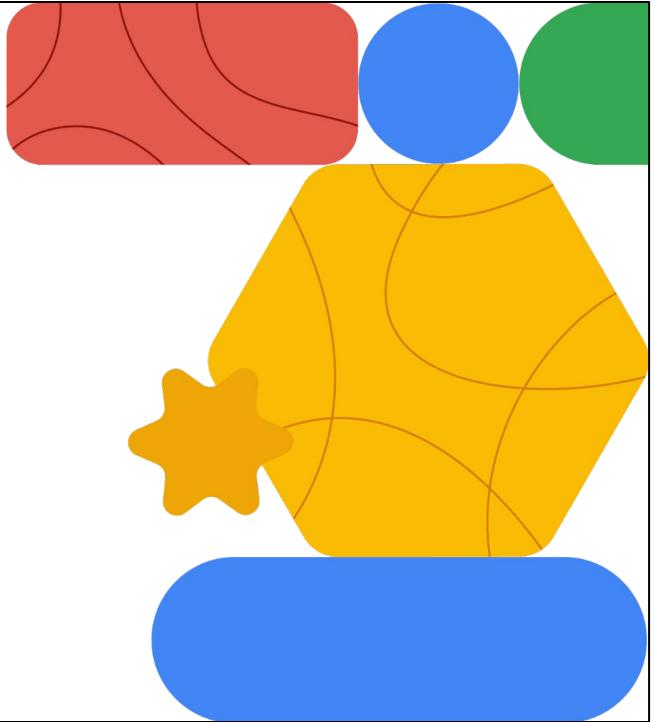
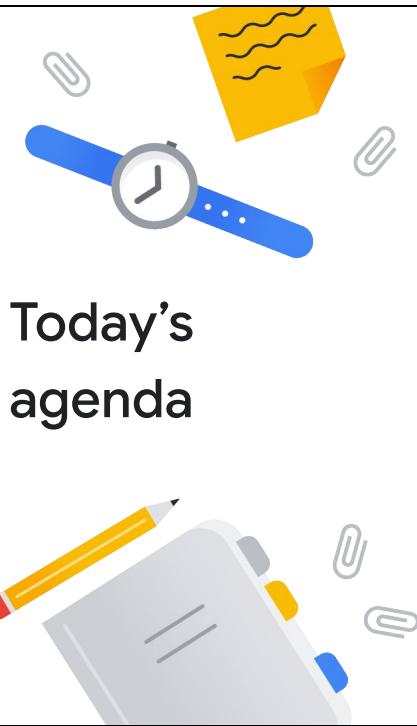


Networking in Google Cloud

Hybrid Load Balancing and Traffic Management



Welcome to the Hybrid Load Balancing and Traffic Management module.



Today's agenda

- 01 Load balancing
- 02 Hybrid load balancing
- 03 Traffic management
- 04 Lab: Configuring Traffic Management with a Load Balancer
- 05 Quiz

Google Cloud

In this module, we will cover the topics listed on the screen.

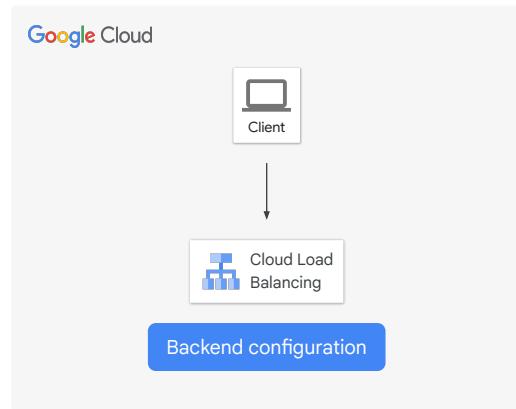
We'll begin with an overview of Cloud Load Balancing. We will continue with a discussion of hybrid load balancing. In other words, load balancing between Google Cloud, other public clouds, and on-premises environments.

We will follow with a discussion on traffic management, which provides enhanced features to route traffic based on criteria that you specify. After that, you will apply what you've learned in a traffic management lab exercise.

Let's get started.

Overview of Cloud Load Balancing

- Cloud Load Balancing receives client traffic.
- The backend can be a backend service or a backend bucket.
- Backend configuration defines:
 - How traffic is distributed.
 - Which health check to use.
 - If session affinity is used.
 - Which other services are used (such as Cloud CDN or Identity-Aware Proxy).



A load balancer, as the name suggests, balances load across multiple instances of your applications.

Cloud Load Balancing receives client traffic. This traffic can be external or internal, depending on the load balancer you use. A backend configuration distributes requests to healthy backends.

Some load balancers also support backend buckets.

One or more backends must be connected to the backend service or backend bucket.

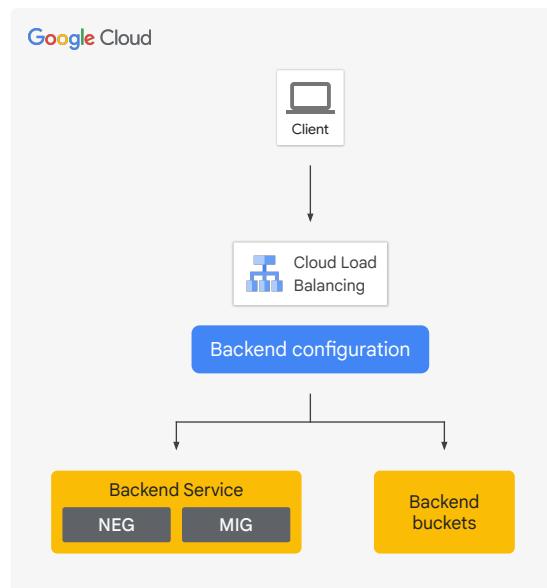
Backend configuration defines:

- How traffic is distributed.
- Which health check to use.
- If session affinity is used.
- Which other services are used (such as Cloud CDN or Identity-Aware Proxy).

Backend configuration

Cloud Load Balancing can route traffic to:

- Backend Services:
 - Managed instance groups: a group of virtual machines created from a template.
 - Network endpoint groups (NEGs): a group of services or workloads.
- Cloud Storage backend buckets.

