Google Cloud

Professional Cloud Architect

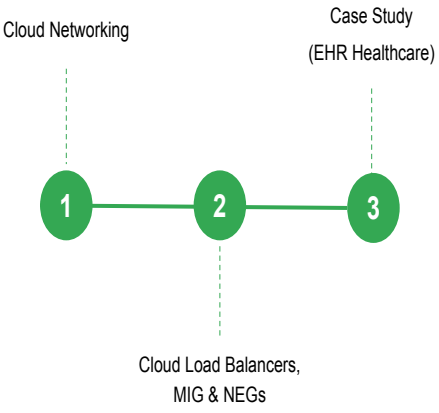
Preparing for Professional Cloud Architect Journey for AWS Professionals

Plan:

- 5 mins Explain VPC global, subnet regional and it's effect on HA/DR
- 5 mins [DEMO] Network Service Tiers -> show on regional external LB
- 10 mins [DEMO] Shared VPCs - show host project, service project, VM creation
    - Shared VPC vs VPC Peering (~slide 66)
- 3 mins [Slides] "Basic Dual-Region HA Network" in AWS vs GCP
- 12 mins [Overview] network reference architecture (~slide 70)
    - Include DNS, Shared VPC, appliances, VPC Peering
- 5 mins [DEMO] Load Balancers & backends
    - Show frontend, backend (incl. Buckets), Serverless as backends, Cloud CDN + Armor as reasons to use LB
- ~5 min [Slides] Diag questions
- ~15 mins: [CASE STUDY] EHR Healthcare with questions

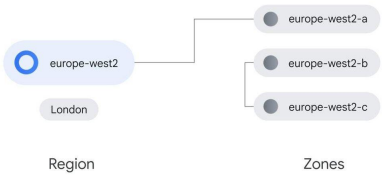
Perfect timing. 60 mins is just enough to cover all

Session 5 topics



Cloud Networking

Networking concepts: Regions



AWS	Google Cloud
• A geographic area where data centers are clustered into availability zones that are interconnected with redundant, low-latency network connections • AWS does not specify the round trip time (RTT) for a region.	• A geographic area where data centers are clustered into zones that are interconnected with redundant, low-latency network connections • The round trip time (RTT) from one virtual machine (VM) to another within a region is typically under 1 millisecond.

Google Cloud and AWS both use **regions** as a way to provide cloud services to customers.

In AWS and Google Cloud, a region refers to a geographic area where data centers are clustered into at least three availability zones that are interconnected with redundant, low-latency network connections.

AWS does not specify the round trip time (RTT) for a region.

In Google Cloud, the round trip time (RTT) from one virtual machine (VM) to another is typically under 1 millisecond.

Networking concepts: Zones

AWS	Google Cloud
• Every region has at least three availability zones. • Each zone is physically separate from the others in the same region with independent utilities and security. • Located within 60 miles of each other • Offers services that are regional and global	• Zones provide data center services. • Every region has at least three zones. • Each zone is physically separate from the others in the same region with independent utilities and security. • Offers services that are multi-regional, global, or zonal

Both platforms also have **zones**.

In AWS, every region has at least three availability zones. Each one is physically separate from the other zones in the same region with independent utilities and security. All Availability Zones for a region are located within 60 miles of each other. AWS offers services that are regional and global.

Google Cloud uses zones to provide data center services and every region will have at least three zones. As in AWS, each one is physically separate from the other zones in the same region with independent utilities and security. It offers services that are multi-regional, global, or zonal.

Networking concepts: Points of Presence (PoPs)

AWS	Google Cloud
• PoPs (Edge Locations) provide CDN and firewall services. • PoPs called Local Zones are an extension of regions with limited subsets of services such as compute and storage.	• PoPs provide Cloud CDN and deliver built-in edge caching for services. • PoPs connect to data centers through Google-owned fiber for fast and reliable access to Google services.

a Point of Presence (PoP) is a physical location where a cloud provider maintains network equipment and infrastructure.

Google Cloud and AWS both have Points of Presence, or PoPs located in many locations around the world. These PoP locations help cache content closer to end users. Each platform uses PoPs in different ways. Let's look at these differences.

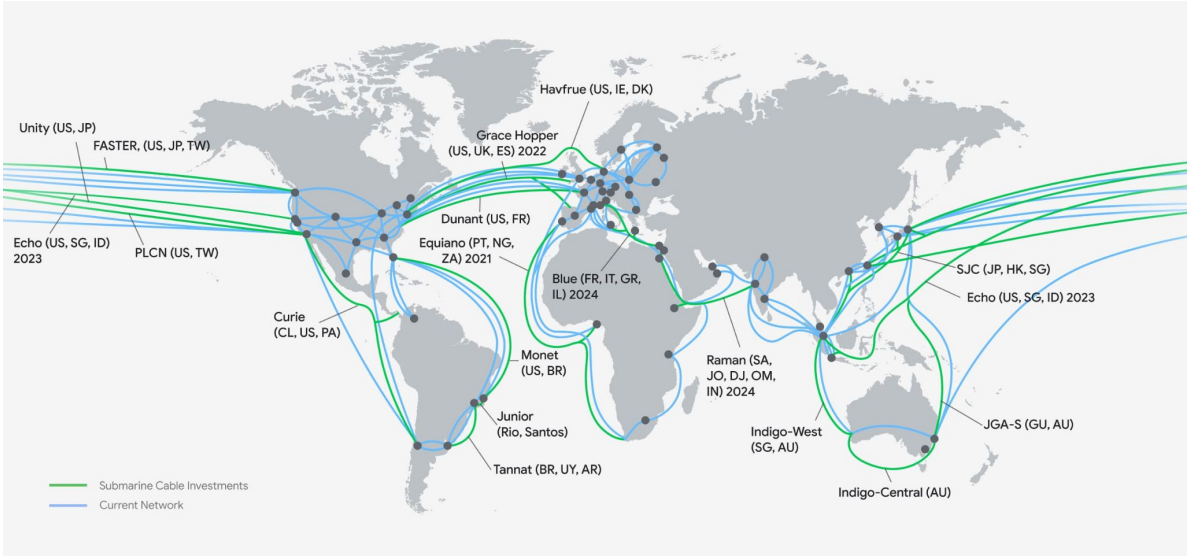
AWS uses PoPs referred to as Edge Locations to provide CDN services through CloudFront, DNS services through Route 53, and firewall services through AWS WAF and AWS Shield.

AWS also provides access to PoPs referred to as Local Zones as an extension of regions with limited subsets of services such as compute and storage for customers that have workloads that require very low latency for end users.

Google Cloud uses PoPs to provide Cloud CDN and to deliver built-in edge caching for services such as App Engine and Cloud Storage.

Google Cloud's PoPs connect to data centers through Google-owned fiber. This unimpeded connection means that Google Cloud-based applications have fast, reliable access to all of the services on Google Cloud.

Google's physical network



Google Cloud and AWS use PoPs in different ways.

Edge Locations: https://cloud.google.com/vpc/docs/edge-locations
These sites are sometimes referred to as points of presence or PoPs.
Cache Locations: https://cloud.google.com/cdn/docs/locations

AWS uses PoPs referred to as Edge Locations to provide CDN services through CloudFront, DNS services through Route 53, and firewall services through AWS WAF and AWS Shield. AWS also provides access to PoPs referred to as Local Zones as an extension of regions with limited subsets of services such as compute and storage for customers that have workloads that require very low latency for end users.

Importance of choosing a zone and region

Handling failures

Google Cloud is designed to help you distribute your resources across multiple zones and regions to minimize the risk of failures caused by physical outages.

Decreased network latency

To decrease network latency, choose a region or zone that is close to your point of service.

Data security and compliance

Google offers regional resources and multiregional resources to provide the necessary flexibility to meet network security requirements around data transfer.

The first reason relates to handling failures.
Distribute your resources across multiple zones and regions to tolerate outages. Google Cloud designs zones to minimize the risk of correlated failures caused by physical infrastructure outages like power, cooling, or networking.

So, if a zone becomes unavailable, you can transfer traffic to another zone in the same region to keep your services running.

Similarly, you can mitigate the impact of a region outage on your application by running backup services in a different region.

The second reason is to decrease network latency.
To do this, you might want to choose a region or zone that is close to your point of service.

For example, if you mostly have customers on the East Coast of the US, then you might want to choose a primary region and zone that is close to that area and a backup region and zone that is also close by.

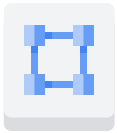
The third and final reason is for data security and compliance.

Some Google Cloud customers have concerns about the transfer of data across different regions and geographies, or must meet compliance requirements for the location of their data.

With this in mind, Google offers regional resources and multiregional resources to provide the necessary flexibility to meet network requirements.

There are several ways to connect to Google Cloud, including Cloud VPN, Interconnect, and peering. You'll learn more about these options later in the module.

VPC objects



Virtual Private Cloud

- Projects (Accounts)
- Networks (VPC)
  - Default, auto mode, custom mode
- Subnetworks
- Regions
- Availability Zones
- IP addresses
  - Internal, external, range
- Virtual machines (VMs)
- Routes (Route tables)
- Firewall rules (Network Security Groups and Application Security Groups)

AWS has the equivalent of External IP addresses in Google Cloud. In AWS, these are known as Public IP addresses.

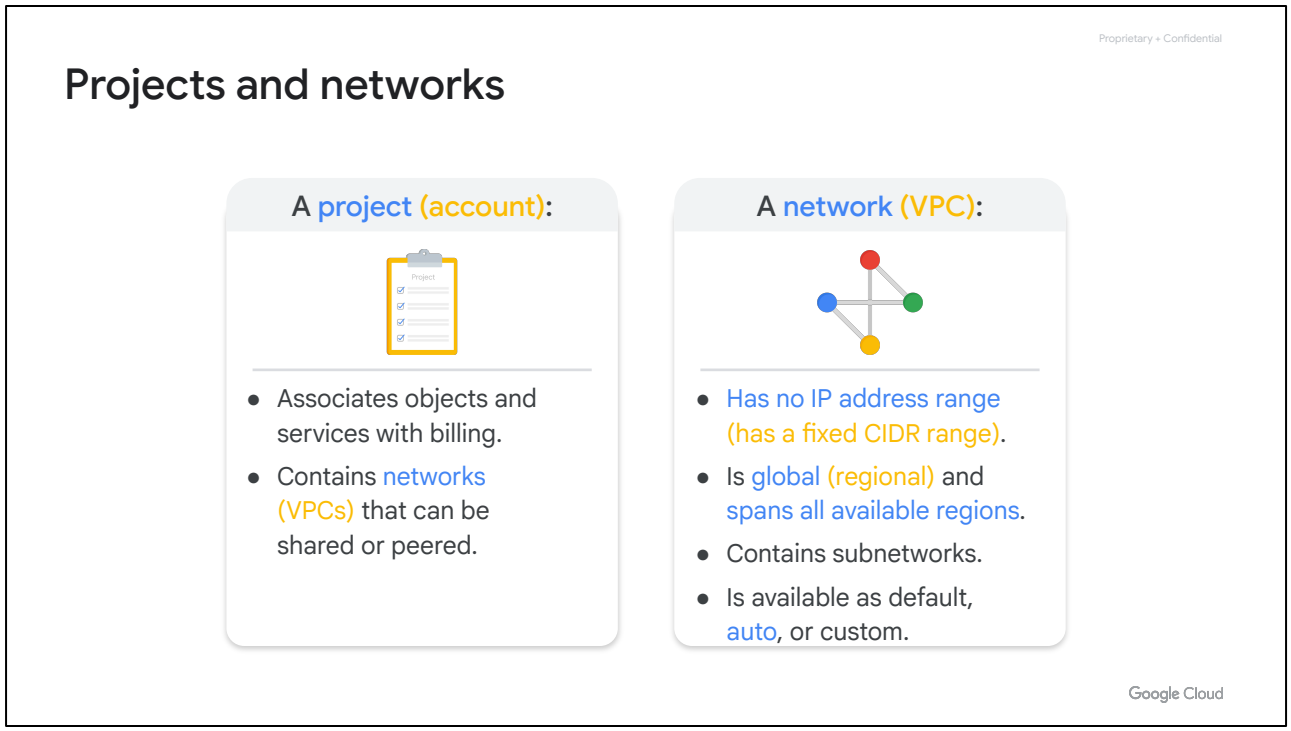
Source: Architecting with Compute Engine slides.

With Google Cloud, you can provision your Google Cloud resources, connect them to each other, and isolate them from each other in a Virtual Private Cloud. You can also define fine-grained networking policies within Google Cloud, and between Google Cloud and on-premises or other public clouds. Essentially, VPC is a comprehensive set of Google-managed networking objects, which we will explore in detail throughout this module.

Let me give you a high-level overview of these objects:

- Projects are going to encompass every single service that you use, including networks.
- Networks come in three different flavors: Default, auto mode, and custom mode.
- Subnetworks allow you to divide or segregate your environment.

- Regions and zones represent Google's data centers, and they provide continuous data protection and high availability.
- VPC provides IP addresses for internal and external use, along with granular IP address range selections.
  As for virtual machines, in this module we will focus on configuring VM instances from a networking perspective.
- We'll also go over routes and firewall rules.



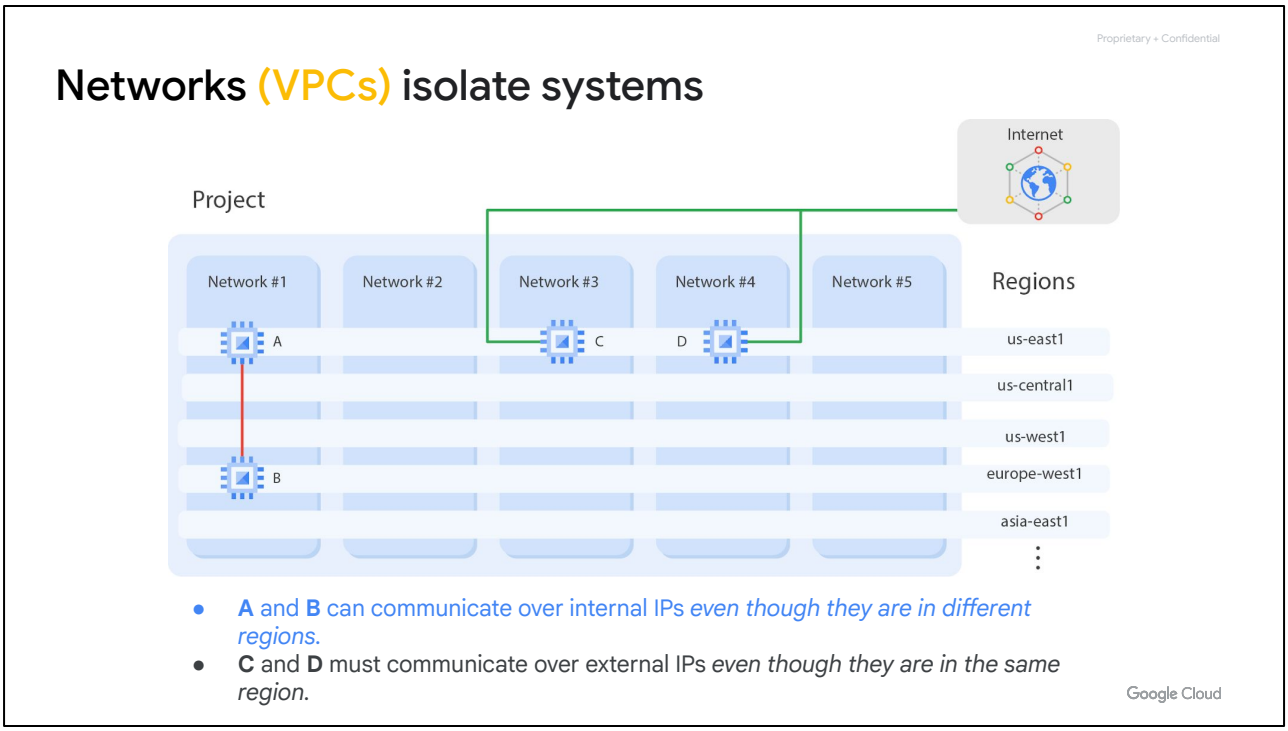
Source: Architecting with Compute Engine slides.

Projects are the key organizer of infrastructure resources in Google Cloud. A project associates objects and services with billing. Now, it's unique that projects actually contain entire networks. The default quota for each project is 5 networks, but you can simply request additional quota using the Cloud Console. These networks can be shared with other projects, or they can be peered with networks in other projects, both of which we will cover later in the Architecting with Google Compute Engine course.

These networks do not have IP ranges but are simply a construct of all of the individual IP addresses and services within that network. Google Cloud's networks are global, spanning all available regions across the world that we showed earlier. So, you can have one network that literally exists anywhere in the world—Asia, Europe, Americas—all simultaneously.
Inside a network, you can segregate your resources with regional subnetworks.

We just mentioned that there are different types of networks: default, auto, and custom. Let's explore these types of networks in more detail.

[More information on setting up a VPC:
https://www.youtube.com/watch?v=cNb7xKyya5c]



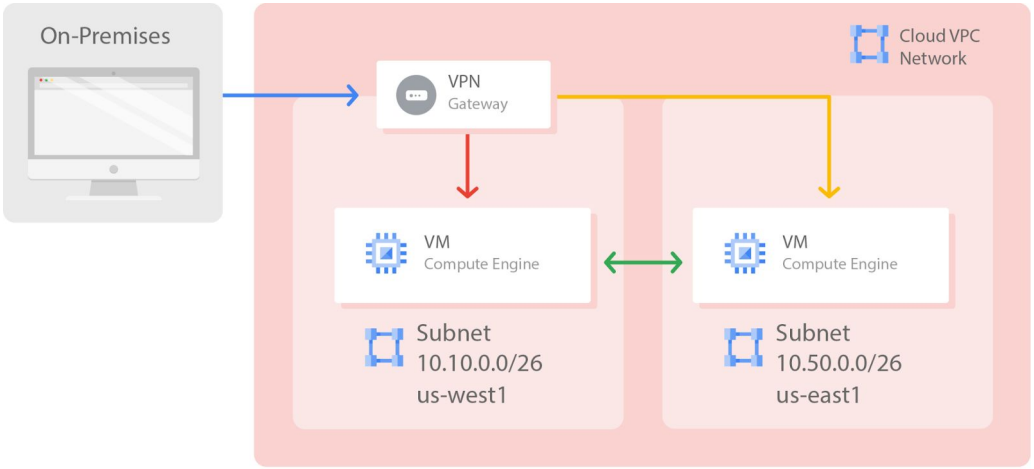
Source: Architecting with Compute Engine slides.

On this slide, we have an example of a project that contains 5 networks. All of these networks span multiple regions across the world, as you can see on the right.

Each network contains separate virtual machines: A, B, C, and D. Because VMs A and B are in the same network, network 1, they can communicate using their internal IP addresses, even though they are in different regions. Essentially, your virtual machines, even if they exist in different locations across the world, take advantage of Google's global fiber network. Those virtual machines appear as though they're sitting in the same rack when it comes to a network configuration protocol.

VMs C and D, however, are not in the same network. Therefore, by default, these VMs must communicate using their external IP addresses, even though they are in the same region. The traffic between VMs C and D isn't actually touching the public internet, but is going through the Google Edge routers. This has different billing and security ramifications that we will explore later.

Google's VPC is global (AWS is regional)

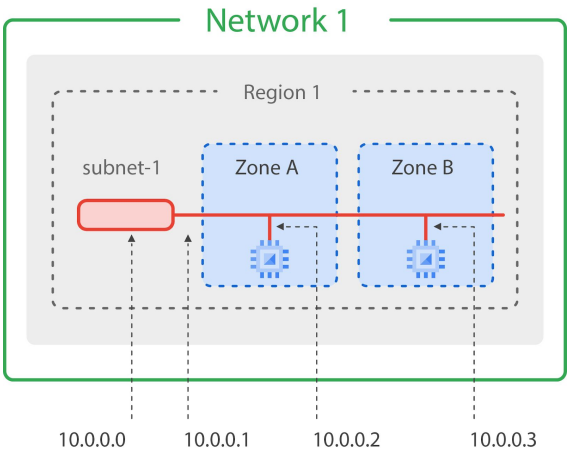


Source: Architecting with Compute Engine slides.

Because VM instances within a VPC network can communicate privately on a global scale, a single VPN can securely connect your on-premises network to your Google Cloud network, as shown in this diagram. Even though the two VM instances are in separate regions (us-west1 and us-east1), they leverage Google's private network to communicate between each other and to an on-premises network through a VPN gateway.

This reduces cost and network management complexity.

Subnetworks cross zones (unique to an AZ)



- VMs can be on the same subnet but in different zones.
- A single firewall rule (security group) can apply to both VMs.
- Private IPv4 (RFC 1918), dual-stack IPv4 and IPv6 (and IPv6-only) subnets.

Source: Architecting with Compute Engine slides.

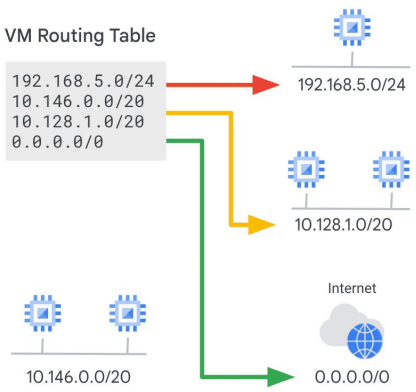
We mentioned that subnetworks work on a regional scale. Because a region contains several zones, subnetworks can cross zones.

This slide has a region, region 1, with two zones, zones A and B. Subnetworks can extend across these zones within the same region, such as, subnet-1. The subnet is simply an IP address range, and you can use IP addresses within that range. Notice that the first and second addresses in the range, .0 and .1, are reserved for the network and the subnet's gateway, respectively. This makes the first and second available addresses .2 and .3, which are assigned to the VM instances. The other reserved addresses in every subnet are the second-to-last address in the range and the last address, which is reserved as the "broadcast" address. To summarize, every subnet has four reserved IP addresses in its primary IP range.

Now, even though the two virtual machines in this example are in different zones, they still communicate with each other using the same subnet IP address. This means that a single firewall rule can be applied to both VMs, even though they are in different zones.

Routing tables: Google Cloud

- In Google Cloud, the routing table defines routes at the VPC network level for both the default and subnet route.
- Routes apply to traffic that is egressing from a VM and forward traffic to the most specific route.
- Routes have the destination in CIDR notation.
- Routes match packets by destination IP address when there is a matching firewall rule.



In Google Cloud, the routing table defines routes at the VPC network level.

When a new VPC network is created, two system-generated routes are created: the default route and a subnet route that defines paths to each subnet in the VPC network.

In Google Cloud, system-generated routes apply to all virtual machines in a VPC network.

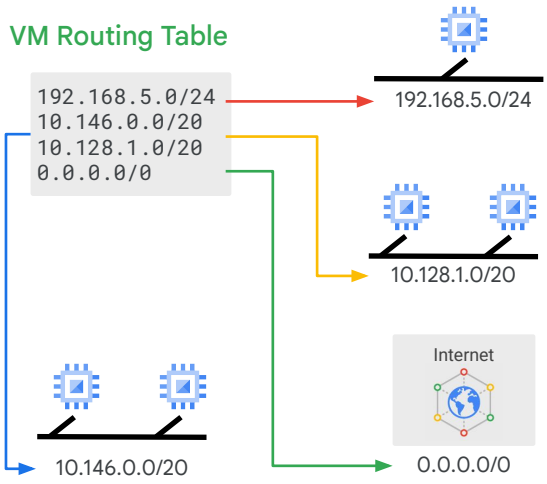
A route is created when a network is created, enabling traffic delivery from “anywhere.” Also, a route is created when a subnet is created. This is what enables VMs on the same network to communicate.

Some characteristic of routes include the following:

- Routes apply to traffic that is egressing from a VM.
- They forward traffic to the most specific route.
- Routes have the destination in CIDR notation.
- And they match packets by destination IP address when there is a matching firewall rule.

Routes (route tables) map traffic to destination networks

- Apply to traffic egressing a VM.
- Forward traffic to most specific route.
- Are created when a subnet is created.
- Enable VMs on same network (VPC) to communicate.
- Destination is in CIDR notation.
- Traffic is delivered only if it is allowed by a firewall rule (security group and NACL).
- Internet route part of the default routes (internet gateway required)
  - All (public) subnets have internet access if external (elastic) IP assigned
  - Tags can modify routing tables



Source: Architecting with Compute Engine slides.

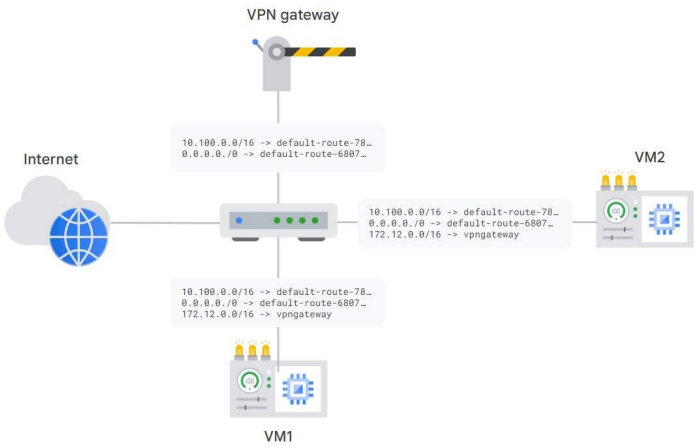
Routes match packets by destination IP address. However, no traffic will flow without also matching a firewall rule.

A route is created when a network is created, enabling traffic delivery from “anywhere.” Also, a route is created when a subnet is created. This is what enables VMs on the same network to communicate.

This slide shows a simplified routing table, but let’s look at this in more detail.

Instance routing tables

- Each route in the routes collection may apply to one or more instances.
- A route applies to an instance if the network and instance tags match.
- Compute Engine uses the routes collection to create individual read-only routing tables for each instance.



Each route in the routes collection may apply to one or more instances.

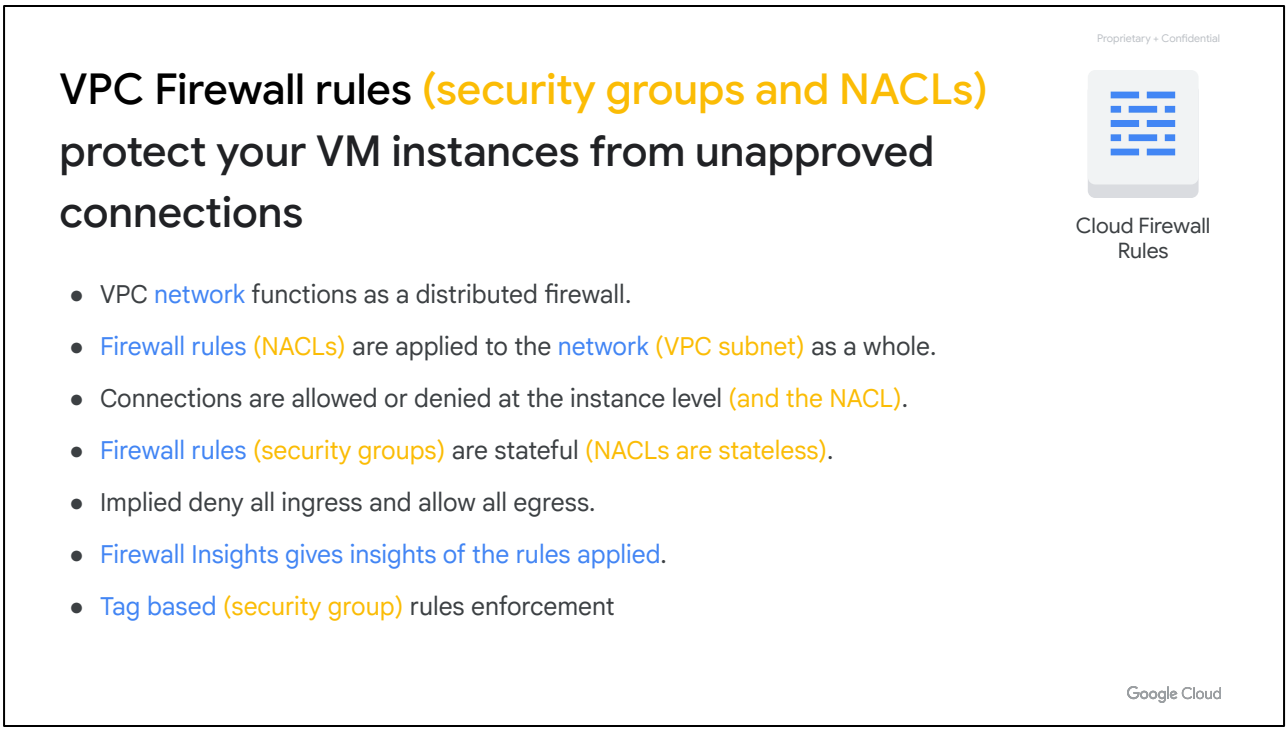
A route applies to an instance if the network and instance tags match.

