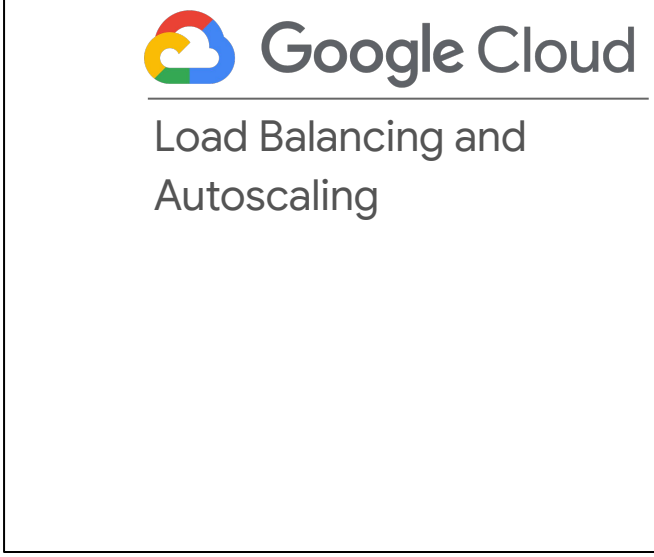
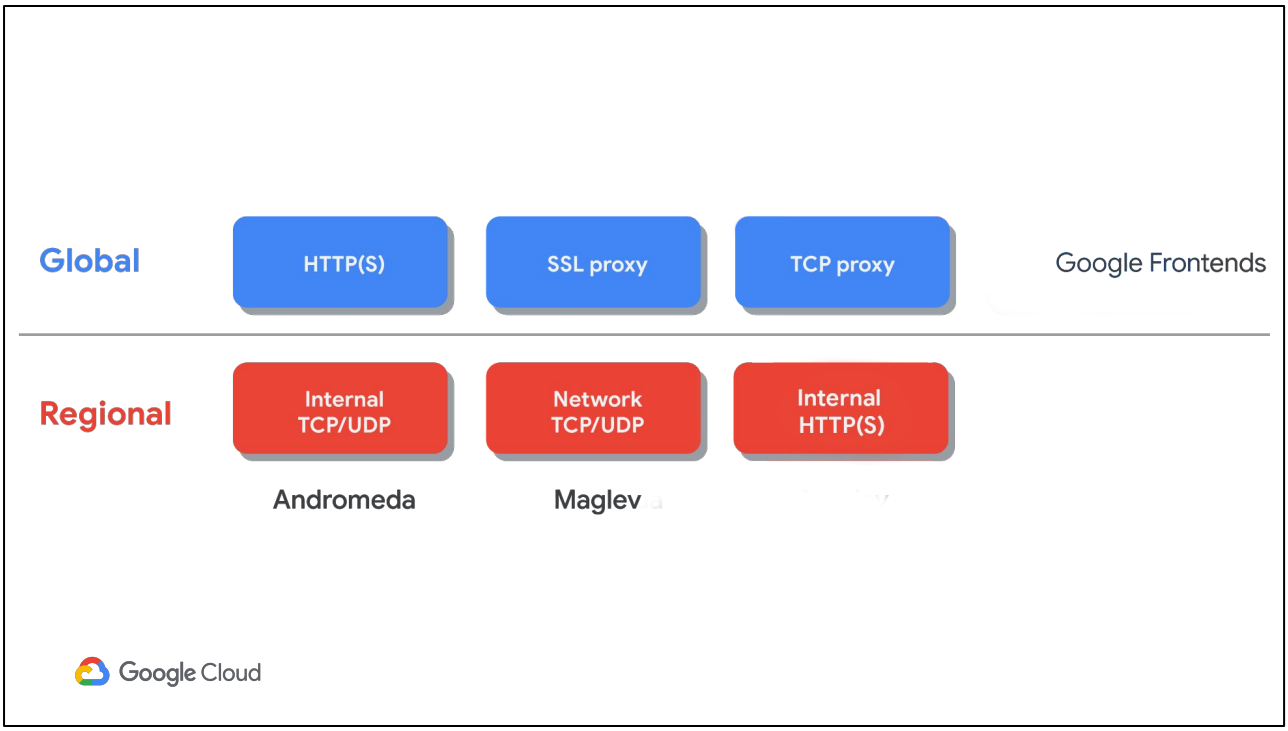


In this module we focus on Load Balancing and Autoscaling.

Cloud Load Balancing gives you the ability to distribute load-balanced compute resources in single or multiple regions to meet your high availability requirements, to put your resources behind a single anycast IP address, and to scale your resources up or down with intelligent autoscaling.

Using Cloud Load Balancing, you can serve content as close as possible to your users on a system that can respond to over 1 million queries per second. Cloud Load Balancing is a fully distributed, software-defined, managed service. It is not instance- or device-based, so you do not need to manage a physical load balancing infrastructure.



GCP offers different types of load balancers that can be divided into two categories: global and regional.

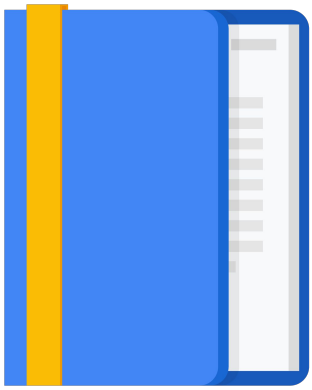
The global load balancers are the HTTP(S), SSL proxy, and TCP proxy load balancers. These load balancers leverage the Google frontends, which are software-defined, distributed systems that sit in Google's points of presence and are distributed globally. Therefore, you want to use a global load balancer when your users and instances are globally distributed, your users need access to the same applications and content, and you want to provide access using a single anycast IP address.

The regional load balancers are the internal and network load balancers, and they distribute traffic to instances that are in a single GCP region. The internal load balancer uses Andromeda, which is GCP's software-defined network virtualization stack, and the network load balancer uses Maglev, which is a large, distributed software system.

There's also another internal load balancer for HTTP(S) traffic, but it's in beta as of this recording. This 6th load balancer is a proxy-based, regional Layer 7 load balancer that enables you to run and scale your services behind a private load balancing IP address that is accessible only in the load balancer's region in your VPC network.

Agenda

Managed Instance Groups
HTTP(S) Load Balancing
Lab
SSL Proxy/TCP Proxy Load Balancing
Network Load Balancing
Internal Load Balancing
Lab
Choosing a Load Balancer



Google Cloud

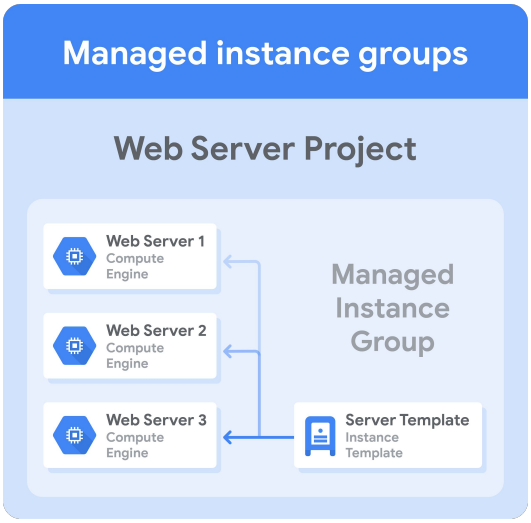
In this module, we will cover the different types of load balancers that are available in GCP. We will also go over managed instance groups and their autoscaling configurations, which can be used by these load balancing configurations.

You will explore many of the covered features and services throughout the two labs of this module, and I will wrap things up by helping you determine which GCP load balancer best meets your needs.

Let’s start by talking about managed instance groups.

Managed instance groups

- Deploy identical instances based on instance template
- Instance group can be resized
- Manager ensures all instances are RUNNING
- Typically used with autoscaler
- Can be single zone or regional