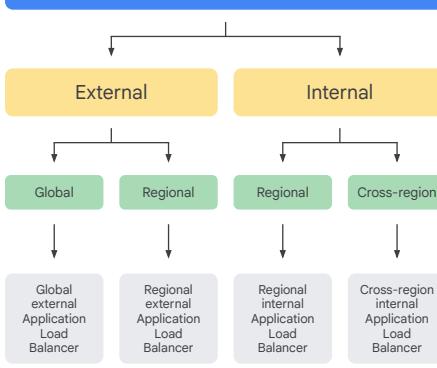


Cloud Load Balancing can route traffic to either a backend service or a backend bucket. The backend services define how to handle the traffic. For example, backend services define how the traffic is distributed, which health check to use, and if session affinity is used. Backend services also define which other Google Cloud services to use, such as Cloud CDN or Identity-Aware Proxy. On the other hand, backend buckets direct incoming traffic to Cloud Storage buckets. Backend buckets are useful in serving static content. We will discuss this in more detail in the upcoming section.

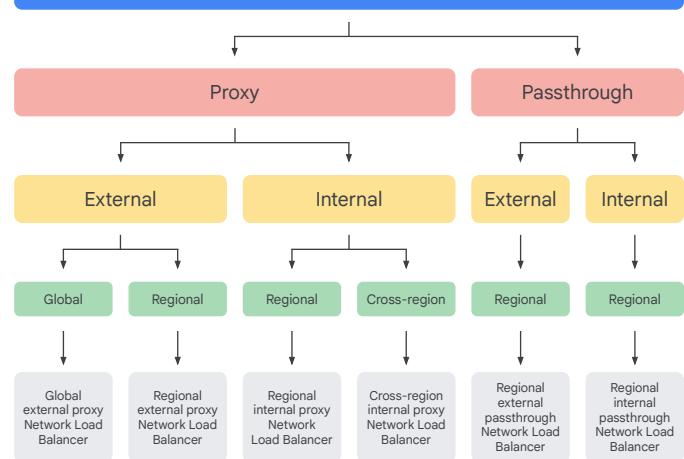
Some of the backend services include a managed instance group, or a network endpoint group (NEG). In this module, we are going to look at some special features related to network endpoint groups.

Types of load balancers

Application Load Balancer (HTTP / HTTPS)



Network Load Balancer (TCP / UDP / other IP protocols)



Google Cloud

Google Cloud Platform offers a range of load balancing solutions that can be classified based on the OSI model layer they operate at and their specific functionalities.

Application Load Balancers

These load balancers are designed to handle HTTP and HTTPS traffic, making them ideal for web applications and services that require advanced features like content-based routing and SSL/TLS termination. Application Load Balancers operate as reverse proxies, distributing incoming traffic across multiple backend instances based on rules you define. They are highly flexible and can be configured for both internet-facing (external) and internal applications.

Network Load Balancers

Network Load Balancers operate at the transport layer and efficiently handle TCP, UDP, and other IP protocols. They can be further classified into two types:

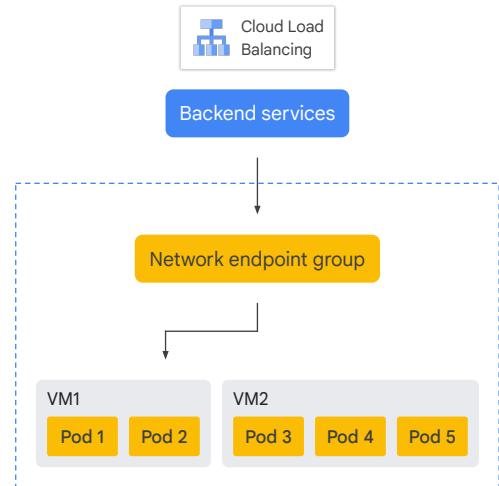
Proxy Load Balancers: These also function as reverse proxies, terminating client connections and establishing new ones to backend services. They offer advanced traffic management capabilities and support backends located both on-premises and in various cloud environments.

Passthrough Load Balancers: Unlike proxy load balancers, these do not modify or

terminate connections. Instead, they directly forward traffic to the backend while preserving the original source IP address. This type is well-suited for applications that require direct server return or need to handle a wider range of IP protocols.

Network endpoint groups (NEGs)

- A NEG is a configuration object that specifies a group of backend endpoints or services.
- A common use case for this configuration is deploying services in GKE.
- There are five types of NEGs:
 - Zonal
 - Internet
 - Serverless
 - Private Service Connect
 - Hybrid connectivity



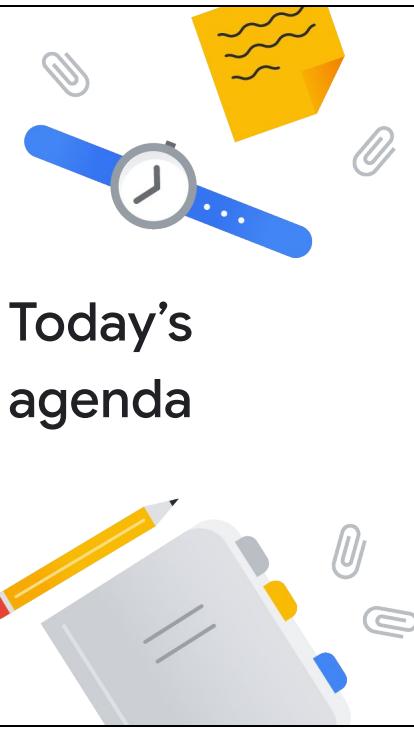
A network endpoint group (NEG) is a configuration object that specifies a group of backend endpoints or services. A common use case for this configuration is deploying services in containers, as in Google Kubernetes Engine. The load balancer must be able to select a pod from the container, as shown in the example. You can also distribute traffic in a granular fashion to workloads and services that run on your backend hosts.

