conditions. If a server becomes overloaded or unavailable, traffic seamlessly shifts to the next closest healthy server with capacity, guaranteeing uninterrupted service. Furthermore, the ability to configure backend services with instance groups across multiple regions unlocks further scalability and resilience, enabling your application to effortlessly handle spikes in user traffic or regional outages.

**Backend with external load balancers**

As depicted above, the external load balancers distribute the traffic coming from a user to specific backend. The backends, which reside in a region, could further have multiple tiers in order to increase the manageability, scalability, redundancy, and operational ease. A typical highly available and resilient architecture will have multiple tiers:

**Web tier:** An external Application Load Balancer distributes traffic from the internet to multiple web servers in different regions. This ensures that no single server gets overloaded.

**Application tier:** An internal Application Load Balancer distributes traffic within your private network to multiple application servers. This helps your application scale up or down as needed.

**Database tier:** An internal passthrough Network Load Balancer distributes traffic to multiple database servers. This ensures that your database can handle high volumes of requests.

The global front end distributes HTTP and HTTPS traffic to backends hosted on a variety of Google Cloud platforms, as well as external publicly accessible HTTP backends hosted in other cloud service providers (CSPs), or in on-premises environments connected over the internet via Google Cloud's premium network or via hybrid connectivity.

**Hosting the backend in Google Cloud:** When hosting the backends in Google Cloud, the options are Google Compute Engine VM instances, Google Kubernetes Engine pods, Google AppEngine, Cloud Run, or Cloud Storage buckets. These backends can be in a single or multiple Google Cloud regions. It is strongly recommended to have all the resources in the same project.

**Hosting the backend in non-Google Cloud location:** When hosting the backends outside of Google Cloud, the connections can be over VPN or Cloud Interconnect for private connectivity using hybrid network endpoint group (hybrid NEG), or through external forwarding gateway for public connectivity using internet network endpoint group (internet NEG). When connecting over the internet, the backend will have to be exposed through an external IP and the traffic will be insecure. Hence it is recommended to use the hybrid NEG method to connect to the backend when the backend is not storage.

**Hybrid backend deployment:** It is possible to host certain modules in Google Cloud and some other modules in other CSPs or on-prem. In this deployment model, the load balancer decides the appropriate backend depending on the URL map. There are various options available from redundancy and high availability perspective. In some deployment models, certain backends may provide better

latency or cost or performance. Such backends can be preferred for a configured capacity before load balancer algorithms distribute traffic to other backends. In such cases, auto-capacity draining can also be enabled to check for the health of the backend's instance or endpoints. This will automatically remove the unhealthy (determined by a threshold) backend from the global load balancing pool.

Choice of backend protocol (HTTP, HTTPS, or HTTP/2) impacts application latency and the network bandwidth available for your application. For example, using HTTP/2 between the load balancer and the backend instance can significantly reduce the TCP connections to the instance. As a result, you might see high backend latencies because backend connections are made more frequently. The backend service protocol impacts how the traffic is encrypted in transit. With external Application load balancers, all the traffic going to backends that reside within Google Cloud are automatically encrypted. However, this is only available for communications within instance groups and zonal NEG backends. For all other backend types, Google recommends using secure protocols such as HTTPS or HTTP/2.

The Google Cloud external load balancer provides advanced capabilities to handle different failover conditions . This could be extended to use in migration scenarios as well.

**Failover and redundancy:** It is possible to have a deployment model where the application on Google Cloud is backed up by the same application running on other CSPs. In such a case, a backend service can be created with a mix of zonal NEG and hybrid NEG. The zonal NEG points to a Google Cloud VM instance while the hybrid NEG points to another CSP's resource. The zonal NEG or the hybrid NEG can be made primary while other acts as backup.

**Migration:** In certain deployment models, there could be a requirement to gracefully migrate the application from another CSP to Google Cloud. In such a case, different backend services having different traffic weights can be created. To start with, Google Cloud can have a small amount of traffic and as the application stabilizes the percentage of users being served through Google Cloud can be increased.