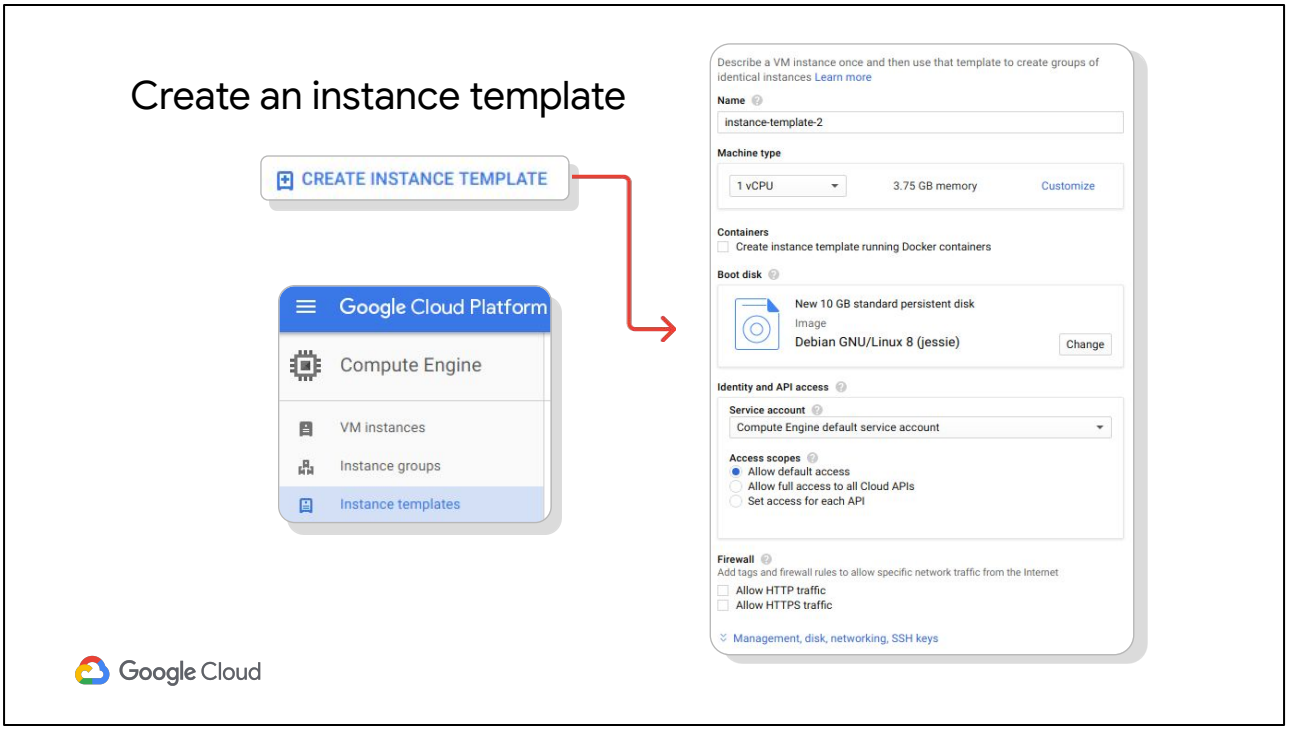
Google Cloud

A managed instance group is a collection of identical VM instances that you control as a single entity, using an instance template. You can easily update all the instances in the group by specifying a new template in a rolling update. Also, when your applications require additional compute resources, managed instance groups can automatically scale the number of instances in the group.

Managed instance groups can work with load balancing services to distribute network traffic to all of the instances in the group. If an instance in the group stops, crashes, or is deleted by an action other than the instance group's commands, the managed instance group automatically recreates the instance so it can resume its processing tasks. The recreated instance uses the same name and the same instance template as the previous instance. Managed instance groups can automatically identify and recreate unhealthy instances in a group to ensure that all the instances are running optimally.

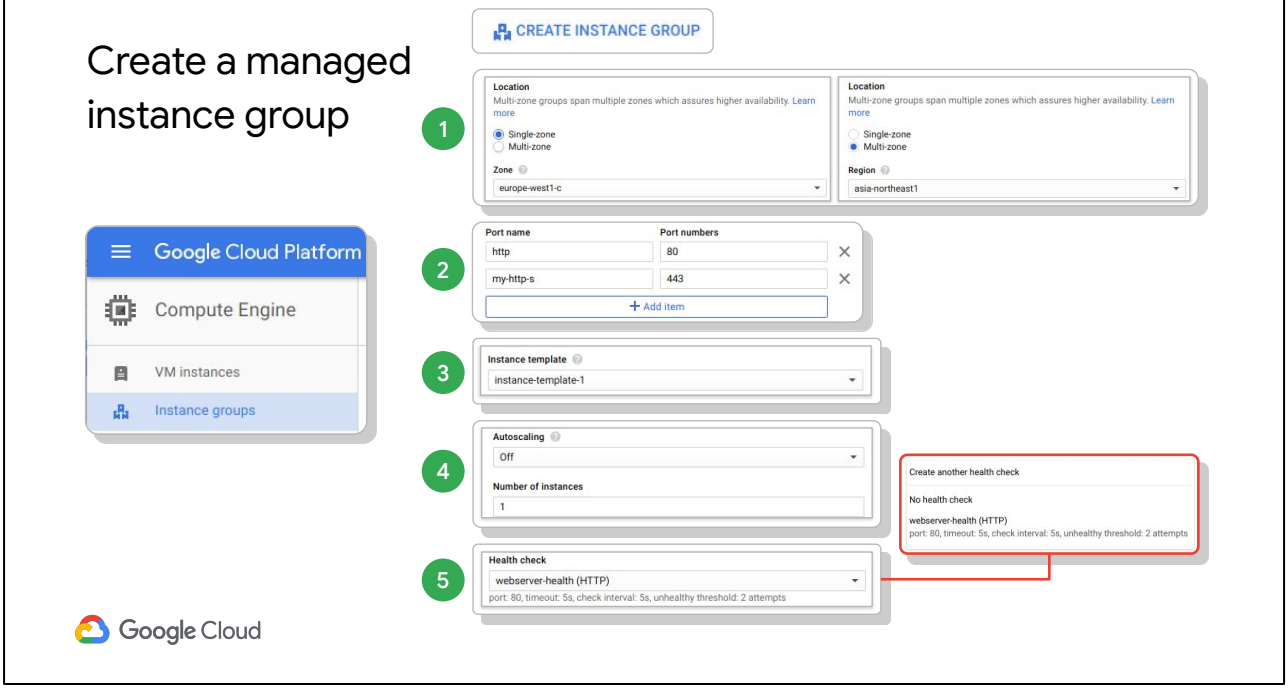
Regional managed instance groups are generally recommended over zonal managed instance groups because they allow you to spread the application load across multiple zones instead of confining your application to a single zone or you having to manage multiple instance groups across different zones. This replication protects against zonal failures and unforeseen scenarios where an entire group of instances in a single zone malfunctions. If that happens, your application can continue serving traffic

from instances running in another zone of the same region.



In order to create a managed instance group, you first need to create an instance template. Next, you're going to create a managed instance group of *n* specified instances. The instance group manager then automatically populates the instance group based on the instance template.

You can easily create instance templates using the GCP Console. The instance template dialog looks and works exactly like creating an instance, except that the choices are recorded so that they can be repeated.



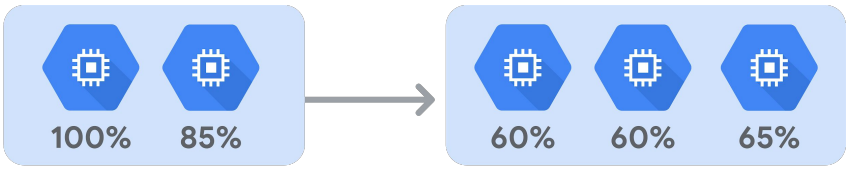
When you create an instance group, you define the specific rules for the instance group.

First, decide whether the instance group is going to be single or multi-zoned, and where those locations will be.
Second, choose the ports that you are going to allow and load balance across.
Third, select the instance template that you want to use.
Fourth, decide whether you want to autoscale, and under what circumstances.
Finally, consider creating a health check to determine which instances are healthy and should receive traffic.

Essentially, you're still creating virtual machines, but you're applying more rules to that instance group.

Managed instance groups offer autoscaling capabilities

- Dynamically add/remove instances:
  - Increases in load
  - Decreases in load
- Autoscaling policy:
  - CPU utilization
  - Load balancing capacity
  - Monitoring metrics
  - Queue-based workload



Target CPU utilization = 75%

Google Cloud

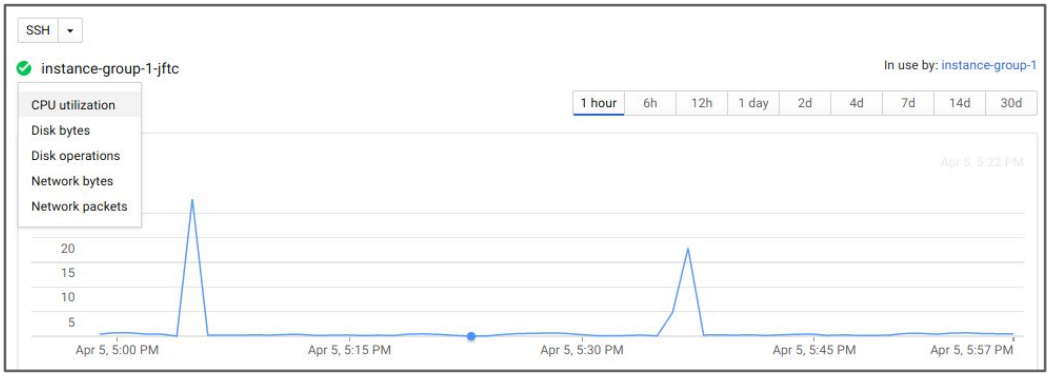
Let me provide more details on the autoscaling and health checks of a managed instance group.

As I mentioned earlier, managed instance groups offer autoscaling capabilities that allow you to automatically add or remove instances from a managed instance group based on increases or decreases in load. Autoscaling helps your applications gracefully handle increases in traffic and reduces cost when the need for resources is lower.

You just define the autoscaling policy, and the autoscaler performs automatic scaling based on the measured load. Applicable autoscaling policies include scaling based on CPU utilization, load balancing capacity, or monitoring metrics, or by a queue-based workload like Cloud Pub/Sub.