You can use NEGs as backends for some load balancers and with Traffic Director. There are five types of NEGs:

- **Zonal NEGs** define how endpoints should be reached, whether they are reachable, and where they are located. **A zonal NEG** contains one or more endpoints that can be Compute Engine virtual machines (VMs) or services that run on the VMs. Each endpoint is specified either by an IP address or an IP:port combination.
- **An internet NEG** contains a single endpoint that is hosted outside of Google Cloud. This endpoint is specified by hostname FQDN:port or IP:port.
- **A serverless NEG** is a backend that points to a Cloud Run, App Engine, Cloud Functions, or API Gateway service that resides in the same region as the NEG.
- **A Private Service Connect NEG** contains a single endpoint. That endpoint resolves to either a Google-managed regional API endpoint or a managed service published by using Private Service Connect.
- **A hybrid connectivity NEG** points to Traffic Director services that run outside

- of Google Cloud (on-premises or other public cloud backends). The focus in this module is on hybrid connectivity NEGs.

For more information on using NEGs, and a complete list of supported load balancers, please refer to the [Network endpoint groups overview](#) in the Google Cloud documentation.



Today's agenda

- 01 Load balancing
- 02 [Hybrid load balancing](#)
- 03 Traffic management
- 04 Lab: Configuring Traffic Management with a Load Balancer
- 05 Quiz

Google Cloud

Let's next discuss hybrid load balancing.

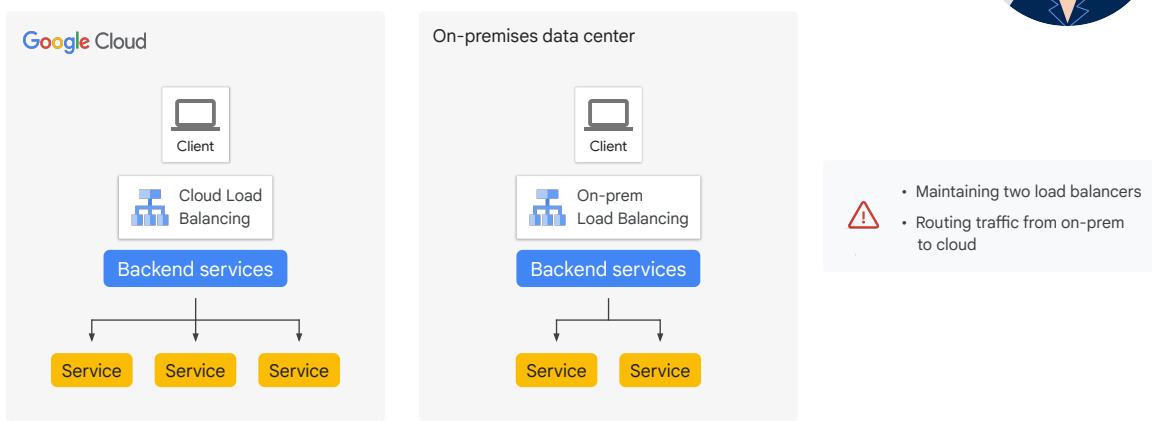
Hybrid connectivity and load balancing

- A hybrid strategy lets you extend Cloud Load Balancing to workloads that run on your existing infrastructure outside of Google Cloud.
- This strategy could be:
 - Permanent to provide multiple platforms for your workloads.
 - Temporary as you prepare to migrate your internal or external workload to Google Cloud.

A hybrid load balancing lets you extend Cloud Load Balancing to workloads that run on your existing infrastructure outside of Google Cloud. A hybrid strategy is a pragmatic solution for you to adapt to changing market demands and incrementally modernize the backend services that run your workloads. You can create a hybrid deployment to enable migration to a modern cloud-based solution or a permanent fixture of your organization IT infrastructure.

Next, let's look at a few general usecases of hybrid load balancing.

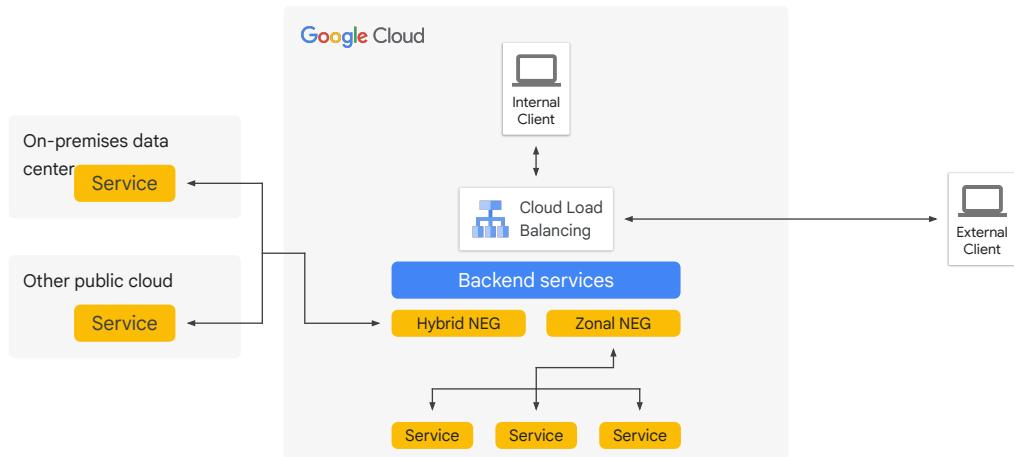
Use case: Complexity distributing load in hybrid environment



Jie, a Cloud Network Engineer, needs to modernize the Cymbal Corporation IT infrastructure to improve application performance and scalability.

Currently, Cymbal runs a mix of on-premises applications and applications deployed in Google Kubernetes Engine (GKE). Traffic is often unpredictable, with sudden surges during sales or promotions overwhelming their on-premises servers. Izumi knows they need a way to handle these spikes without compromising performance or spending a fortune on excess capacity that sits idle most of the time. Additionally, managing two separate load balancing solutions for the on-premises and GKE environments is becoming increasingly cumbersome.

