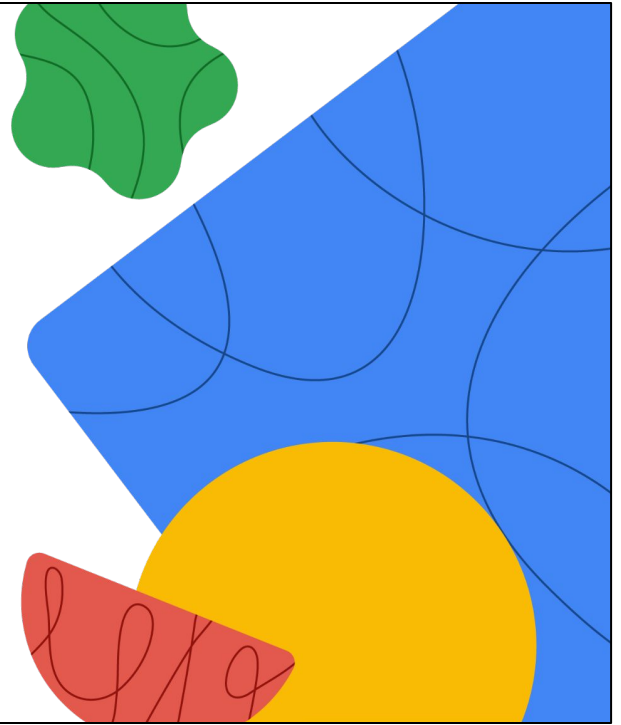




# Networking in Google Cloud


Caching and Optimizing Load  
Balancing



Welcome to the Caching and Optimizing Load Balancing module



# Today's agenda



- 01 [Internal Network Load Balancers as next hops](#)
- 02 [Cloud CDN](#)
- 03 [Lab: Defending Edge Cache with Cloud Armor](#)
- 04 [Load balancer optimization strategies](#)
- 05 [Quiz](#)

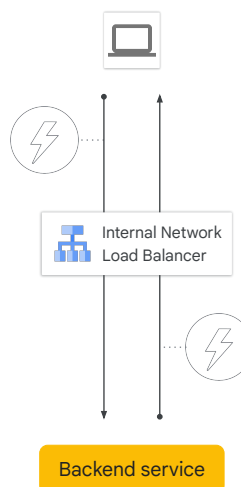
In this module, we will discuss using internal Network Load Balancers as next hops, including benefits, caveats, and some use cases.

We will also introduce Google Cloud Armor and Cloud CDN. We will cover content caching and share some strategies to optimize load balancing.

Let's get started.

## Internal Network Load Balancers are fast

- Internal Network LBs are high-performance, pass-through Layer 4 load balancers.
- Client requests to the load balancer IP address go directly to the healthy backend VMs. Responses from the healthy backend VMs go directly to the clients, not back through the load balancer.



Before we cover how to use an internal Network Load Balancer as a routing next hop, let's discuss why these load balancers are useful: they're fast.

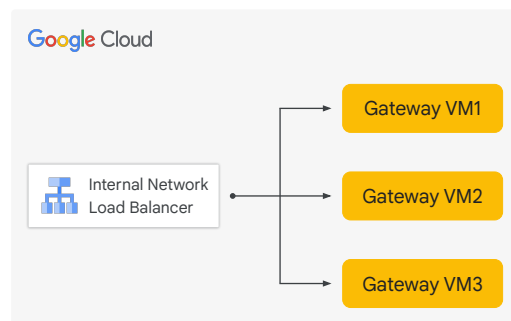
Internal Network Load Balancers don't have the overhead associated with other types of Cloud load balancers; the reduced overhead makes them fast.

An internal Network Load Balancer routes connections directly from clients to the healthy backend without any interruption. There's no intermediate device or single point of failure. Client requests to the load balancer IP address go directly to the healthy backend VMs. Unlike other types of load balancers, there's minimal processing of the incoming traffic.

Responses from the healthy backend VMs go directly to the clients, not back through the load balancer. TCP responses use direct server return. For more information, see [IP addresses for request and return packets](#) in the Google Cloud documentation.

## Use cases

- Load-balance traffic across multiple VMs that are functioning as gateway or router VMs.
- Use gateway virtual appliances as a next hop for a default route.
- Send traffic through multiple load balancers in two or more directions by using the same set of multi-NIC gateway or router VMs as backends.



Let's consider some use cases for internal Network Load Balancers.

You can load-balance traffic across multiple VMs that are functioning as gateway or router VMs.

You can use gateway virtual appliances as the next hop for a default route. With this configuration, VM instances in your virtual private cloud (VPC) network send traffic to the internet through a set of load balanced virtual gateway VMs.