For example, let's assume you have 2 instances that are at 100% and 85% CPU utilization as shown on this slide. If your target CPU utilization is 75%, the autoscaler will add another instance to spread out the CPU load and stay below the 75% target CPU utilization. Similarly, if the overall load is much lower than the target, the autoscaler will remove instances as long as that keeps the overall utilization below the target. Now, you might ask yourself how do I monitor the utilization of my instance group.

VM graph helps set CPU utilization



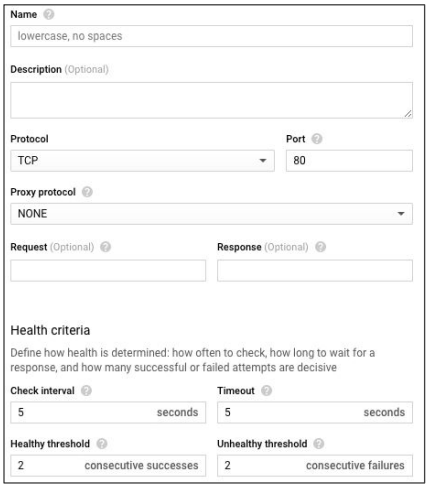
Google Cloud

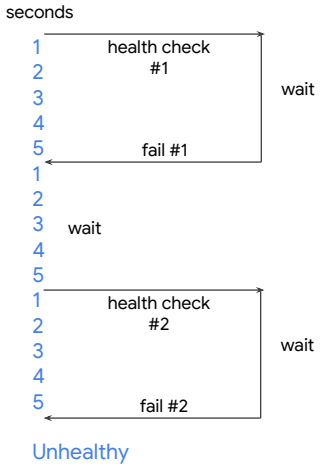
When you click on an instance group (or even an individual VM), a graph is presented. By default you'll see the CPU utilization over the past hour, but you can change the time frame and visualize other metrics like disk and network usage. These graphs are very useful for monitoring your instances' utilization and for determining how best to configure your Autoscaling policy to meet changing demand.

If you monitor the utilization of your VM instances in Stackdriver Monitoring, you can even set up alerts through several notification channels.

For more information on autoscaling, see the links section of this video. [https://cloud.google.com/compute/docs/autoscaler/]

Create a health check





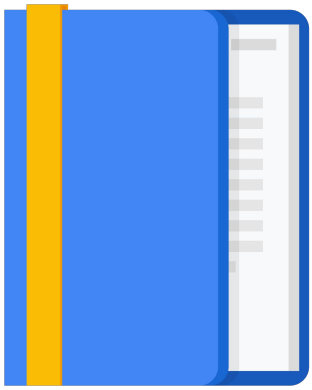
Google Cloud

Another important configuration for a managed instance group and load balancer is a health check. A health check is very similar to an uptime check in Stackdriver. You just define a protocol, port, and health criteria, as shown in this screenshot. Based on this configuration, GCP computes a health state for each instance.

The health criteria define how often to check whether an instance is healthy (that's the check interval); how long to wait for a response (that's the timeout); how many successful attempts are decisive (that's the healthy threshold); and how many failed attempts are decisive (that's the unhealthy threshold). In the example on this slide, the health check would have to fail twice over a total of 15 seconds before an instance is considered unhealthy.

Agenda

- Managed Instance Groups
- HTTP(S) Load Balancing
- Lab
- SSL Proxy/TCP Proxy Load Balancing
- Network Load Balancing
- Internal Load Balancing
- Lab
- Choosing a Load Balancer

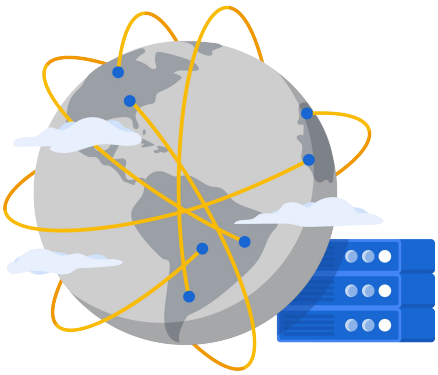


Google Cloud

Now, let's talk about HTTP(S) load balancing, which acts at Layer 7 of the OSI model. This is the application layer, which deals with the actual content of each message, allowing for routing decisions based on the URL.

HTTP(S) load balancing

- Global load balancing
- Anycast IP address
- HTTP or port 80 or 8080
- HTTPs on port 443
- IPv4 or IPv6
- Autoscaling
- URL maps

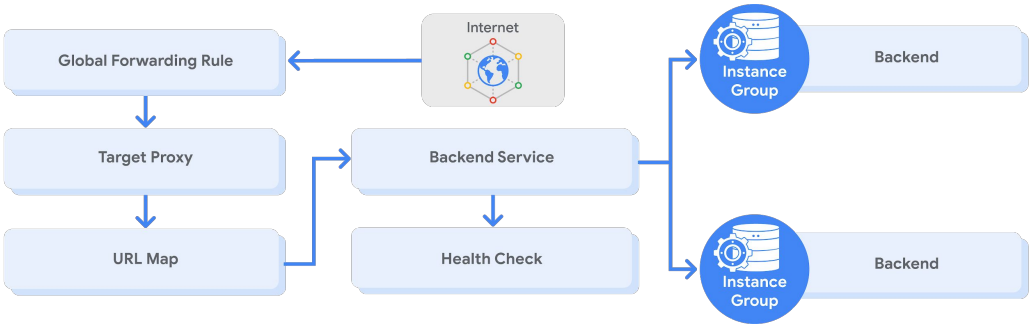


Google Cloud

GCP's HTTP(S) load balancing provides global load balancing for HTTP(S) requests destined for your instances. This means that your applications are available to your customers at a single anycast IP address, which simplifies your DNS setup. HTTP(S) load balancing balances HTTP and HTTPS traffic across multiple backend instances and across multiple regions.
HTTP requests are load balanced on port 80 or 8080, and HTTPS requests are load balanced on port 443.
This load balancer supports both IPv4 and IPv6 clients, is scalable, requires no pre-warming, and enables content-based and cross-region load balancing.

You can configure URL maps that route some URLs to one set of instances and route other URLs to other instances. Requests are generally routed to the instance group that is closest to the user. If the closest instance group does not have sufficient capacity, the request is sent to the next closest instance group that does have capacity. You will get to explore most of these benefits in the first lab of the module.

Architecture of an HTTP(S) load balancer

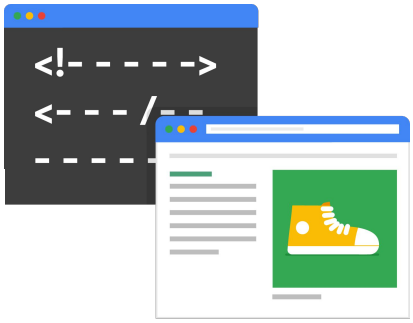


Let me walk through the complete architecture of an HTTP(S) load balancer, by using this diagram:

- A global forwarding rule directs incoming requests from the internet to a target HTTP proxy.
- The target HTTP proxy checks each request against a URL map to determine the appropriate backend service for the request. For example, you can send requests for www.example.com/audio to one backend service, which contains instances configured to deliver audio files, and requests for www.example.com/video to another backend service, which contains instances configured to deliver video files.
- The backend service directs each request to an appropriate backend based on serving capacity, zone, and instance health of its attached backends.

Backend services

- Health check
- Session affinity (optional)
- Time out setting (30-sec default)
- One or more backends
  - An instance group (managed or unmanaged)
  - A balancing mode (CPU utilization or RPS)
  - A capacity scaler (ceiling percentage of CPU/Rate targets)



Google Cloud

The backend services contain a health check, session affinity, a timeout setting, and one or more backends.

A health check polls instances attached to the backend service at configured intervals. Instances that pass the health check are allowed to receive new requests. Unhealthy instances are not sent requests until they are healthy again.
Normally, HTTP(S) load balancing uses a round-robin algorithm to distribute requests among available instances.
This can be overridden with session affinity. Session affinity attempts to send all requests from the same client to the same virtual machine instance.