Use case: Jie can benefit from hybrid load balancing



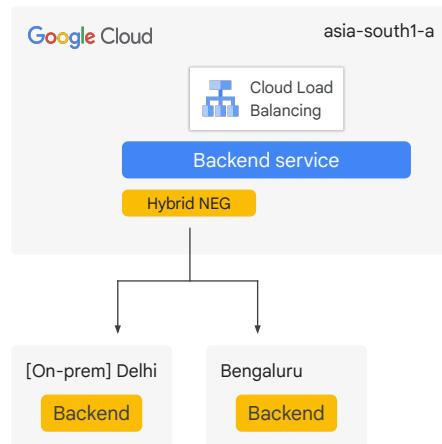
In this example, traffic from clients on the public internet enters your private on-premises environment, and traffic from another public cloud provider enters through a Cloud load balancer. The load balancer also gets requests from internal clients.

The load balancer sends requests to the services that run your workloads. These services are the load balancer endpoints, and they can be located inside or outside of Google Cloud. You configure a load balancer backend service to communicate to the external endpoints by using a hybrid NEG. The external environments can use Cloud Interconnect or Cloud VPN to communicate with Google Cloud. The load balancer must be able to reach each service with a valid IP address:Port combination.

The example shows a load balancer backend service with a hybrid and a zonal NEG. The hybrid NEG connects to endpoints that are on-premises and in other public clouds. The zonal NEG points to Cloud Endpoints in the same subnet and zone.

Configuring backend services outside of Google Cloud

- Configure one or more hybrid connectivity network endpoint groups (NEGs):
 - Add the IP address
 - Specify a Google Cloud zone
 - Add a health check to the NEG.
- Add the hybrid connectivity NEGs to a hybrid load balancer backend service.



A hybrid load balancer requires special configuration only for the backend service. The frontend configuration is the same as any other load balancer.

To configure backend services outside of Google Cloud, first configure one or more hybrid connectivity network endpoint groups (NEG).

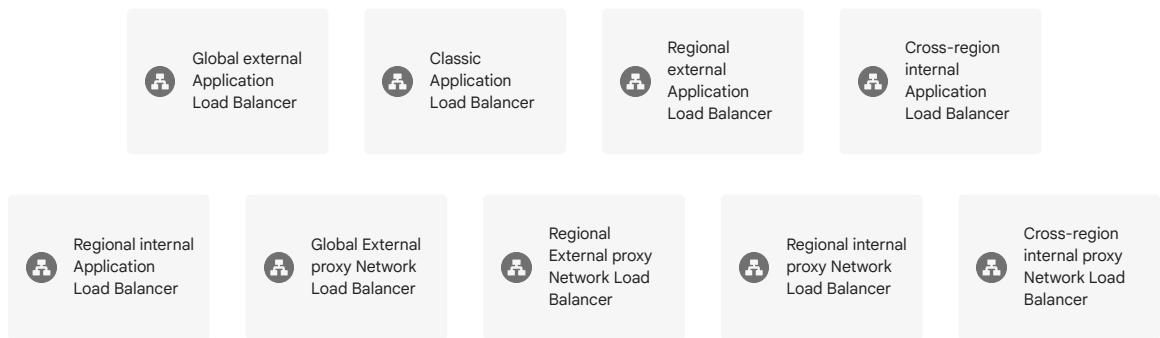
Add each non-Google Cloud network endpoint IP:Port combination to a hybrid connectivity network endpoint group (NEG). Ensure that the IP address and port are reachable from Google Cloud. For hybrid connectivity NEGs, you set the network endpoint type to `NON_GCP_PRIVATE_IP_PORT`.

Create the NEG in a Google Cloud zone that is as close as possible to your other environment. For example, if you're hosting a service in an on-premises environment in Bengaluru, India, you can place the NEG in the `asia-south1-a` Google Cloud zone, as shown in the example.

Next, add a health check to the NEG.

Add the hybrid connectivity NEGs to a hybrid load balancer backend. A hybrid connectivity NEG must only include endpoints outside Google Cloud. Traffic might be dropped if a hybrid NEG includes endpoints for resources within a Google Cloud VPC network.

Types of load balancers that support hybrid load balancing



You can use hybrid load balancing with the following:

- Global external Application Load Balancer
- Classic Application Load Balancer
- Regional external Application Load Balancer
- Cross-region internal Application Load Balancer
- Regional internal Application Load Balancer
- External proxy Network Load Balancer (global and regional)
- Regional internal proxy Network Load Balancer
- Cross-region internal proxy Network Load Balancer

You choose a load balancer depending on your needs, such as where the clients and workloads are located.

Caveats: Hybrid load balancing

01

To create, delete, or manage a load balancer with mixed zonal and hybrid connectivity NEGs backends in a single backend service, use the Google Cloud CLI or the REST API.

02

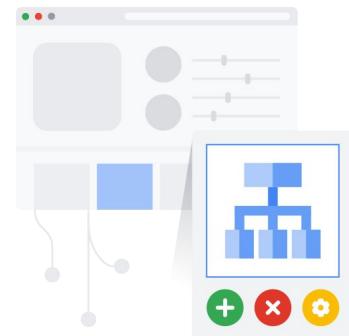
Regional dynamic routing and static routes are not supported.

03

The internal Application Load Balancer and hybrid connectivity must be configured in the same region.

04

Ensure that you also review the security settings on your hybrid connectivity configuration.

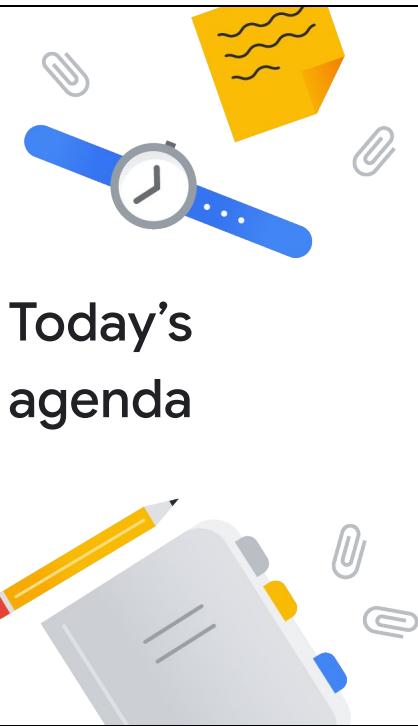


To create, delete, or manage a load balancer with mixed zonal and hybrid connectivity NEGs backends in a single backend service, you must use the Google Cloud CLI or the REST API.

