



Google Cloud VPC Networking Fundamentals

Philipp Maier

In this module, we will be covering Google Cloud VPC Networking Fundamentals.

GCP uses a software-defined network that is built on a global fiber infrastructure. This infrastructure makes GCP one of the world's largest and fastest networks. Thinking about resources as services rather than as hardware will help you understand the options that are available, and their behavior.

Agenda

Virtual Private Cloud (VPC)

Projects, networks, and subnetworks

IP addresses

Routes and firewall rules

Lab

Multiple network interfaces

Lab

In this module, we will start by introducing Virtual Private Cloud, or VPC, which is Google's managed networking functionality for your Cloud Platform resources. Then, we are going to dissect networking into its fundamental components, which are projects, networks, subnetworks, IP addresses, routes, and firewall rules.

You will explore GCP's network structure in two labs by creating networks and subnetworks of many different varieties and exploring the network relationships between them, including how to create a VM instance with multiple network interfaces.

Agenda

Virtual Private Cloud (VPC)

Projects, networks, and subnetworks

IP addresses

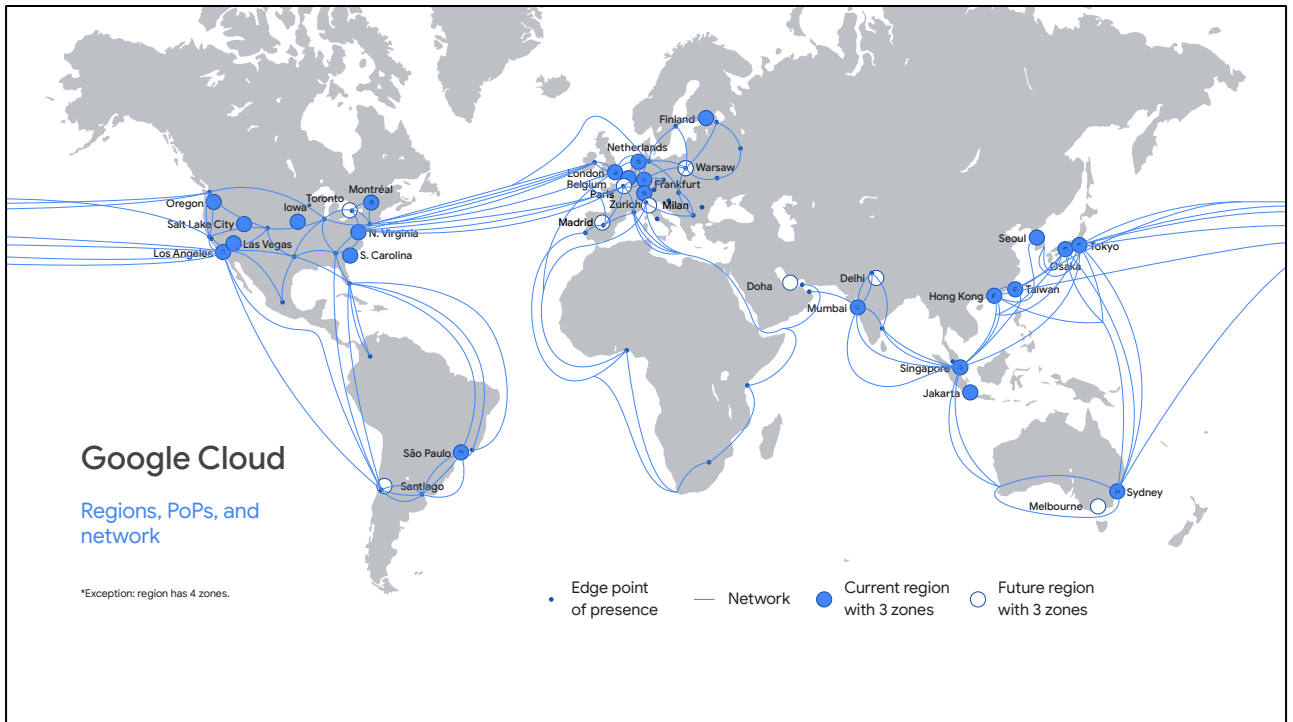
Routes and firewall rules

Lab

Multiple network interfaces

Lab

Let's start by talking about GCP's network and specifically Virtual Private Cloud, or VPC.