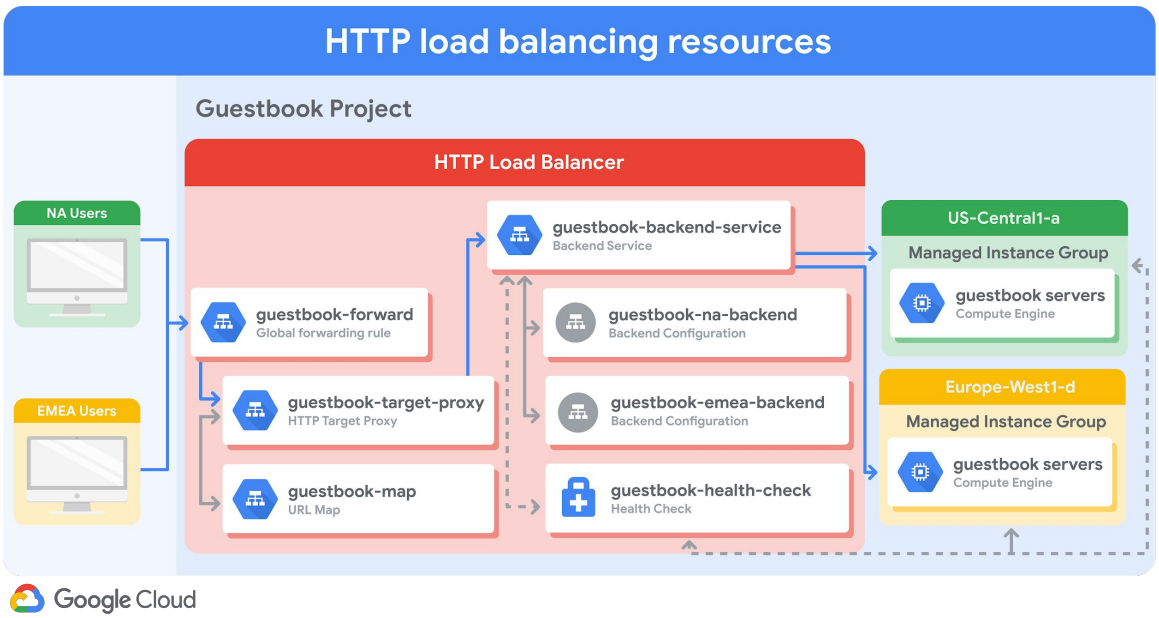
Backend services also have a timeout setting, which is set to 30 seconds by default. This is the amount of time the backend service will wait on the backend before considering the request a failure. This is a fixed timeout, not an idle timeout. If you require longer-lived connections, set this value appropriately.

The backends themselves contain an instance group, a balancing mode and a capacity scaler.

- An instance group contains virtual machine instances. The instance group may be a managed instance group with or without autoscaling or an unmanaged instance group.

- A balancing mode tells the load balancing system how to determine when the backend is at full usage. If all the backends for the backend service in a region are at full usage, new requests are automatically routed to the nearest region that can still handle requests. The balancing mode can be based on CPU utilization or requests per second (RPS).
- A capacity setting is an additional control that interacts with the balancing mode setting. For example, if you normally want your instances to operate at a maximum of 80% CPU utilization, you would set your balancing mode to 80% CPU utilization and your capacity to 100%. If you want to cut instance utilization in half, you could leave the balancing mode at 80% CPU utilization and set capacity to 50%.

Now, any changes to your backend services are not instantaneous. So, don't be surprised if it takes several minutes for your changes to propagate throughout the network.



Let me walk through an HTTP load balancer in action. The project on this slide has a single global IP address, but users enter the Google Cloud network from two different locations: one in North America and one in EMEA.

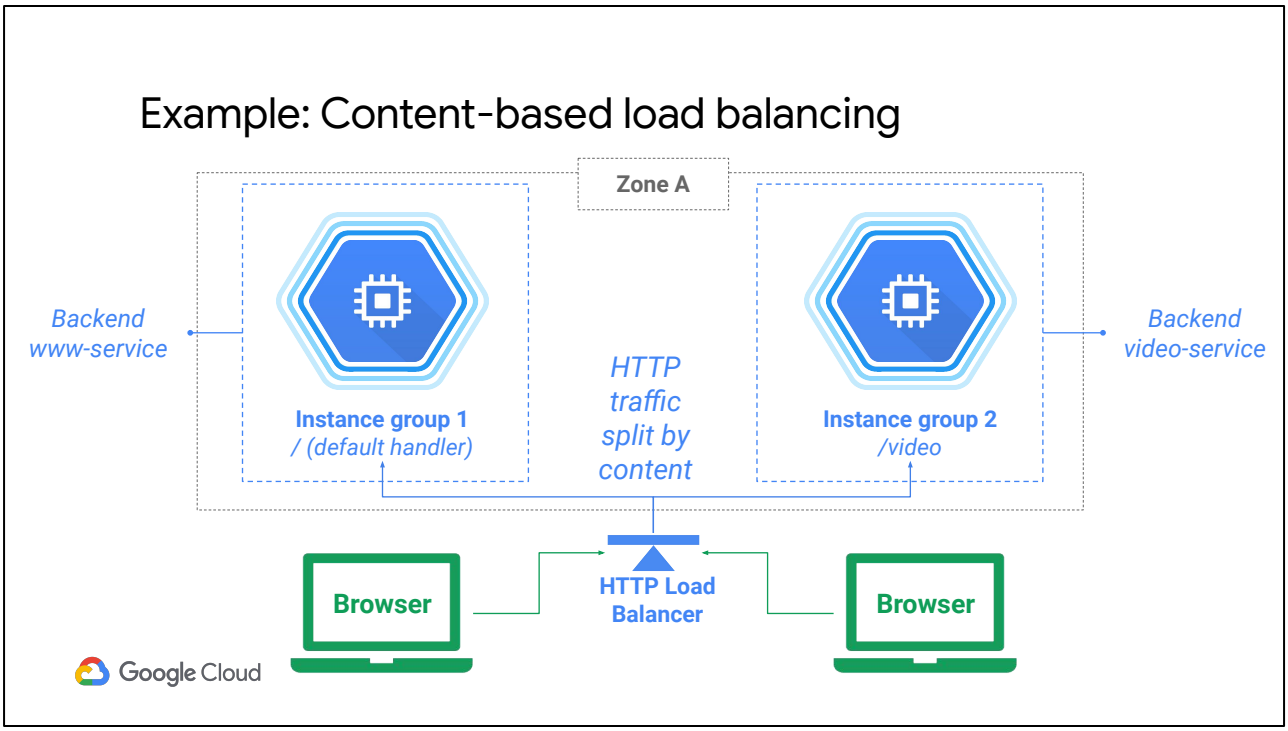
First, the global forwarding rule directs incoming requests to the target HTTP proxy. The proxy checks the URL map to determine the appropriate backend service for the request. In this case, we are serving a guestbook application with only one backend service.

The backend service has two backends: one in us-central1-a and one in europe-west1-d. Each of those backends consists of a managed instance group. Now, when a user request comes in, the load balancing service determines the approximate origin of the request from the source IP address. The load balancing service also knows the locations of the instances owned by the backend service, their overall capacity, and their overall current usage. Therefore, if the instances closest to the user have available capacity, the request is forwarded to that closest set of instances.

In our example, traffic from the user in North America would be forwarded to the managed instance group in us-central1-a, and traffic from the user in EMEA would be forwarded to the managed instance group in europe-west1-d. If there are several

users in each region, the incoming requests to the given region are distributed evenly across all available backend services and instances in that region.

If there are no healthy instances with available capacity in a given region, the load balancer instead sends the request to the next closest region with available capacity. Therefore, traffic from the EMEA user could be forwarded to the us-central1-a backend if the europe-west1-d backend does not have capacity or has no healthy instances as determined by the health checker. This is referred to as cross-region load balancing.

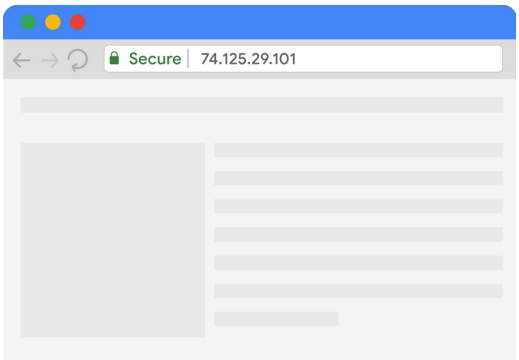


Another example of an HTTP load balancer is a content-based load balancer. In this case, there are two separate backend services that handle either web or video traffic.

The traffic is split by the load balancer based on the URL header as specified in the URL map. If a user is navigating to /video, the traffic is sent to the backend video-service, and if a user is navigating anywhere else, the traffic is sent to the web-service backend. All of that is achieved with a single global IP address.

HTTP(S) load balancing

- Target HTTP(S) proxy
- One signed SSL certificate installed (minimum)
- Client SSL session terminates at the load balancer
- Support the QUIC transport layer protocol



Google Cloud

An HTTP(S) load balancer has the same basic structure as an HTTP load balancer, but differs in the following ways:

- An HTTP(S) load balancer uses a target HTTPS proxy instead of a target HTTP proxy.
- An HTTP(S) load balancer requires at least one signed SSL certificate installed on the target HTTPS proxy for the load balancer.
- The client SSL session terminates at the load balancer.
- HTTP(S) load balancers support the QUIC transport layer protocol.

QUIC is a transport layer protocol that allows faster client connection initiation, eliminates head-of-line blocking in multiplexed streams, and supports connection migration when a client's IP address changes. For more information on the QUIC protocol, see the links section of this video. [https://www.chromium.org/quic]

SSL certificates

- Required for HTTP(S) load balancing
- Up to 10 SSL certificates (per target proxy)
- Create an SSL certificate resource

Google Cloud

To use HTTPS, you must create at least one SSL certificate that can be used by the target proxy for the load balancer.
You can configure the target proxy with up to ten SSL certificates.
For each SSL certificate, you first create an SSL certificate resource, which contains the SSL certificate information. SSL certificate resources are used only with load balancing proxies such as a target HTTPS proxy or target SSL proxy, which we will discuss later in this module.