You can send traffic through multiple load balancers in two or more directions by using the same set of multi-NIC gateway or router VMs as backends. To accomplish this result, you create a load balancer and use it as the next hop for a custom static route in each VPC network. Each internal Network Load Balancer operates within a single VPC network; distributing traffic to the network interfaces of backend VMs in that network.

In these use cases, the backend services are the gateway VMs, gateway virtual appliances, multi-NIC gateways, and router VMs. Because these resources are all internal, it makes sense to access them through an internal Network Load Balancer. As we discussed a moment ago, these load balancers have lower overhead than other load balancers that Google Cloud offers.

Next, let's consider how to access to these backends even faster.

## Specifying the next hop

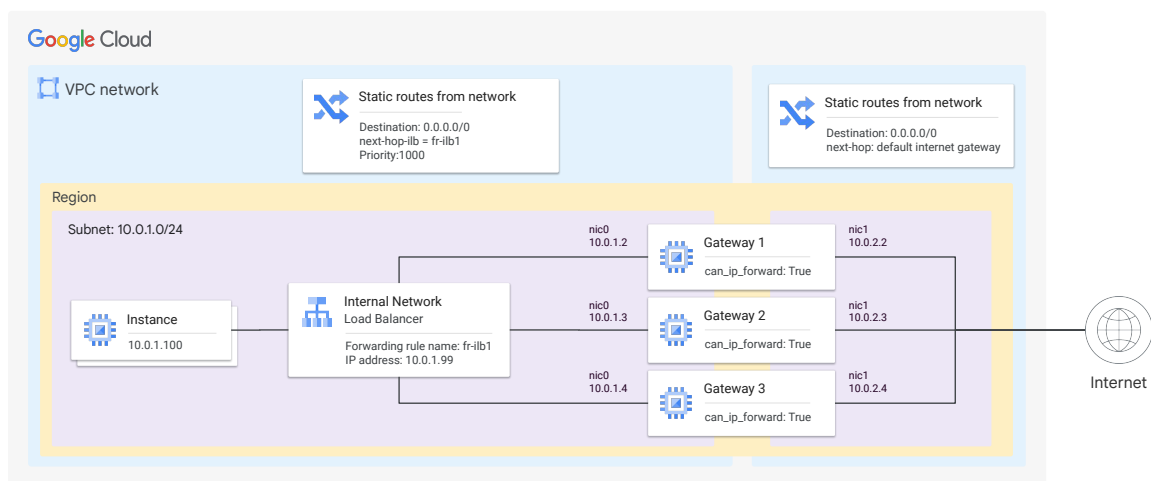
Specification option	Next hop network
Forwarding rule name and the load balancer region	The next hop load balancer and route must be in the same VPC network.
Internal IP address of the forwarding rule	The next hop load balancer can be in the same VPC network as the route or in a peered VPC network.
Forwarding resource link	The forwarding rule's network must match the route's VPC network.

To specify the next hop, you have three choices as shown in the table. The main difference concerns the location of the next hop load balancer.

If the next hop load balancer is in the same VPC network, you can specify the forwarding rule name and the load balancer region. To use a next hop load balancer in a peered VPC network, specify the internal IP address of the forwarding rule.

You can also specify a next hop forwarding rule by its resource link. The forwarding rule's network must match the route's VPC network. The forwarding rule can be located in either the project that contains the forwarding rule's network (a standalone project or a Shared VPC host project) or a Shared VPC service project.