This map represents Google Cloud. On a high level, Google Cloud consists of regions which are the icons in blue, points of presence or PoPs which are the dots in blue, a global private network which is represented by the blue lines, and services.

A region is a specific geographical location where you can run your resources. This map shows several regions that are currently operating, as well as future regions. The number on each region represents the zones within that region. For example, in Iowa there is a region called us-central1 which has four zones: us-central1-a, us-central1-b, us-central1-c and us-central1-f. For more information on current regions and zones, please refer to documentation linked below this video.

<https://cloud.google.com/compute/docs/regions-zones/>

The PoPs are where Google's network is connected to the rest of the internet. Google Cloud can bring its traffic closer to its peers because it operates an extensive global network of interconnection points. This reduces costs and provides users with a better experience.

The network connects regions and PoPs and is composed of hundreds of thousands of miles of fiber optic cable and several submarine cable investments.

For more information on Google's networking infrastructure, please refer to link below

this video. [<https://peering.google.com/#/infrastructure>]

Next, let's focus on services by starting with Google Cloud's Virtual Private Network or VPC.

VPC objects



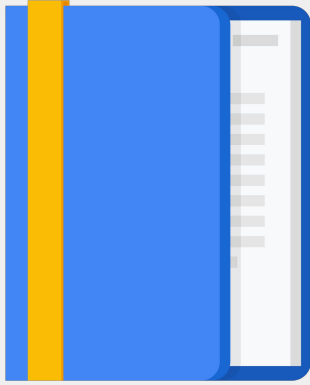
- Projects
- Networks
 - Default, auto mode, custom mode
- Subnetworks
- Regions
- Zones
- IP addresses
 - Internal, external, range
- Virtual machines (VMs)
- Routes
- Firewall rules

With GCP, you can provision your GCP resources, connect them to each other, and isolate them from one another in a Virtual Private Cloud. You can also define fine-grained networking policies within GCP, and between GCP and on-premises or other public clouds. Essentially, VPC is a comprehensive set of Google-managed networking objects that 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 course we will focus on configuring VM instances from a networking perspective.

We'll also go over routes and firewall rules.

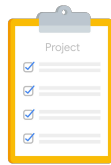
Agenda



- Virtual Private Cloud (VPC)
- **Projects, networks, and subnetworks**
- IP addresses
- Routes and firewall rules
- Lab
- Multiple network interfaces
- Lab

Let's start exploring the VPC objects by looking at projects, networks, and subnetworks.

Projects and networks



A project:

- Associates objects and services with billing
- Contains networks (up to 5)
- Networks can be shared/peered



A network:

- No IP address range
- Global and spans all available regions
- Contains subnetworks
- Type: default, auto, or custom




Projects are the key organizer of infrastructure resources in GCP. A project associates objects and services with billing. Now, what's unique, is the fact that projects actually contain entire networks. The default quota for each project is 5 networks but you can simply request additional quota using the GCP 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 in a later module.

These networks do not have IP ranges but are simply a construct of all of the individual IP addresses and services within that network. GCP's networks are global, spanning all available regions across the world, as shown 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.

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

There are 3 VPC network types

 Default	 Auto Mode	 Custom Mode
<ul style="list-style-type: none">• Every project• One subnet per region• Default firewall Rules	<ul style="list-style-type: none">• Default network• One subnet per region• Regional IP allocation• Fixed /20 subnetwork per region• Expandable up to /16	<ul style="list-style-type: none">• No default subnets created• Full control of IP ranges• Regional IP allocation• Expandable to any RFC 1918 size

Every project is provided with a default VPC network with predefined subnets and firewall rules. Specifically, a subnet is allocated for each region with non-overlapping CIDR blocks. Also, each default network has default firewall rules. These rules are configured to allow ingress traffic for ICMP, RDP and SSH traffic from anywhere, as well as ingress traffic from within the default network for all protocols and ports.