Lab

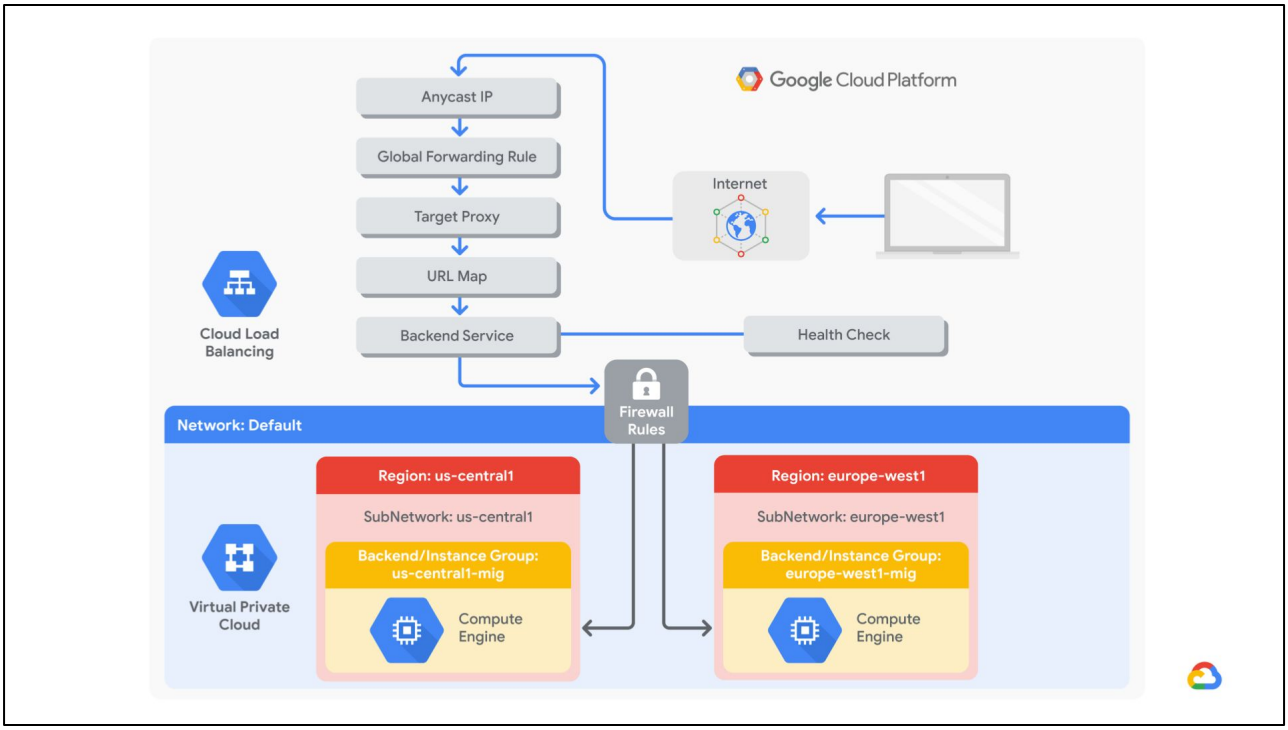
Configuring an HTTP Load Balancer with Autoscaling

Philipp Maier

Google Cloud

Let’s apply what we just covered.

In this lab, you configure an HTTP Load Balancer with autoscaling.



Specifically, you create two managed instance groups that serve as backends in us-central1 and europe-west1. Then, you create and stress test a load balancer to demonstrate global load balancing and autoscaling.

Lab review

Configuring an HTTP Load Balancer with Autoscaling

Philipp Maier

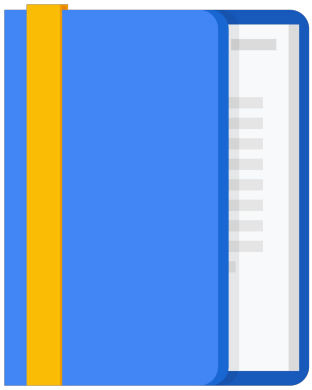
Google Cloud

In this lab, you configured an HTTP load balancer with backends in us-central1 and europe-west1. Then you stress-tested the load balancer with a VM to demonstrate global load balancing and autoscaling.

You can stay for a lab walkthrough, but remember that GCP's user interface can change, so your environment might look slightly different.

Agenda

- Managed Instance Groups
- HTTP(S) Load Balancing
- Lab
- SSL Proxy/TCP Proxy Load Balancing
- Network Load Balancing
- Internal Load Balancing
- Lab
- Choosing a Load Balancer



Google Cloud

Let’s talk about SSL proxy and TCP proxy load balancing.

SSL proxy load balancing

- Global load balancing for encrypted, non-HTTP traffic
- Terminates SSL session at load balancing layer
- IPv4 or IPv6 clients
- Benefits:
  - Intelligent routing
  - Certificate management
  - Security patching
  - SSL policies

Google Cloud

SSL proxy is a global load balancing service for encrypted, non-HTTP traffic. This load balancer terminates user SSL connections at the load balancing layer, then balances the connections across your instances using the SSL or TCP protocols. These instances can be in multiple regions, and the load balancer automatically directs traffic to the closest region that has capacity.

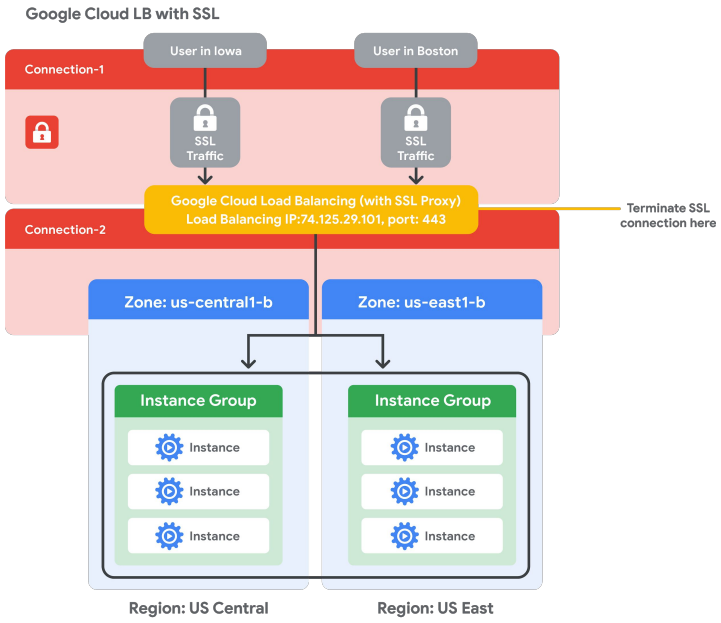
SSL proxy load balancing supports both IPv4 and IPv6 addresses for client traffic and provides Intelligent routing, Certificate management, Security patching and SSL policies.

- Intelligent routing means that this load balances can route requests to backend locations where there is capacity.
- From a certificate management perspective, you only need to update your customer-facing certificate in one place when you need to switch certificates. Also, you can reduce the management overhead for your virtual machine instances by using self-signed certificates on your instances.
- In addition, if vulnerabilities arise in the SSL or TCP stack, GCP will apply patches at the load balancer automatically in order to keep your instances safe.

For the full list of ports supported by SSL proxy load balancing and other benefits,

please refer to the links section of this video.
[https://cloud.google.com/load-balancing/docs/ssl/#overview]

Example: SSL proxy load balancing



Google Cloud

This network diagram illustrates SSL proxy load balancing. In this example, traffic from users in Iowa and Boston is terminated at the global load balancing layer. From there, a separate connection is established to the closest backend instance. In other words, the user in Boston would reach the us east region, and the user in Iowa would reach the us central region, if there is enough capacity.

Now, the traffic between the proxy and the backends can use SSL or TCP. I recommend using SSL.

TCP proxy load balancing

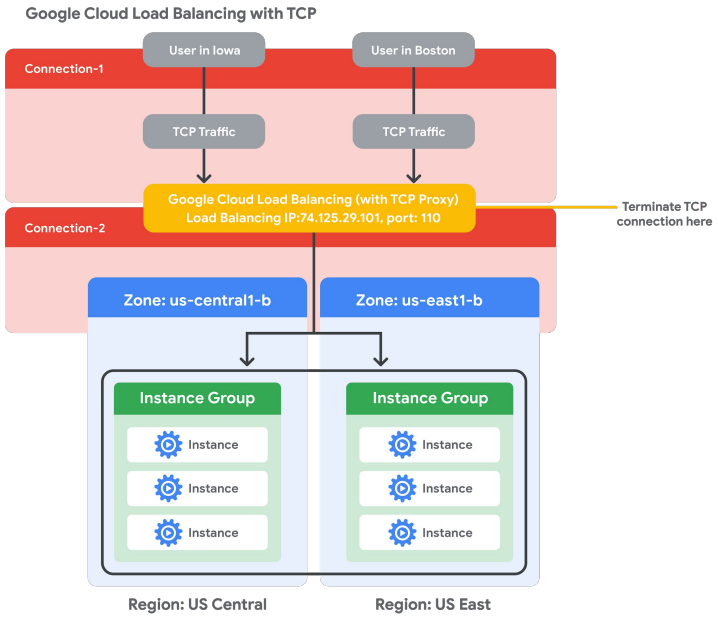
- Global load balancing for unencrypted, non-HTTP traffic
- Terminates TCP sessions at load balancing layer
- IPv4 or IPv6 clients
- Benefits:
  - Intelligent routing
  - Security patching

TCP proxy is a global load balancing service for unencrypted, non-HTTP traffic. This load balancer terminates your customers' TCP sessions at the load balancing layer, then forwards the traffic to your virtual machine instances using TCP or SSL. These instances can be in multiple regions, and the load balancer automatically directs traffic to the closest region that has capacity.