If the network matches and there are no instance tags specified, the route applies to all instances in that network.

Compute Engine then uses the routes collection to create individual read-only routing tables for each instance.

In the diagram, every virtual machine instance in the network is directly connected to this router.

All packets leaving a virtual machine instance are first handled at this layer before they are forwarded to their next hop. The virtual network router selects the next hop for a packet by consulting the routing table for that instance.



Source: Architecting with Compute Engine slides.

Google Cloud firewall rules protect your virtual machine instances from unapproved connections, both inbound and outbound, known as ingress and egress, respectively. Essentially, every VPC network functions as a distributed firewall.

Although firewall rules are applied to the network as a whole, connections are allowed or denied at the instance level. You can think of the firewall as existing not only between your instances and other networks, but between individual instances within the same network.

Google Cloud firewall rules are stateful. This means that if a connection is allowed between a source and a target or a target and a destination, all subsequent traffic in either direction will be allowed. In other words, firewall rules allow bidirectional communication once a session is established.

Also, if for some reason, all firewall rules in a network are deleted, there is still an implied "Deny all" ingress rule and an implied "Allow all" egress rule for the network.

A firewall rule (Security Group [SG] or Network ACL [NACL]) is composed of...

Parameter	Details
direction	Inbound connections are matched against ingress (inbound) rules only.
	Outbound connections are matched against egress (outbound) rules only.
source or destination	For the ingress direction, sources can be specified as part of the rule with IP addresses (IPv4 or v6), source tags or a source service account (Prefix list or security group [SGs only]).
	For the egress direction, destinations can be specified as part of the rule with one or more ranges of IP addresses (Prefix list or security group [SGs only]).
protocol and port	Any rule can be restricted to apply to specific protocols only or specific combinations of protocols and ports only.
action	To allow or deny (deny for NACLs only) packets that match the direction, protocol, port, and source or destination of the rule.
priority	Governs the order in which rules are evaluated; the first matching rule is applied (rule number in NACLs, N/A for SGs).
Rule assignment	All rules are assigned to all instances, but you can assign certain rules to certain instances only.

Source: Architecting with Compute Engine slides.

Speaker Notes:
One best practice: Use hierarchical firewall policy rules to block traffic that should never be allowed at an organization or folder level

You can express your desired firewall configuration as a set of firewall rules. Conceptually, a firewall rule is composed of the following parameters:

- The **direction** of the rule. Inbound connections are matched against ingress rules only, and outbound connections are matched against egress rules only.
- The **source** of the connection for ingress packets, or the **destination** of the connection for egress packets.
- The **protocol** and **port** of the connection, where any rule can be restricted to apply to specific protocols only or specific combinations of protocols and ports only.
- The **action** of the rule, which is to allow or deny packets that match the direction, protocol, port, and source or destination of the rule.
- The **priority** of the rule, which governs the order in which rules are evaluated. The first matching rule is applied.
- The **rule assignment**. By default, all rules are assigned to all instances, but

- you can assign certain rules to certain instances only.

For more information on firewall rule components, refer to the documentation page.

Let’s look at some Google Cloud firewall use cases for both egress and ingress.

Firewall rule definition: AWS and Google Cloud

AWS

- Network Firewalls can regulate a much more expansive scope of network traffic and manages the traffic entering and exiting through the VPC.
- Security groups are associated with resources, providing much finer-grained control.

Google Cloud

- All VPC networks are distributed firewalls, and can be configured to apply to particular compute resources.
- You can configure your required firewall as a set of firewall rules, or parameters (outlined in the additional resource).
  - **Additional resource:** Firewall rules parameters

In AWS, security groups are narrower in scope than AWS Network Firewalls.

- AWS Network Firewalls can regulate a much more expansive scope of network traffic because the firewall is set at the VPC level, and manages the traffic entering and exiting through the VPC.
- Security groups are associated with resources, providing much finer-grained control.

AWS Web Application Firewall can be associated with CloudFront, Application Load Balancer (ALB), and API Gateway. AWS Shield is a managed Distributed Denial of Service (DDoS) protection service that is available on all AWS services. Amazon Route 53 Resolver DNS Firewall can be associated with VPCs to provide filtering for outbound DNS queries.

In Google Cloud, every VPC network essentially functions as a distributed firewall. Similar to routes, firewall rules are defined at the network level. However, it is also possible to configure them to apply to particular compute resources in the VPC network, offering a high degree of flexibility.

You can express your desired firewall configuration as a set of firewall rules, composed of parameters. For an overview of Google Cloud firewall parameters, refer to the table in the additional resource, “Firewall rules parameters”.

Networking: GCP vs AWS

Networking service	Google Cloud VPC	Amazon VPC
Virtual networks	VPC networks (global)	VPCs (regional)
IP address ranges	Subnets (regional)	Subnets (Availability Zone)
Routing entries	Routes (global)	Routes (regional)
Security boundaries	Firewall rules (global)	Network Access Control Lists (VPC) Security Groups (VPC) AWS Network Firewall (regional)

Main differences:

- Regional vs global (VPC, Subnet)
- When creating an Amazon VPC, you must specify a custom private IP address space.
- In AWS, it is not possible to reduce the IPv4 range of the subnet after it is created.

_____

Let's summarise the networking services that AWS and Google Cloud provide. These services vary in terms of their scope, as indicated in the table in parentheses.

The key takeaways are that AWS maintains the scope of its networks to a regional level; Google Cloud VPC networks are global, and subnets span regions.

Global networks offer farther out-of-the-box reach, which means you can create a single private network that is global without having to connect multiple private networks and manage those spaces separately.

You can also define multiple networks per project for added flexibility.

Proprietary + Confidential

Network Service Tiers (no AWS equivalent)

Optimize your network for performance or cost

- **Premium**
  - Delivers traffic on Google's premium backbone
  - Optimized for performance, Higher cost
  - Backed by SLA
- **Standard**
  - Delivers traffic using regular ISP networks
  - Optimized for cost, Lower performance
  - **Available for regional load balancing only**, not global
  - No SLA

Network Service Tiers

Google Cloud

Network Service Tiers
https://cloud.google.com/network-tiers

Network Service Tiers Release Notes
https://cloud.google.com/network-tiers/docs/release-notes

GA March 28, 2019

Network Service Tiers Overview
https://cloud.google.com/network-tiers/docs/overview

Networking: Technical Deep Dive | Technical | Y20
Slide 6
https://docs.google.com/presentation/d/1-AxfQ1LyBVy5XsWUk2nbG95Vnlx6oxD7iwc9QAqfyEI/edit#slide=id.g9637aec922_0_2748

GCP Networking Portfolio overview - LATAM (slides) | Partners | Pre-Sales | Y20
Slide 46
https://docs.google.com/presentation/d/1UGANDG787A11YLP1cBsDt45bUpMxj27YpBshEZUTpAc/edit#slide=id.g78d3264ce9_0_12740

Blog Post

Google Cloud networking in depth: Understanding Network Service Tiers (May 15, 2019)
https://cloud.google.com/blog/products/networking/google-cloud-networking-in-depth-understanding-network-service-tiers

Graphic source:
GCP Networking Portfolio overview - LATAM (slides) | Partners | Pre-Sales | Y20
Slide 46
https://docs.google.com/presentation/d/1UGANDG787A11YLP1cBsDt45bUpMxj27YpBshEZUTpAc/edit#slide=id.g78d3264ce9_0_12740

Premium Tier

https://cloud.google.com/network-tiers/docs/overview#premium_tier
Premium Tier delivers traffic from external systems to Google Cloud resources by using Google's low latency, highly reliable global network. This network consists of an extensive private fiber network with over 100 points of presence (PoPs) around the globe. This network is designed to tolerate multiple failures and disruptions while still delivering traffic.

Premium Tier supports both regional external IP addresses and global external IP addresses for VM instances and load balancers. All global external IP addresses must use Premium Tier. Applications that require high performance and availability, such as those that use HTTP(S), TCP proxy, and SSL proxy load balancers with backends in more than one region, require Premium Tier.

Premium Tier is ideal for customers with users in multiple locations worldwide who need the best network performance and reliability.

Standard Tier
https://cloud.google.com/network-tiers/docs/overview#standard_tier

Standard Tier delivers traffic from external systems to Google Cloud resources by routing it over the internet. It leverages the double redundancy of Google's network only up to the point where Google's data center connects to a peering PoP. Packets that leave Google's network are delivered using the public internet and are subject to the reliability of intervening transit providers and ISPs. Standard Tier provides network quality and reliability comparable to that of other cloud providers.

Standard Tier is priced lower than Premium Tier because traffic from systems on the internet is routed over transit (ISP) networks before being sent to VMs in your VPC network or regional Cloud Storage buckets. Standard Tier outbound traffic normally exits Google's network from the same region used by the sending VM or Cloud Storage bucket, regardless of its destination. In rare cases, such as during a network event, traffic might not be able to travel out the closest exit and might be sent out another exit, perhaps in another region.

Standard Tier offers a lower-cost alternative for the following use cases:

- You have applications that are not latency or performance sensitive.
- You're deploying VM instances or using Cloud Storage that can all be within a single region.

Network pricing (subject to change)

Traffic type	Price
Ingress	No charge
Egress to the same zone (internal IP address)	No charge
Egress to Google products (YouTube, Maps, Drive)	No charge
Egress to a different Google Cloud (AWS) service (within same region; exceptions)	No charge
Egress between zones in the same region (per GB)	$0.01
Egress to the same zone (external IP address, per GB)	$0.01
Egress between regions within the US and Canada (per GB)	$0.01
Egress between regions, not including traffic between US regions	Varies by region

Source: Architecting with Compute Engine slides.

This table is from the Compute Engine documentation, and it lists the price of each traffic type.

First of all, ingress or traffic coming into Google Cloud's network is not charged, unless there is a resource such as a load balancer that is processing ingress traffic. Responses to requests count as egress and are charged.

The rest of this table lists egress or traffic leaving a virtual machine. Egress traffic to the same zone is not charged, as long as that egress is through the internal IP address of an instance. Also, egress traffic to Google products, like YouTube, Maps, Drive, or traffic to a different Google Cloud service within the same region is not charged for.

However, there is a charge for egress between zones in the same region, egress within a zone if the traffic is through the external IP address of an instance, and egress between regions.

As for the difference in egress traffic to the same zone, Compute Engine cannot determine the zone of a virtual machine through the external IP address. Therefore,

this traffic is treated like egress between zones in the same region.

Also, there are some exceptions, and pricing can always change, so please refer to the documentation page.

IP addresses in GCP and AWS: differences

IP address type	AWS	Google Cloud
Publicly routable	Public IP address Elastic IP address	External IP address (Ephemeral) External IP address (Static)
Not publicly routable	Private IP address	Internal/Private IP address

There are several key differences in how IP addresses are used between AWS and Google Cloud. These differences are summarized in the table on screen.

AWS has private, public, and elastic IP addresses.

- Private IP addresses communicate with other resources in AWS and networks that are connected with AWS. They can connect to the internet for outbound communication using a network address translation (NAT) gateway in the VPCs public subnet, making a potential security breach more difficult.
- Elastic IP addresses are static, public IP addresses that can be associated and dissociated with instances or network interfaces.
- In AWS, a public IP address is distinct from a private IP address in that it allows both inbound and outbound communication.

In Google Cloud, you have a similar concept but the terminology is different.

The Google Cloud IP address is more nuanced. IP addresses are distinguished by whether or not the address is routed publicly and whether or not the address can be reached from the internet.

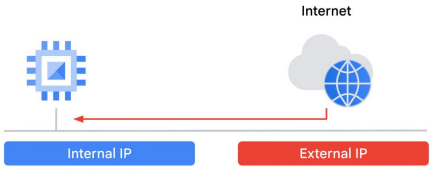
For example, a public IP address type can be used as an internal address configured as the primary or secondary IPv4 address range of a subnet in a VPC network.

This is sometimes known as a privately used public IP address.

Internal and external IP addresses in Google Cloud

Internal IP address

- Allocated from subnet range to VMs by DHCP
- DHCP lease is renewed every 24 hours
- VM name and IP address are registered with network-scoped DNS



External IP address

- Assigned from pool (ephemeral) or reserved (static)
- Bring Your Own IP address (BYOIP).
- VM doesn't know the external IP, so it's mapped to the internal IP.

Let's examine how internal and external IP addresses are assigned in Google Cloud.

An internal IP address is assigned as follows:

- It's allocated from subnet range to VMs by DHCP.
- The DHCP lease is renewed every 24 hours.
- And the VM name and IP address is registered with network-scoped DNS.

An external IP address is:

- You can assign an external IP address if your device or your machine is externally facing. That external IP address can be assigned from a pool, making it ephemeral.Or it can be assigned a reserved external IP address, making it static. Regardless of whether you use an ephemeral or static IP address, the external address is unknown to the OS of the VM..
- You must bring Your Own IP address (BYOIP).
- And the VM doesn't know the external IP address, so it's mapped to the VM's internal IP address transparently by VPC.

Cloud NAT and Private Google Access

AWS	Google Cloud
AWS NAT Gateway	Cloud NAT
AWS PrivateLink	Private Google Access

In AWS or Google Cloud, you need to set up resources to deliver services to your customers, partners, or colleagues.

These resources will need to create outbound connections to the internet so that they can be accessed.

Let's review the services in AWS.

AWS Internet Gateway is the resource that enables resources in a VPC to connect to the internet if the resource has a public IP address.

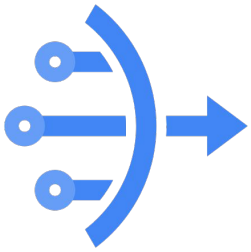
AWS NAT Gateway is a managed network address translation (NAT) service that enables resources in a private subnet to connect to the internet. Using this service means that the AWS resource does not need to have a public IP address to have outbound connectivity. AWS NAT Gateways can be public or private. A private NAT Gateway will not have a public IP associated with it. Private NAT Gateways enable resources to connect to other VPCs, but not the public internet.

AWS PrivateLink is an AWS product that enables the creation of VPC endpoints. VPC endpoints can be used to privately access AWS resources without a public IP address, Internet Gateway, or NAT Gateway.

In this section, you will learn how you can use the Google Cloud NAT service and Private Google Access to help set up networking infrastructure in Google.

Google Cloud NAT (NAT Gateway): Overview (1/2)

- Google Cloud NAT is a distributed, software-defined managed service.
- Cloud NAT provides outgoing connectivity for:
  - Compute Engine VM instances without external IP addresses
  - Private GKE clusters
  - Cloud Run instances through Serverless VPC Access
  - Cloud Functions instances through Serverless VPC Access
  - App Engine standard environment instances through Serverless VPC Access



Google Cloud NAT is a distributed, software-defined managed service.

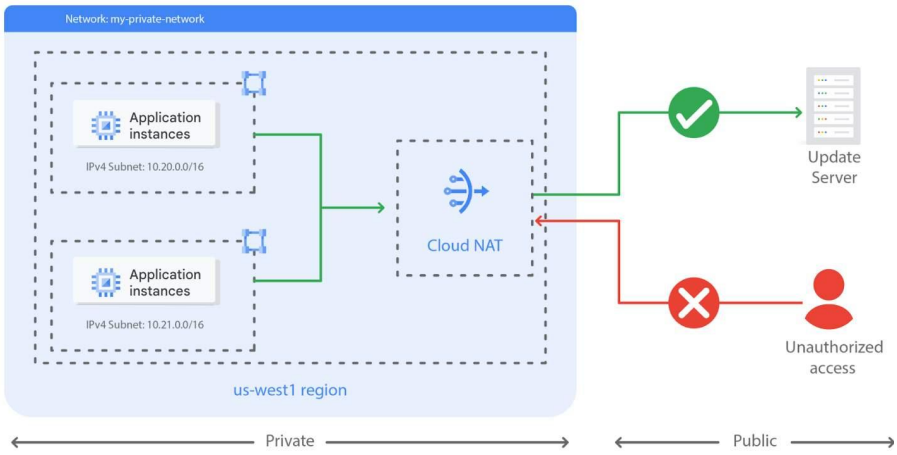
It configures the Andromeda software that powers your VPC network so that it provides source network address translation (source NAT or SNAT) for VMs without external IP addresses.

Cloud NAT also provides destination network address translation (destination NAT or DNAT) for established inbound response packets.

Cloud NAT provides outgoing connectivity for the following resources:

- Compute Engine VM instances without external IP addresses
- Private Google Kubernetes Engine (GKE) clusters
- Cloud Run instances through Serverless VPC Access
- Cloud Functions instances through Serverless VPC Access
- App Engine standard environment instances through Serverless VPC Access

Google Cloud NAT (NAT Gateway): Overview (2/2)



This diagram illustrates that Cloud NAT lets resources without external IP addresses create outbound connections to the internet.

Cloud NAT is Google's managed network address translation service. It lets you provision your application instances without public IP addresses, while also allowing them to access the internet in a controlled and efficient manner.

This means your private instances can access the internet for updates, patching, configuration management, and more.

In this diagram, Cloud NAT enables two private instances to access an update server on the internet, which is referred to as outbound NAT.

However, Cloud NAT does not implement inbound NAT.

In other words, hosts outside your VPC network cannot directly access any of the private instances behind the Cloud NAT gateway. This helps you keep your VPC networks isolated and secure.

Private Google Access (AWS PrivateLink)

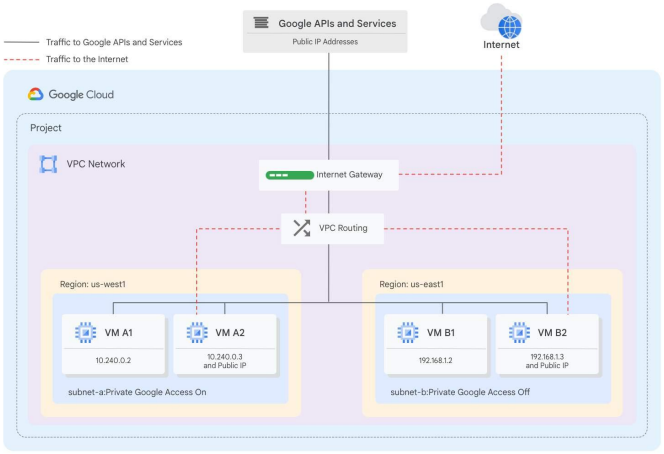
- When a Compute Engine VM lacks an external IP address assigned to its network interface, by default it can only send packets to other internal IP address destinations
- You can allow these VMs to connect to the Google managed services running on external IP addresses space e.g. Google Cloud Storage service.
- You enable Private Google Access on a subnet-by-subnet basis.

In Google Cloud, when a Compute Engine VM lacks an external IP address assigned to its network interface, by default it can only send packets to other internal IP address destinations.

You can allow these VMs to connect to the Google managed services running on external IP addresses space, such as Google Cloud Storage service.

For example, if your private VM instance needs to access a Cloud Storage bucket, you need to enable Private Google Access on a subnet-by-subnet basis.

Private Google Access (AWS PrivateLink): Example



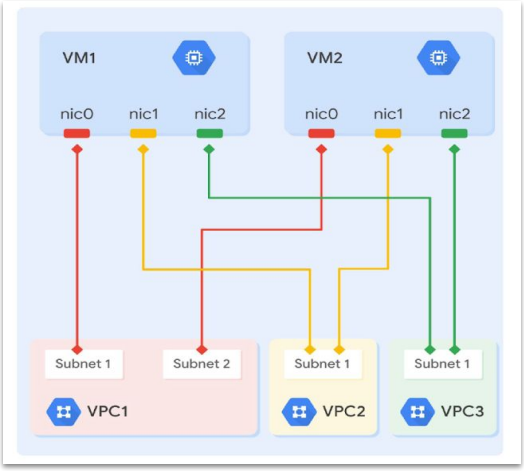
Additional resource: Comparison of AWS PrivateLink and Private Google Access

Let's explore an example of how Private Google Access affects a private VM instance's ability to access resources.

- Subnet-a has Private Google Access enabled. This allows VM A1 to access Google APIs and services, even though it has no external IP address.
- VM A2 has a public IP address. Private Google Access has no effect on instances that have external IP addresses.
- VM B1 has no public IP address, and it is in subnet-b where Google Private access is disabled. So it cannot access Google APIs and services.
- Subnet-b has Google Private Access disabled. However, VM B2 can still access Google APIs and services because it has an external IP address.

Multi-NIC VMs

- VMs have Multi-NIC support (8 max)
  - Each NIC must connect to a different VPC network
  - Allows communication between VPCs using private IPs
- Are other ways to accomplish private IP communication between VPCs, such as
  - VPC Peering
  - VPN
  - These will be discussed later



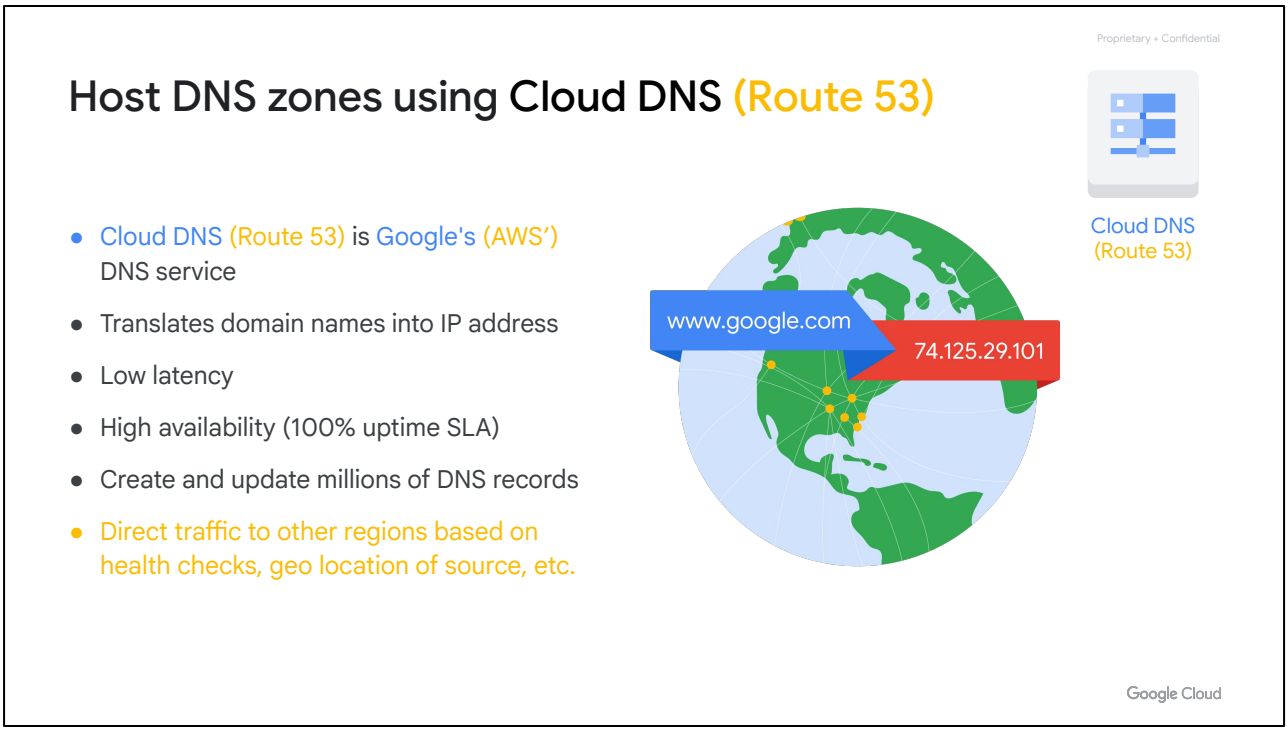
Creating instances with multiple network interfaces
https://cloud.google.com/vpc/docs/create-use-multiple-interfaces

Cloud DNS (Route 53)

- Managed DNS service that runs on the same infrastructure as Google (AWS)
- Low latency, high availability, and cost-effective way to make applications and services available to users
- The DNS information you publish is served from redundant locations around the world.
- Cloud DNS is programmable. You can publish and manage millions of DNS zones and records using the Google Cloud (AWS) console, the command-line interface, or the API.

Google Cloud offers **Cloud DNS** to help the world find them. It's a managed DNS service that runs on the same infrastructure as Google. It has low latency and high availability, and it's a cost-effective way to make your applications and services available to your users. The DNS information you publish is served from redundant locations around the world.

Cloud DNS is also programmable. You can publish and manage millions of DNS zones and records using the Google Cloud console, the command-line interface, or the API.



Source: Architecting with Compute Engine slides.

Cloud DNS is a scalable, reliable, and managed authoritative Domain Name System, or DNS, service running on the same infrastructure as Google. Cloud DNS translates requests for domain names like google.com into IP addresses.

Cloud DNS uses Google's global network of Anycast name servers to serve your DNS zones from redundant locations around the world, providing lower latency and high availability for your users. High availability is very important because if you can't look up a domain name, the internet might as well be down. That's why Google Cloud offers a 100% uptime Service Level Agreement, or SLA, for domains configured in Cloud DNS. For more information about this SLA, refer to the documentation page.

Cloud DNS lets you create and update millions of DNS records without the burden of managing your own DNS servers and software. Instead, you use a simple user interface, command-line interface, or API. For more information about Cloud DNS, refer to the documentation page.

Cloud CDN (CloudFront)

Content Delivery Network

- Use Google's globally distributed edge caches to cache content close to your users.
- Or use CDN Interconnect if you'd prefer to use a different CDN.

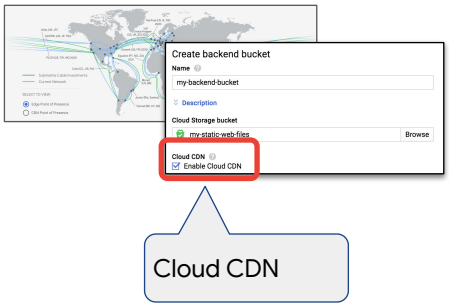
Source: Demo Template

Google has a global system of edge caches. You can use this system to accelerate content delivery in your application using Cloud CDN. Your customers will experience lower network latency, the origins of your content will experience reduced load, and you can save money too. Once you've set up HTTP(S) Load Balancing, simply enable Cloud CDN with a single checkbox.