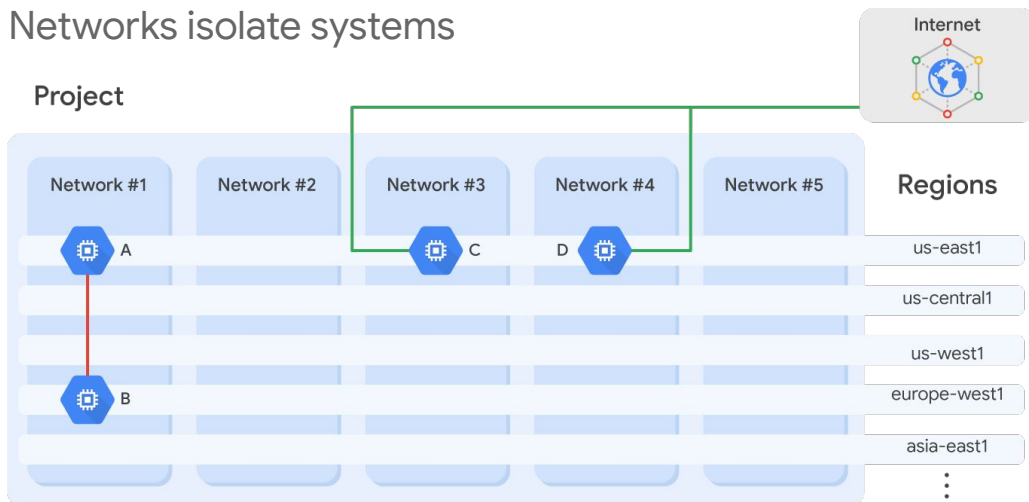
An auto mode network, also has one subnet per region. The default network is actually an auto mode network. Now, these automatically created subnets use a set of predefined IP ranges with a /20 mask that can be expanded to /16. All of these subnets fit within the 10.128.0.0/9 CIDR block. As new GCP regions become available, new subnets in those regions are automatically added to auto mode networks using an IP range from that block.

A custom mode network does not automatically create subnets. This type of network provides you with complete control over its subnets and IP ranges. You decide which subnets to create, in regions you choose, and using IP ranges you specify within the RFC 1918 address space. These IP ranges cannot overlap between subnets of the same network.

Now, you can convert an auto mode network to a custom mode network to take advantage of the control that custom mode networks provide. However, this

conversion is one way, meaning that custom mode networks cannot be changed to auto mode networks. So, carefully review the considerations for auto mode networks to help you decide which type of network meets your needs.

Networks isolate systems



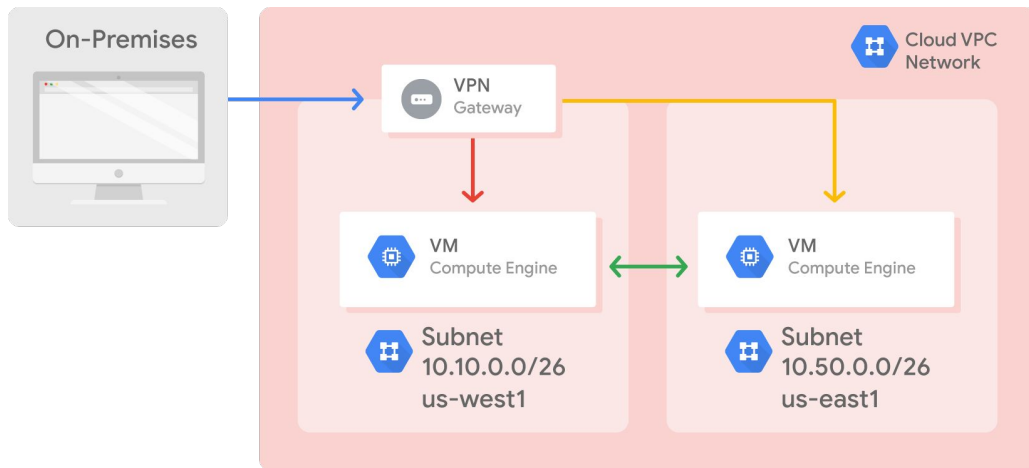
- A and B can communicate over internal IPs even though they are in different regions.
- C and D must communicate over external IPs even though they are in the same region.