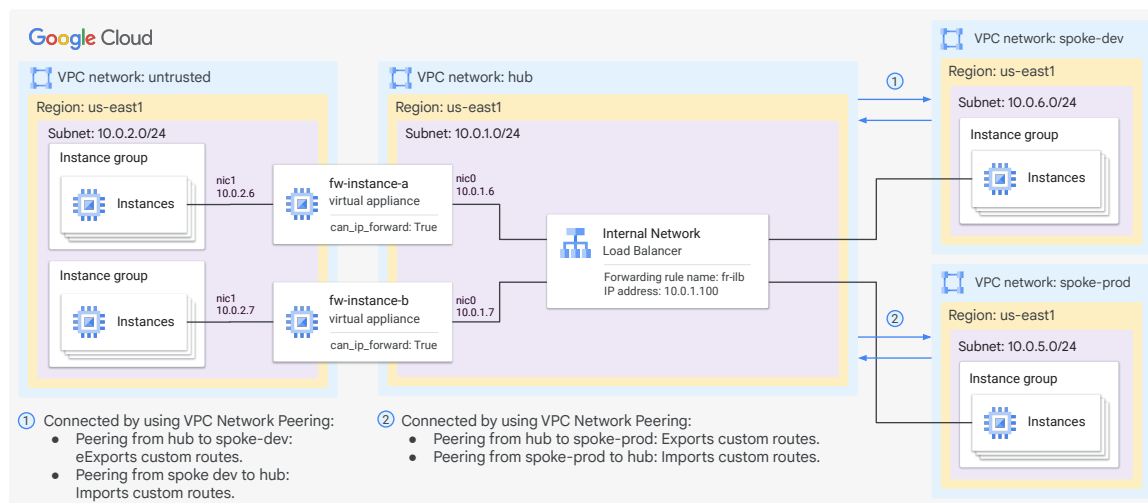
# Next hop to a NAT gateway



Google Cloud

This use case load balances traffic from internal VMs to multiple network address translation (NAT) gateway instances that route traffic to the internet. In this example, an internal Network Load Balancer has next hops configured to three Compute Engine VMs. Each Compute Engine VM has a NAT gateway that runs on it, and has `can_ip_forward` set to `true`. These VMs then forward traffic to the internet. Optionally, you can set up the gateways to apply custom logic to fine-tune access to the internet.

## Using a hub and spoke topology



Google Cloud

In addition to exchanging subnet routes, you can configure VPC Network Peering to export and import custom static and dynamic routes. Custom static routes that have a next hop of the default internet gateway are excluded. Custom static routes that use next-hop internal Network Load Balancers are included.

You can configure a hub-and-spoke topology with your next-hop firewall virtual appliances located in the hub VPC network by doing the following:

1. In the hub VPC network, create an internal Network Load Balancer with firewall virtual appliances as the backends.
2. In the hub VPC network, create a custom static route (with the destination subnet: 10.0.2.0/24), and set the next hop to be the internal Network Load Balancer (10.0.1.100).
3. Use VPC Network Peering to connect the hub VPC network to each of the spoke VPC networks.

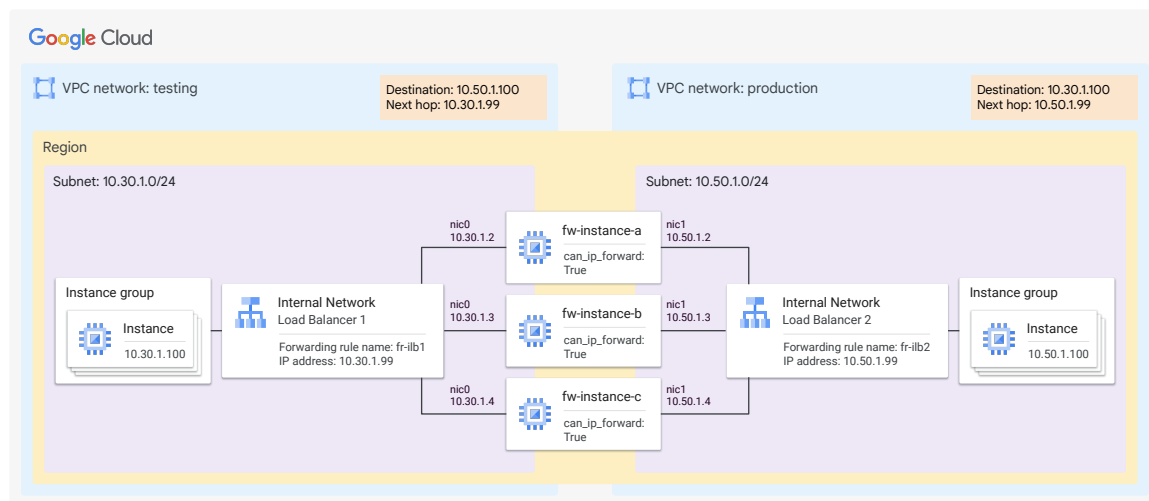
For each peering, configure the hub network to export its custom routes, and configure the corresponding spoke networks to import custom routes (custom static route created in step 2 with destination subnet 10.0.2.0/24). The route with the load balancer next hop is one of the routes that the hub network exports.

Subject to the routing order, the next hop firewall appliance load balancer in the hub

VPC network is available in the spoke networks. If global access is enabled, the firewall appliance is available according to the routing order. If global access is disabled, then resources are only available to requestors in the same region.



# Load balancing to multiple NICs



Google Cloud

Internal Network Load Balancer 1, shown on the left, distributes traffic from the clients to nic0, the primary interface on the backend services. The internal Network Load Balancer 2, shown on the right, distributes traffic from the clients to nic1, the secondary interface on the backend services. The result is that clients can connect to the backend services through nic0 or nic1.

## Benefits

When the load balancer is a next hop for a static route:

- No special configuration is needed within the guest operating systems of the client VMs in the VPC network where the route is defined.
- Client VMs send packets to the load balancer backends through VPC network routing, in a bump-in-the-wire fashion.
- It also provides the same benefits as standalone internal passthrough Network Load Balancer.