There are lots of other CDNs out there, of course. If you are already using one, chances are, it is a part of Google Cloud's CDN Interconnect partner program, and you can continue to use it.

Google Cloud provides two CDN options

- Cloud CDN (CloudFront) caches regularly accessed static content
  - Optimized for serving static web assets such as CSS, JavaScript, or non-dynamic HTML files
- Media CDN (CloudFront) is a media delivery platform
  - Optimized for high-throughput egress workloads such as streaming video and large file downloads
- Both use Google's (AWS') globally distributed Edge locations to **cache content** close to your users
- **170+ locations**, with **single IP** across multiple regions



Choose a CDN product

Google Cloud

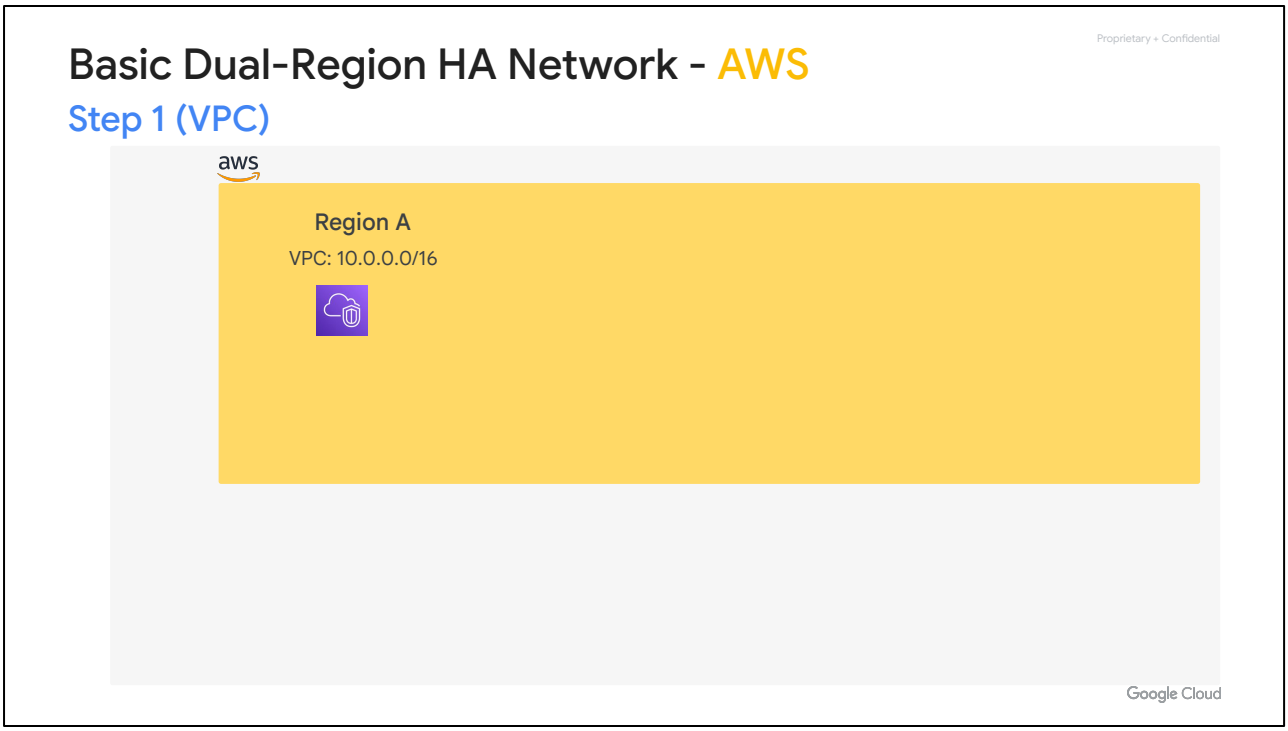
https://cloud.google.com/media-cdn/docs/choose-cdn-product

The load balancer can also be configured to leverage Google's content delivery network, called Cloud CDN.

By enabling Cloud CDN, you can cache content all over the world, closer to the user.

Introducing Media CDN—the modern extensible platform for delivering immersive experiences
https://cloud.google.com/blog/products/networking/introducing-media-cdn/

Choose a CDN product
https://cloud.google.com/media-cdn/docs/choose-cdn-product

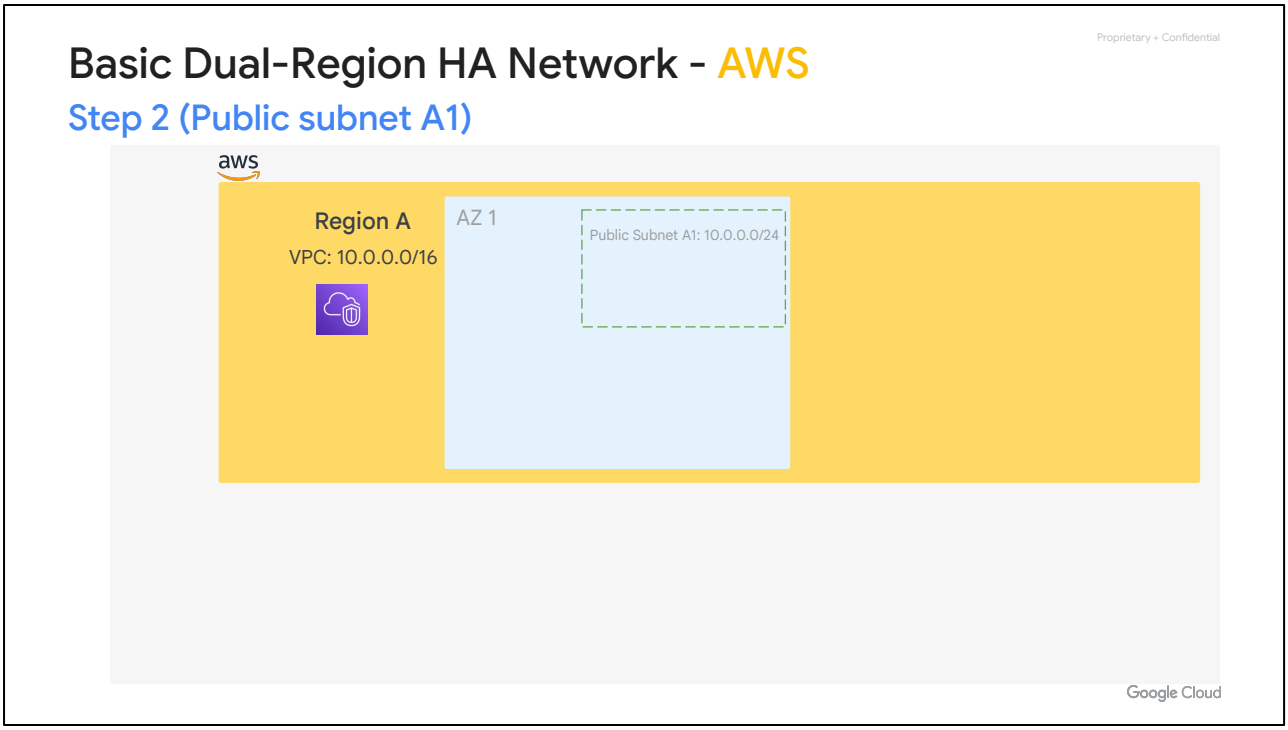


Minimum Network Resources for Region A

- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.



Minimum Network Resources for Region A

- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 3 (Public subnet A2)

Minimum Network Resources for Region A

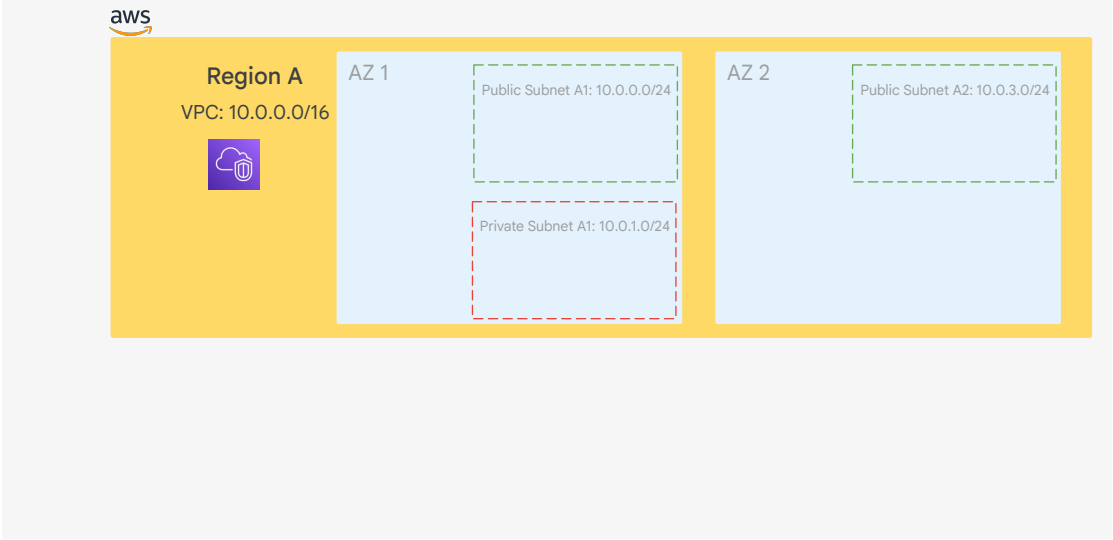
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 4 (Private subnet A1)



Minimum Network Resources for Region A

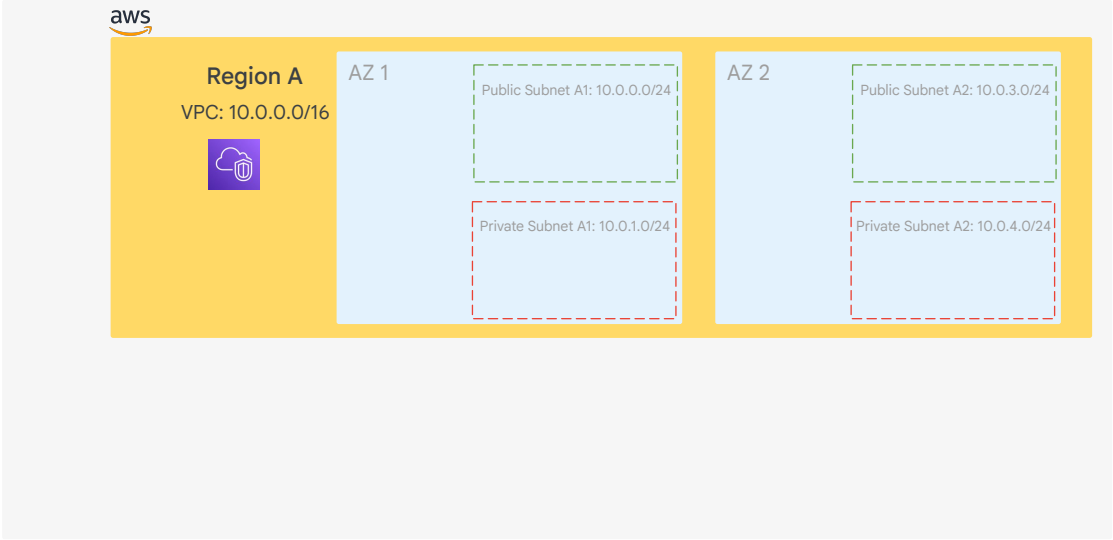
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 5 (Private Subnet A2)



Minimum Network Resources for Region A

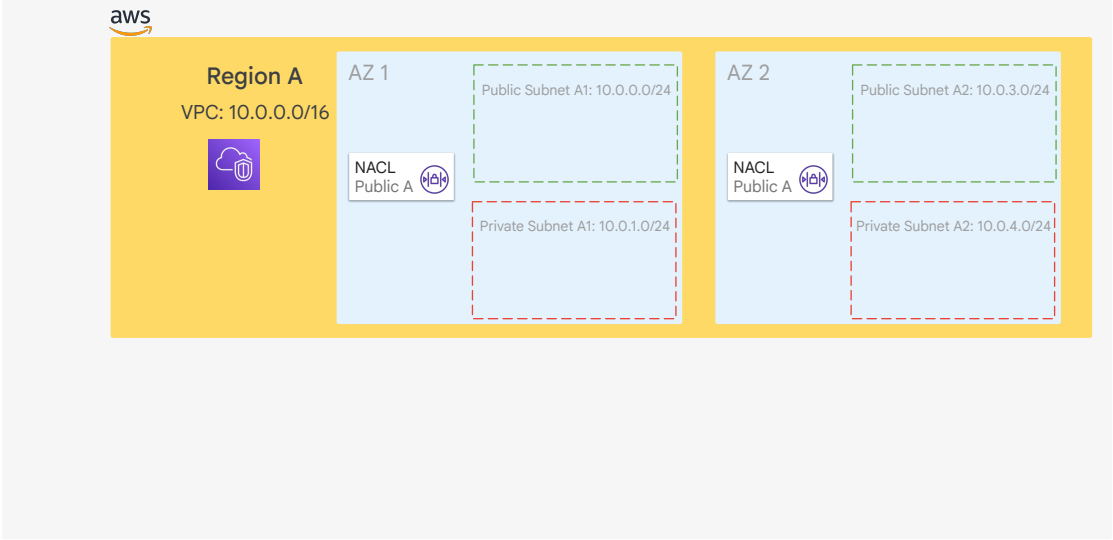
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 6 (Public NACL)



Minimum Network Resources for Region A

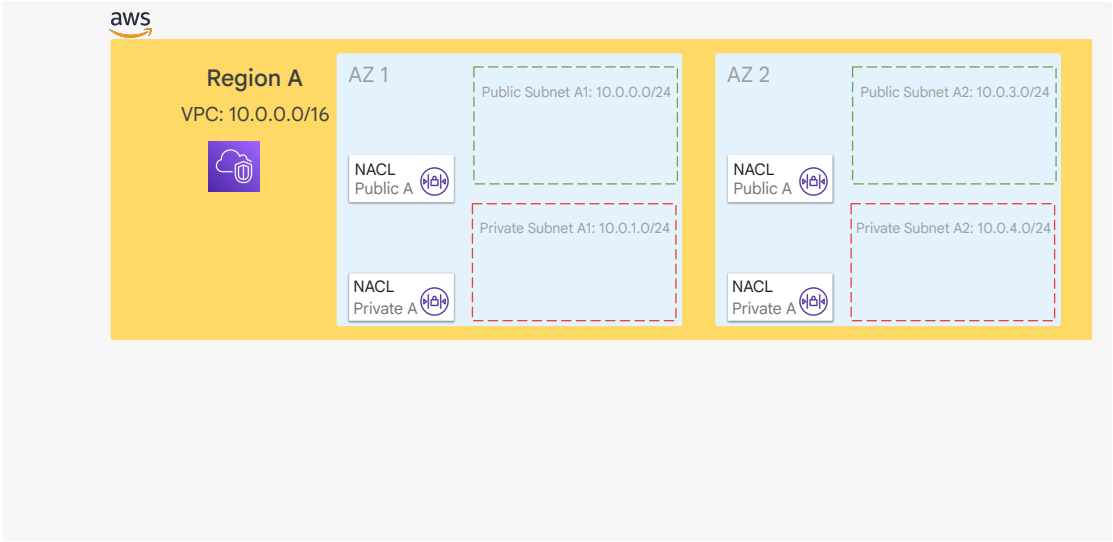
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 7 (Private NACL)



Minimum Network Resources for Region A

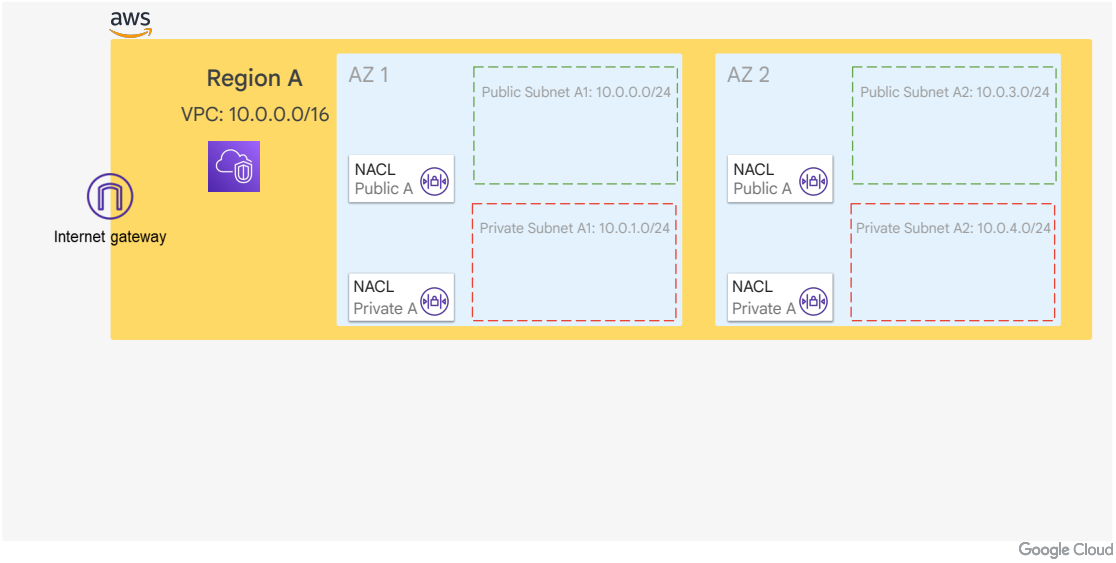
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 8: Internet Gateway



Minimum Network Resources for Region A

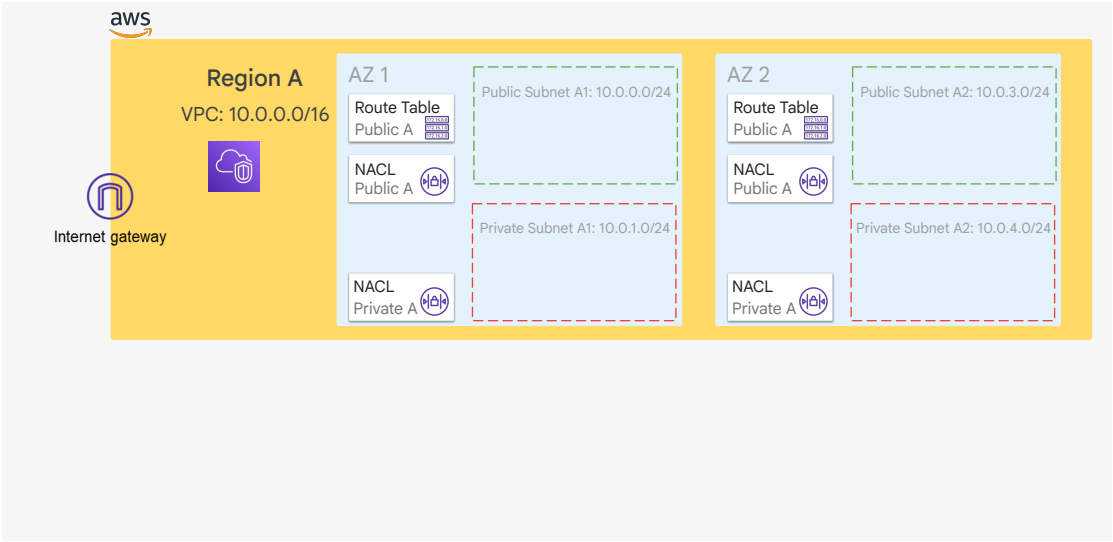
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 9: Public Route Table



Minimum Network Resources for Region A

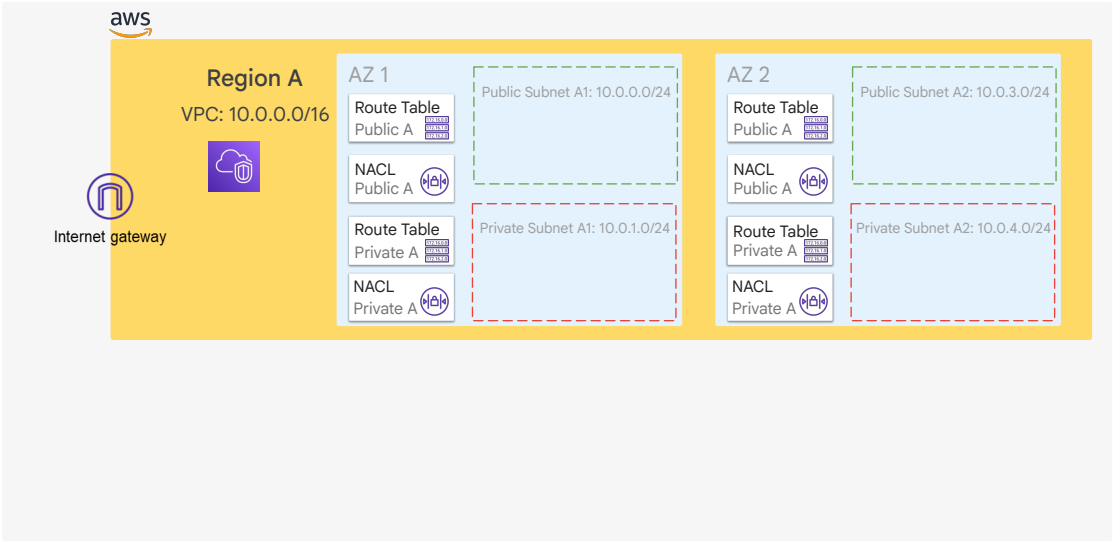
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 10: Private Route Table



Minimum Network Resources for Region A

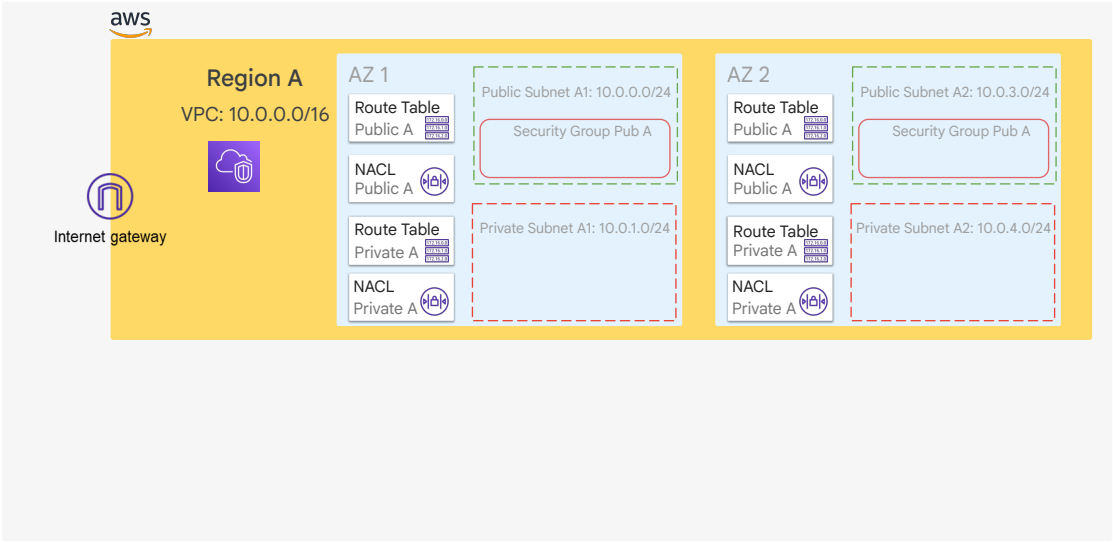
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 11: Public Security Group



Minimum Network Resources for Region A

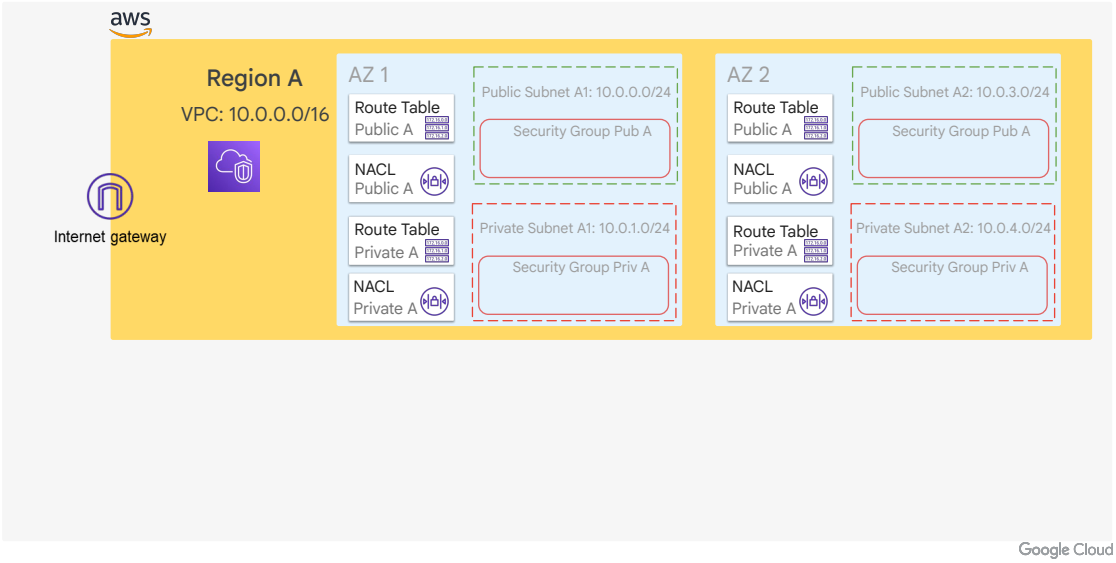
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 12: Private Security Group



Minimum Network Resources for Region A

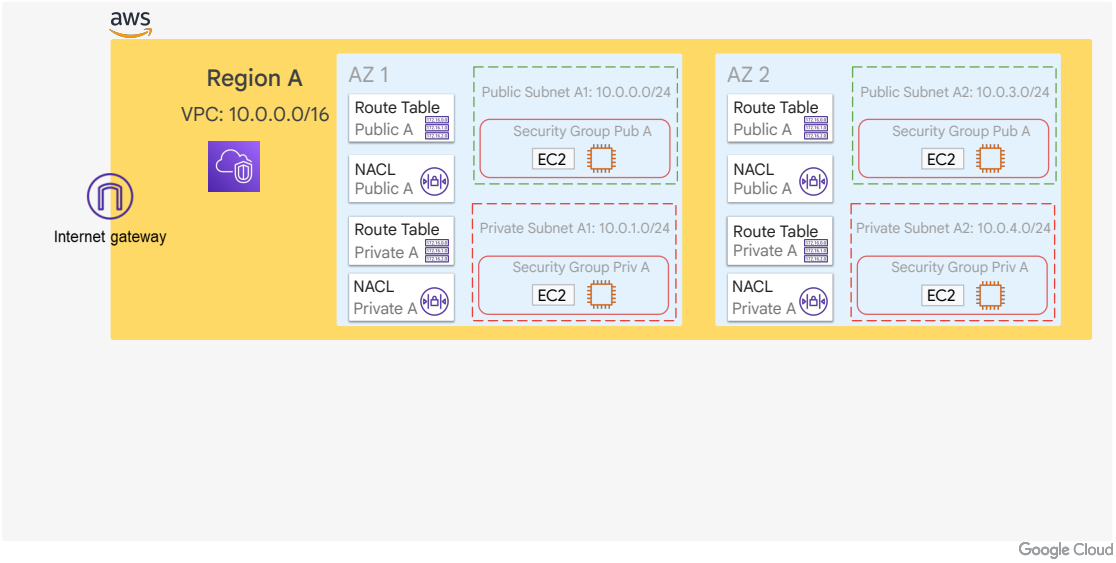
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 13: EC2s