**Performance/latency**

The global front end includes the Integrations with Cloud CDN and is highly recommended to enable Cloud CDN for content delivery for faster delivery. By default, the cache mode is set to "cache static content". However, for best performance of an application which has a large number of static objects, set the cache mode to "force cache all content" with desired TTLs. Additional cost savings can be achieved by setting this compression mode to "automatic". Dynamic compression automatically compresses responses served by Cloud CDN. The size of the data sent over the network is reduced by 60% to 85% in typical cases. The size reduction reduces the time it takes to download content. For important assets like stylesheets (CSS), scripts (JavaScript), and video manifests (HLS/DASH), this can reduce page load and video start times. Optionally set serve while stale to at least "1 min". To improve the performance further, enable 'negative caching', Negative caching lets you set a different TTL for each status code. The reason to do this is to apply fine-grained control over caching for common errors or redirects, which in turn can reduce the load on your origins and improve the end-user experience by reducing response latency.

In migration scenarios, it is possible to cascade Google Cloud CDN in front of other CDNs. When the applications are in other cloud service providers, there could be cost implications when content is fetched from the other CSP to the Google Cloud global front end. In such a case, it is possible to make

another CSP's CDN as a backend to the Google Cloud global front end. This will bring the cost down for the content which is available in the other CSP's CDN.

### Security

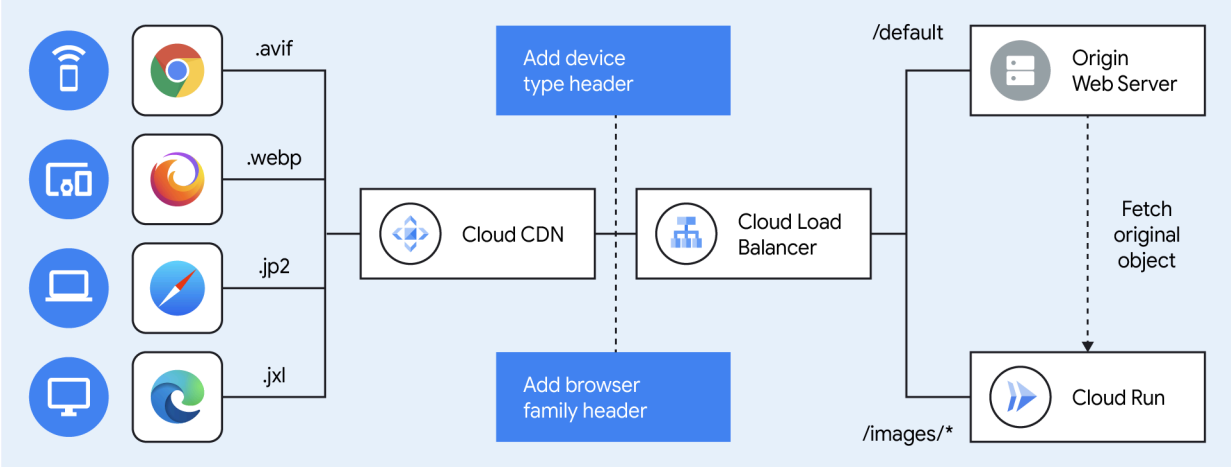
Internet facing services have increasingly become targets for malicious or fraudulent bot activities such as scraping, stock out attacks, credential stuffing, and L7 DDoS attacks. Securing both the backend and front end is critical . From a backend security perspective, it's strongly recommended to enable HTTPS delivery and protect the backend by restricting public access by enabling signed URL/ signed cookies. Cloud CDN's private origin authentication for object storage origin should also be enabled. From a front end security perspective, if there is any business requirement to restrict users on the basis of geolocation, Cloud Armor edge security policies can be configured to enforce actions based on the request's source IP/source geography. Cloud Armor rule evaluation stops at the first rule match, so the placement of rules is an important consideration. Rate limiting rules should come after your explicit allow and deny rules. We recommend the following structure of rule priority, from greatest-priority to least-priority:

- Explicit deny rules (ASN, region, IP ranges)
- Trusted explicit allow rules (scanners, trusted systems — use with extreme caution)
- Adaptive protection auto deploy rule (Recommended to keep in preview deny mode)
- OWASP, custom deny rules
- Explicit allow rules (ASN, presence of header value, IP range)
- Rate limiting rules
- Default deny rules

The threat landscape is changing as more businesses require that their customers have user accounts and login credentials to make purchases. reCAPTCHA Enterprise is useful in such cases where you want to protect your websites from bots, and abusive or fraudulent behavior that are either carried out through automated attacks or done by humans. Using Cloud Armor security policy at the edge of the network, reCAPTCHA Enterprise can be easily enabled.