When the load balancer is a next hop for a static route, no special configuration is needed within the client VMs. Client VMs send packets to the load balancer backends through VPC network routing, in a bump-in-the-wire fashion.

Using an internal passthrough Network Load Balancer as a next hop for a static route provides the same benefits as a standalone internal passthrough Network Load Balancer. The health check ensures that new connections are routed to healthy backend VMs. By using a managed instance group as a backend, you can configure autoscaling to grow or shrink the set of VMs based on service demand.

## Caveats: Internal Network Load Balancers as next hops

01

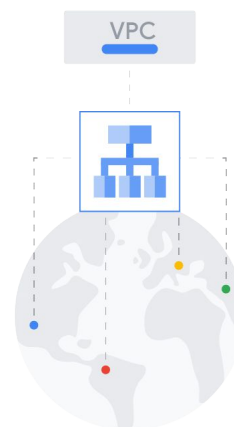
Enable global access on the VPC network so that the next hop is usable from all regions.

02

Even if all health checks fail, the load balancer next hop is still in effect.

03

The load balancer must use an IP address that is unique to a load balancer forwarding rule.



You must enable global access on the VPC network so that the next hop is usable from all regions. Whether the next hop is usable depends on the global access setting of the load balancer. With global access enabled, the load balancer next hop is accessible in all regions of the VPC network. With global access disabled, the load balancer next hop is only accessible in the same region as the load balancer. With global access disabled, packets sent from another region to a route that uses an internal Network Load Balancer next hop are dropped.

Even if all health checks fail, the load balancer next hop is still in effect. Packets processed by the route are sent to one of the next hop load balancer backends. If needed, configure a failover policy.

A next hop internal Network Load Balancer must use an IP address that is unique to a load balancer forwarding rule. Only one backend service is unambiguously referenced.

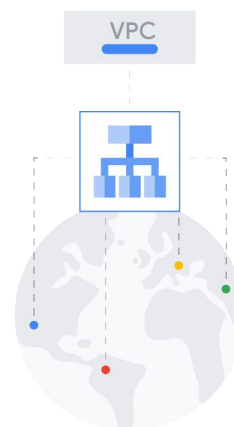
## Caveats: Internal Network Load Balancers as next hops

04

Two or more custom static route next hops with the same destination that use different load balancers are never distributed by using ECMP.

05

To route identical source IP addresses to the same backend, use the client IP, no destination (CLIENT\_IP\_NO\_DESTINATION) session affinity option.




Two or more custom static route next hops with the same destination that use different load balancers are never distributed by using ECMP. If the routes have unique priorities, Google Cloud uses the next hop internal Network Load Balancer from the route with the highest priority. If the routes have equal priorities, Google Cloud still selects just one next hop internal Network Load Balancer.

For packets with identical source IP addresses routed to the same backend, use the client IP, no destination (CLIENT\_IP\_NO\_DESTINATION) session affinity option.

There are some additional caveats for using an internal Network Load Balancer as a next hop, for example, pertaining to the use of network tags. For additional information on this and other caveats, refer to [Additional considerations](#) on the Internal Network Load Balancers as next hops page in the Google Cloud documentation.



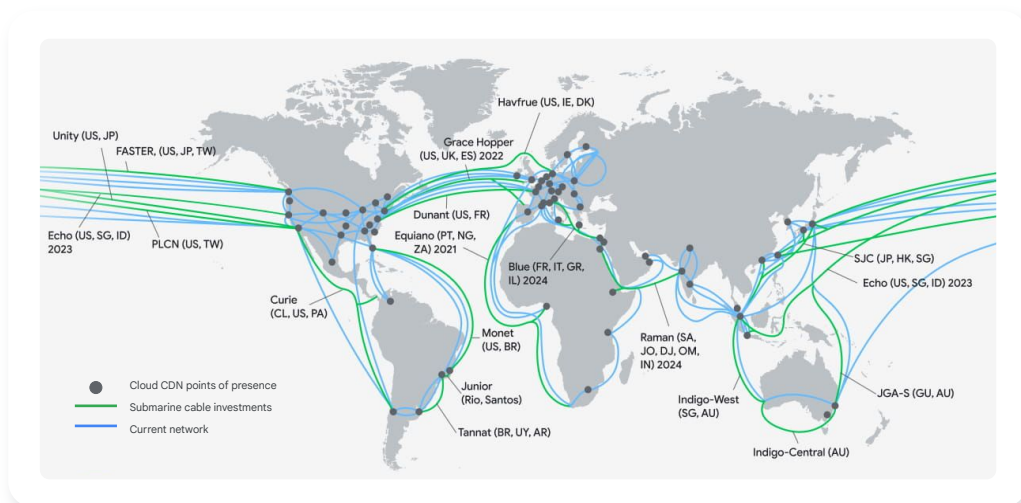
# Today's agenda



- 01 Internal Network Load Balancers as next hops
- 02 [Cloud CDN](#)
- 03 Lab: Defending Edge Cache with Cloud Armor
- 04 Load balancer optimization strategies
- 05 Quiz

Next, let's discuss content delivery network (CDN), which caches content nearer to users. We'll also talk about CDN Interconnect, which lets select third-party content delivery network (CDN) providers establish Direct Interconnect links at edge locations in the Google network.