Minimum Network Resources for Region A

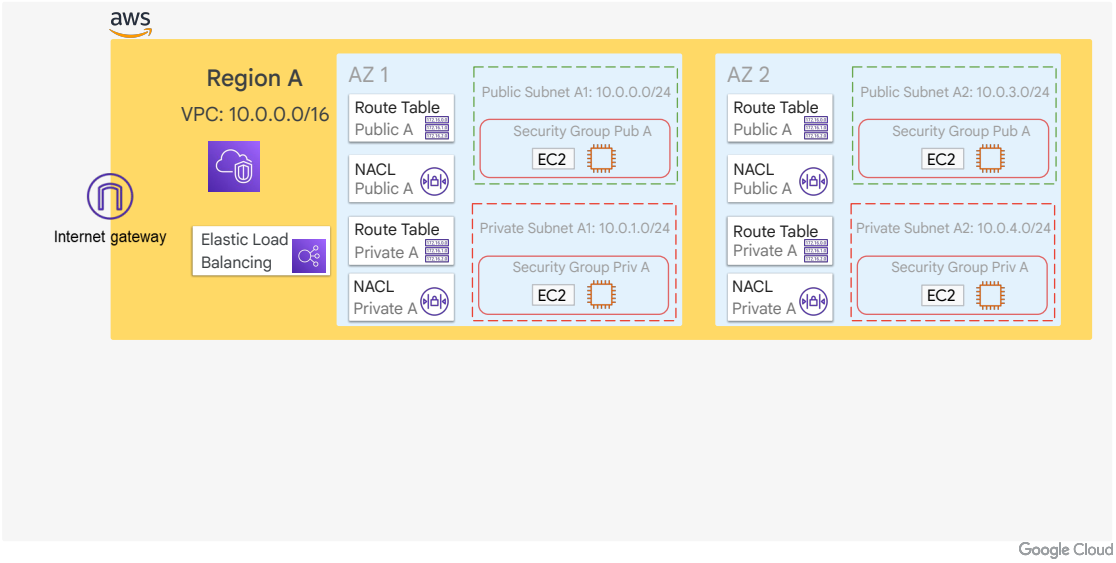
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 14: Elastic Load Balancer

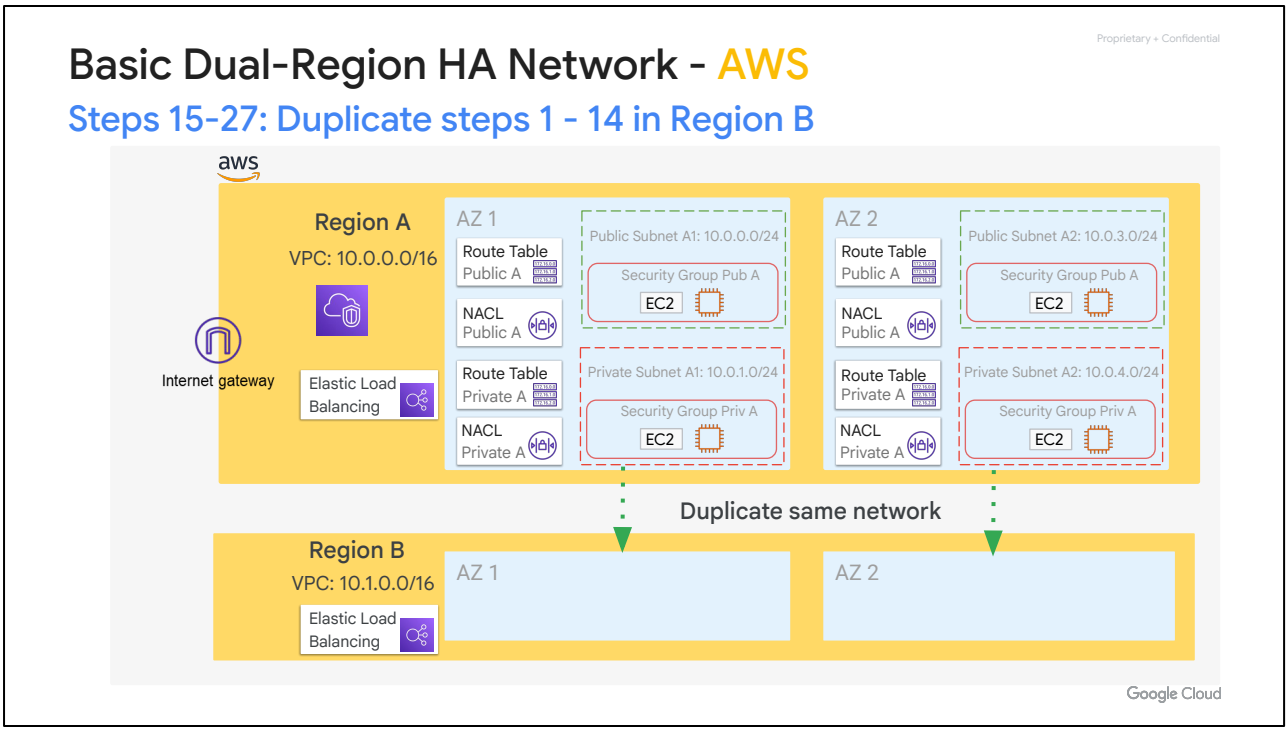


Minimum Network Resources for Region A

- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.



Minimum Network Resources for Region A

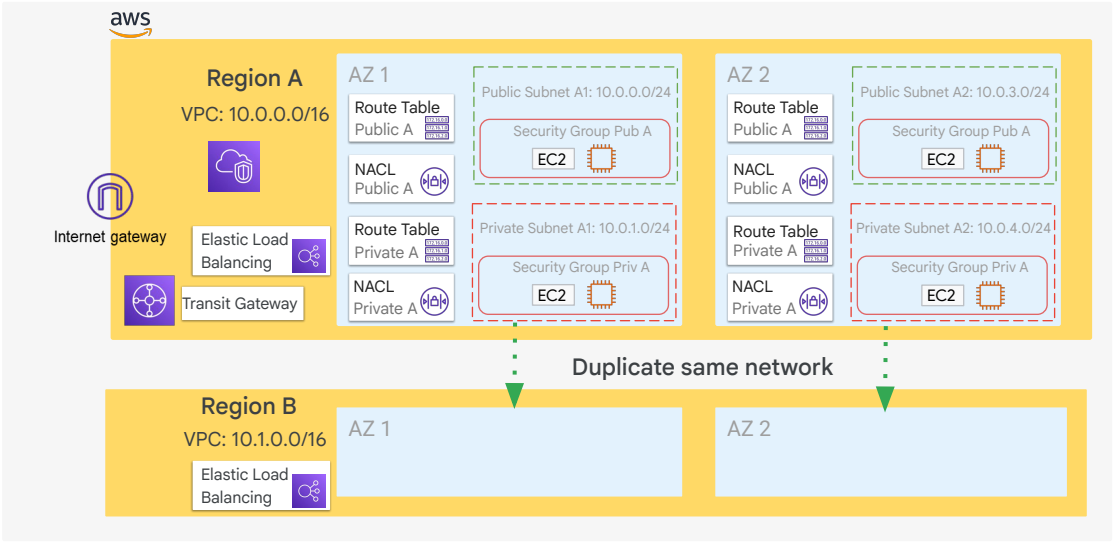
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 29: Transit Gateway - Region A



Minimum Network Resources for Region A

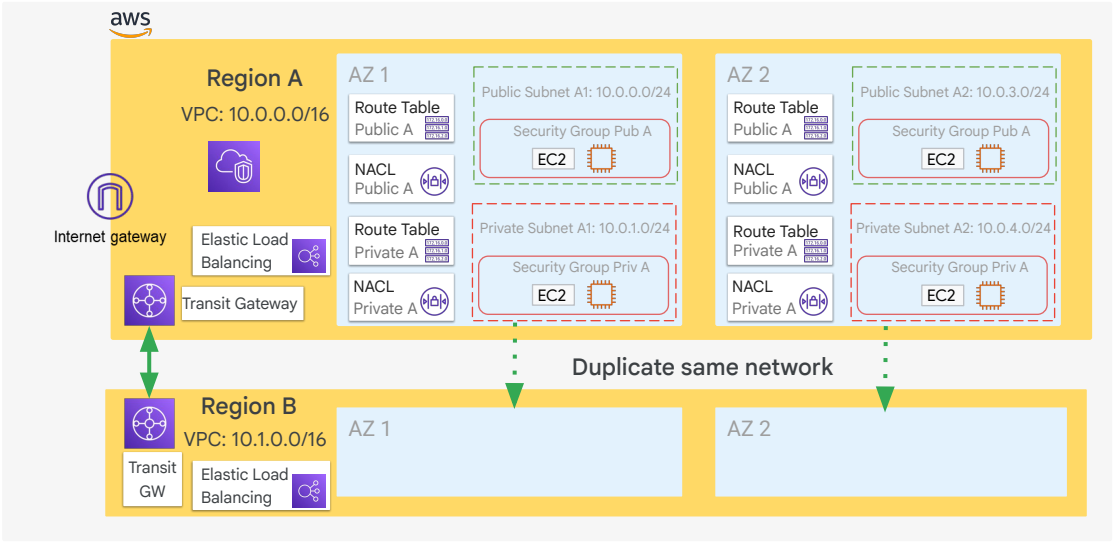
- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

Basic Dual-Region HA Network - AWS

Step 30: Transit Gateway - Region B

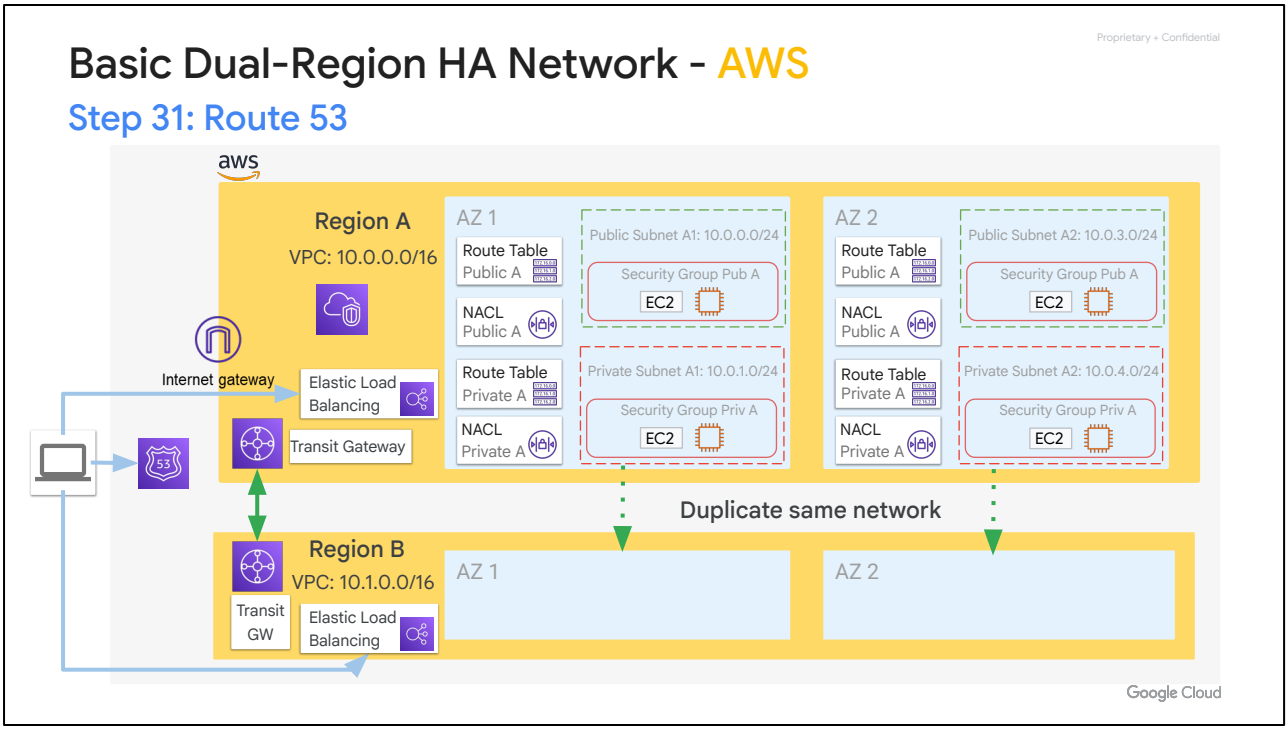


Minimum Network Resources for Region A

- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.

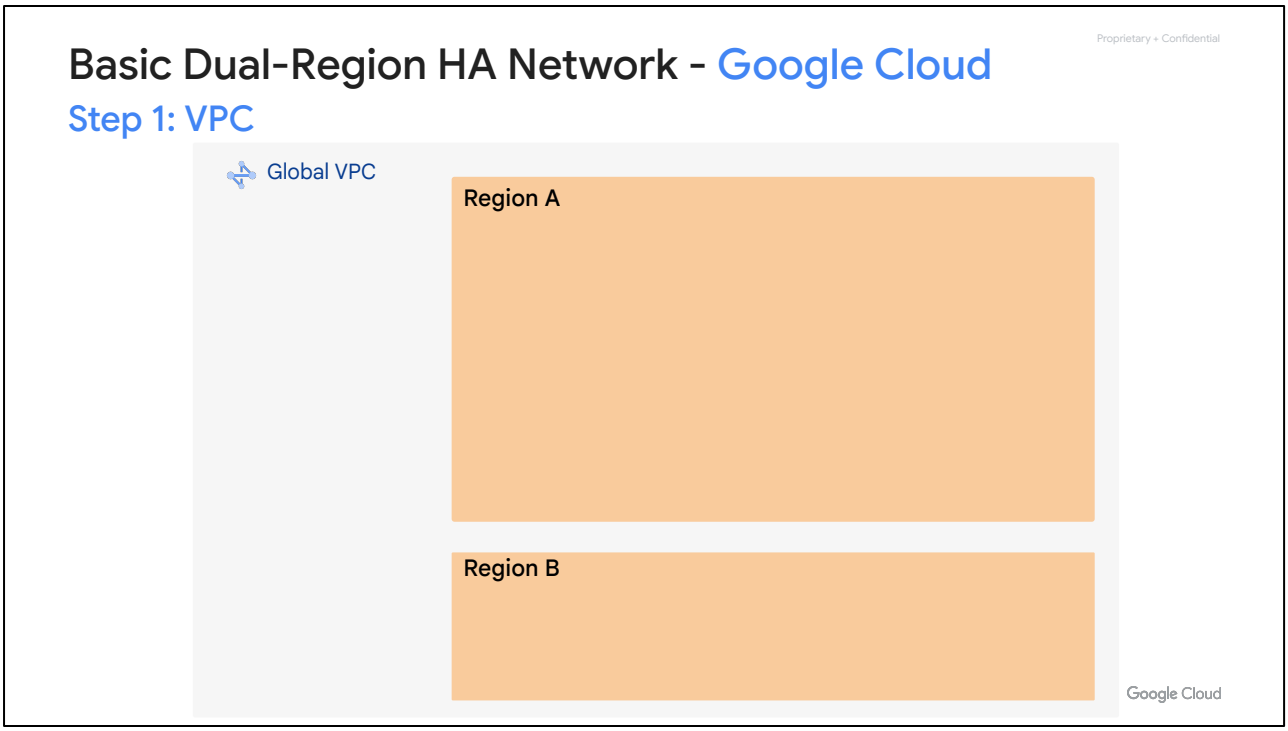


Minimum Network Resources for Region A

- 1 VPC with IP range
- 1 Load Balancer
- 4 Subnets spanning 2 AZs
- 1 Route table for Public Subnets
- 1 Route Table for Private Subnets
- 1 NACL for Private Subnets
- 1 NACL for Public Subnets
- 1 Security Group for Public Instances
- 1 Security Group for Private Instances

Region B

Duplicate the entire network.



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

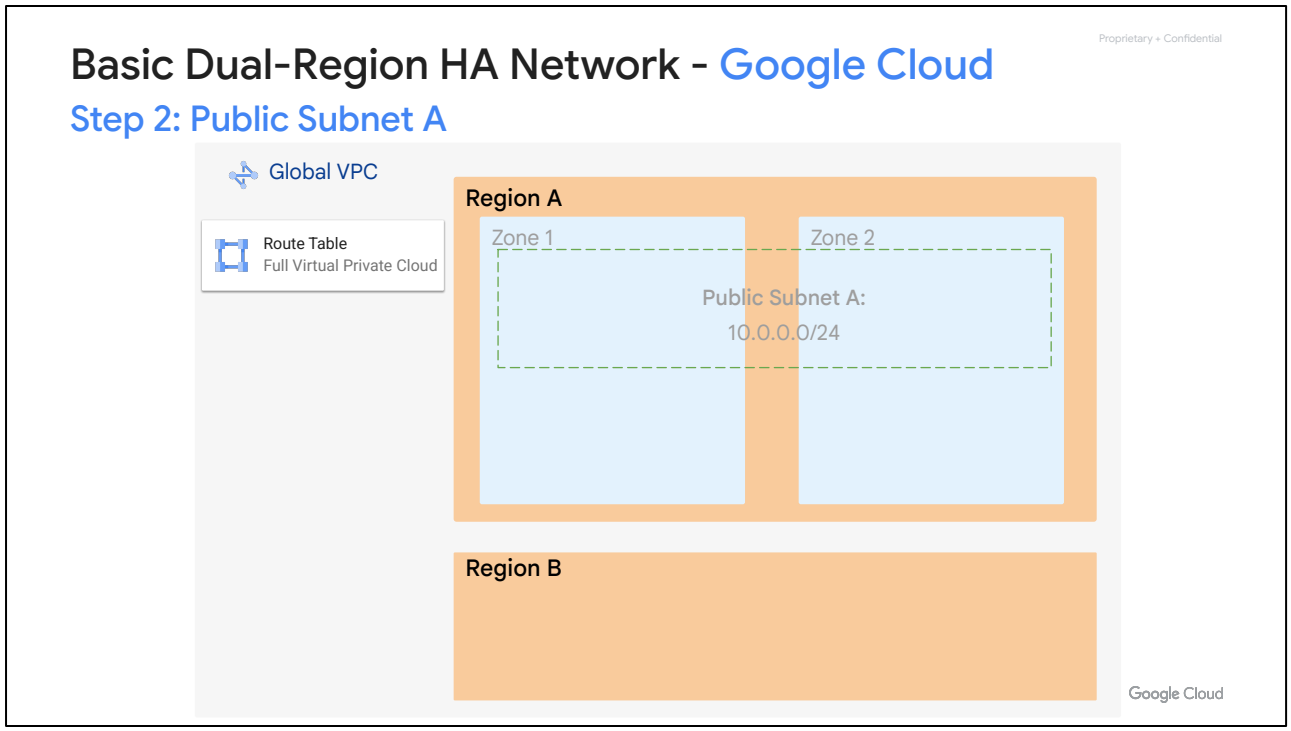
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

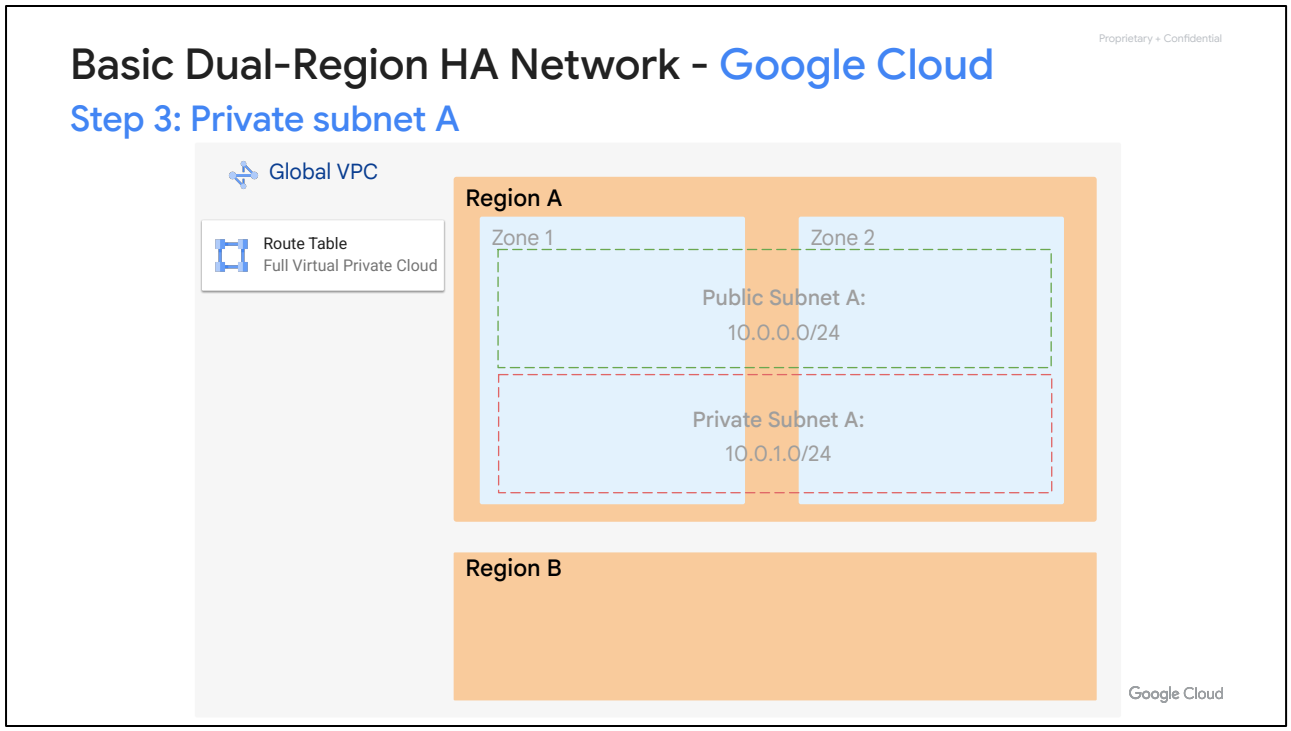
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

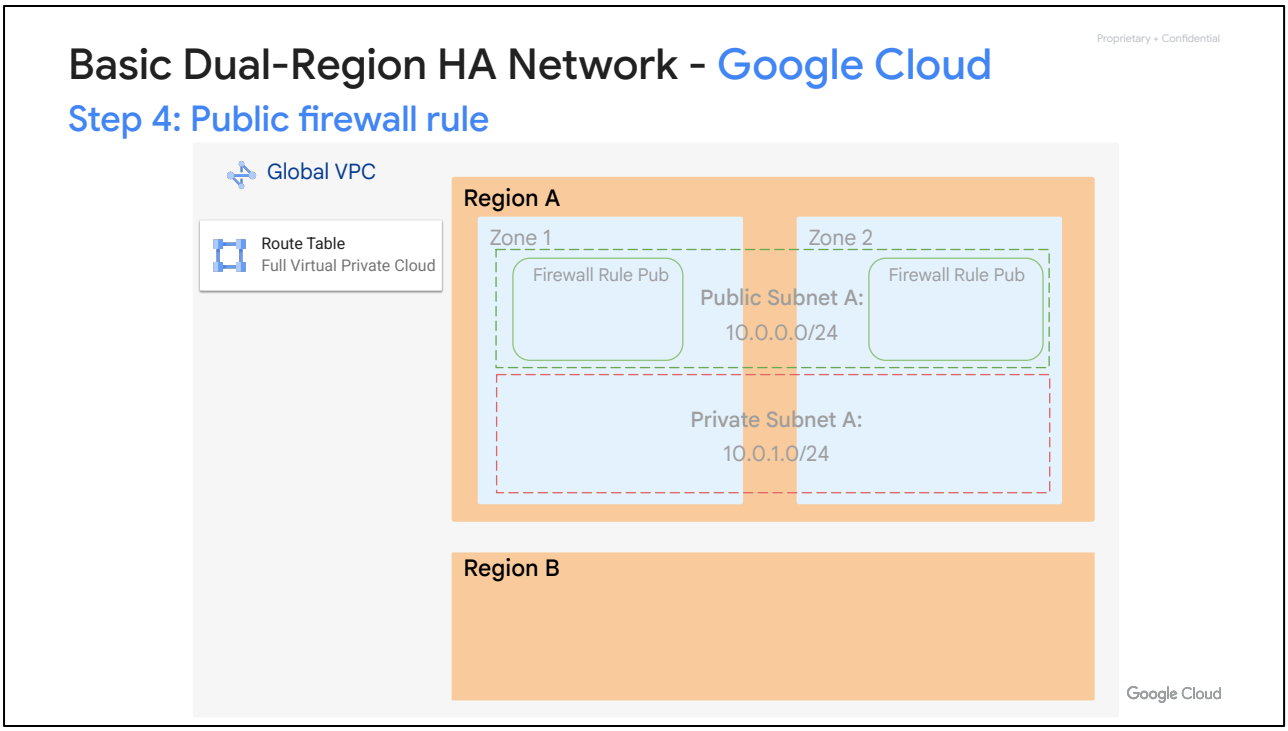
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

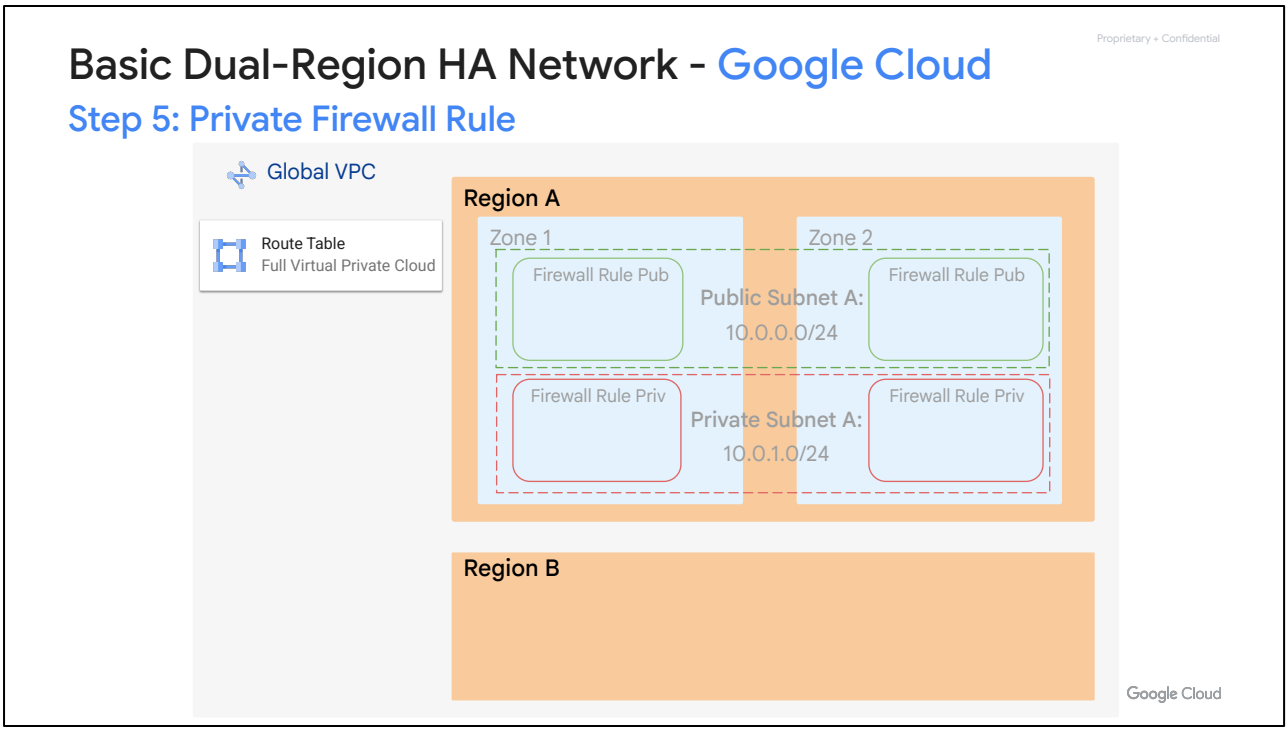
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