TCP proxy load balancing supports both IPv4 and IPv6 addresses for client traffic. Similar to the SSL proxy load balancer, the TCP proxy load balancer provides Intelligent routing and Security patching.

For the full list of ports supported by TCP proxy load balancing and other benefits, please refer to the links section of this video.
[https://cloud.google.com/load-balancing/docs/tcp/#overview]

Example: TCP proxy load balancing



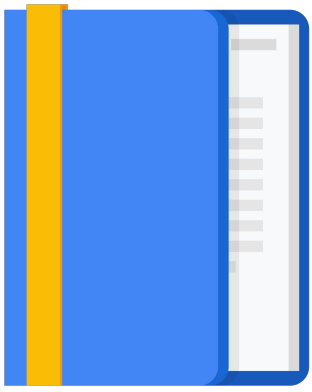
Google Cloud

This network diagram illustrates TCP proxy load balancing. In this example, traffic from users in Iowa and Boston is terminated at the global load balancing layer. From there, a separate connection is established to the closest backend instance. As in the SSL proxy load balancing example, the user in Boston would reach the us east region, and the user in Iowa would reach the us central region, if there is enough capacity.

Now, the traffic between the proxy and the backends can use SSL or TCP, and I also recommend using SSL here.

Agenda

- Managed Instance Groups
- HTTP(S) Load Balancing
- Lab
- SSL Proxy/TCP Proxy Load Balancing
- **Network Load Balancing**
- Internal Load Balancing
- Lab
- Choosing a Load Balancer

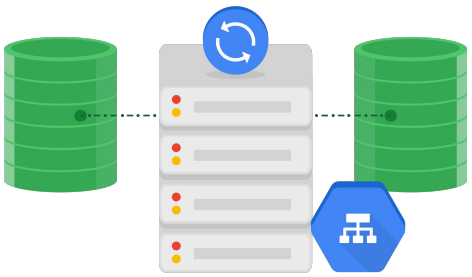


Google Cloud

Next, let's talk about network load balancing, which is a regional load balancing service.

Network load balancing

- Regional, *non-proxied* load balancer
- Forwarding rules (IP protocol data)
- Traffic:
  - UDP
  - TCP/SSL ports
- Backends:
  - Instance group
  - Target pool



Google Cloud

Network load balancing is a regional, non-proxied load balancing service. In other words, all traffic is passed through the load balancer, instead of being proxied, and traffic can only be balanced between VM instances that are in the same region, unlike a global load balancer.

This load balancing service uses forwarding rules to balance the load on your systems based on incoming IP protocol data, such as address, port, and protocol type. You can use it to load balance UDP traffic and to load balance TCP and SSL traffic on ports that are not supported by the TCP proxy and SSL proxy load balancers.

The backends of a network load balancer can be a template-based instance group or target pool resource. But what is a target pool resource?

Target pool resource defines a group of instances that receive incoming traffic from forwarding rules

- Forwarding rules (TCP and UDP)
- Up to 50 per project
- One health check
- Instances must be in the same region

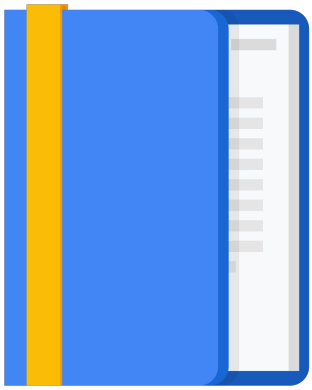
Google Cloud

A target pool resource defines a group of instances that receive incoming traffic from forwarding rules. When a forwarding rule directs traffic to a target pool, the load balancer picks an instance from these target pools based on a hash of the source IP and port and the destination IP and port. These target pools can only be used with forwarding rules that handle TCP and UDP traffic.

Now, each project can have up to 50 target pools, and each target pool can have only one health check.
Also, all the instances of a target pool must be in the same region, which is the same limitation as for the network load balancer.

Agenda

- Managed Instance Groups
- HTTP(S) Load Balancing
- Lab
- SSL Proxy/TCP Proxy Load Balancing
- Network Load Balancing
- **Internal Load Balancing**
- Lab
- Choosing a Load Balancer



Google Cloud

Next, let's talk about internal load balancing.

Internal load balancing

- Regional, private load balancing
  - VM instances in same region
  - RFC 1918 IP addresses
- TCP/UDP traffic
- Reduced latency, simpler configuration
- Software-defined, fully distributed load balancing

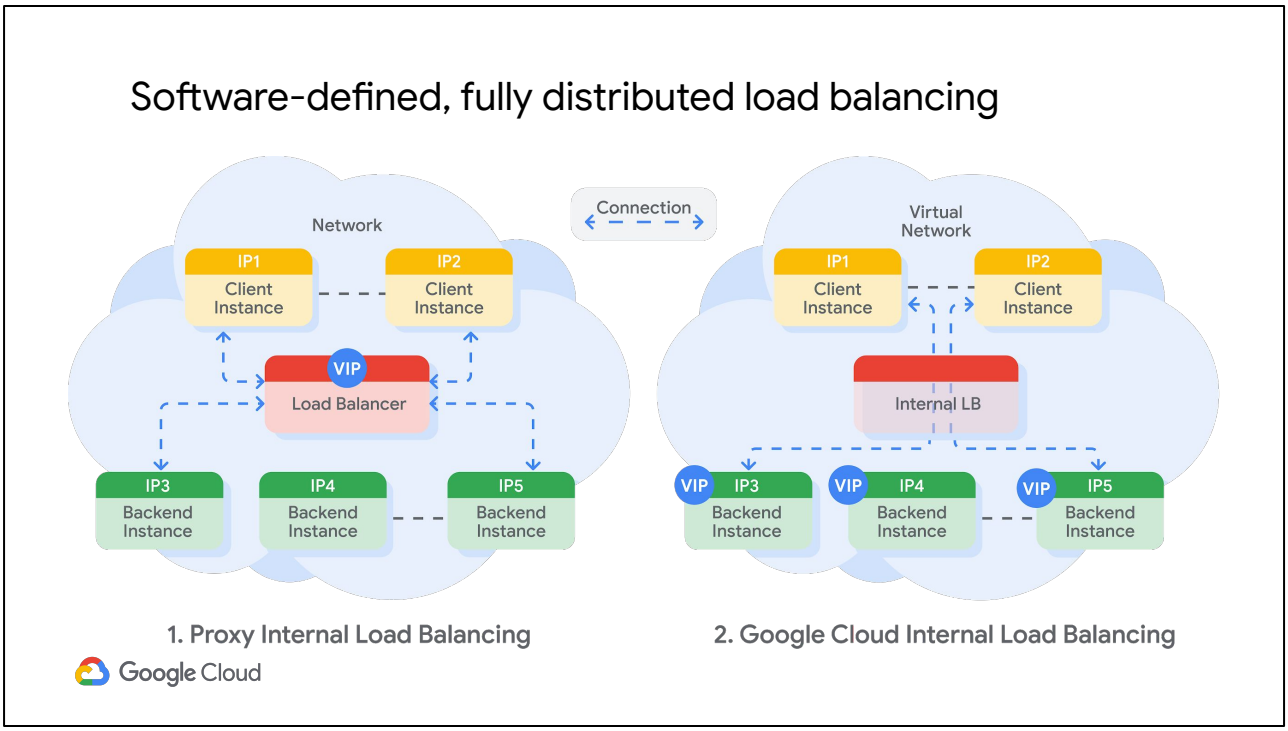
Google Cloud

Internal load balancing is a regional, private load balancing service for TCP- and UDP-based traffic. In other words, this load balancer enables you to run and scale your services behind a private load balancing IP address This means that it is only accessible through the internal IP addresses of virtual machine instances that are in the same region.

Therefore, use internal load balancing to configure an internal load balancing IP address to act as the frontend to your private backend instances. Because you don't need a public IP address for your load-balanced service, your internal client requests stay internal to your VPC network and region.

This often results in lowered latency, because all your load-balanced traffic will stay within Google's network, making your configuration much simpler.

Let's talk more about the benefit of using a software-defined internal load balancing service.