## Cloud CDN (content delivery network)



Google Cloud

Cloud CDN (content delivery network) caches content at the edges of the Google network. This caching provides faster content delivery to users while reducing transmission costs.

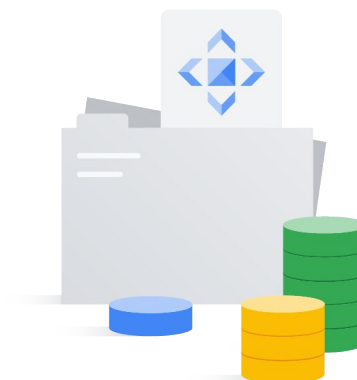
Content can be cached at CDN nodes as shown on this map. There are over 90 of these cache sites spread across metropolitan areas in Asia Pacific, the Americas, and EMEA. For an updated list, please see [Cache locations](#) in the Google Cloud documentation.

For Cloud CDN performance measured by Cedexis, please see these [reports](#) on the Citrix website.

When setting up the backend service of a Application Load Balancer, you can enable Cloud CDN with a checkbox.

# Cloud CDN cache modes

- Cache modes control the factors that determine whether Cloud CDN caches your content.
- Cloud CDN offers three cache modes:
  - `USE_ORIGIN_HEADERS`
  - `CACHE_ALL_STATIC`
  - `FORCE_CACHE_ALL`



Google Cloud

Using cache modes, you can control the factors that determine whether Cloud CDN caches your content.

Cloud CDN offers three cache modes. The cache modes define how responses are cached, whether Cloud CDN respects cache directives sent by the origin, and how cache TTLs are applied.

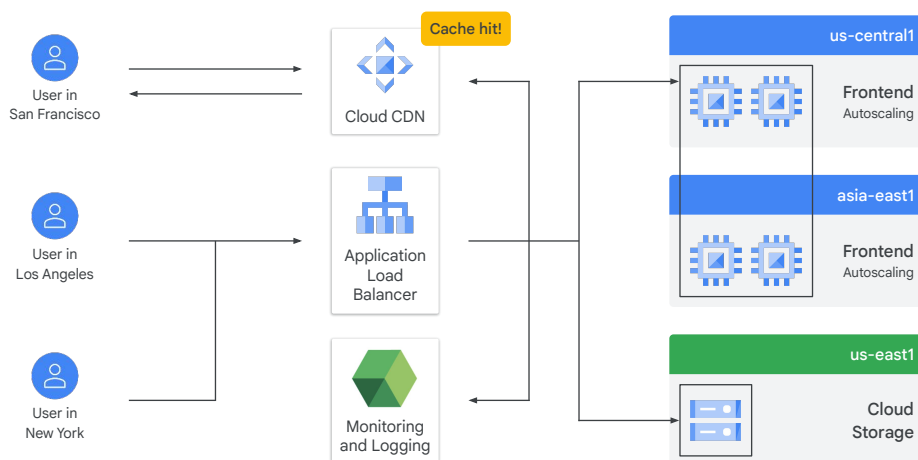
The available cache modes are `USE_ORIGIN_HEADERS`, `CACHE_ALL_STATIC` and `FORCE_CACHE_ALL`.

`USE_ORIGIN_HEADERS` mode requires origin responses to set valid cache directives and valid caching headers.

`CACHE_ALL_STATIC` mode automatically caches static content that doesn't have the no-store, private, or no-cache directive. Origin responses that set valid caching directives are also cached.

`FORCE_CACHE_ALL` mode unconditionally caches responses; overriding any cache directives set by the origin. If you use a shared backend with this mode configured, ensure that you don't cache private, per-user content (such as dynamic HTML or API responses).

# Caching Content with Cloud CDN



Google Cloud

Let's walk through the Cloud CDN response flow with this diagram.

In this example, the Application Load Balancer has two types of backends. There are managed VM instance groups in the us-central1 and asia-east1 regions, and there's a Cloud Storage bucket in us-east1. A URL map decides which backend to send the content to. The Cloud Storage bucket could be used to serve static content and the instance groups could handle PHP traffic.