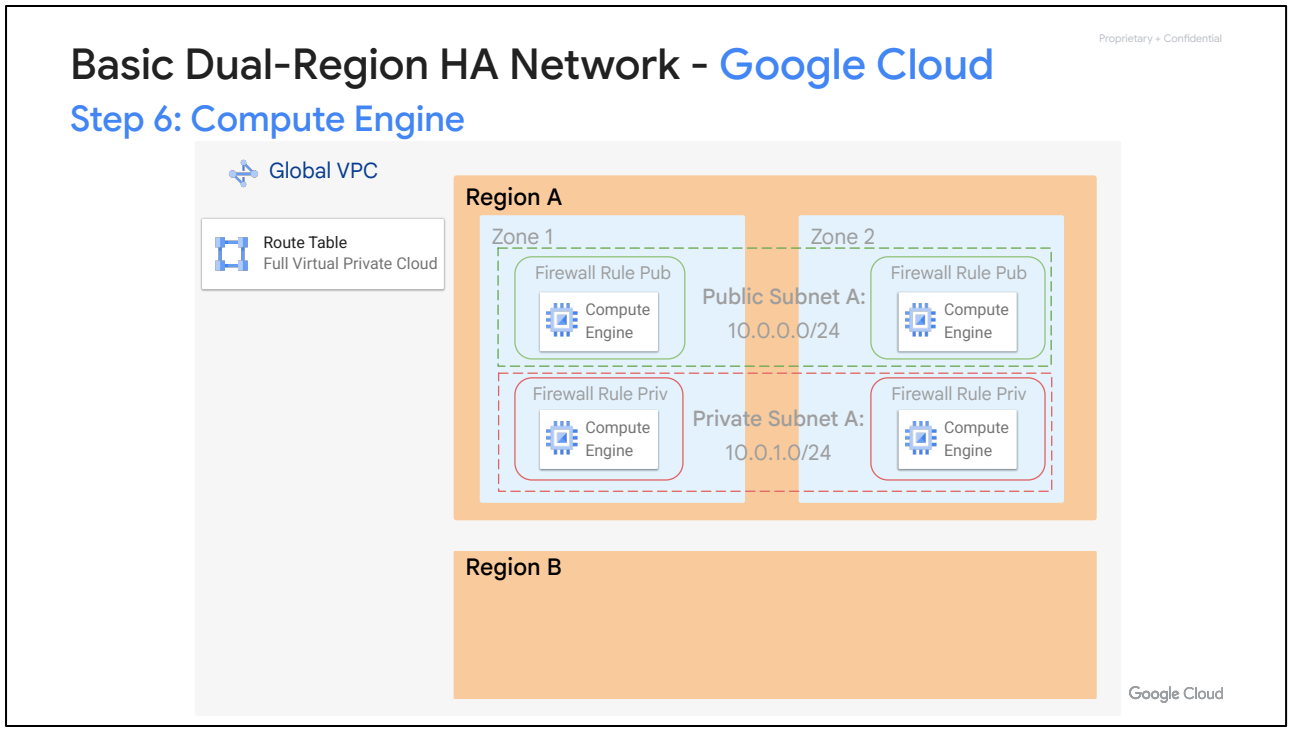
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

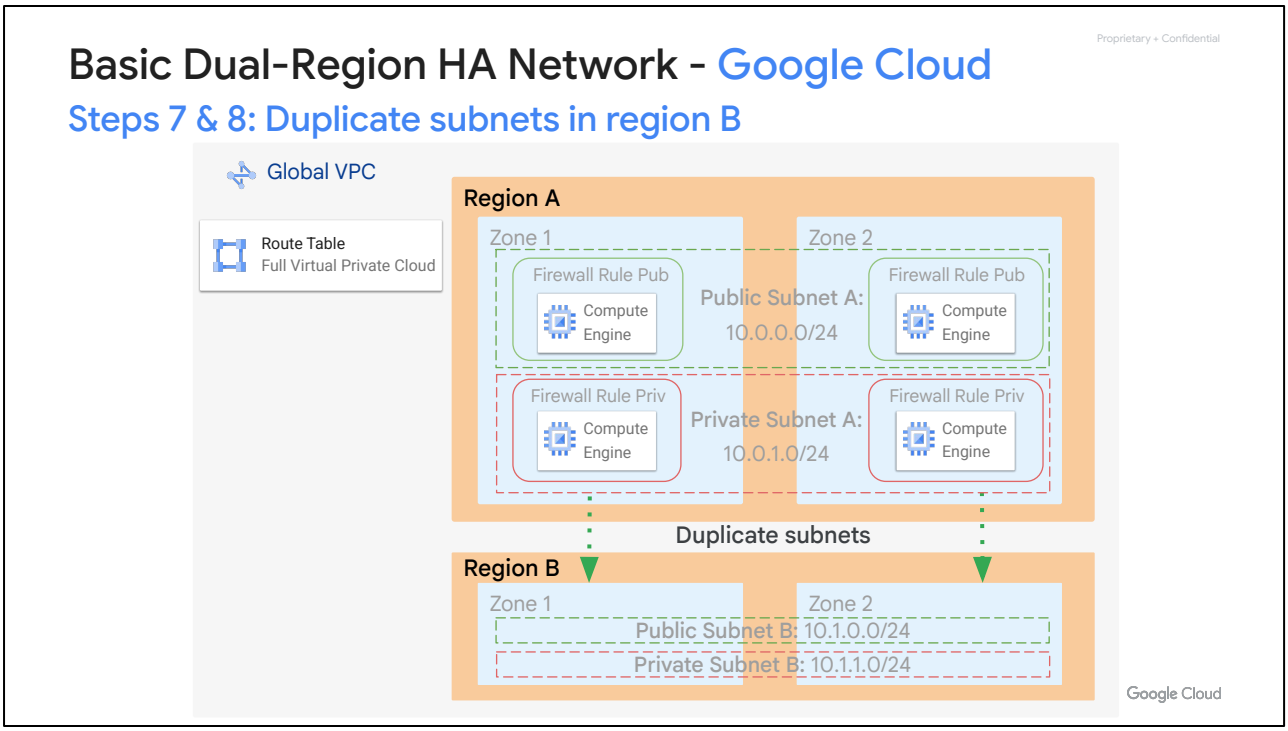
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

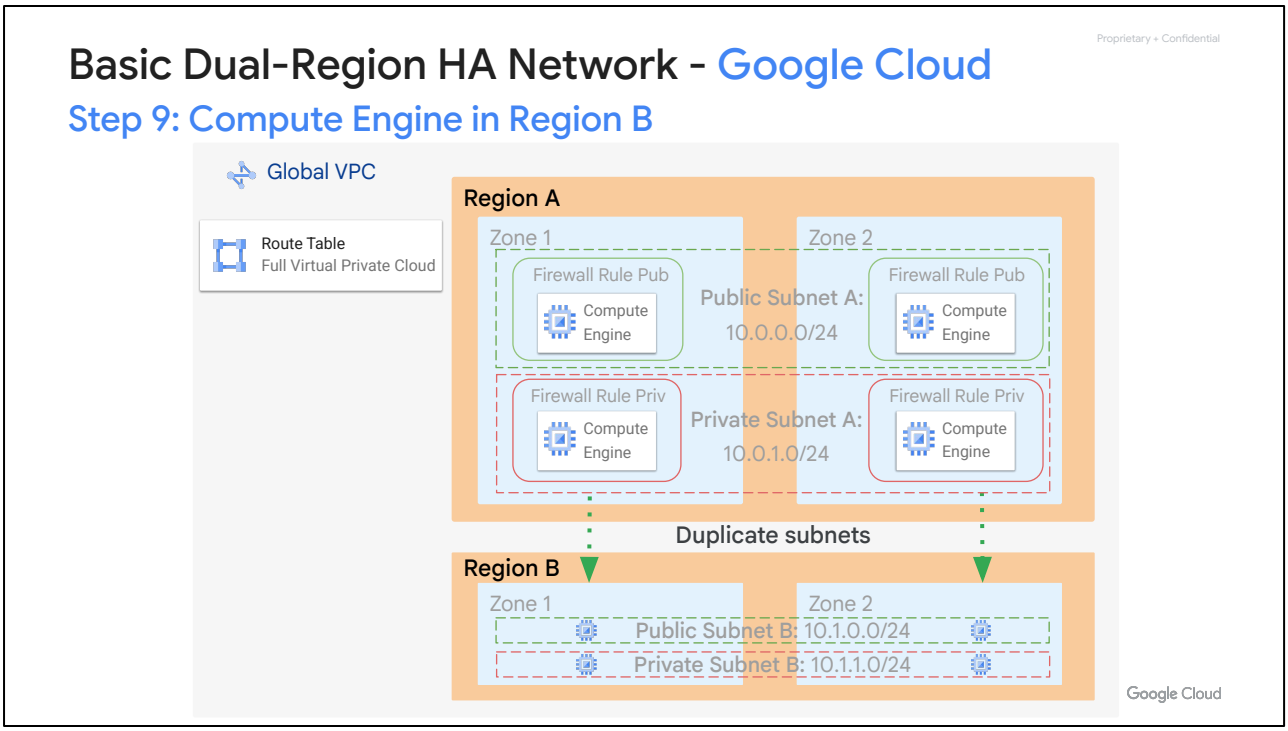
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

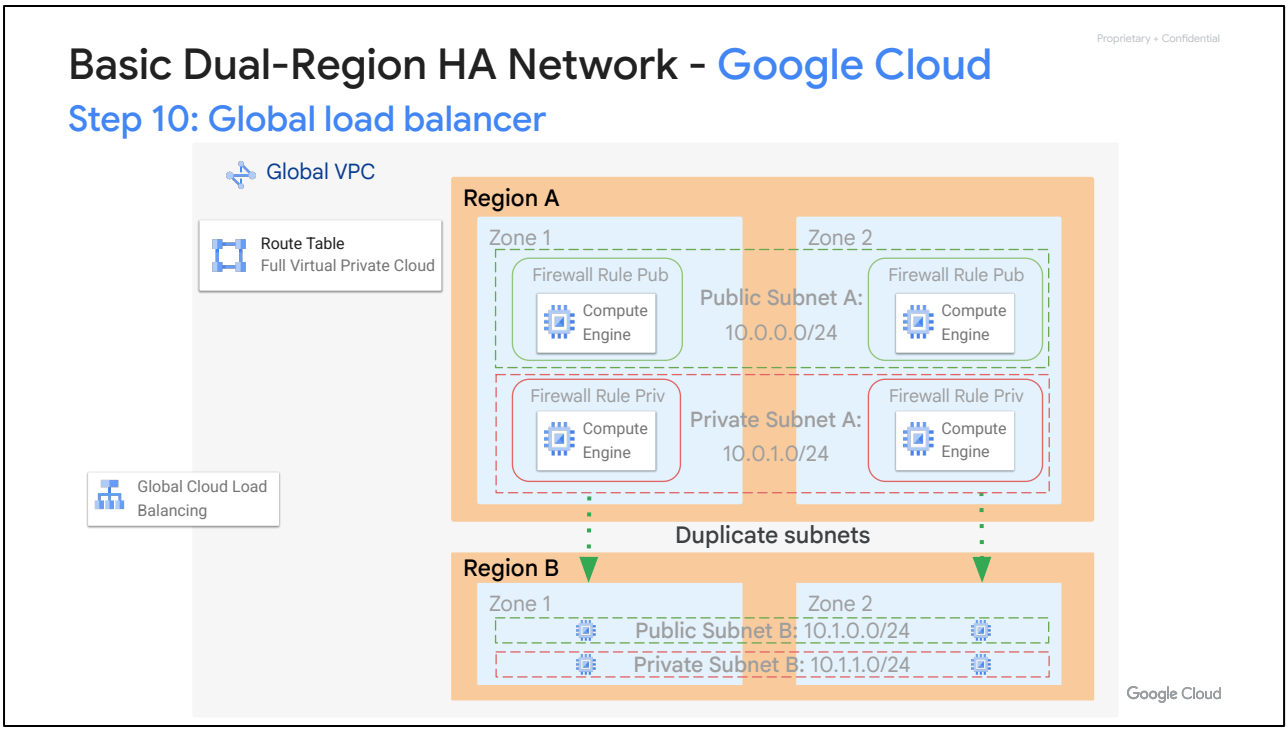
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

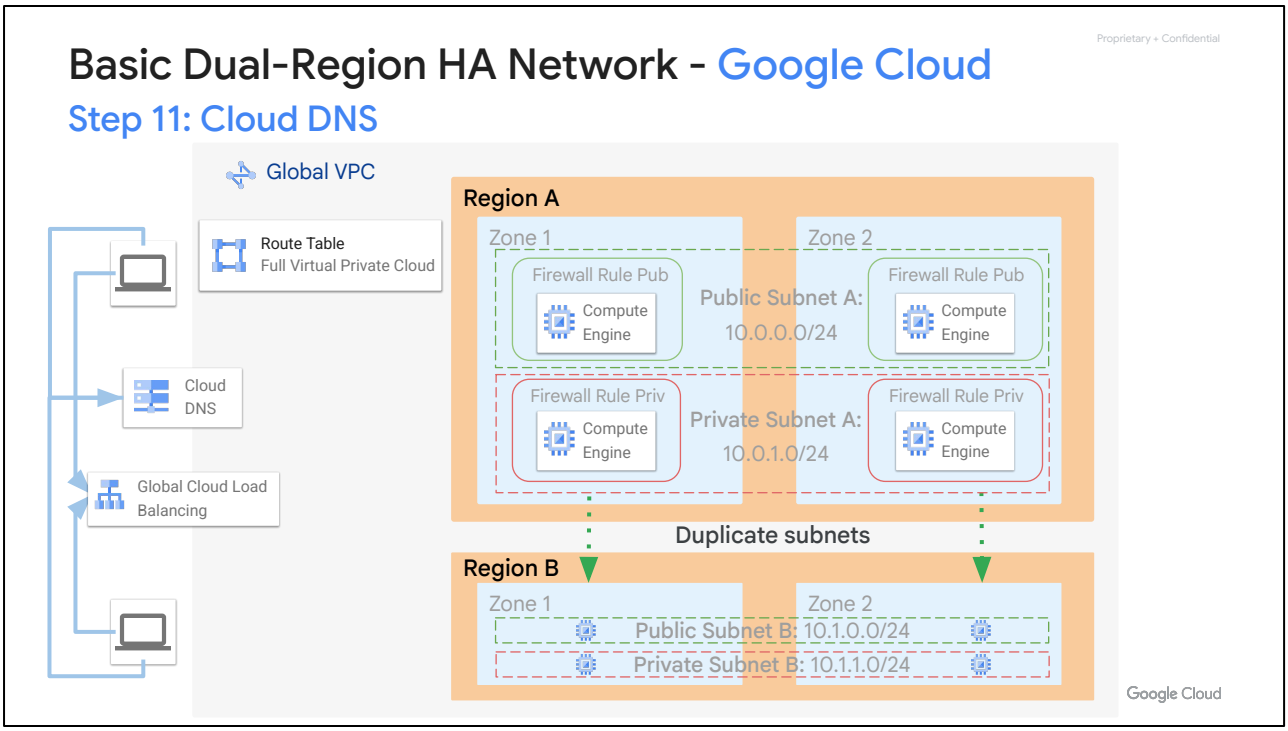
Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions



- Why only the subnets need to be created for the second region ?
  - Because Route Table and Firewall Rules apply at VPC level (so Global)
  - Global Load Balancer can direct traffic to different regions

Explain Firewall rules can applied across the entire network (different zone or region) based on Tags or service account which is great to manage at scale compare to Security Group.

Minimum Network Resources

- 1 Global VPC
- 1 Global Load Balancer
- 2 Subnets spanning 2 Zones
- 1 Route table for the full VPC
- 1 firewall rule for Public Instances
- 1 firewall rule for Private Instances

Second Region

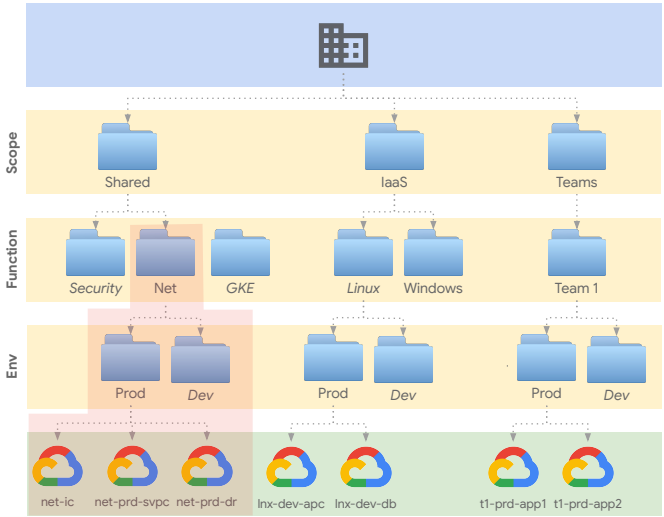
- Only the subnets need to be created in the other region
- Note: Make sure there is no overlap in Subnet IP Range across regions

Network provisioning example

If networking is managed by a central team, a separate folder allows assigning a **single set of permissions across environments**.

Lower-level folders for each environment still allow efficient assignment of roles to support, application teams, etc., **even when multiple projects are needed for network resources**.

Project separation allows high-level split of consumption.

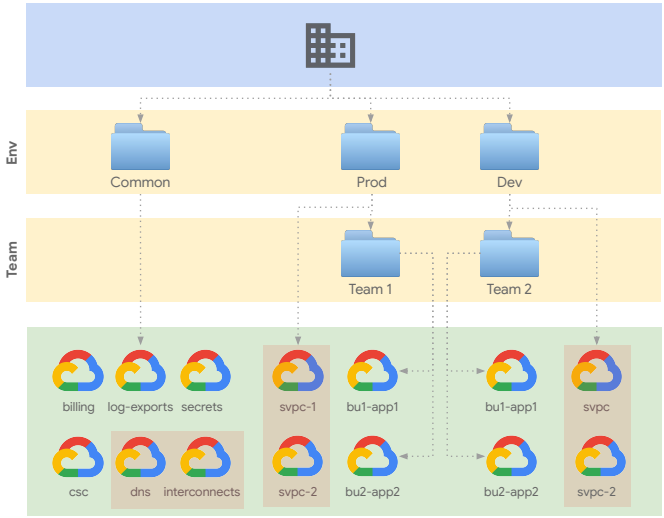


Antipattern - Network provisioning example

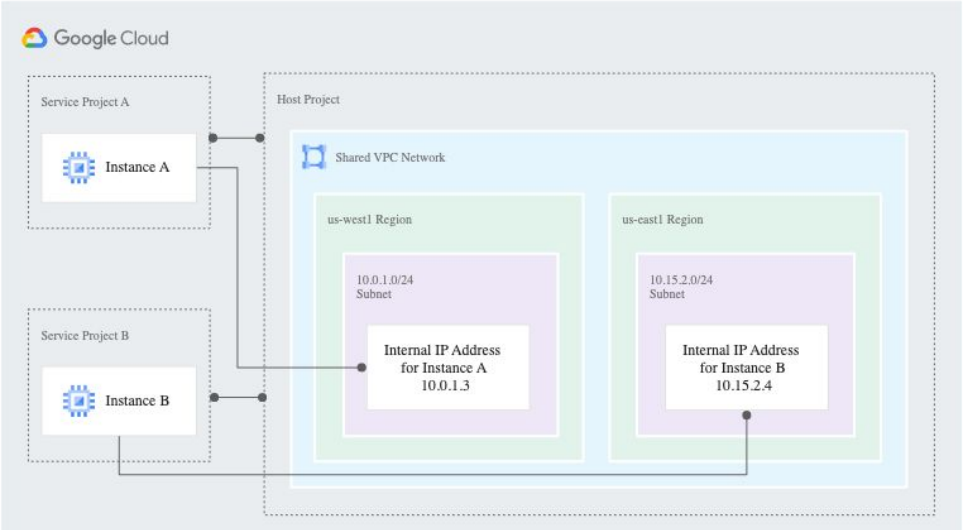
Networking still managed by a central team, but network resources are scattered all over the organization

No easy way to leverage folders to group permissions. IAM bindings have to be applied at the project level.

Any change involves additional operational complexity, increased risk of errors, and increased potential of environment drift.



Shared VPC (Shared VPC)



Host Project has the VPC; Service Project uses the shared subnets; Projects must be in the same Organization Google Cloud

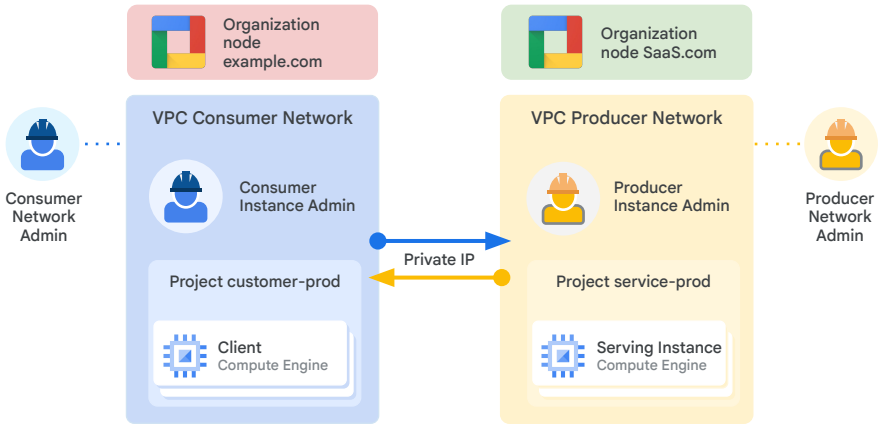
Source: Architecting with Compute Engine Slides

Shared VPC allows an organization to connect resources from multiple projects to a common VPC network. This allows the resources to communicate with each other securely and efficiently using internal IPs from that network.

For example, in this diagram there is one network that belongs to the Web Application Server's project. This network is shared with three other projects, namely the Recommendation Service, the Personalization Service, and the Analytics Service. Each of those service projects has instances that are in the same network as the Web Application Server and allow for private communication to that server, using internal IP addresses. The Web Application Server communicates with clients and on-premises using the server's external IP address. The backend services, in contrast, cannot be reached externally because they only communicate using internal IP addresses.

When you use shared VPC, you designate a project as a host project and attach one or more other service projects to it. In this case, the Web Application Server's project is the host project, and the three other projects are the service projects. The overall VPC network is called the shared VPC network.

VPC Network Peering (VNet Peering)



VPCs are global (regional) so VPC peering is used less in Google Cloud than AWS

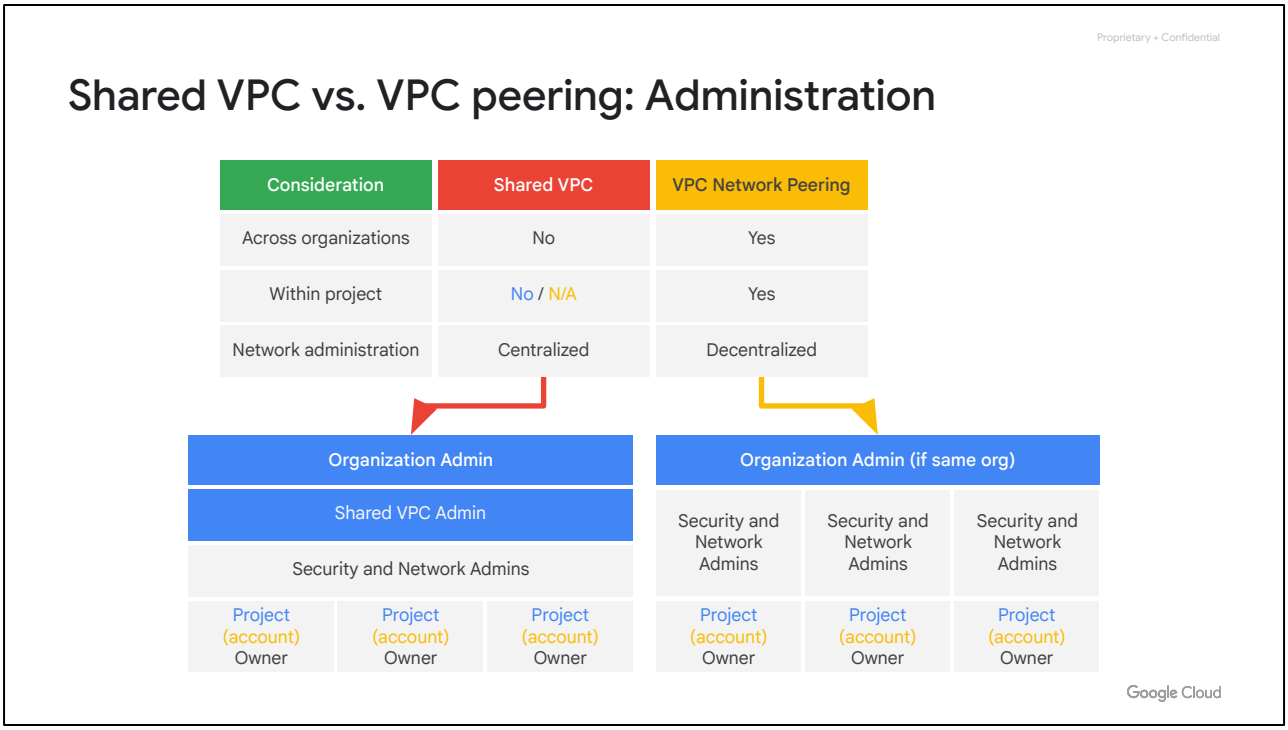
Source: Architecting with Compute Engine Slides

VPC Network Peering, in contrast, allows private RFC 1918 connectivity across two VPC networks, regardless of whether they belong to the same project or the same organization. Now, remember that each VPC network will have firewall rules that define what traffic is allowed or denied between the networks.

For example, in this diagram there are two organizations that represent a consumer and a producer, respectively. Each organization has its own organization node, VPC network, VM instances, Network Admin, and Instance Admin. In order for VPC Network Peering to be established successfully, the Producer Network Admin needs to peer the Producer Network with the Consumer Network, and the Consumer Network Admin needs to peer the Consumer Network with the Producer Network. When both peering connections are created, the VPC Network Peering session becomes Active and routes are exchanged. This allows the virtual machine instances to communicate privately using their internal IP addresses.

VPC Network Peering is a decentralized or distributed approach to multi-project networking, because each VPC network may remain under the control of separate administrator groups and maintains its own global firewall and routing tables. Historically, such projects would consider external IP addresses or VPNs to facilitate private communication between VPC networks. However, VPC Network Peering does

not incur the network latency, security, and cost drawbacks that are present when using external IP addresses or VPNs.