Regional dynamic routing and static routes are not supported. The Cloud Router used for hybrid connectivity must be enabled with global dynamic routing.

The internal Application Load Balancer and hybrid connectivity must be configured in the same region. If they are configured in different regions, you might see backends as healthy, but client requests will not be forwarded to the backend.

Ensure that you also review the security settings on your hybrid connectivity configuration. Currently, HA Cloud VPN connections are encrypted by default, using IPsec encryption. Cloud Interconnect connections are not encrypted by default. For more details, go to [Encryption in Transit in Google Cloud](#) on the Google Cloud website.



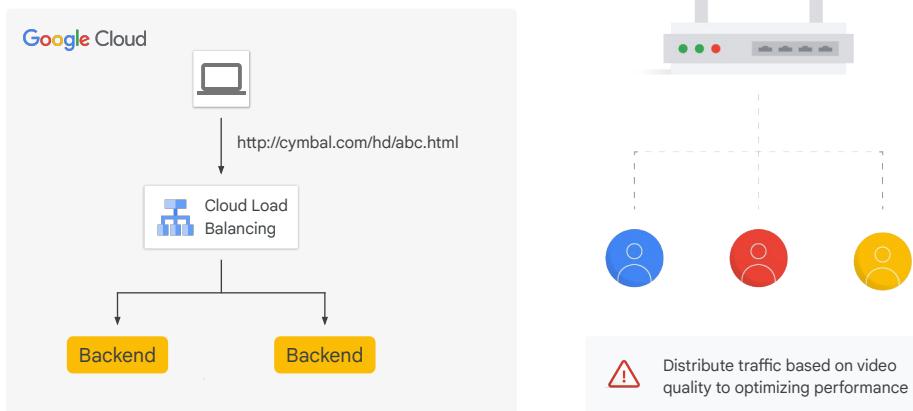
Today's agenda

- 01 Load balancing
- 02 Hybrid load balancing
- 03 [Traffic management](#)
- 04 Lab: Configuring Traffic Management with a Load Balancer
- 05 Quiz

Google Cloud

Traffic management is key to ensuring optimal network performance and user experience. In this section, we'll delve into a real-world use case and its implementation.

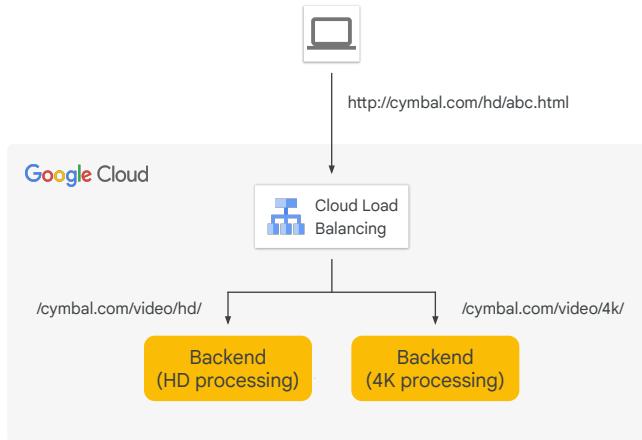
Use case: Distribute traffic by using URL map



Bola, a Cloud Network Engineer at Cymbal Corporation, is responsible for maintaining their video streaming platform. The website accesses the Cymbal store infrastructure using Cloud Load Balancing. To optimize performance and costs, they want to route user traffic to different server backends based on the requested video quality (e.g., 4K versus HD). What should Bola do?

Traffic management

- Traffic management provides enhanced features to route load balancer traffic based on criteria that you specify.
- With traffic management, you can:
 - Direct traffic to a backend based on HTTPS parameters.
 - Perform request-based and response-based actions.
 - Use traffic policies to fine-tune load balancing behavior.



Bola can benefit from Traffic management . Traffic management provides enhanced features to route load balancer traffic based on criteria that you specify.

With traffic management, you can:

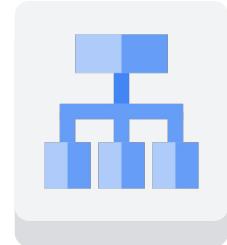
- Implement traffic steering based on HTTPS parameters, such as the host, path, headers, and other request parameters.
- Perform request-based and response-based actions, such as redirects and header transformations.
- Use traffic policies to fine-tune load balancing behavior, such as retry policies, request mirroring, and cross-origin resource sharing (CORS).

The traffic features that are available can vary per load balancer. Check the Google Cloud documentation for details, for example, [Traffic management overview for a classic Application Load Balancer](#), [Traffic management overview for global external Application Load Balancers](#), and [Traffic management overview for regional external Application Load Balancers](#).

Recall that, in addition to traffic management, Cloud Load Balancing offers backend services like health checks, session affinity, balancing mode, and capacity scaling.

Supported load balancers

- These load balancers support traffic management features:
 - Global external Application Load Balancer
 - Global external classic Application Load Balancer
 - Regional external Application Load Balancer
 - Internal Application Load Balancer
- Other load balancers have access only to traffic features that are available in backend services, such as balancing mode and session affinity.
- For a complete list of features supported by each load balancer, refer to [Routing and traffic management](#).

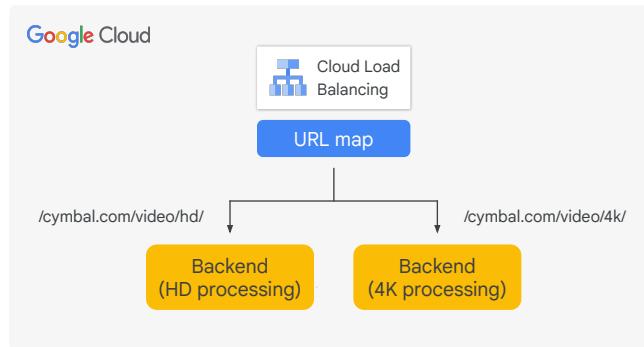


These load balancers support traffic management: global external Application Load Balancer, global external classic Application Load Balancer, internal Application Load Balancer and the regional external Application Load Balancer. Other load balancers have access to only traffic features available in backend services, such as balancing mode and session affinity.