When a user in San Francisco is the first to access content, the cache site in San Francisco sees that it can't fulfill the request. This situation is called a cache miss. If content is in a nearby cache, Cloud CDN might attempt to get the content from it, for example, if a user in Los Angeles has already accessed the content. Otherwise, the request is forwarded to the Application Load Balancer, which in turn forwards the request to one of your backends.

Depending on the content that is being served, the request will be forwarded to the us-central1 instance group or the us-east1 storage bucket.

If the content from the backend is cacheable, the cache site in San Francisco can store it for future requests. In other words, if another user requests the same content in San Francisco, the cache site might now be able to serve that content. This approach shortens the round trip time and saves the origin server from having to process the request. This is called a cache hit.



For more information on what content can be cached, please refer to [Caching overview](#) in the Google Cloud documentation.

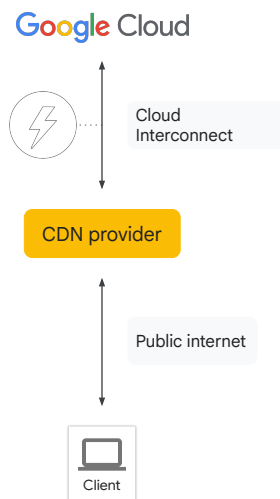
Each Cloud CDN request is automatically logged within Google Cloud. These logs will indicate a “Cache hit” or “Cache miss” status for each HTTP request of the load balancer. You will explore such logs in the next lab.

Cache modes let you control how content is cached.

# CDN Interconnect

CDN Interconnect lets you:

- Select third-party Cloud CDN providers to establish Direct Interconnect links at edge locations in the Google network.
- Direct your traffic from your VPC networks to a provider network.
- Optimize your Cloud CDN cache population costs.



Google Cloud

CDN Interconnect lets select third-party content delivery network (CDN) providers establish Direct Interconnect links at edge locations in the Google edge network. These connections let you direct your traffic from your VPC networks to a CDN provider network. For a complete list of CDN providers, refer to [CDN Interconnect overview](#) in the Google Cloud documentation.

CDN Interconnect lets you connect directly to select CDN providers from Google Cloud. Your network traffic that egresses from Google Cloud through one of these links benefits from the direct connectivity to supported CDN providers.

CDN Interconnect reduces your Cloud CDN cache population costs.

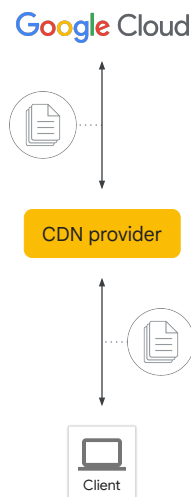
## Typical use cases for CDN Interconnect

01

High-volume egress traffic.

02

Frequent content updates.



If you have a high-volume of egress traffic, consider using CDN Interconnect. You can use the CDN Interconnect links between Google Cloud and selected providers to automatically optimize the egress traffic and save money. If you're populating the Cloud CDN cache locations with large data files from Google Cloud, this optimization can be especially helpful.

Frequent content updates are another typical CDN Interconnect use case. Cloud workloads that frequently update data stored in Cloud CDN cache locations benefit from using CDN Interconnect. The direct link to the Cloud CDN provider reduces latency.

## CDN Interconnect traffic billing

- Ingress traffic is free for all regions.
- Egress traffic rates apply only to data that leaves Compute Engine or Cloud Storage.
- The reduced price applies only to IPv4 traffic.
- Egress charges for CDN Interconnect appear on the invoice as Compute Engine Network Egress via Carrier Peering Network.



Ingress traffic is free for all regions.

Egress traffic rates apply only to data that leaves Compute Engine or Cloud Storage. Egress charges for CDN Interconnect appear on the invoice as *Compute Engine Network Egress via Carrier Peering Network*.

The special pricing for your traffic that egresses from Google Cloud to a CDN provider is automatic. Google works with approved CDN partners in supported locations to accept provider IP addresses. Any data that you send to your allowlisted CDN provider from Google Cloud is charged at the reduced price.

This reduced price applies only to IPv4 traffic. It does not apply to IPv6 traffic.

Intra-region pricing for CDN Interconnect applies only to intra-region egress traffic that is sent to Google-approved CDN providers at specific locations.

## Setting up CDN Interconnect



CDN Interconnect does not require any configuration or integration with Cloud Load Balancing.



**Work with your supported CDN provider to:**




Learn which locations are supported.




Correctly configure your deployment to use intra-region egress routes.

CDN Interconnect does not require any configuration or integration with Cloud Load Balancing. If your CDN provider is already part of the program, you don't need to do anything. Traffic from supported Google Cloud locations to your CDN provider automatically benefits from the direct connection and reduced pricing.