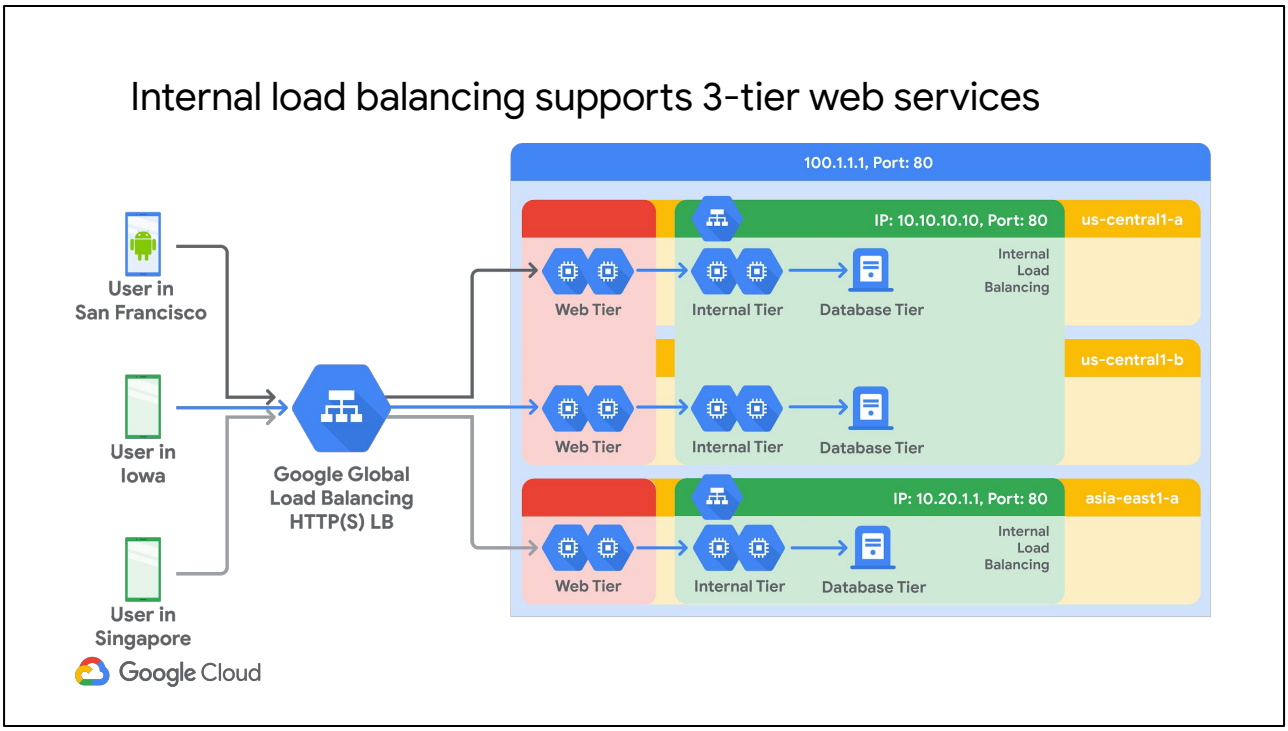


GCP internal load balancing is not based on a device or a VM instance. Instead, it is a software-defined, fully distributed load balancing solution.

In the traditional proxy model of internal load balancing, as shown on the left, you configure an internal IP address on a load balancing device or instances, and your client instance connects to this IP address. Traffic coming to the IP address is terminated at the load balancer, and the load balancer selects a backend to establish a new connection to. Essentially, there are two connections: one between the Client and the Load Balancer, and one between the Load Balancer and the Backend.

GCP internal load balancing distributes client instance requests to the backends using a different approach, as shown on the right. It uses lightweight load balancing built on top of Andromeda (Google's network virtualization stack) to provide software-defined load balancing that directly delivers the traffic from the client instance to a backend instance.

To learn more about Andromeda, I recommend the blog that is in the links section of this video.
[https://cloudplatform.googleblog.com/2014/04/enter-andromeda-zone-google-cloud-platforms-latest-networking-stack.html]



Now, internal load balancing enables you to support use cases such as the traditional 3-tier web services.

In this example, the web tier uses an external HTTP(S) load balancer that provides a single global IP address for users in San Francisco, Iowa, Singapore, and so on. The backends of this load balancer are located in the us-central1 and asia-east1 regions because this is a global load balancer.

These backends then access an internal load balancer in each region as the application or internal tier. The backends of this internal tier are located in us-central1-a, us-central1-b, and asia-east1-b. The last tier is the database tier in each of those zones.

The benefit of this 3-tier approach is that neither the database tier nor the application tier is exposed externally. This simplifies security and network pricing.

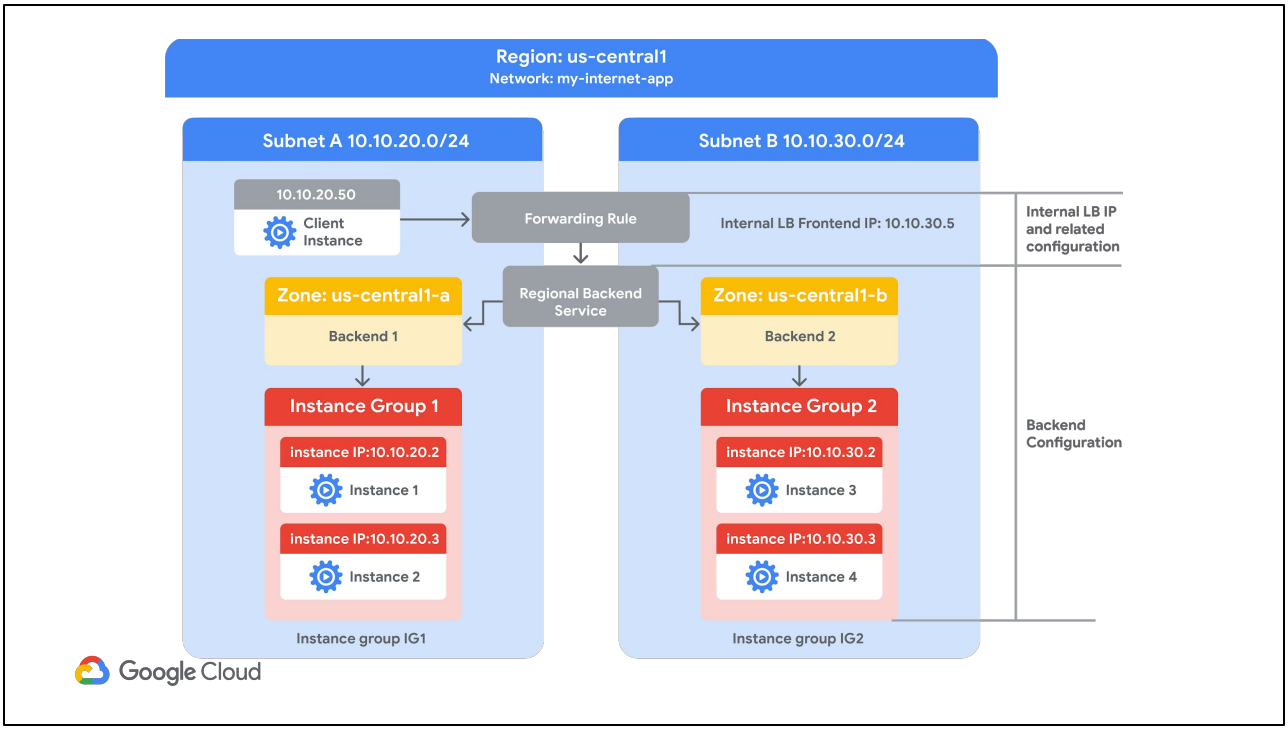
Lab

Configuring an Internal Load Balancer

Philipp Maier

Google Cloud

Let’s apply some of the internal load balancer concepts that we just discussed in a lab.



In this lab, you create two managed instance groups in the same region. Then, you configure and test an internal load balancer with the instances groups as the backends, as shown in this network diagram.

Lab review

Configuring an Internal Load Balancer

Philipp Maier

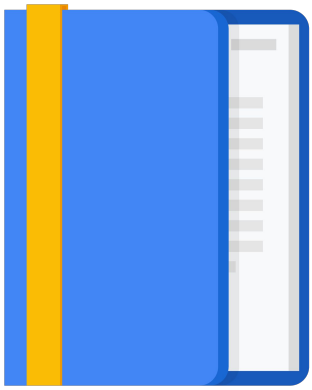
Google Cloud

In this lab, you created two managed instance groups in the us-central1 region, along with firewall rules to allow HTTP traffic to those instances and TCP traffic from the GCP health checker. Then, you configured and tested an internal load balancer for those instance groups.

You can stay for a lab walkthrough, but remember that GCP's user interface can change, so your environment might look slightly different.