### Programmability

In many deployments, there may be a requirement to have more control over the packet to solve some specific business requirements. In such cases, Service Extensions can be leveraged to provide advanced HTTP header manipulation and normalization. One of the commonly asked requirements is to optimize the image depending on the network and end devices. In the below diagram, the image optimization utility is built and run on Cloud Run. Cloud CDN inserts the client-device-type and client-ua-family attributes to the incoming request. In the case of cache-miss, Cloud Load Balancing routes the traffic to the image optimization utility running on Cloud Run which fetches the image from the origin, if required, and mutates the image to the size/format desired by the solution.




Observability

- Enable Cloud Logging
- Use a custom monitoring dashboard for Cloud CDN. Cloud Monitoring provides a set of dashboard definitions available on GitHub in the monitoring-dashboard-samples repository as JSON files. In the networking file, there is a Cloud CDN-specific dashboard called cloud-cdn-monitoring.json. Upload this custom dashboard to Cloud Monitoring.
- If cost is a consideration, then enable logging during initial deployment of the project. Once the deployment stability is reached, logging can be turned off.
- If cost if a consideration, then optionally sample rate can be enabled — which lets the administrator control the probability of request being logged (value of 1 would mean that all requests are log)

## 2.2 Use-Cases

The Google Cloud global front end can help with many use cases and deployments of applications. Some of the prominent common use cases are below:

**Whole site delivery:** Modern web experiences demand lightning-fast load times, the ability to handle surges in traffic and protect the application, regardless of a user's location. Whole site delivery (WSD) addressed these challenges by caching not only static website assets but also dynamic content, ensuring optimal performance and reliability. This can be used for ecommerce applications, news and media websites, high traffic blogs, and other web applications.

The users of this use case may be spread across the globe. This is where an external Application Load Balancer's use of a single, global anycast IP address could simplify the solution. The solution would consist of both static and dynamic content which needs intelligent routing that can be accelerated and protected using the global front end. Some of the techniques mentioned in the performance section above can be used to achieve optimal results. The application backend, or origin, can reside anywhere (Google Cloud or other CSPs or on-prem). The use of hybrid NEG can greatly simplify the experience as it can connect the backend resources over Cloud VPN or Cloud Interconnect to another CSP or on-prem. From a security perspective, Cloud Armor automatically detects and helps mitigate high volume Layer 7 DDoS attacks with an ML system trained locally on your applications. The deployment

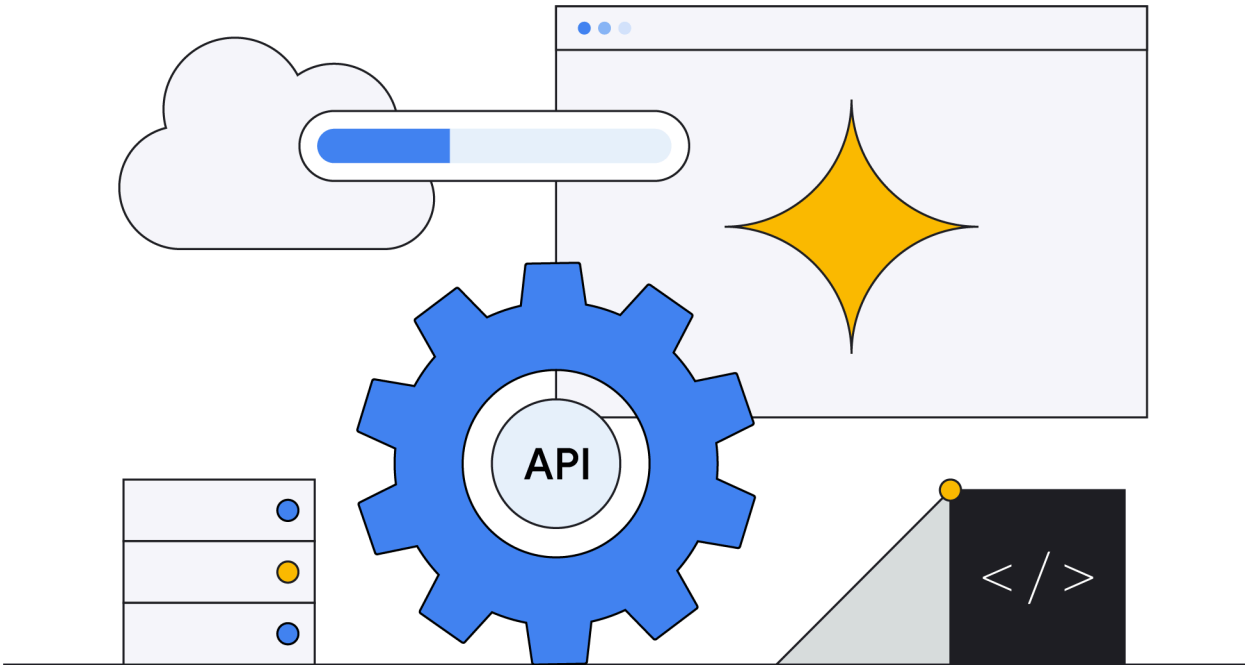
can also consider native integration of reCaptcha Enterprise to provide automated protection for your apps from bots and to help stop fraud both in line and at the edge.