Source: Architecting with Compute Engine Slides

In my opinion, the biggest difference between the two configurations is the network administration models. Shared VPC is a centralized approach to multi-project networking, because security and network policy occurs in a single designated VPC network. In contrast, VPC Network Peering is a decentralized approach, because each VPC network can remain under the control of separate administrator groups and maintains its own global firewall and routing tables.

If you want to learn more about Shared VPC and VPC peering, we recommend the Networking in Google Cloud course.

Deciding between Shared VPC and VPC peering

Consideration	Shared VPC	VPC Network Peering
Across organizations	No	Yes
Within project	No	Yes
Network administration	Centralized	Decentralized

If you want to configure private communication between VPC networks in different organizations, you have to use VPC Network Peering. Shared VPC only works within the same organization.

Somewhat similarly, if you want to configure private communication between VPC networks in the same project, you have to use VPC Network Peering.

This doesn't mean that the networks need to be in the same project, but they can be. Shared VPC only works across projects.

One key difference between the two configurations is the network administration models:

- Shared VPC is a centralized approach to multi-project networking, because security and network policy occurs in a single designated VPC network.
- VPC Network Peering is a decentralized approach. This is because each VPC network can remain under the control of separate administrator groups and maintains its own global firewall and routing tables.

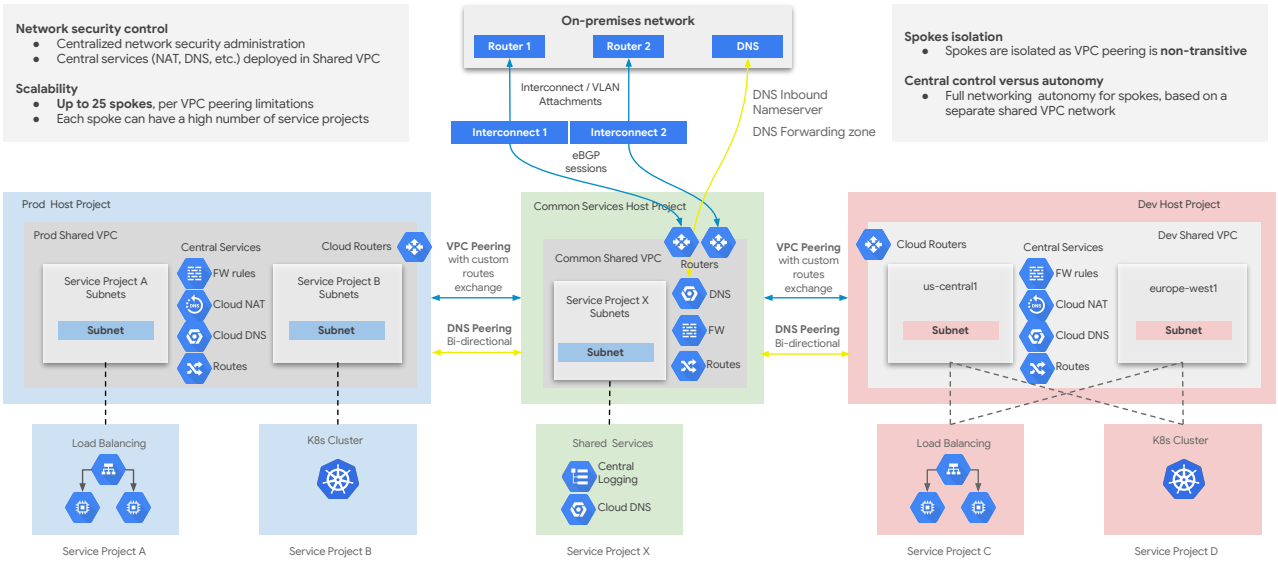
AWS v/s Google

Resource	AWS	Google Cloud
VPC	Regional	Global
VPN	AWS VPN Site-to-Site	Cloud VPN (Classic or HA)
VPC Peering	VPC Peering, Transit Gateway	VPC Peering
VPC sharing	Shared VPC	Shared VPC
Cloud External (WAN) Connection	Dedicated or Hosted Direct Connect, Direct Connect Gateway	Dedicated, Partner, or Cross-Cloud Interconnect

Source :

Reference architecture

Hub-and-spoke with VPC peering - Segmentation based on environments



TL;DR / Purpose of the slide:

- **Hub-and-spoke with VPC peering** for **environments segmentation**

Key points:

- **Use case**
  - hub-and-spoke model
  - Each spoke represents a **larger network segment**, e.g environments (prod, test, dev), business units, etc. We will need a **handful of spokes.**
  - Spokes shouldn't be able to communicate with each other.
  - Within each spoke, it should be possible to separate connectivity between workloads.
- **High-level**
  - In this design, the hub is a **Shared VPC** and is **VPC peered** to the spokes, which are in turn **Shared VPCs** as well.
  - Making use of **Shared VPC** helps keeping the design scalable and simple.
  - **VPC peering** is useful for **lower latency**, fits the spokes **scale requirements,** and helps **isolating connectivity between the spokes.**
- **Network connectivity**
  - **Hub**
    - Common services host project hosts a **VPC** that is **connected with on-premise** (VPN / interconnect)

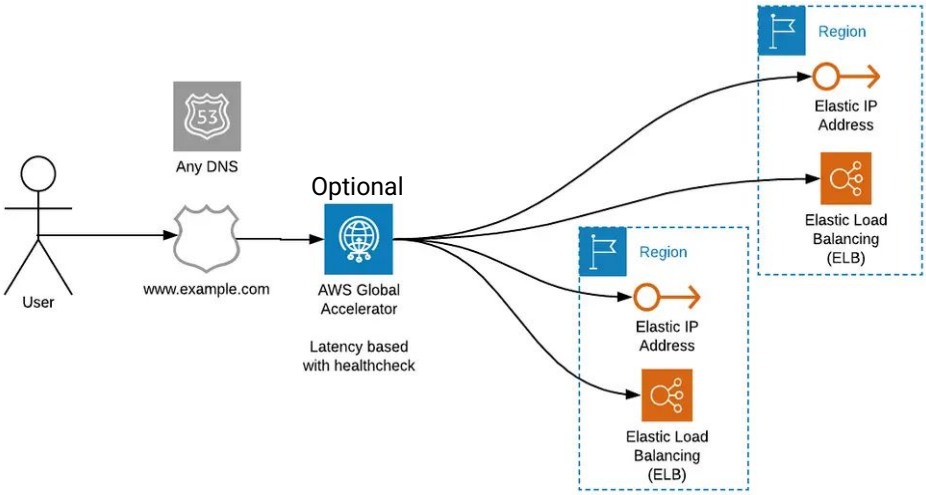
        - Centralizes **administration** of **network services** for connectivity **between spokes and on-premise**, such as FW, DNS, routes, etc.
        - We could also have a **security appliance** in the hub for **east-west** and/or **north-south** communication, depending on the need.
    - **Spokes**
        - Spokes are connected to the hub with **VPC peering** to ensure **low latency**, and **minimal management overhead**.
        - **Separation between spokes** is ensured due to **non-transitivity**.
        - **# of spokes**: As spokes represent larger network segments, the **upper limit of 25 peered networks**/spokes is **sufficient**.
        - **Each spoke** is essentially a **Shared VPC host project** with **network administration autonomy**: FW rules, NAT, DNS, routes, etc.
        - To allow a **path between spokes and on-prem, custom routes exchange** is configured between the hub and each spoke.
        - Controlling **network access between workloads** within each spoke is done **based on VPC firewall,** which we will discuss in a later slide.
- **DNS topology**
    - Similar to the hub-and-spoke model we have shown in the DNS topology section
    - **GCP to on-prem**: **DNS forwarding zone**
    - **On-prem to GCP: Inbound server-policy** allows access to inbound nameservers from on-prem
    - **Spokes to hub**: **DNS peering** to allow making use of the forwarding zone and querying on-prem
    - **Hub to spokes**: **DNS peering** to allow on-prem query spokes, and not the hub only
- **Additional design considerations** regarding **VPC Peering**
    - Communication between **on-prem** and **products** (e.g: Cloud SQL Private IP, GKE private masters) is not possible when **using VPC peering** due to **non-transitivity**.
        - **Workaround** by going through a **proxy** in the environment VPC. See link.
    - When using **VPC Peering**, some **limits are shared** between the VPCs. E.g: forwarding rules

Probing questions (optional):

- None

Cloud Load Balancing

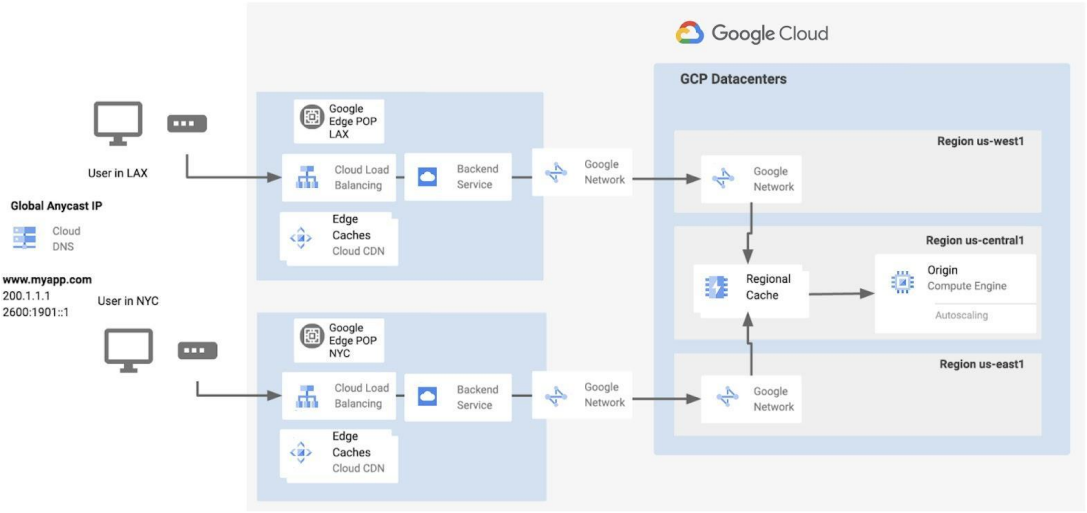
Typical Load Balancing in AWS



Source: Architecting with Compute Engine slides.

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.

Load Balancing on Google Cloud



Source: Architecting with Compute Engine slides.

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.

Key differences between AWS and Google Cloud load balancing

AWS	Google Cloud
Does not offer global load balancing	Option for global load balancing, allowing you to manage traffic globally instead of regionally
AWS offers function-specific load balancers	Protocol-specific load balancers
Supports Transport Layer Security (TLS)	You can use HTTP(S) Load Balancing
Automatically provisions and manages certificates via Certificate Manager	Automatically provisions and manages certificates via Certificate Manager

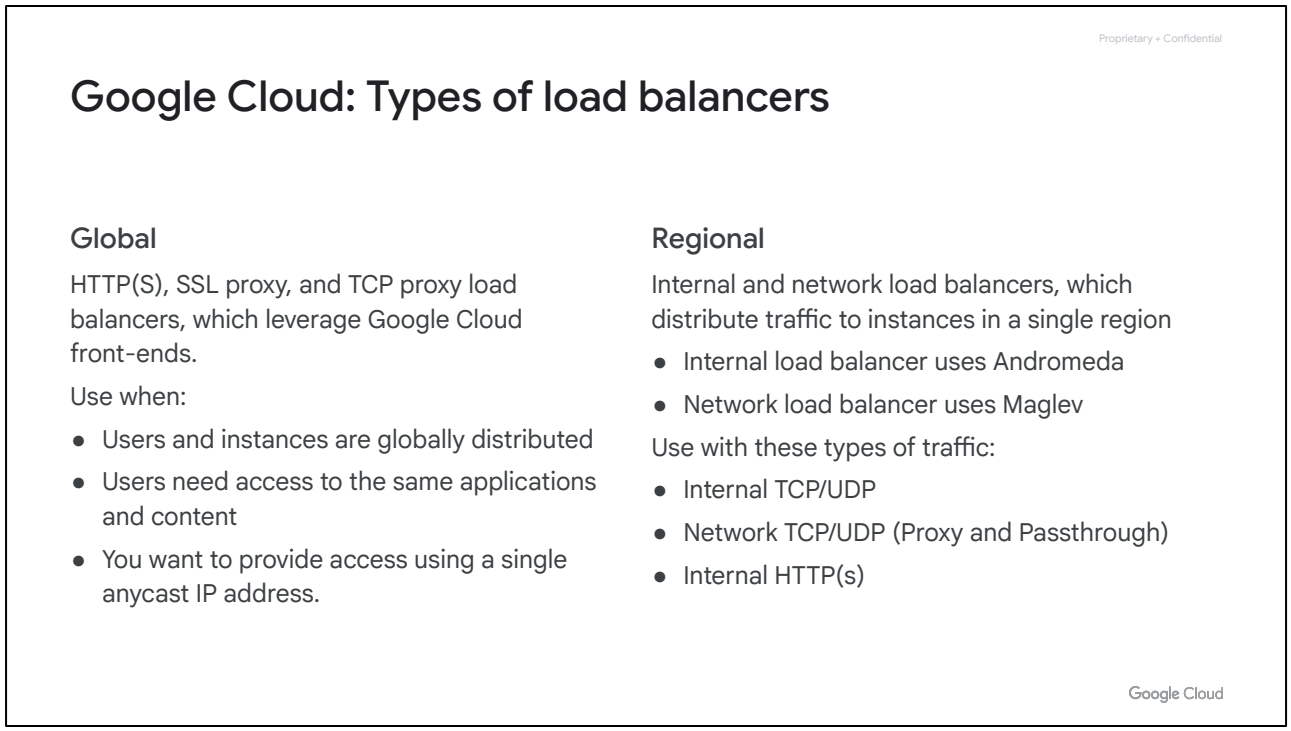
Now let’s look at some key differences between AWS and Google Cloud load balancing.

The biggest difference in load balancing is that Google Cloud provides an option for global load balancing. Using a global load balancer allows you to manage traffic globally instead of regionally, reducing the complexity of deployment and simplifying the application deployment process.

Google Cloud offers protocol-specific load balancers, while AWS offers function-specific load balancers. AWS's Application Load Balancer does not currently supports Transport Layer Security (TLS). If TLS is in use, a network load balancer is required, even for an application. In Google Cloud, you can use HTTP(S) Load Balancing.

Lastly, both AWS Google Cloud automatically provisions and manages certificates via Certificate Manager.

Refer to the additional resource "Comparison of load balancing features and products in AWS and Google Cloud” to learn more about the features and products in AWS and Google Cloud.



Google Cloud offers different types of load balancers that can be divided into two categories: global and regional. Let's explore each category.

The global load balancers are the HTTP(S), SSL proxy, and TCP proxy load balancers. These load balancers leverage the Google Cloud frontends, which are software-defined, distributed systems that sit in Google Cloud's points of presence and are distributed globally.

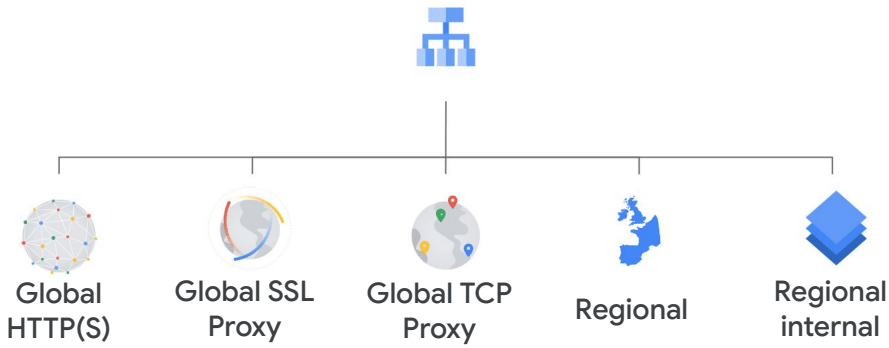
You want to use a global load balancer when:

- Your users and instances are globally distributed.
- Your users need access to the same applications and content.
- 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 Google Cloud region. The internal load balancer uses Andromeda, which is Google Cloud's software-defined network virtualization stack. The network load balancer uses Maglev, which is a large, distributed software system. You can use these load balancers for these types of traffic:

- Internal TCP/UDP
- Network TCP/UDP (Proxy and Passthrough)
- Internal HTTP(s)

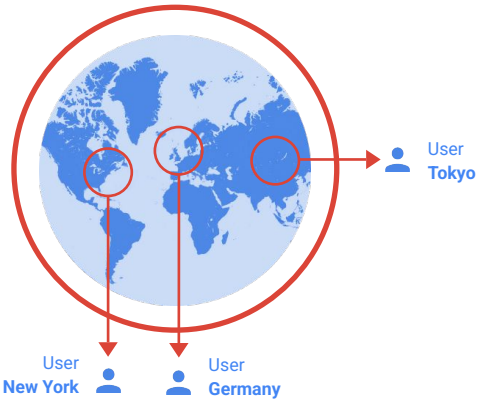
Load-balancing options with Cloud Load Balancing



There are also several load-balancing options available with Cloud Load Balancing:

- If you need cross-regional load balancing for a web application, use Global HTTP(S) load balancing
- For Secure Sockets Layer traffic that is not HTTP, use the Global SSL Proxy load balancer
- If you have other TCP traffic that does not use Secure Sockets Layer, use the Global TCP Proxy load balancer
- The two proxy services only work for specific port numbers, and they only work for TCP. If you want to load balance UDP traffic, or traffic on any port number, you can still load balance across a Google Cloud region with the Regional load balancer.
- If you want to load balance traffic inside your project, use the Regional internal load balancer. It accepts traffic on a Google Cloud internal IP address and load balances it across Compute Engine VMs.

With global Cloud Load Balancing (AWS Global Accelerator), your application presents a single front-end to the world



- Users get a single, global anycast IP address.
- Traffic goes over the Google (AWS) backbone from the closest point-of-presence to the user.
- Backends are selected based on client location, load (and user-created policies).
- Only healthy backends receive traffic.
- No pre-warming is required (except classic ELB backends).

Source: Demo Template

A few slides back, we talked about how virtual machines can autoscale to respond to changing load. But how do your customers get to your application when it might be provided by four VMs one moment and forty VMs at another? Cloud Load Balancing is the answer.

Cloud Load Balancing is a fully distributed, software-defined, managed service for all your traffic. And because the load balancers don't run in VMs you have to manage, you don't have to worry about scaling or managing them. You can put Cloud Load Balancing in front of all of your traffic: HTTP(S), other TCP and SSL traffic, and UDP traffic too.

With Cloud Load Balancing, a single anycast IP front-ends all your backend instances in regions around the world. It provides cross-region load balancing, including automatic multi-region failover, which gently moves traffic in fractions if backends become unhealthy. Cloud Load Balancing reacts quickly to changes in users, traffic, network, backend health, and other related conditions.

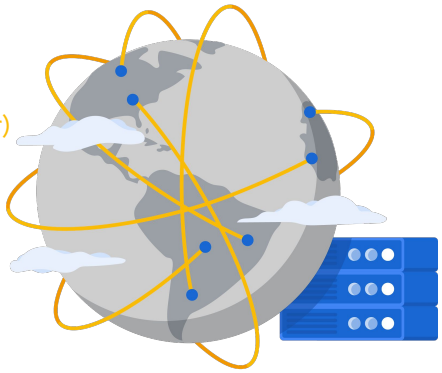
And what if you anticipate a huge spike in demand? Say, your online game is already a hit; do you need to file a support ticket to warn Google of the incoming load? No. No

so-called “pre-warming” is required.

HTTP(S) load balancing

HTTP(S) Load Balancing

- Global or cross-regional or regional load balancing
- Anycast IP address (via Global accelerator)
- HTTP on port 80 or 8080 (any port)
- HTTPs on port 443 (any port)
- IPv4 or IPv6
- Autoscaling
- URL maps



Source: Architecting with Compute Engine slides.

Google Cloud'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.

Global Load Balancing provides Anycast IP

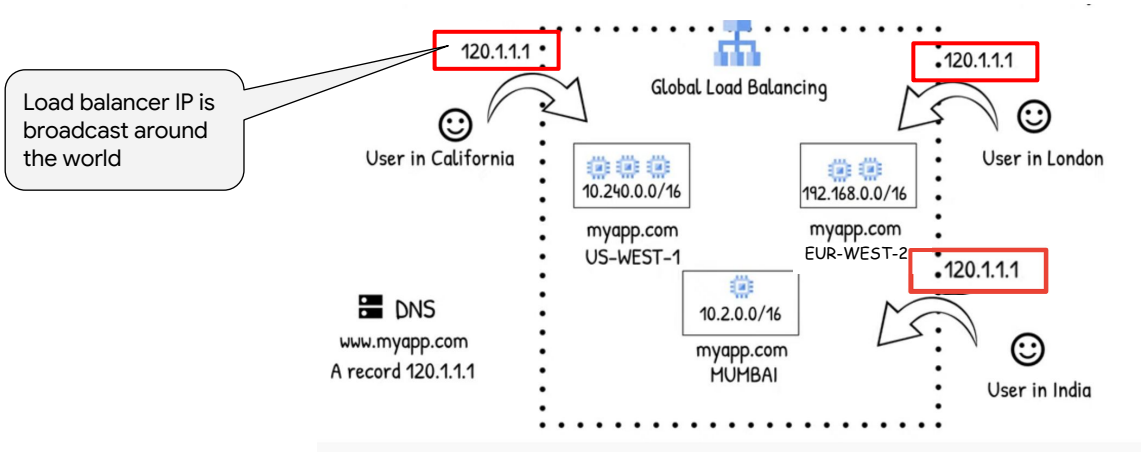
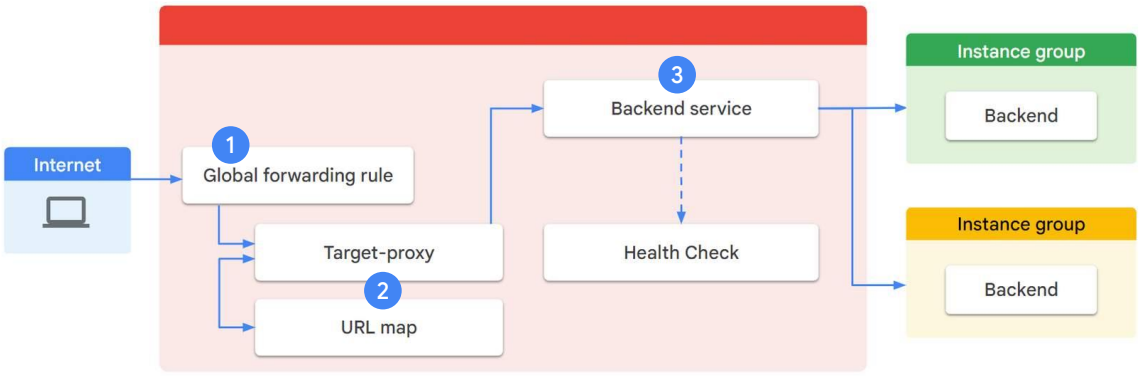


Image from video:
https://www.youtube.com/watch?v=0fQr7TRhnnU&list=PLTWE_Imu2InBzuPmOcgAYP7U80a87cpJd

Corresponding blog:
https://cloud.google.com/blog/topics/developers-practitioners/what-cloud-load-balancing

Architecture of an HTTP(S) load balancer



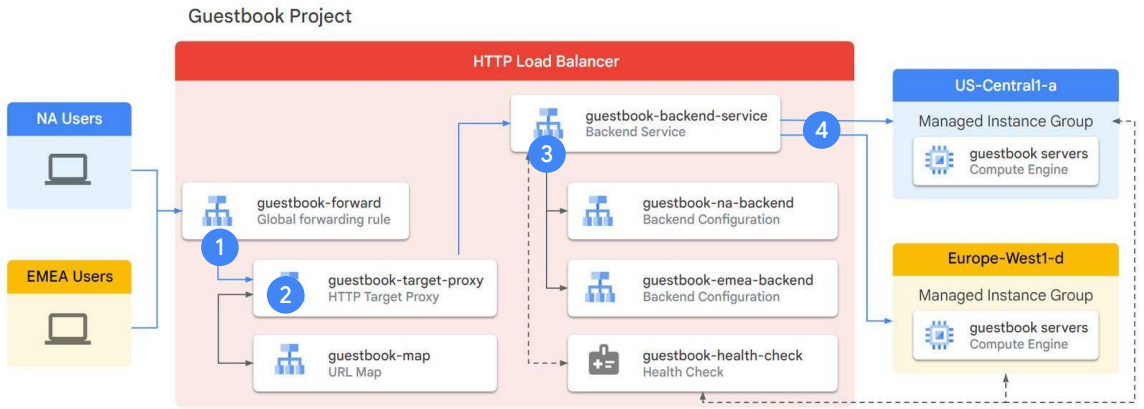
Now, let's explore the complete architecture of an HTTP(S) load balancer.

First, a global forwarding rule directs incoming requests from the internet to a target HTTP proxy.

Next, the target HTTP proxy checks each request against a URL map to determine the appropriate backend service for the request.

Finally, the backend service directs each request to an appropriate backend based on serving capacity, zone, and instance health of its attached backends.

HTTP load balancing resources: Cross-region example



Let's examine an HTTP load balancer in action using this image as an example.

The project in this diagram 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.

1. First, the global forwarding rule directs incoming requests to the target HTTP proxy.
2. Then the proxy checks the URL map to determine the appropriate backend service for the request. In this case, the application has only one backend service.
3. 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.
4. 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 this 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

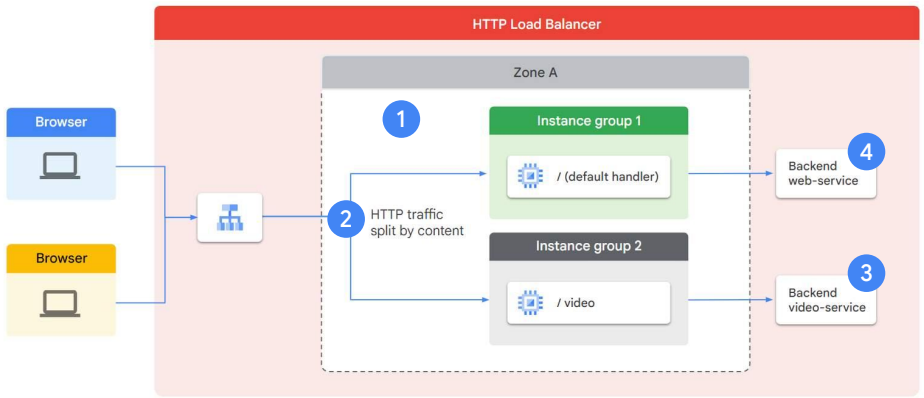
across all available backend services and instances in that region.

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.

HTTP load balancing resources: Content-based example



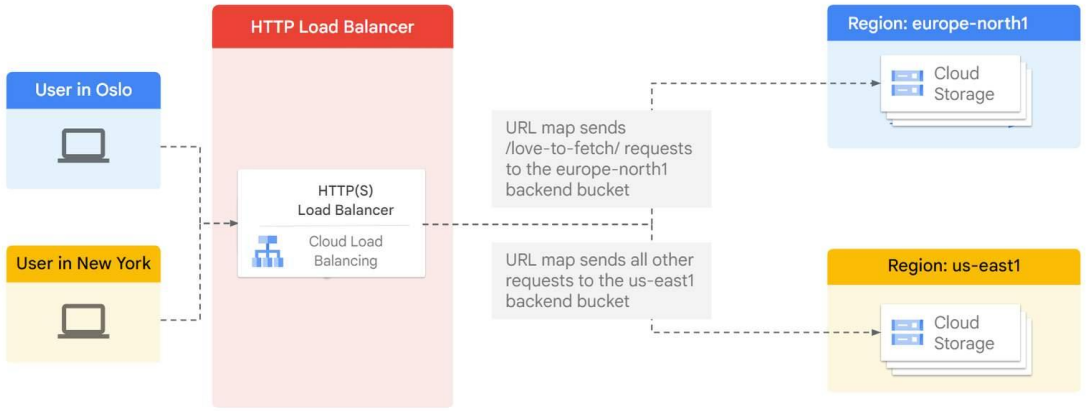
Another example of an HTTP load balancer is a content-based load balancer.