Not all load balancers support all traffic management features. For a complete list of traffic management features supported for each load balancer, refer to [Routing and traffic management](#) in the Google Cloud documentation.

URL map

- The URL map contains rules that define the criteria to use to route incoming traffic to a backend service.
- Traffic management features are configured in a URL map.
- You can choose between the simple and the advanced host mode.

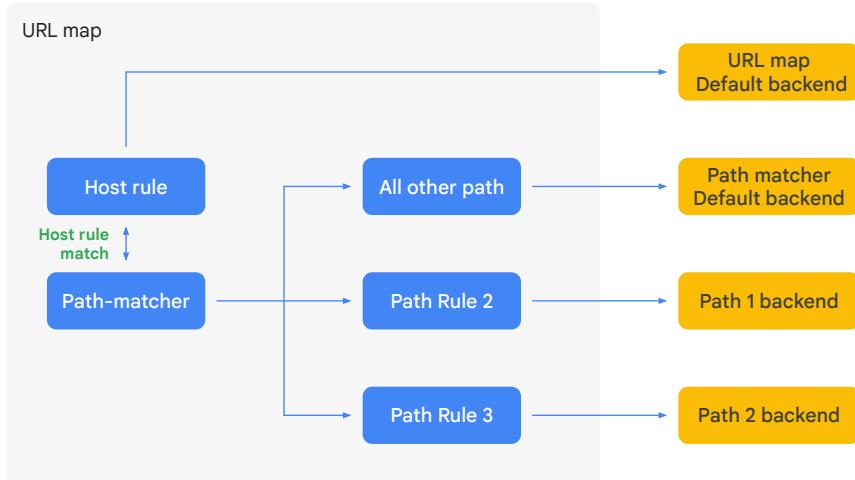


The URL map contains rules that define the criteria to use to route incoming traffic to a backend service or backend bucket. Traffic management features are configured in a URL map. In other words, the load balancer uses the URL map to determine where to route incoming traffic. When you configure routing, you can choose between the following modes:

- Simple host and path rule
- Advanced host, path, and route rule

Each URL map can contain only one mode or the other mode.

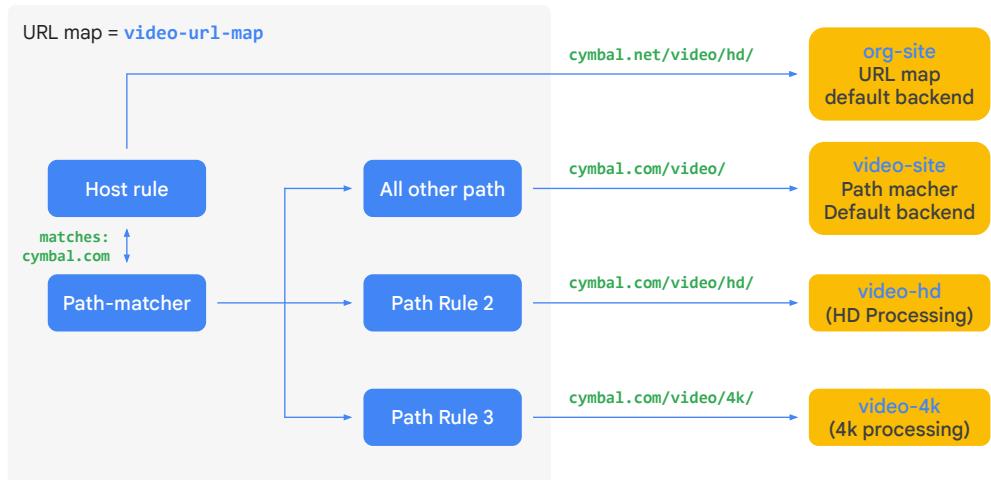
URL map workflow



A request is first evaluated based on the host rule. If the domain matches a defined host rule, the system uses its associated path matcher to further refine routing. Each host rule consists of a list of one or more hosts and a single path matcher (pathMatcher). If no hostRules are defined, the request is routed to the defaultService. The request's path is compared against available path rules, with the longest, most specific match taking priority. Path rules are evaluated on a longest-path-matches-first basis. You can specify the path rules in any order.

Once the best match is found, the request is sent to the corresponding backend service. If no specific host and path rules apply, the request is handled by the default backend service. This setup allows you to create custom routing rules, like sending video-related requests to a dedicated service while directing general traffic to a different backend.

Bola can use URL maps to distribute traffic



Going back to the scenario, Bola can use the URL map to distribute traffic. The host rule is cymbal.com, which means any host other than cymbal.com (example, cymbal.org, cymbal.net) are directed to the default service. Once the host name, cymbal.com matches, the URL is matched for a Path rule:

- The default backend service is video-site.
- Requests with the exact URL path /video/hd/ are directed to the video-hd backend service.
- Requests with the exact URL path /video/4k/ are directed to the video-4k backend service.

A simple URL map

```
defaultService:https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/org-site
fingerprint: mfyJIT7Zurs=
hostRules:
- hosts:
  - '*'
  pathMatcher: pathmap
name: video-org-url-map
pathMatchers:
- defaultService:https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/video-site
  name: pathmap
  pathRules:
  - paths:
    - /video/hd
    service: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/video-hd
  - paths:
    - /video/4K
    service: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/video-4k
```

On this slide, you see a URL map that routes video traffic to the example we covered in the previous slides. This example is shown by using a YAML file. You can also use the Google Cloud console to configure URL maps.

Let's look at how this example works.

Default Service: The `defaultService` defines a service where traffic should be routed when no matching URL rule is found. You must specify a `defaultService` or a `backendBucket`.

The `hostRules` defines a list of hostnames that are processed by this rule. In this example, there's only one item in the list, which means that only one host rule is defined. This host rule contains an asterisk (*). The asterisk is a wildcard, which matches all hosts.

To see where to find the matching logic to use, look at the value of **`pathMatcher`**. For this host rule, `pathMatcher` is set to `pathmap`. Here's a `pathMatchers` list, which contains a list of path matching rules. The only element in this list is `pathmap`. Each match rule defines logic to process the traffic that is sent to the backend service.