**API acceleration:** An application programming Interface (API) is the backbone of any modern application. They provide a structured way for the client to access different software components and data of the application. While the use of an API is pervasive in modern day applications, it can also suffer from performance/latency and security challenges. Hosting the complete application in Google Cloud can provide an optimal solution for this use case. The latency can be significantly improved with the global front end's large number of edge locations for caching and premium network access across the Google Cloud backbone. From a security perspective, the deployment can enable TLS everywhere in the network. With Apigee X, customers can easily and seamlessly apply Cloud Armor web application firewall (WAF) to APIs, adding another layer of security to ensure that corporate digital assets are accessed only by authorized users. If API personalization is a requirement, then Service Extensions callout can be used which allows user to instruct Google Cloud networking products to make gRPC 'callouts' to custom services running in Google Cloud, multicloud, or on-premises from within the data processing path

**Software downloads:** Distributing software files can be challenging and costly as the file sizes can be large. Releases and major updates can cause significant traffic spikes which can overwhelm your infrastructure, leading to slowdowns, failed downloads, and frustrated users. With its large number of edge locations, and planet scale security, the Google Cloud global front end offers an excellent solution for this use case.

## 2.3 Deployment

To deploy a global front end, use the Terraform example which is available on GitHub.
For more information, see the README.



## 2.4 Summary Best Practices and Recommendations

**Optimize for latency:** Create your load balancer backends in the region closest to where you anticipate your users' traffic to arrive at the Google Cloud global front end. In many cases, running your backends in multiple regions is necessary to minimize latency to clients in different parts of the world.

**Optimize for health checks:**

- Autoscaling and autohealing are independant. Autohealing is a more expensive operation and hence it's recommended to be careful with timers.
- Health checks can be chatty which can have implications on logging