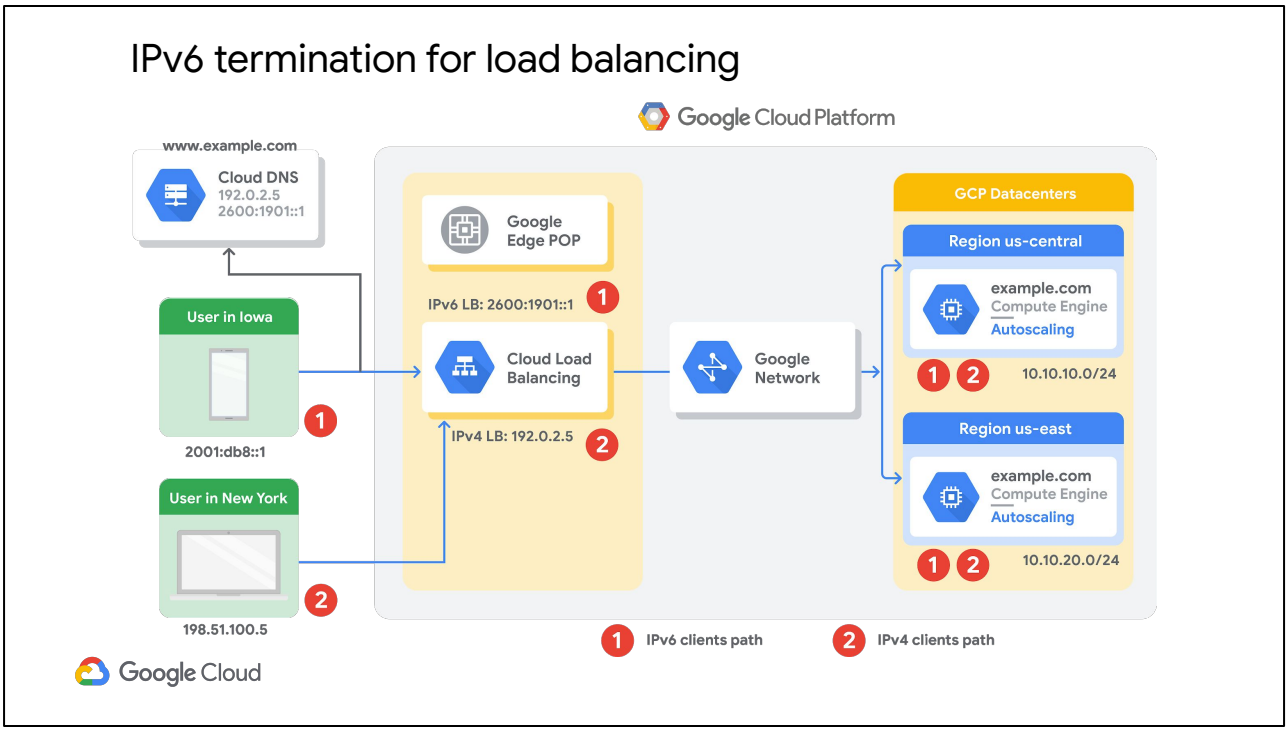
Agenda

- Managed Instance Groups
- HTTP(S) Load Balancing
- Lab
- SSL Proxy/TCP Proxy Load Balancing
- Network Load Balancing
- Internal Load Balancing
- Lab
- Choosing a Load Balancer



Google Cloud

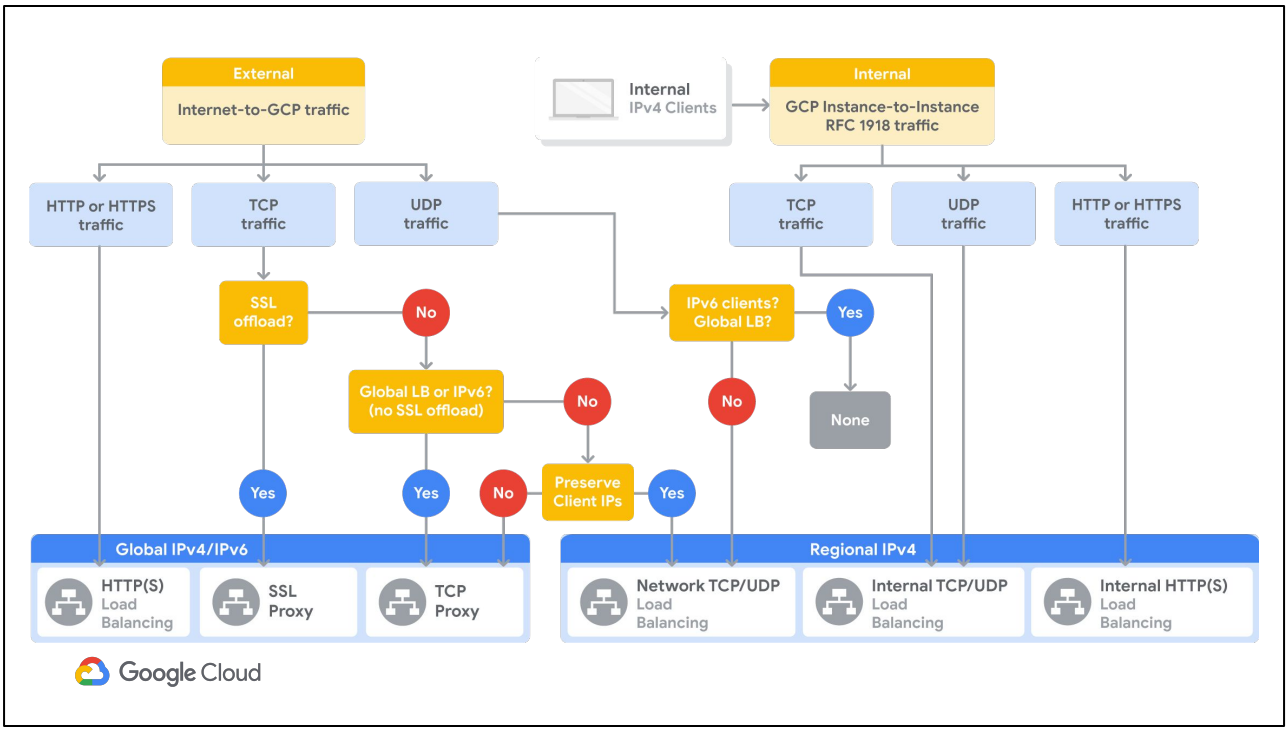
Now that we have discussed all the different load balancing services within GCP, let me help you determine which load balancer best meets your needs.



One differentiator between the different GCP load balancers is the support for IPv6 clients. Only the HTTP(S), SSL proxy, and TCP proxy load balancing services support IPv6 clients. IPv6 termination for these load balancers enables you to handle IPv6 requests from your users and proxy them over IPv4 to your backends.

For example, in this diagram there is a website www.example.com that is translated by Cloud DNS to both an IPv4 and IPv6 address. This allows a desktop user in New York and a mobile user in Iowa to access the load balancer through the IPv4 and IPv6 addresses, respectively. But how does the traffic get to the backends and their IPv4 addresses?

Well, the load balancer acts as a reverse proxy, terminates the IPv6 client connection, and places the request into an IPv4 connection to a backend. On the reverse path, the load balancer receives the IPv4 response from the backend and places it into the IPv6 connection back to the original client. In other words, configuring IPv6 termination for your load balancers lets your backend instances appear as IPv6 applications to your IPv6 clients.



Now, in order to decide which load balancer best suits your implementation of GCP, consider the following aspects of Cloud Load Balancing: global versus regional load balancing, external versus internal load balancing, and the traffic type.

If you need an external load balancing service, start on the top left of this flow chart. First, choose the type of traffic that your load balancer must handle. If that is HTTP or HTTPS traffic, I recommend using the HTTP(S) load balancing service as a Layer 7 load balancer. Otherwise, use the TCP and UDP traffic paths of this flow chart to determine whether the SSL proxy, TCP proxy, or network load balancing service meets your needs.

If you need an internal load balancing service, you have the internal load balancing service available, and it supports both TCP and UDP traffic. As I mentioned at the beginning of this module, there's actually another internal load balancer for HTTP(S) traffic but it's in beta as of this recording. This 6th load balancer is for HTTP or HTTPS traffic and it's regional, meaning for IPv4 clients.

Summary of load balancers

Load balancer	Traffic type	Global/ Regional	External/ Internal	External ports for load balancing
HTTP(S)	HTTP or HTTPS	Global IPv4 IPv6	External	HTTP on 80 or 8080; HTTPS on 443
SSL Proxy	TCP with SSL offload			25, 43, 110, 143, 195, 443, 465, 587, 700, 993, 995, 1883, 5222
TCP Proxy	• TCP without SSL offload • Does not preserve client IP addresses			25, 43, 110, 143, 195, 443, 465, 587, 700, 993, 995, 1883, 5222
Network TCP/UDP	• TCP/UDP without SSL offload • Preserves client IP addresses	Regional IPv4		Any
Internal TCP/UDP	TCP or UDP		Internal	Any
Internal HTTP(S)	HTTP or HTTPS			HTTP on 80 or 8080; HTTPS on 443

Google Cloud

If you prefer a table over a flow chart, I recommend this summary table.

This table helps you identify the right load balancer based on the traffic type, the distribution of your backends (global or regional), and the type of IP addresses of your backends (external or internal). This table also lists the available ports for load balancing and highlights that only the global load balancers support both IPv4 and IPv6 clients.

Review

Load Balancing and Autoscaling

Google Cloud

In this module, we looked at the different types of load balancers that are available in GCP, along with managed instance groups and autoscaling. You were able to apply most of the covered services and concepts by working through the two labs of this module.

We also discussed the criteria for choosing between the different load balancers and looked at a flow chart and a summary table to help you pick the right load balancers. Remember, sometimes it's useful to combine an internal and an external load balancer to support 3-tier web services.