In this example, there are two sets of paths. One `paths` list defines valid URL paths for the `video-hd`. The other `paths` list defines valid URL paths for the `video-4k`. If the URL contains a match for one of these paths lists, the load balancer routes the traffic to the corresponding service.

If the traffic contains a path that matches none of the paths lists, then it's sent to the default backend service, `video-site`. In other words, the traffic is sent to the service denoted by `pathMatchers/defaultService`. For additional details, refer to the [documentation](#).

Advanced routing mode

- The advanced routing mode:
 - Can choose a rule based on a defined priority.
 - Includes additional configuration options.
 - Uses route rules instead of path rules.
 - Can't include any path rules if a URL map includes route rules.

The advanced routing mode can choose a rule based on a defined priority and includes additional configuration options. Instead of path rules, advanced routing uses route rules.

Each URL map can include either simple or advanced rules, but not both.

An advanced routing mode URL map

```
defaultService: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-a
hostRules:
- hosts:
  - '*'
  pathMatcher: matcher1
name: lb-map
pathMatchers:
- defaultService: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-b
  name: matcher1
  routeRules:
- matchRules:
  - prefixMatch: ''
  routeAction:
    weightedBackendServices:
    - backendService: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-a
      weight: 95
    - backendService: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-b
      weight: 5
```

This URL map contains rules that route 95% of the traffic to service-a, and 5% of the traffic is routed to service-b.

The example shows a YAML implementation of an advanced routing mode. You can also use the Google Cloud console to configure URL maps.

Let's look at how this example works.

When no matching host rule is found, the `defaultService` defines a service to use. The field `defaultService` is required.

The `hostsRules` works the same way as for simple routing mode. As in the previous example, this host rule uses the asterisk to match all hosts. Because `pathMatcher` is set to `matcher1`, `/pathMatchers/matcher1` defines the matching logic.

Advanced routing mode: pathMatchers

```
pathMatchers:  
- defaultService:  
  https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-b  
    name: matcher1  
    routeRules:  
      - matchRules:  
        - prefixMatch: ''  
        routeAction:  
          weightedBackendServices:  
            - backendService:  
              https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-a  
                weight: 95  
            - backendService:  
              https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-b  
                weight: 5
```

/pathMatchers/matcher1 contains a list of routeRules. The routeRules contain a list of one or more matchRules and a routeAction. When URLs satisfy the matchRules, their traffic is processed by the routeAction.

In this example, there's only one item in matchRules, where prefixMatch equals an empty string. The prefixMatch condition matches the URL path prefix; URLs that start with the same string match. In the example, the prefixMatch is the empty string, which matches all URLs. In other words, all URLs trigger this match rule, and the routeAction is applied.

The routeAction defines how the traffic is routed. In the example, the routeAction is set to weightedBackendServices.

weightedBackendServices is a list of backend services. A weight value is specified for each backend service; representing a percentage of the total traffic. 95% of the traffic is sent to service-a, and 5% of the traffic is sent to service-b.

The routeAction can also define traffic policies, such as retry policies and CORS.

For a complete list of routeAction values, refer to the Google Cloud documentation for the load balancer that you're using.

defaultService

```
defaultService: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-a
hostRules:
- hosts:
  - '*'
  pathMatcher: matcher1
name: lb-map
pathMatchers:
- defaultService: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-b
  name: matcher1
  routeRules:
  - matchRules:
    - prefixMatch: ''
    routeAction:
      weightedBackendServices:
      - backendService: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-a
        weight: 95
      - backendService: https://www.googleapis.com/compute/v1/projects/PROJECT_ID/global/backendServices/service-b
        weight: 5
```

Used if there's no matching host rule.

Used if there's a matching host rule but there's no matching route rule.

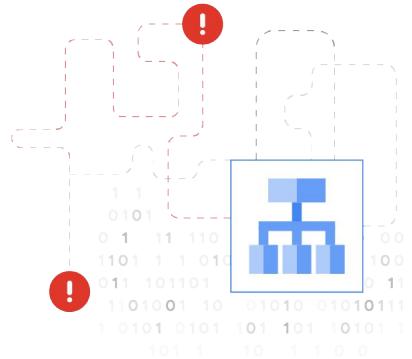
In this example, you might notice that there are two `defaultService` key-value pairs. One `defaultService` is associated with the `hostRules`, and the other is associated with the `routeRules`.

If there's no matching host rule, the first `defaultService` is used.

If there's no matching route rule, the second `defaultService` is used.

Caveats: Traffic routing

- Not all load balancers support all traffic management features.
- Wildcards are supported, but only after a forward slash (/), for example:
 - Valid: /video/*
 - Invalid: /video*
- Substring matching and regular expressions are not supported, for example:
 - /videos/hd* doesn't match /videos/hd-pdq.
 - /videos/* does match /videos/hd-pdq.



Not all load balancers support all traffic management features. For a complete list of traffic management features supported for each load balancer, refer to [Routing and traffic management](#) in the Google Cloud documentation.

Wildcards are supported, but only after a forward slash (/). For example, /video/* is valid and /video* is invalid.

Rule matching does not use regular expressions or substring matching. For example, /videos/hd/* does not match /videos/hd-pdq, because -pdq is a substring and also because it comes after the forward slash. /videos/* matches /videos/hd-pdq.

Let's ask Gemini



You are a patient and friendly Google Cloud technical support engineer who specializes in cloud networking and responds to customer's questions.

What is the most **reliable Google Cloud load balancer**?



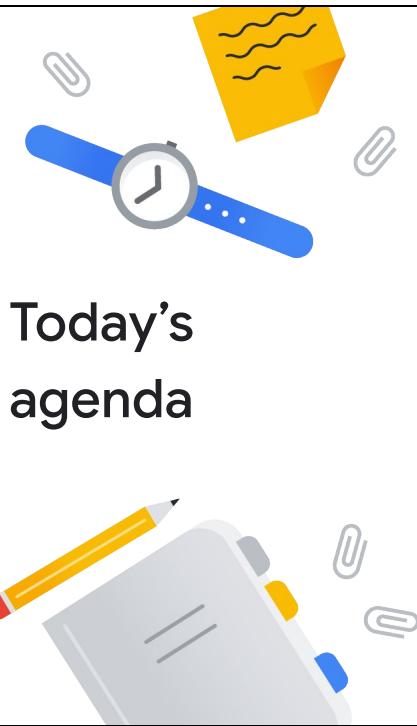
As a Google Cloud technical support engineer who specializes in cloud networking, I can confirm that the most reliable load balancer in Google Cloud is **Network Load Balancing (NLB)**.