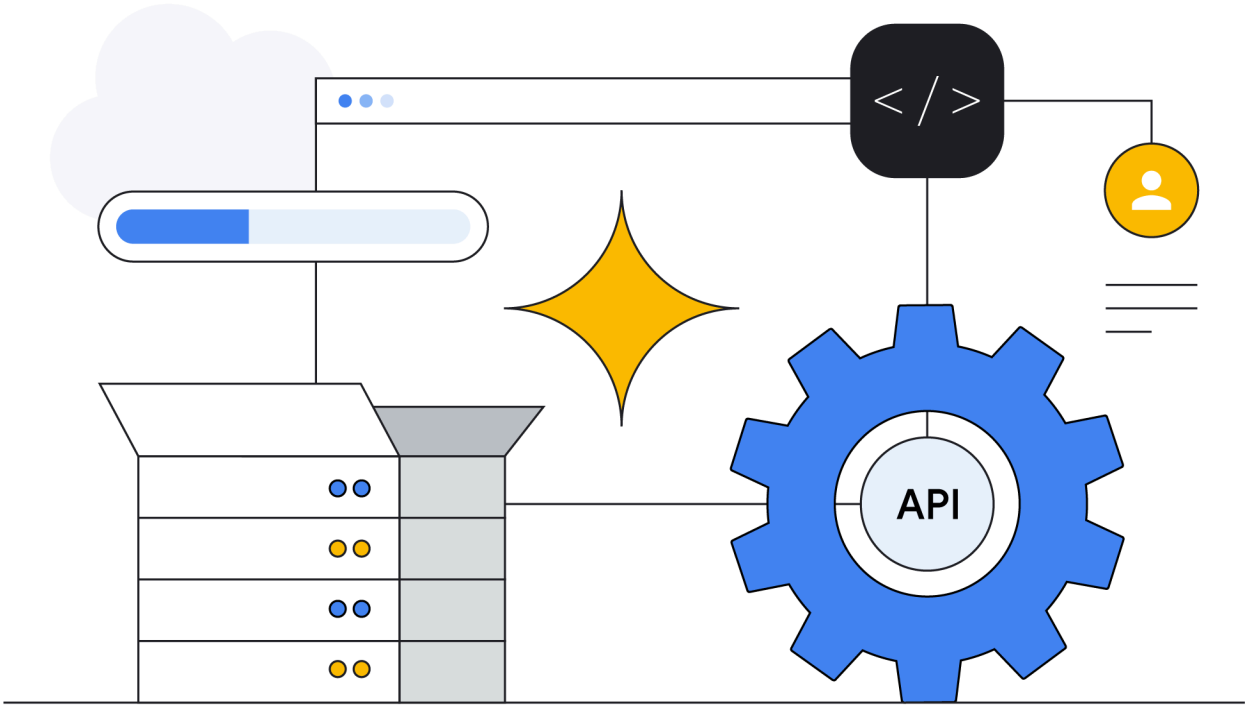
**Secure the service:**

- Enable TLS everywhere in the design
- Secure at edge with cloud armor
- It's recommended to not use public IPs on backends
- Use signed URLS
- Authenticate private origins

Operational recommendations:

- Group like applications into backend services - with similar request characteristics, security, health checks, etc.
- Understand the difference between TTFB (Time to First Byte) and TTLB (Time to Last Byte) parameters. Depending on the application, TTFB or TTLB might be important. Evaluate and optimize your CDN depending on what parameter is important for your application.
- Reduce the TTL value in Cloud DNS while onboarding. Increase the TTL value after integration is completed.
- Use Terraform to configure as it provides better control on version management and also helps recover to last known good configuration.
- Objects that don't match route rules won't be cached. This is the default behavior.
- Set appropriate connection timeouts for long lived connections.
- Use cookies or IP based affinity for persistence
- Use HTTP/3 for forwarding rule protocol selection
- Create alerts for origin/edge failure or for any latency increments.
- Enable full logging during integration which helps troubleshoot and optimize faster. Remember to turn off the full logging after the integration as it may result in increased cost.

- Enable dynamic compression which automatically compress text based content into Brotli or Gzip compressed file format
- Don't use Cloud CDN to cache user-specific content.
- Use the --cache-mode=CACHE_ALL_STATIC setting (default). This setting lets Cloud CDN cache common static content types when the origin does not specify any caching directives in the response headers. Ensure that your content matches the categories outlined; otherwise, content is not cached.
- Use versioned URLs to update content instead of invalidation
- Use custom cache-keys to improve cache hit ratio
- Use negative caching
- Use custom monitoring dashboard for Cloud CDN



# 3.5 References

https://cloud.google.com/load-balancing/docs/https/use-cases

https://cloud.google.com/architecture/framework/reliability/design-scale-high-availability

https://github.com/GoogleCloudPlatform/terraform-google-waap/tree/main/examples/web_app_protection_example

https://cloud.google.com/certificate-manager/docs/overview

https://cloud.google.com/dns/

https://cloud.google.com/load-balancing/docs/https

https://cloud.google.com/security/products/armor

https://cloud.google.com/cdn

https://cloud.google.com/load-balancing/docs/service-lb-policy#auto-capacity-draining