Work with your supported CDN provider to learn what locations are supported. Your supported CDN service provider can also help you correctly configure your deployment to use intra-region egress routes, which cost less than inter-region egress traffic.



# Today's agenda

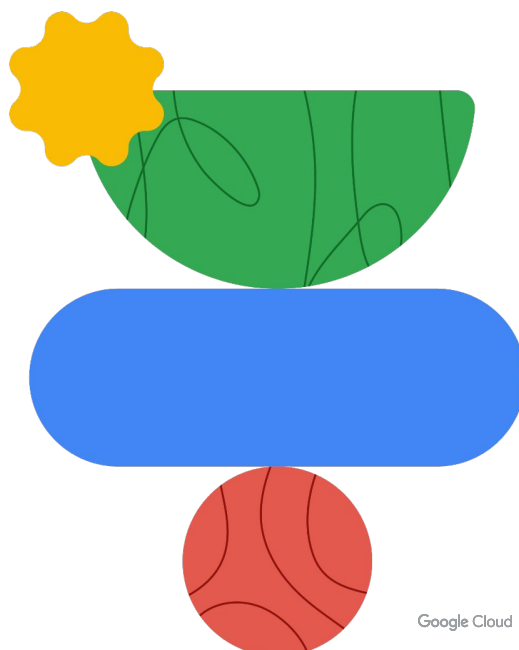


- 01 Internal Network Load Balancers as next hops
- 02 Cloud CDN
- 03 [Lab: Defending Edge Cache with Cloud Armor](#)
- 04 Load balancer optimization strategies
- 05 Quiz

Next, you will explore a lab covering how to use Google Cloud Armor to defend edge cache.

# Lab intro


Defending Edge Cache with  
Cloud Armor




Google Cloud

In this lab, you learn how to perform the following tasks:

- Create and populate a Cloud Storage bucket.
- Create an Application Load Balancer with Cloud CDN.
- Verify the caching of your bucket's content.
- Invalidate the cached content.



# Today's agenda



- 01 Internal Network Load Balancers as next hops
- 02 Cloud CDN
- 03 Lab: Defending Edge Cache with Cloud Armor
- 04 [Load balancer optimization strategies](#)
- 05 Quiz

Next, let's discuss some load balancer optimization strategies.



## Cost optimization strategies: Autoscaling

01 Dynamically adjust resources.

02 Define scaling threshold.

03 Utilize custom metrics.



Google Cloud

Optimizing your cloud load balancer configuration can significantly reduce costs without impacting performance or availability. Here are some key strategies:

### Autoscaling

- **Dynamically adjust resources:** Enable autoscaling to automatically scale backend instances based on real-time traffic demands. This ensures you only pay for the resources you need, eliminating overprovisioning and unnecessary costs.
- **Define scaling thresholds:** Set clear thresholds for scaling up and down to optimize resource utilization and prevent unnecessary scaling events.
- **Utilize custom metrics:** Leverage custom metrics like CPU usage or response time to trigger autoscaling, ensuring resources are allocated based on specific performance indicators.

## Cost optimization strategies: Rightsizing

01

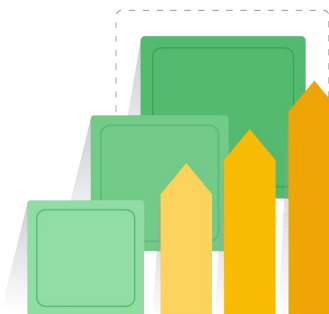
Choose the right load balancer based on your traffic type and requirements.

02

Match resources to workload.

03

Regularly review resource utilization.



### Rightsizing resources:

- Choose the right load balancer type: Select the most appropriate load balancer type (for example, Application, Proxy Network, or Passthrough Network) based on your traffic type and requirements. Avoid using a more expensive type than necessary.
- Match resources to workload: Select the appropriate machine type and size for your backend instances based on their anticipated workload. Overprovisioning resources will lead to higher costs, while underprovisioning can impact performance.
- Regularly review resource utilization: Monitor the CPU, memory, and network utilization of your backend instances and load balancers. If utilization is consistently low, consider downsizing resources for cost savings.

## Cost optimization strategies: Using monitoring and management tools

**01** Use Cloud Monitoring to gain insights into load balancer and backend instance performance.

**02** Leverage Cloud Billing and Cloud cost management tools to track load balancing costs.

**03** Implement cost allocation tags to categorize costs for tracking and optimization.



Google Cloud

### Use Cloud Monitoring and cost management tools:

- **Use Cloud Monitoring:** Use Cloud Monitoring to gain insights into your load balancer and backend instance performance. Analyze metrics like CPU usage, memory consumption, and network bandwidth to identify areas for optimization.
- **Leverage Cloud cost management tools:** Use tools like Cloud Billing and Cloud cost management to track your load balancing costs and identify potential savings opportunities. These tools can provide detailed cost breakdowns by project, service, and resource, helping you identify underutilized resources or areas for potential consolidation.
- **Cost allocation tags:** Implement cost allocation tags to categorize your load balancing costs by department, project, or any other relevant criteria. This allows for more granular cost tracking and facilitates cost optimization efforts.