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-hand side.

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 over 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 over external IPs, even though they are in the same region. The traffic between VMs C and D isn't actually touching the internet, but is going through the Google Edge routers, which has different billing and security ramifications that we are going to explore later.

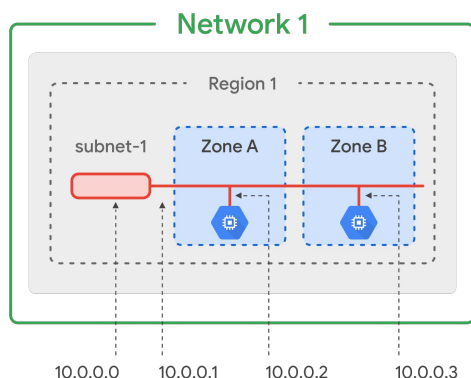
Google's VPC is global



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 GCP network, as shown in this diagram. Even though the two VM instances are in separate regions, namely 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



- VMs can be on the same subnet but in different zones
- A single firewall rule can apply to both VMs

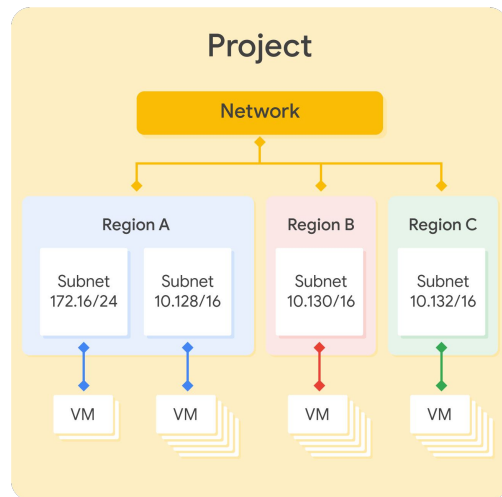
I mentioned that subnetworks work on a regional scale. Because a region contains several zones, subnetworks can cross zones.

This slide has a region, region 1, with two zones, zone A and zone 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 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 will 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.

Expand subnets without re-creating instances

- Cannot overlap with other subnets
- Inside the RFC 1918 address spaces
- Can expand but not shrink
- Auto mode can be expanded from /20 to /16
- Avoid large subnets



Speaking of IP addresses of a subnet, Google Cloud VPCs let you increase the IP space of any subnets without any workload shutdown or downtime.

This diagram illustrates a network with subnets that have different subnet masks, allowing for more instances in some subnets than others. This gives you flexibility and growth options to meet your needs, but there are some things to remember:

- The new subnet must not overlap with other subnets in the same VPC network in any region.
- Also, the new subnet must stay inside the RFC 1918 address spaces.
- The new network range must be larger than the original, which means the prefix length value must be a smaller number. In other words, you cannot undo an expansion.
- Now, auto mode subnets start with a /20 IP range. They can be expanded to a /16 IP range, but no larger. Alternatively, you can convert the auto mode subnetwork to a custom mode subnetwork to increase the IP range further.
- Also, avoid creating large subnets. Overly large subnets are more likely to cause CIDR range collisions when using Multiple Network Interfaces and VPC Network Peering, or when using a VPN or other connections to an on-premises network. Therefore, do not scale your subnet beyond what you actually need.

Migrate a VM between networks

- From legacy network to a VPC network in the same project.
- From one VPC network to another VPC network in the same project.
- From one subnet of a VPC network to another subnet of the same network.
- From a service project network to the shared network of a Shared VPC host project.

You can also migrate a VM instance from one network to another. In the case of a VM that is connected to more than one network using multiple network interfaces, this process updates one of the interfaces and leaves the rest in place.