*Gemini responses may vary

You can also use Gemini to learn more. One trick to get a better response is to assign Gemini a role.

Prime the model to assume a specific role (also known as a persona). Adding a role is not always necessary but can enforce a certain level of expertise when generating a response, improve performance, and tailor its communication style. This technique is particularly useful for getting the model to perform highly technical tasks or enforcing specific communication styles.

The example on slide shows a sample prompt where Gemini assumes the role of a Google Cloud technical support engineer.



Today's agenda

- 01 Load balancing
- 02 Hybrid load balancing
- 03 Traffic management
- 04 [Lab: Configuring Traffic Management with a Load Balancer](#)
- 05 Quiz

Google Cloud

In this module, we will cover the topics listed on the screen.

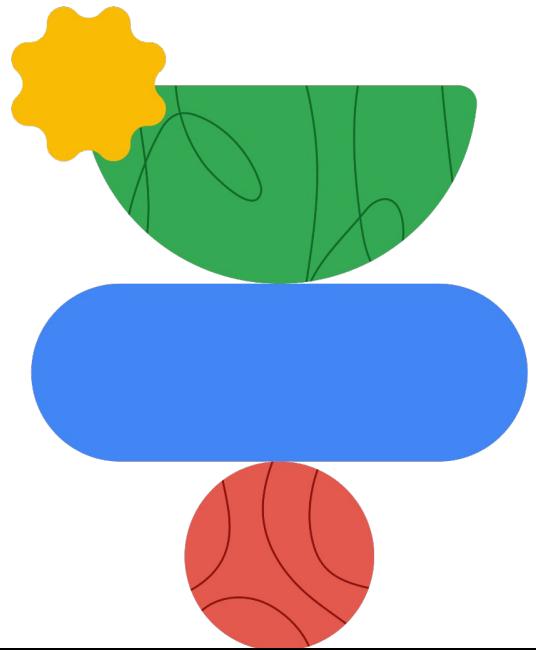
We'll begin with an overview of Cloud Load Balancing. We will continue with a discussion of hybrid load balancing. In other words, load balancing between Google Cloud, other public clouds, and on-premises environments.

We will follow with a discussion on traffic management, which provides enhanced features to route traffic based on criteria that you specify. After that, you will apply what you've learned in a traffic management lab exercise.

Let's get started.

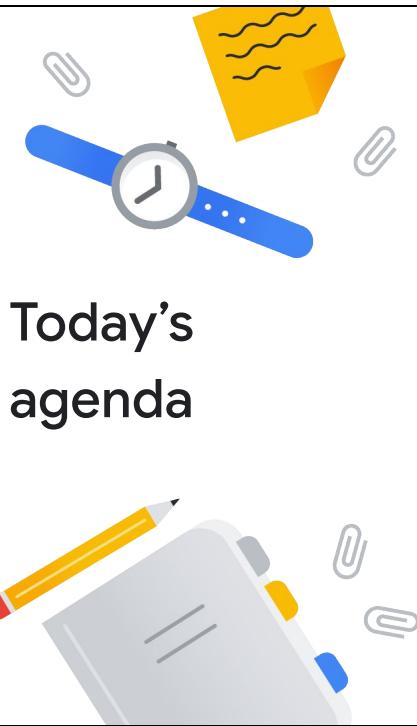
Lab intro

Configuring Traffic Management
with a Load Balancer



In this lab, you create a regional internal Application Load Balancer with two backends. Each backend will be an instance group. You will configure the load balancer to create a blue-green deployment.

The blue deployment refers to the current version of your application. The green deployment refers to a new application version. In this lab exercise, you configure the load balancer to send 70% of the traffic to the blue deployment and 30% to the green deployment.



Today's agenda

- 01 Load balancing
- 02 Hybrid load balancing
- 03 Traffic management
- 04 Lab: Configuring Traffic Management with a Load Balancer
- 05 Quiz

Google Cloud

In this module, we will cover the topics listed on the screen.

We'll begin with an overview of Cloud Load Balancing. We will continue with a discussion of hybrid load balancing. In other words, load balancing between Google Cloud, other public clouds, and on-premises environments.

We will follow with a discussion on traffic management, which provides enhanced features to route traffic based on criteria that you specify. After that, you will apply what you've learned in a traffic management lab exercise.

Let's get started.

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 2

Question

Where would you configure traffic management for a load balancer?

- A. In the load balancer frontend
- B. In the load balancer backend
- C. In the load descriptor
- D. In the URL map

Quiz | Question 3

Question

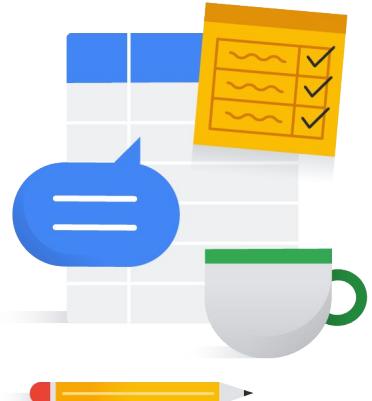
You can use hybrid load balancing to connect these environments:

- A. Google Cloud and on-premises
- B. Google Cloud and AWS
- C. Google Cloud, AWS, and on-premises
- D. Google Cloud, other public clouds, and on-premises

Explanation:

- A. That's incorrect. Although you can connect these environments, this answer is not complete.
- B. That's incorrect. Although you can connect these environments, this answer is not complete.
- C. That's incorrect. Although you can connect these environments, this answer is not complete.
- D. Correct. You can connect any destination that you can reach by using a Google hybrid connectivity product and that can be reached with a valid IP:Port combination.

Debrief



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 against incoming traffic and specify where the traffic should be sent.

You then applied what you learned in a lab exercise.

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



Thank you.