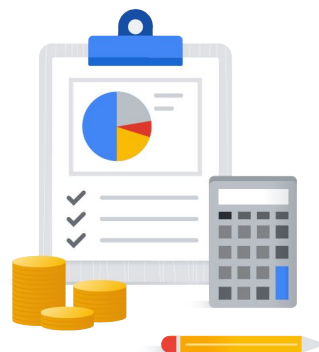
## Other cost optimization strategies

**01** Schedule downtimes for non-critical workload during low traffic time.

**02** Use reserved instances for predictable workloads.

**03** Use spot instances for non-critical workloads.


**04** Integrate Cloud CDN with a load balancer to reduce load on backend instances.




### Additional strategies:

- **Scheduled downtimes:** Consider scheduling downtimes for non-critical workloads during periods of low traffic to reduce costs.
- **Reserved instances:** Use reserved instances for predictable workloads to obtain significant discounts on load balancer resources.
- **Spot instances:** Explore using spot instances for non-critical workloads to take advantage of discounted compute resources.
- **Cloud CDN integration:** Integrate Cloud CDN with your load balancer to reduce the load on your backend instances and potentially reduce load balancer costs.

Remember, cost optimization is an ongoing process. By implementing these strategies and continuously monitoring your cloud resources, you can achieve significant cost savings while ensuring your cloud infrastructure remains efficient and scalable.



# Today's agenda



- 01 Internal Network Load Balancers as next hops
- 02 Cloud CDN
- 03 Lab: Defending Edge Cache with Cloud Armor
- 04 Load balancer optimization strategies
- 05 [Quiz](#)

## Quiz | Question 1

### Question

When you use the internal IP address of the forwarding rule to specify an internal Network Load Balancer next hop, the load balancer can only be:

- A. In the same VPC network as the next hop route.
- B. In the same VPC network as the next hop route or in a peered VPC network.
- C. In the same subnet as the next hop route.
- D. In the same subnet as the next hop route or a shared VPC network.

## Quiz | Question 3

### Question

CDN Interconnect provides:

- A. A direct connection between your origin servers and Google's Cloud Load Balancing service.
- B. A direct peering connection between third-party content delivery networks (CDNs) and Google's edge network.
- C. A private connection between your on-premises network and Google Cloud
- D. A virtual private network (VPN) tunnel between your VPC network and Google's global network.

## Quiz | Question 3

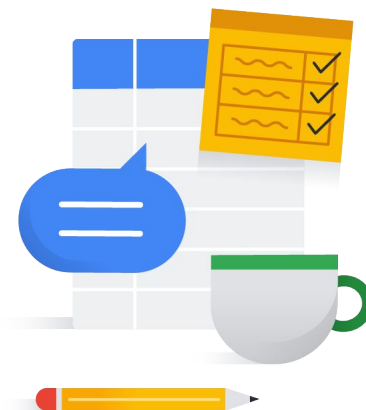
### Question

Which of the following best practices help optimize load balancing cost?

- A. Choosing a load balancer based on your traffic type.
- B. Choosing a load balancer type that closely matches your traffic patterns.
- C. Implementing a caching layer with a content delivery network (CDN).
- D. Increasing your timeout periods for load balancer health checks.



## Debrief



Google Cloud

In this module, we began with an overview of load balancing in Google Cloud. We continued with a discussion of hybrid load balancing. You learned that hybrid load balancing can be used to migrate your workloads into Google Cloud or to provide multiple platforms for your workloads. We covered the load balancers that support hybrid load balancing and an overview of the components that you must configure.

We then talked about using traffic management with your load balancers. You learned which load balancers support traffic management features. You were introduced to the URL map, where you configure traffic management features. We walked through a simple example of traffic management. In the example, you saw how to configure a URL map to match incoming traffic and specify where the traffic should be sent.

You then applied what you learned in a lab exercise.

Next, we covered using internal Network Load Balancers as next hops. You learned about some of the major use cases, and some simple topologies were shown.

We continued by discussing using Cloud CDN to get content to your clients faster. You also learned about CDN Interconnect to direct traffic from your VPC networks to a supported CDN provider network. You also learned how CDN Interconnect optimizes your Cloud CDN cache population costs.

Finally, you took a brief quiz to test your knowledge.



Thank you.