Let's explore how this works using this diagram.

1. In this case, there are two separate backend services that handle either web or video traffic.
2. Traffic is split by the load balancer based on the URL header as specified in the URL map.
3. If a user is navigating to video, the traffic is sent to the backend video-service.
4. If a user is navigating to anywhere other than video, the traffic is sent to the web-service backend.

All of this is achieved with a single global IP address.

HTTP(S) load balancing: Backend buckets



Backend buckets allow you to use Google Cloud Storage buckets with HTTP(S) Load Balancing.

An external HTTP(S) load balancer uses a URL map to direct traffic from specified URLs to either a backend service or a backend bucket.

Common use cases include:

- Send requests for dynamic content, such as data, to a backend service
- Send requests for static content, such as images, to a backend bucket

In this diagram, the load balancer sends traffic with a path of /love-to-fetch/ to a Cloud Storage bucket in the europe-north region.

All other requests go to a Cloud Storage bucket in the us-east region.

After you configure a load balancer with the backend buckets, requests to URL paths that begin with /love-to-fetch are sent to the europe-north Cloud Storage bucket.

All other requests are sent to the us-east Cloud Storage bucket.

SSL proxy load balancing (Google Cloud only)

- 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

Layer 4! No equivalent… maybe not a lot of use-cases?

Source: Architecting with Compute Engine slides.

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.

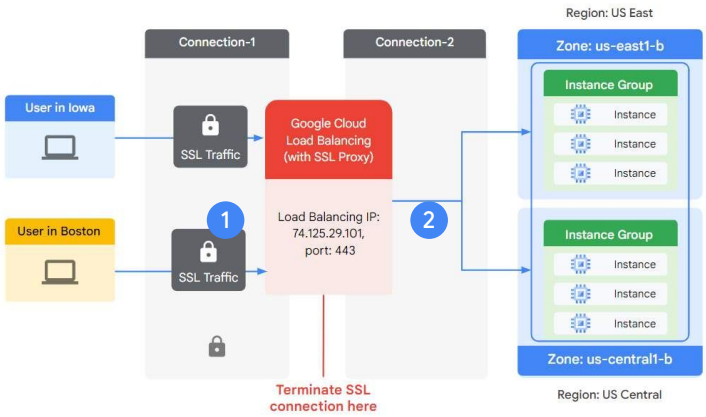
SSL proxy load balancing supports both IPv4 and IPv6 addresses for client traffic and provides Intelligent routing, Certificate management, Security patching and SSL policies.

- 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, Google Cloud 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 documentation page.

External SSL Proxy Load Balancing: Example



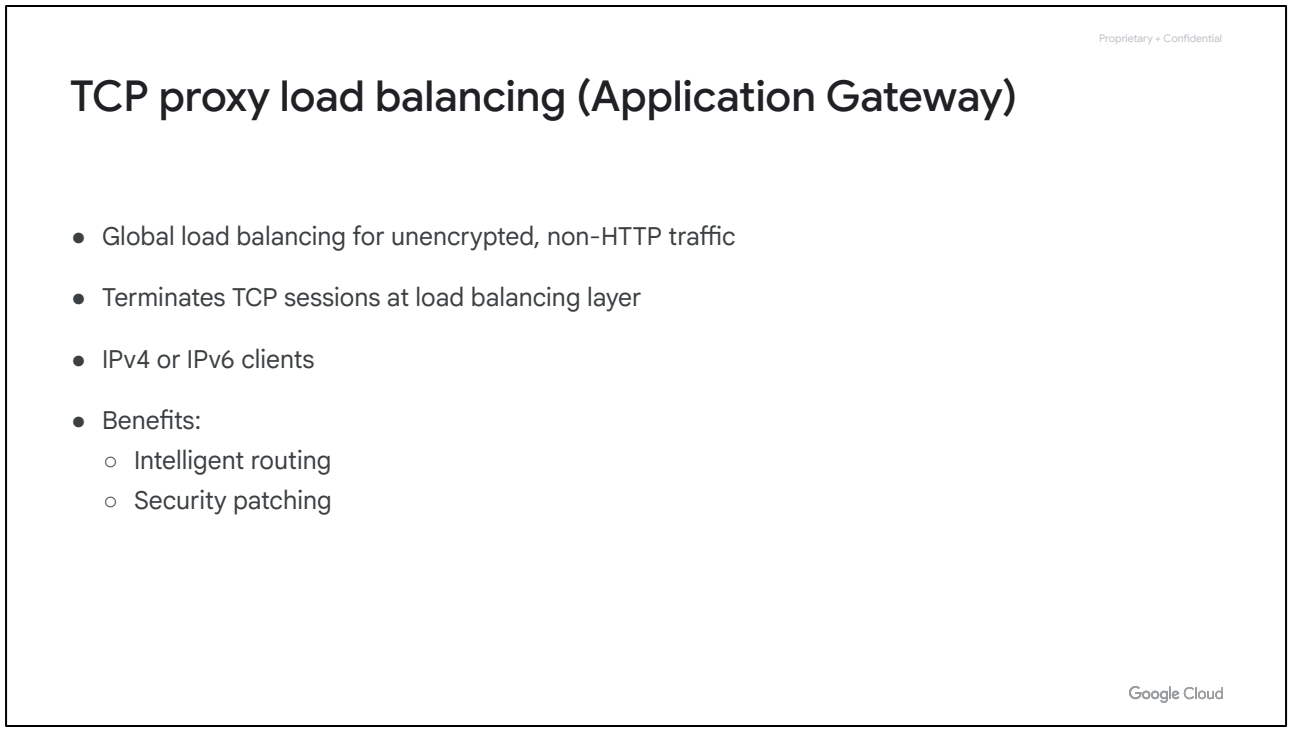
Let's use this diagram to examine an example of SSL proxy load balancing.

In this example, at Connection 1, SSL traffic from users in Iowa and Boston is terminated at the global load balancing layer.

A separate connection (Connection 2) is established to the closest backend instance.

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.

In this example, the traffic between the proxy and the backends can use SSL or TCP. Google recommends using SSL.



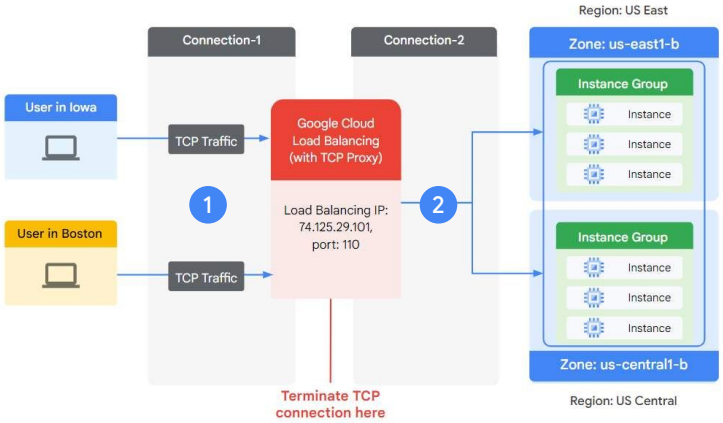
Source: Architecting with Compute Engine slides.

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 documentation page.

External TCP Proxy Load Balancing: Overview and example



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.

Let's use this diagram to explore an example with TCP proxy load balancing.

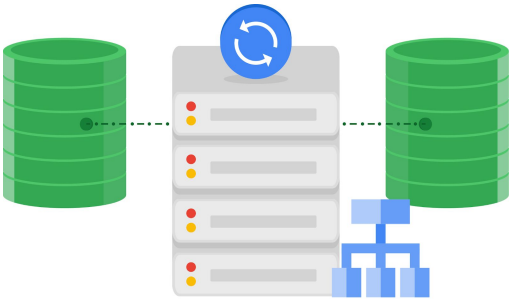
In this example, at connection one, TCP traffic from users in Iowa and Boston is terminated at the global load balancing layer.

A separate connection (connection two) 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.

In this example, the traffic between the proxy and the backends can use SSL or TCP. Google recommends using SSL.

Network load balancing (Network Load Balancer)

- Regional, non-proxied or proxied load balancer
- Forwarding rules (IP protocol data)
- Traffic:
  - UDP
  - TCP/SSL ports & many more
- Architecture:
  - Backend service-based
  - Target pool-based
  - Listener / Target Group

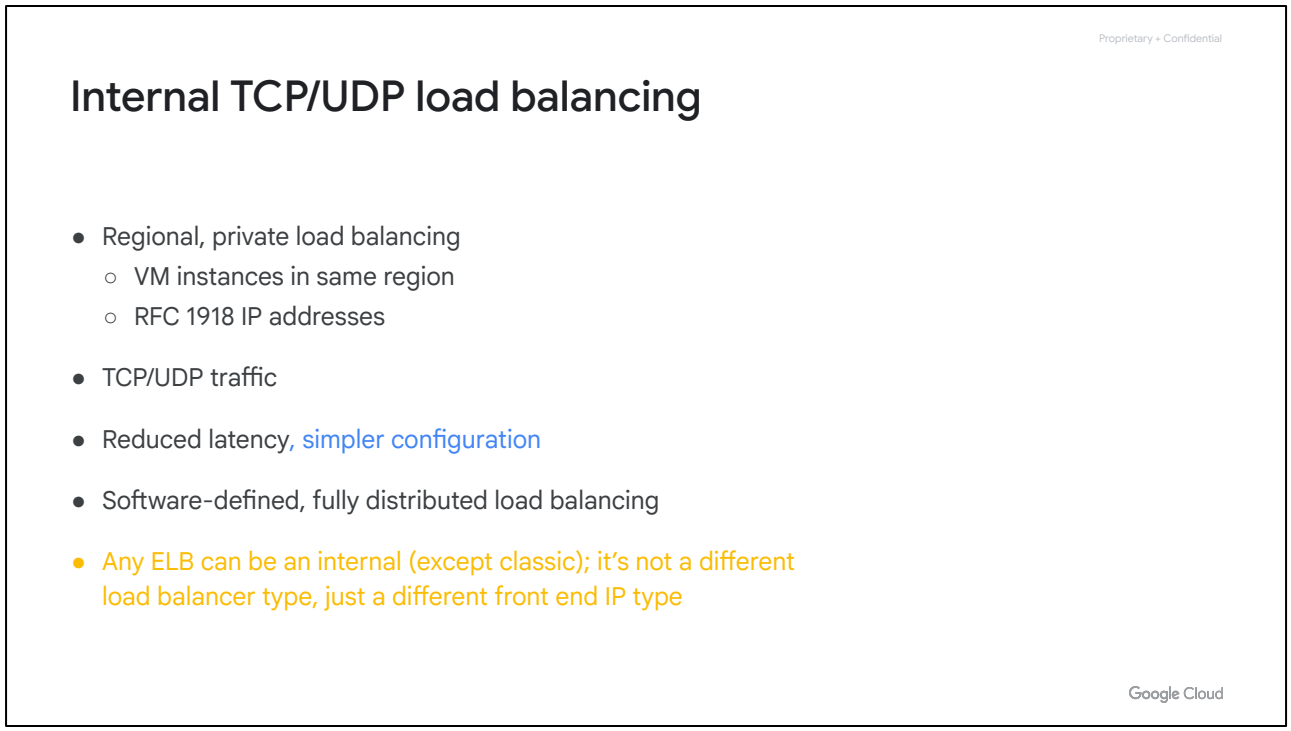


Source: Architecting with Compute Engine slides.

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 architecture of a network load balancer depends on whether you use a backend service-based network load balancer or a target pool-based network load balancer. Let's explore these in more detail.



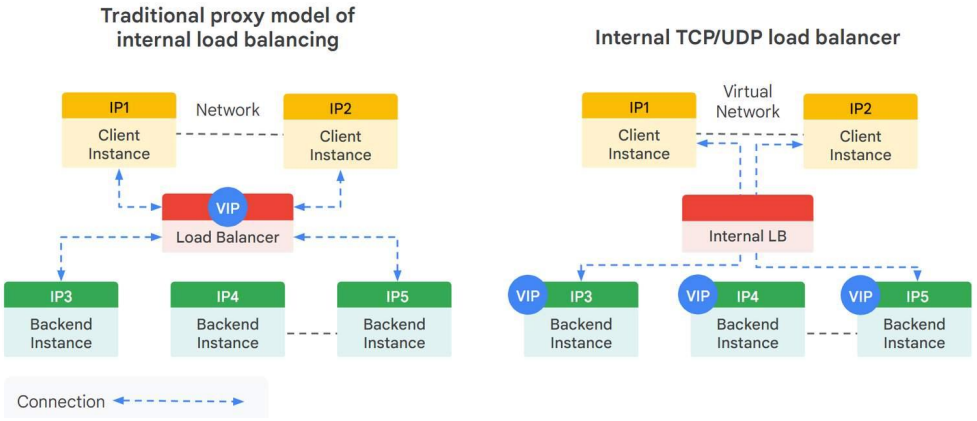
Source: Architecting with Compute Engine slides.

The internal TCP/UDP load balancer 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, configure an internal TCP/UDP load balancing IP address to act as the frontend to your private backend instances. Because you don't need a public IP 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 TCP/UDP load balancing service.

Internal TCP/UDP Load Balancing service: Benefits



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

Let's examine the benefit of using a software-defined internal TCP/UDP load balancing service.

In the traditional proxy model of internal load balancing, 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.

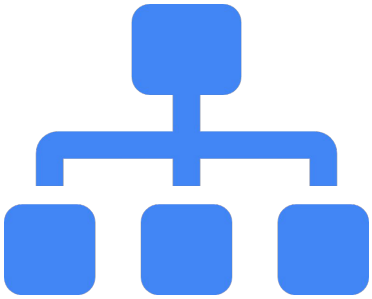
Essentially, there are two connections: one between the client and the load balancer, and one between the load balancer and the backend.

Google Cloud internal TCP/UDP load balancing distributes client instance requests to the backends using a different approach.

It uses lightweight load balancing built on top of Andromeda (Google Cloud's network virtualization stack) to provide software-defined load balancing that directly delivers the traffic from the client instance to a backend instance.

Internal HTTP(S) Load Balancing

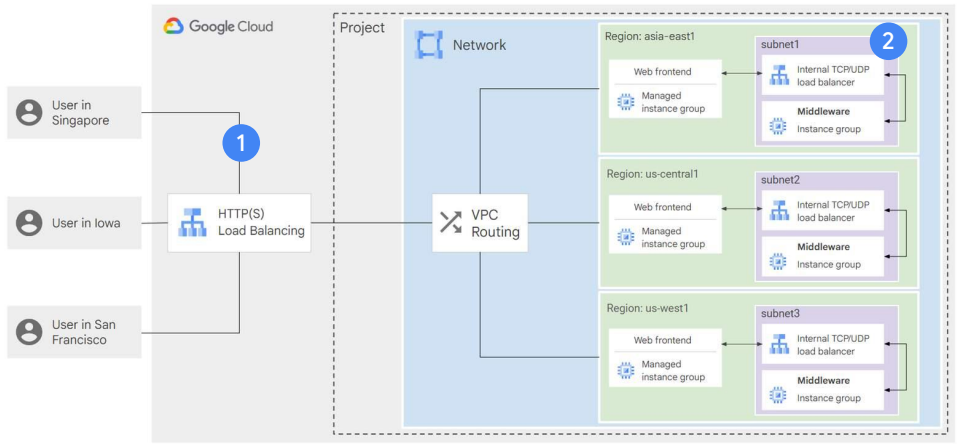
- A proxy-based, regional Layer 7 load balancer that also enables you to run and scale your services behind an internal load balancing IP address.
- Backend services support the HTTP, HTTPS, or HTTP/2 protocols.
- It is a managed service based on the open source Envoy proxy, enabling rich traffic control capabilities based on HTTP(S) parameters.



Google Cloud Internal HTTP(S) Load Balancing is a proxy-based, regional Layer 7 load balancer that also enables you to run and scale your services behind an internal load balancing IP address. Backend services support the HTTP, HTTPS, or HTTP/2 protocols.

Internal HTTP(S) Load Balancing is a managed service based on the open source Envoy proxy. This enables rich traffic control capabilities based on HTTP(S) parameters. After the load balancer has been configured, it automatically allocates Envoy proxies to meet your traffic needs.

Internal HTTP(S) Load Balancing: Example



Internal load balancing enables you to support use cases such as traditional 3-tier web services.

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.

Let's explore an example using this diagram.

First, is External HTTP(S) Load Balancing.

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-west1, us-central1 and asia-east1 regions because this is a global load balancer.

Internal TCP/UDP load balancer

These backends access an internal load balancer in each region as the application or internal tier.

The backends of this internal tier are located in us-west1-a, us-central1-b, and asia-east1-b.

The last tier is the database tier in each of those zones.

Summary of load balancers

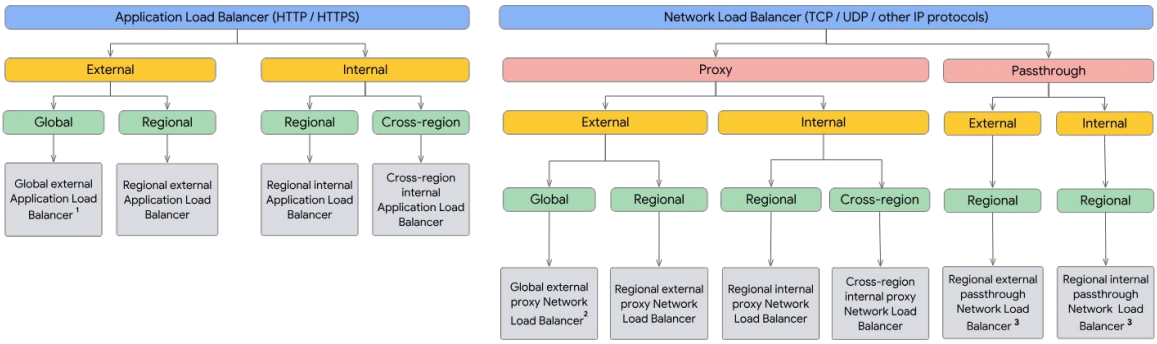
Load balancer	Traffic type	Global/ Regional	External/ Internal	External ports for load balancing
HTTP(S) (ALB)	HTTP or HTTPS	Global [regional for HTTP(s)] IPv4 IPv6	External (external or internal)	HTTP on 80 or 8080; HTTPS on 443 (any port)
SSL Proxy	TCP with SSL offload			Any
TCP Proxy	• TCP without SSL offload • Does not preserve client IP addresses			Any
Network TCP/UDP	• TCP/UDP without (optional) SSL offload • Can preserve client IP addresses	Regional IPv4 and/or IPv6		Any
Internal TCP/UDP	TCP or UDP		Internal	Any
Internal HTTP(S)	HTTP or HTTPS			HTTP on 80 or 8080; HTTPS on 443

Source: Architecting with Compute Engine slides.

If you prefer a table over a flow chart, we 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.

Global versus regional load balancing



To determine which Cloud Load Balancing product to use, you must first determine what traffic type your load balancers must handle.

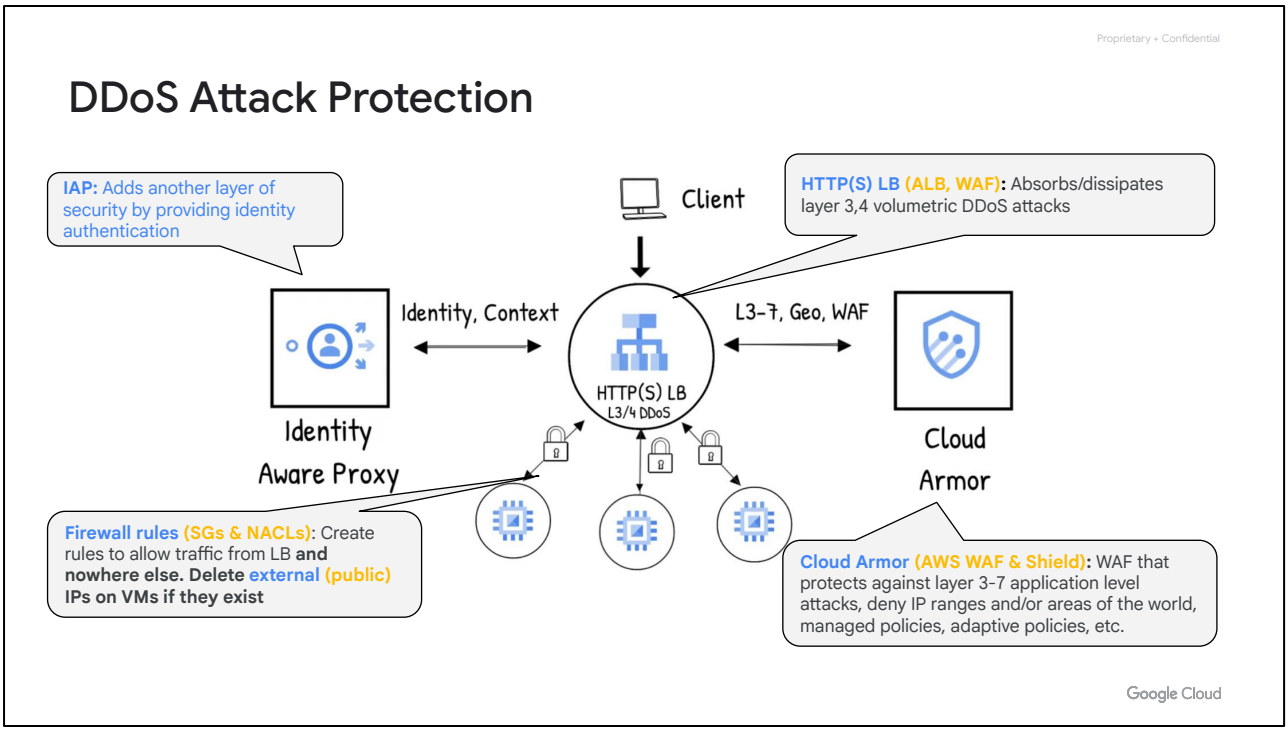
As a general rule, you'd choose an Application Load Balancer when you need a flexible feature set for your applications with HTTP(S) traffic.

You'd choose a proxy Network Load Balancer to implement TCP proxy load balancing to backends in one or more regions.

And, you'd choose a passthrough Network Load Balancer to preserve client source IP addresses, avoid the overhead of proxies, and to support additional protocols like UDP, ESP, and ICMP.

You can further narrow down your choices depending on your application's requirements: whether your application is external (internet-facing) or internal and whether you need backends deployed globally or regionally.

This diagram summarizes all the available deployment modes for Cloud Load Balancing.



WAF: web application firewall

What is a Volumetric Attack? A volumetric attack sends a high amount of traffic, or request packets, to a targeted network in an effort to overwhelm its bandwidth capabilities. These attacks work to flood the target in the hopes of slowing or stopping their services.

Video;
https://www.youtube.com/watch?v=0fQr7TRhnnU&list=PLTWE_lmu2InBzuPmOcgAYP7U80a87cpJd

Corresponding blog

- https://cloud.google.com/blog/topics/developers-practitioners/what-cloud-load-balancing

Google Cloud Armor (AWS WAF & Shield)

Cloud Armor Network Security

Mitigate infrastructure DDoS attacks with any Global Load balancer (ALB and/or AWS Cloudfront) (TCP SYN floods, Amplification attacks, IP fragmentation attacks, and more)

Allow or block traffic based on IP, Geo, and custom match parameters (L3–L7)

Defend against application layer attacks with OWASP Top 10 (for example, SQL injection and XSS). Use in combination with IAP

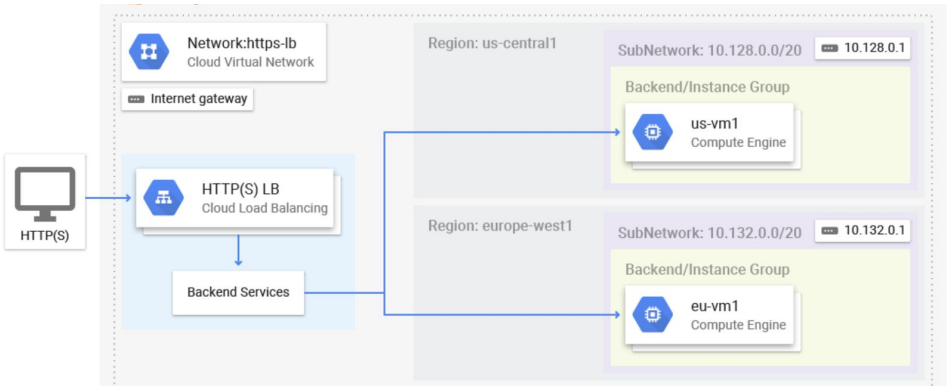
Telemetry: Decisions logged to Cloud Logging and Monitoring (AWS CloudWatch) dashboards, and Cloud Security Command Center

Google Cloud Armor Managed Protection Plus

Feature	Standard	Managed Protection Plus
Billing Method	Pay As You Go	Monthly Subscription + Data Processing (Pay As You Go)
DDoS Protection for	HTTP(S) Load Balancer, TCP/SSL Proxies	HTTP(S) LB, TCP/SSL Proxies (ALB, CDN)
Cloud Armor WAF	A-la-carte (policy, rule, request)	Included
Time Commitment	None	1 year
Named IP Lists	No	Yes
Adaptive Protection	No	Yes
DDoS Response Support	No	Yes (w/Premium Support)
DDoS Bill Protection	No	Yes

Instance Groups, NEGs

Scalable, Fault Tolerant Architecture Illustrated



This illustration shows a fault tolerant architecture using load balancing and autoscaling.

In this example, HTTP(S) traffic is being delivered to an Global Load Balancer. The load balancer is a managed service, so performance and high availability is built into the service.

The backend service is responsible for distributing traffic to the appropriate region based on latency.

Each region, in this example, has a managed instance group of instances that will be increased or decreased based on a predefined metric you choose based on the application.

The infrastructure shown can survive a region outage and still function.

Scalable, Fault-Tolerant Architecture Creation

To set up what is diagrammed on the previous slide, you need to create:

An instance (launch) template	One or more instance groups (EC2 Autoscaling)	A global (multiple regional) load balancers and Route 53 to distribute across them	A backend service (Target groups for each load balancer)

In order to build the infrastructure in the previous slide, you would need to:

- Build an instance template
- Create one or more managed instance groups(MIG) at the regions required
- Build a global load balancer with the MIG as the backend service

Backend options

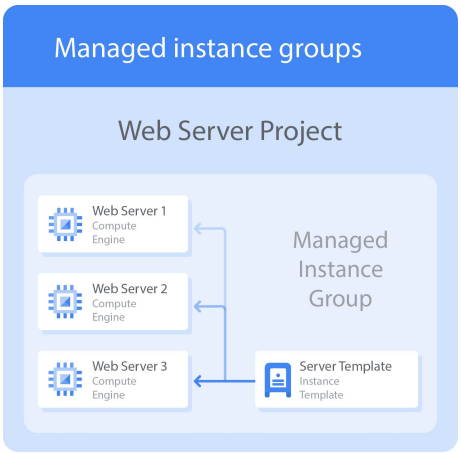
- Backend (Target Group) - one or more endpoints that receive traffic from a load balancer
- Options include:
  - Instance (Autoscaling) groups
  - Cloud Storage (S3) buckets
  - Network endpoint groups (NEG) (No AWS equivalent)
    - A group of backend endpoints or services
    - Common use case: Services running in containers
  - Zonal NEG
    - A group of VMs in the same network, subnet and zone
  - Serverless NEG (Lambda function)
    - Cloud Run, Cloud (Lambda) Function, App Engine, API Gateway
  - Hybrid connectivity NEG (on-prem IP or CIDR block)
    - Routing traffic to an on-premises location or another cloud

Backend services overview

Backend services overview
https://cloud.google.com/load-balancing/docs/backend-service

Managed Instance Groups (auto scaling groups)

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



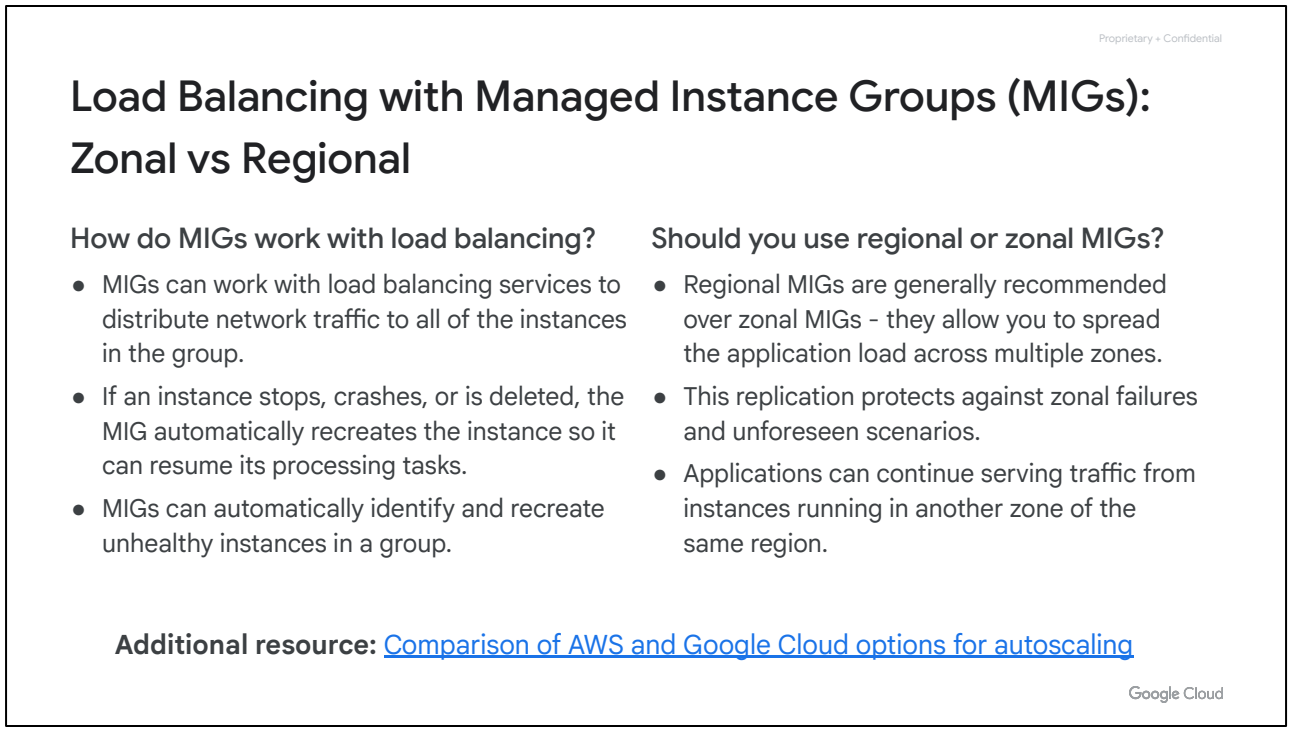
Source: Architecting with Compute Engine slides.

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.



Let's explore MIGs further.

So how do MIGs work with load balancing?

MIGs 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 MIG 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. MIGs can automatically identify and recreate unhealthy instances in a group to ensure that all the instances are running optimally.

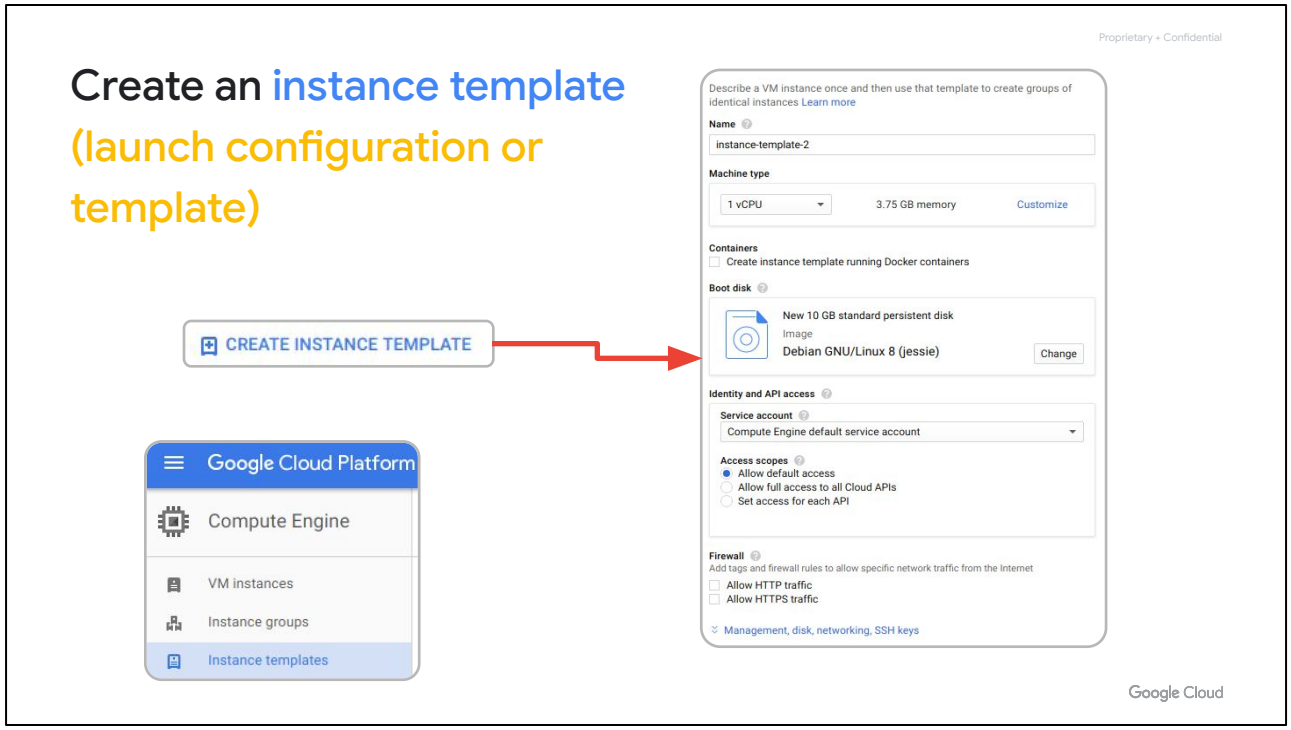
Next let's discuss if you should use regional or zonal MIGs.

Regional MIGs are generally recommended over zonal MIGs 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 MIGs 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.



Source: Architecting with Compute Engine slides.

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 Cloud 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.

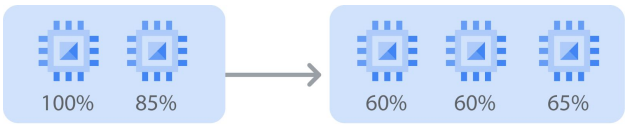
Managed instance groups (auto scaling groups) offer auto scaling capabilities

Dynamically add/remove instances:

- Increases in load
- Decreases in load

Autoscaling policy:

- CPU utilization
- Load balancing capacity
- Monitoring (Cloud watch) metrics
- Queue-based workload
- Prediction of demand based on past history



Target CPU utilization = 75%

Source: Architecting with Compute Engine slides.

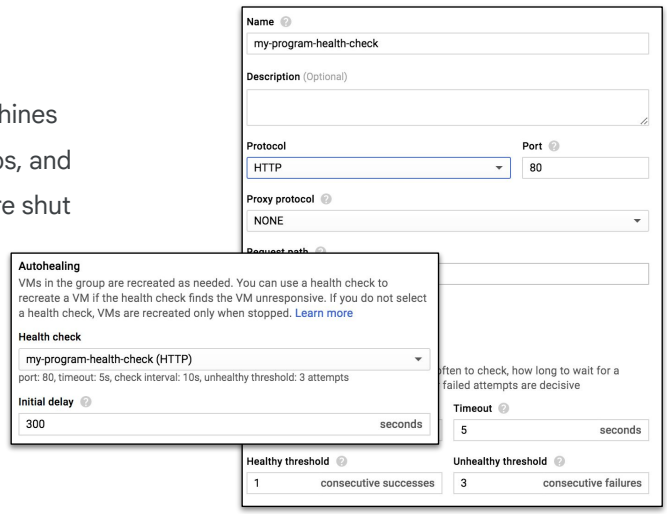
Let me provide more details on the autoscaling and health checks of a managed instance group. As we mentioned earlier, managed instance groups offer autoscaling capabilities that allow you to automatically add or remove instances from a managed instance group based on increases or decreases in load. 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 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?

Health Checks

- Simply makes requests to the machines in the instance (autoscaling) groups, and if the machines don't work, they are shut off and new ones created
- Parameters control:
  - Where to make request
  - Via what port
  - How often



Set up an application health check and autohealing

Google Cloud

Health checks ensure that only fully operable instances are being used from the group. With health checks, instances that fail the health check can be excluded and new instances can take their place.

Parameters required to configure the health check include:

- Where to make the request
- What port to use
- And how often to run the check

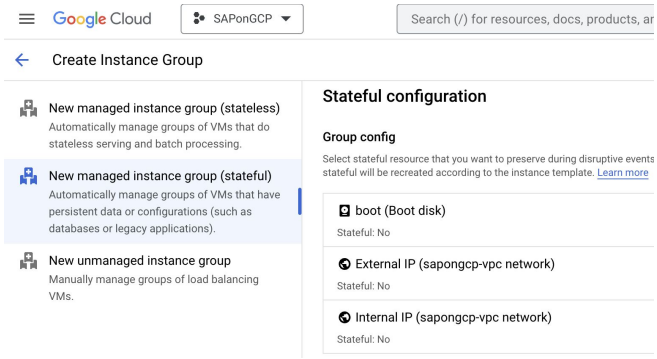
Health checks need a firewall rule to allow incoming probes from certain ports. See https://cloud.google.com/compute/docs/instance-groups/autohealing-instances-in-migs

Stateful vs stateless

And why stateless is usually preferred...

Exam Tips:

- *Here a look at this document.*
- *Prefer stateless. Use stateful only when necessary, eg:*
  - *Databases*
  - *Data processing apps (Kafka etc)*
  - *Legacy monoliths*



Stateful	Stateless (PREFERRED!)
Each server retains information about it's client sessions, such as the current state of an application or the content of a user shopping cart.	Server does not retain any information about the client sessions. Each request made by the client is treated as an independent transaction, and the server does not maintain any memory of previous requests.
Not perfect if we have multiple backends that can serve the requests…	Greater scalability and flexibility.
Can scale up easily. Can't scale down easily since each server keeps its' state.	Ability to scale up & down easily

Google Cloud

[Some of those may be about differentiating between a Stateful and Stateless MIGs. If possible, use stateless as with stateful it's much harder to scale down, since each VM keep its state locally]

Stateful Managed Instance Groups

- MIGs support both stateful and stateless workloads
  - Stateful workloads preserve individual VM state (for example, a database shard, or app configuration) on the VM's disks
    - Not easy to scale horizontally (add or remove nodes)
      - Could require data replication, creation or deletion of data shards, or changing the overall application configuration
    - If VM is reported as “unhealthy”
      - Need to retain VM identity (name), IP address, metadata, and data to be used when a new one is created
  - Stateless workloads, like a web frontend, do not retain any state on the individual VMs
    - Easy to scale up/down as needed

Stateful managed instance groups

Network Endpoint Groups (NEGs) (no AWS equivalent)

Type	Purpose	Scope	Health checks
Zonal	1+ Google Cloud GCE VMs or GKE pods	Zone	Y
Internet	1 Public IP or FQDN outside Google Cloud (ex. AWS ELB, on-prem LB, etc.)	Global	N/A
Serverless	1 FQDN belonging to an App Engine, Cloud Functions, API Gateway, or Cloud Run service	Region	N/A
Hybrid Connectivity	1+ endpoints that resolve to on-premises services, server applications in another cloud, and other internet-reachable services outside Google Cloud using hybrid connectivity (VPN or Interconnect)	Zone	Y
Private Service Connect	1 endpoint that resolves to either a Google-managed regional API endpoint or a managed service published using Private Service Connect	Region	N/A

See https://cloud.google.com/load-balancing/docs/negs for details on which load balancer types each NEG is compatible with.

Diagnostic Question Discussion

How can application parts, owned by different project teams, communicate over RFC1918 addresses?

A. Single project, same VPC

B. Shared VPC, each team's project a service of the Shared VPC project

C. Parts communicate using HTTPS

D. Communicate over global load balancers, one per project

Google Cloud

B

Diagnostic Question Discussion

How can application parts, owned by different project teams, communicate over RFC1918 addresses?

A. Single project, same VPC

B. Shared VPC, each team's project a service of the Shared VPC project

C. Parts communicate using HTTPS

D. Communicate over global load balancers, one per project

B

Diagnostic Question Discussion

Your company is running its application workloads on Compute Engine. The applications have been deployed in production, acceptance, and development environments. The production environment is business-critical and is used 24/7, while the acceptance and development environments are only critical during office hours. Your CFO has asked you to optimize these environments to achieve cost savings during idle times.

What should you do?

A. Create a shell script that uses the gcloud command to change the machine type of the development and acceptance instances to a smaller machine type outside of office hours. Schedule the shell script on one of the production instances to automate the task.

B. Use Cloud Scheduler to trigger a Cloud Function that will stop the development and acceptance environments after office hours and start them just before office hours.

C. Deploy the applications using managed instance groups and enable autoscaling based on appropriate metrics.

D. Use regular Compute Engine instances for the production environment, and use spot VMs for the acceptance and development environments.

Eliminate A -> it introduces downtime, and also shell script + NO MIGs is not really a cloud-native approach
Eliminate B -> “not critical” does not mean “not needed at all”.
Eliminate D -> business-Critical, so can’t use Spot anywhere, even dev/uat

C - the only choice that makes sense. It will cost-optimize since we switch from VMs to MIGs with autoscaling

Diagnostic Question Discussion

Your company is running its application workloads on Compute Engine. The applications have been deployed in production, acceptance, and development environments. The production environment is business-critical and is used 24/7, while the acceptance and development environments are only critical during office hours. Your CFO has asked you to optimize these environments to achieve cost savings during idle times.

What should you do?

A. Create a shell script that uses the gcloud command to change the machine type of the development and acceptance instances to a smaller machine type outside of office hours. Schedule the shell script on one of the production instances to automate the task.

B. Use Cloud Scheduler to trigger a Cloud Function that will stop the development and acceptance environments after office hours and start them just before office hours.

C. Deploy the applications using managed instance groups and enable autoscaling based on appropriate metrics.

D. Use regular Compute Engine instances for the production environment, and use spot VMs for the acceptance and development environments.

Google Cloud

Eliminate A -> it introduces downtime, and also shell script + NO MIGs is not really a cloud-native approach
Eliminate B -> “not critical” does not mean “not needed at all”.
Eliminate D -> business-Critical, so can’t use Spot anywhere, even dev/uat

C - the only choice that makes sense. It will cost-optimize since we switch from VMs to MIGs with autoscaling

EHR Healthcare
case study

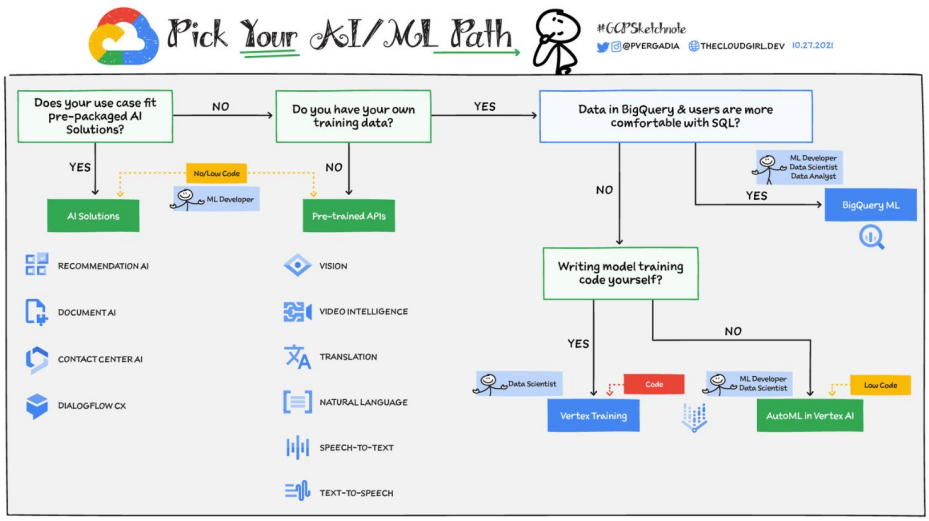
https://services.google.com/fh/files/misc/v6.1_pca_ehr_healthcare_case_study_english.pdf

The ONLY one of “old” case studies.
AI use-cases limited to “classical” AI (not Gen AI)

Proposed Technical Solution

- Data protection: HIPAA, Sensitive Data Protection, data encryption (possibly CMEK, KMS, HSM, EKM), least privilege approach (IAM, custom roles, IAP, ...), secure access to VMs and services, audit logs, bucket locks, Organization Policy Service.
- Kubernetes + "a group of Kubernetes clusters": GKE (possibly Autopilot mode), plus strong arguments for Anthos / Cloud Service Mesh ("multiple, potentially different environments")
  - consistent management, possibly from a single system: Config Controller
  - Manage traffic and execute resiliency tests with Service Mesh: Fault Injection, Circuit Breaking, Request Timeouts
- MySQL + MS SQL Server -> Cloud SQL (DMS for migration); Redis -> Memorystore; MongoDB -> MongoDB on GKE -> Firestore
- APIs for integration: Apigee (since it's integration with on-prem)
- Active Directory:
  - GCDS: Replication AD -> Cloud Identity, possibly also ADFS: AD Federation Services for AD-based single sign-on.
- Email-based alerting and Telemetry modernization: Cloud Operations Suite, uptime checks, SLIs and SLOs, dashboards and different notification channels. Alerting overview.
- Secure and high-performance connection between on-premises and GCP: Interconnect + Cloud VPN (HA) as backup
- CI/CD: Cloud Build + Artifact Registry.
- Ingesting and processing data from new providers: ETL pipeline (possibly Pub/Sub -> Dataflow -> BigQuery)
- Dynamic provisioning of new environments: IaaC (Terraform).
- Making predictions: ML in the form of Vertex AI / AutoML / BigQuery ML / pre-built models (nothing very concrete)
- Generate reports on industry trends based on provider data: Vertex AI Search for Healthcare
- Security products: Cloud Armor, Security Command Center

Google Cloud (pre Gen-AI) machine learning spectrum



Exam Tip: *4 high-level ways to approach ML in GCP. All depends on particular use-case, but the approach should be like with compute: start from more managed / easier approaches and choose more advanced ones only if needed*

And here are the alternatives you have.
As we know, Data Scientists and ML experts are scarce and thus Google tries to support us even if don't have a big know-how in this area, but we'd like to implement some Machine Learning capabilities to our systems. What we can do here is we can try to create our own models by using Tensorflow library that Google created, possibly using Vertex AI Notebooks I mentioned earlier.
But if your use-case is more standard, like you want to do some pattern detection, recommendations, text translation, image recognition etc, you can use AutoML. Which is nothing more than some models that can be joined with your data. Let's assume you have a factory that produces some items and you'd like to have some automated quality control for each item you produce. There is no chance that a model already exists that can differentiate between a flawless item and one that has some slight defect. But there are models that are able to detect general patterns based on pictures and you can train them with your data. So you upload lots of pictures of the flawless items and some with defects, allowing the model to learn on this data, and then you use this customized model productively to make real-time decisions.
But there are even simpler use-cases, where you'd have an online shop and you'd like to have all the comments automatically translated to English, or you'd like to check semantics - so automatically validate if the comments are positive or negative. Then you don't need to create a model or even train anything, but you can use something that's already able to give you that insight. And these are pre-trained models that you can just use from within your code by just calling a specific API and asking for the result… so these are models that can be consumed as a service.

[EHR case study] Diagnostic Question #1

For this question, refer to the EHR Healthcare case study. You need to define the technical architecture for hybrid connectivity between EHR's on-premises systems and Google Cloud. You want to follow Google's recommended practices for production-level applications.

Considering the EHR Healthcare business and technical requirements, what should you do?

A. Configure two Partner Interconnect connections in one metro (City), and make sure the Interconnect connections are placed in different metro zones.

B. Configure two VPN connections from on-premises to Google Cloud, and make sure the VPN devices on-premises are in separate racks.

C. Configure Direct Peering between EHR Healthcare and Google Cloud, and make sure you are peering at least two Google locations.

D. Configure two Dedicated Interconnect connections in one metro (City) and two connections in another metro, and make sure the Interconnect connections are placed in different metro zones.

Google Cloud

https://cloud.google.com/network-connectivity/docs/interconnect/tutorials/dedicated-creating-9999-availability

D

[EHR case study] Diagnostic Question #1

For this question, refer to the EHR Healthcare case study. You need to define the technical architecture for hybrid connectivity between EHR's on-premises systems and Google Cloud. You want to follow Google's recommended practices for production-level applications.

Considering the EHR Healthcare business and technical requirements, what should you do?

A. Configure two Partner Interconnect connections in one metro (City), and make sure the Interconnect connections are placed in different metro zones.

B. Configure two VPN connections from on-premises to Google Cloud, and make sure the VPN devices on-premises are in separate racks.

C. Configure Direct Peering between EHR Healthcare and Google Cloud, and make sure you are peering at least two Google locations.

D. Configure two Dedicated Interconnect connections in one metro (City) and two connections in another metro, and make sure the Interconnect connections are placed in different metro zones.

Google Cloud

https://cloud.google.com/network-connectivity/docs/interconnect/tutorials/dedicated-creating-9999-availability

D

[EHR case study] Diagnostic Question #2

For this question, refer to the EHR Healthcare case study. In the past, configuration errors put public IP addresses on backend servers that should not have been accessible from the Internet. You need to ensure that no one can put external IP addresses on backend Compute Engine instances and that external IP addresses can only be configured on frontend Compute Engine instances.

What should you do?

A. Create an Organizational Policy with a constraint to allow external IP addresses only on the frontend Compute Engine instances.

B. Revoke the compute.networkAdmin role from all users in the project with front end instances.

C. Create an Identity and Access Management (IAM) policy that maps the IT staff to the compute.networkAdmin role for the organization.

D. Create a custom Identity and Access Management (IAM) role named GCE_FRONTEND with the compute.addresses.create permission.

https://cloud.google.com/compute/docs/ip-addresses/reserve-static-external-ip-address#disableexternalip

[EHR case study] Diagnostic Question #2

For this question, refer to the EHR Healthcare case study. In the past, configuration errors put public IP addresses on backend servers that should not have been accessible from the Internet. You need to ensure that no one can put external IP addresses on backend Compute Engine instances and that external IP addresses can only be configured on frontend Compute Engine instances.

What should you do?

A. Create an Organizational Policy with a constraint to allow external IP addresses only on the frontend Compute Engine instances.

B. Revoke the compute.networkAdmin role from all users in the project with front end instances.

C. Create an Identity and Access Management (IAM) policy that maps the IT staff to the compute.networkAdmin role for the organization.

D. Create a custom Identity and Access Management (IAM) role named GCE_FRONTEND with the compute.addresses.create permission.

Google Cloud

https://cloud.google.com/compute/docs/ip-addresses/reserve-static-external-ip-address#disableexternalip

[EHR case study] Diagnostic Question #3

For this question, refer to the EHR Healthcare case study. You are responsible for ensuring that EHR's use of Google Cloud will pass an upcoming privacy compliance audit.

What should you do? (Choose two.)

A. Verify EHR's product usage against the list of compliant products on the Google Cloud compliance page.

B. Advise EHR to execute a Business Associate Agreement (BAA) with Google Cloud.

C. Use Firebase Authentication for EHR's user facing applications.

D. Implement Prometheus to detect and prevent security breaches on EHR's web-based applications.

E. Use GKE private clusters for all Kubernetes workloads.

Google Cloud

A - OK (Google Cloud compliance page will give list of products those are HIPAA compliant https://cloud.google.com/security/compliance/offerings?skip_cache=true#/regions=USA&industries=Healthcare_and_life_sciences&focusArea=Privacy)
B - OK (BAA means HIPAA Business Associate amendment or Business Associate Agreement entered into between Google and Customer. With EHR being a leading provider of health record software, this agreement is required.
https://cloud.google.com/files/gcp-hipaa-overview-guide.pdf?hl=en)
https://cloud.google.com/security/compliance/hipaa#overview

C - Eliminated (Firebase authentication provides backend services, easy-to-use SDKs and ready-made libraries to users on App. https://firebase.google.com/docs/auth)
D - Eliminated (more of an observability platform)
E - Eliminated (Running distributed services in GKE private clusters gives enterprises both secure and reliable services. Not sure how this may help with Private Compliance Audit)

[EHR case study] Diagnostic Question #3

For this question, refer to the EHR Healthcare case study. You are responsible for ensuring that EHR's use of Google Cloud will pass an upcoming privacy compliance audit.

What should you do? (Choose two.)

A. Verify EHR's product usage against the list of compliant products on the Google Cloud compliance page.

B. Advise EHR to execute a Business Associate Agreement (BAA) with Google Cloud.

C. Use Firebase Authentication for EHR's user facing applications.

D. Implement Prometheus to detect and prevent security breaches on EHR's web-based applications.

E. Use GKE private clusters for all Kubernetes workloads.

Google Cloud

A - OK (Google Cloud compliance page will give list of products those are HIPAA compliant https://cloud.google.com/security/compliance/offerings?skip_cache=true#/regions=USA&industries=Healthcare_and_life_sciences&focusArea=Privacy)
B - OK (BAA means HIPAA Business Associate amendment or Business Associate Agreement entered into between Google and Customer. With EHR being a leading provider of health record software, this agreement is required. https://cloud.google.com/files/gcp-hipaa-overview-guide.pdf?hl=en)
https://cloud.google.com/security/compliance/hipaa#overview

C - Eliminated (Firebase authentication provides backend services, easy-to-use SDKs and ready-made libraries to users on App. https://firebase.google.com/docs/auth)
D - Eliminated (more of an observability platform)
E - Eliminated (Running distributed services in GKE private clusters gives enterprises both secure and reliable services. Not sure how this may help with Private Compliance Audit)

Make sure to...

Enjoy the journey as much as the destination!

Now that you know about the overall setup of this course and how to use the workbook, let's get started by exploring section 1 of the exam guide.