The following migrations are supported:

- From legacy network to a VPC network in the same project
- From one VPC network to another VPC network in the same project
- From one subnet of a VPC network to another subnet of the same network
- From a service project network to the shared network of a Shared VPC host project

In all cases, the VM stays in the region and zone where it was before. Only the attached network changes.

[Migrating a VM between networks:

<https://cloud.google.com/compute/docs/instances/migrating-interfaces-between-networks#migrating-a-vm>]

Demo

Expand a subnet

Philipp Maier

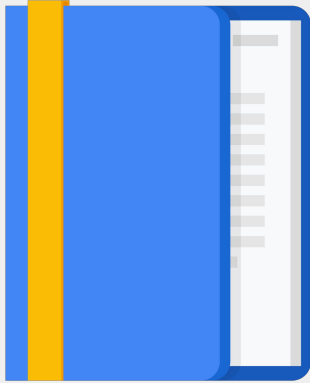
Let me show you how to expand a subnet within GCP.

I have gone ahead and created a custom subnet with a /29 mask. A /29 mask provides 8 addresses but of those 4 are reserved, leaving me with another 4 for my VM instances. Let's try to create another VM instance in this subnet.

[Demo]

That's how it easy it is to expand a subnet in GCP without any workload shutdown or downtime.

Agenda



- Virtual Private Cloud (VPC)
- Projects, networks, and subnetworks
- **IP addresses**
- Routes and firewall rules
- Lab
- Multiple network interfaces
- Lab

Now that we've covered Google Cloud networks at a high level, let's go deeper by exploring IP addresses.

VMs can have internal and external IP addresses



Internal IP

Allocated from subnet range to VMs by DHCP

DHCP lease is renewed every 24 hours

VM name + IP is registered with network-scoped DNS

External IP

Assigned from pool (ephemeral)

Reserved (static)

Bring Your Own IP address (BYOIP)

VM doesn't know external IP; it is mapped to the internal IP

In Google Cloud, each virtual machine can have two IP addresses assigned. One of them is an internal IP address, which is going to be assigned via DHCP. Every VM that starts up and any service that depends on virtual machines gets an internal IP address. Examples of such services are App Engine and Google Kubernetes Engine, which are explored in other courses.

When you create a VM in Google Cloud, its symbolic name is registered with an internal DNS service that translates the name to the internal IP address. DNS is scoped to the network, so it can translate web URLs and VM names of hosts in the same network, but it can't translate host names from VMs in a different network.

The other IP address is the external IP address, but this is optional. If your device or your machine is externally facing, you can assign an external IP address. 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. If you reserve a static external IP address and do not assign it to a resource such as a VM instance or a forwarding rule, you are charged at a higher rate than for static and ephemeral external IP addresses that are in use. You can use your own publicly routable IP address prefixes as Google Cloud external IP addresses and advertise them on the internet. In order to be eligible, you must own and bring a /24 block or larger.

[\[https://cloud.google.com/compute/network-pricing#ipaddress\]](https://cloud.google.com/compute/network-pricing#ipaddress)

Demo

Internal and external IP

Philipp Maier

I just mentioned that VMs can have internal and external IP addresses. Let's explore this in the GCP Console.

[Demo]

This demonstrates that every VM needs an internal IP address but external IP addresses are optional and by default they are ephemeral.

External IPs are mapped to internal IPs

Name ^	Zone	Machine type	Recommendation	In use by	Internal IP	External IP	Connect
<input checked="" type="checkbox"/> instance-1	us-east1-d	1vCPU, 3.75 GB			10.142.0.2	104.196.149.82	SSH ▾ ⋮

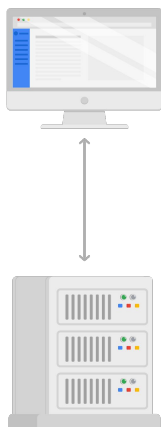
```
$ sudo /sbin/ifconfig
eth0
  Link encap:Ethernet  HWaddr 42:01:0a:8e:00:02
  inet addr:10.142.0.2  Bcast:10.142.0.2  Mask:255.255.255.255
  UP BROADCAST RUNNING MULTICAST  MTU:1460  Metric:1
  RX packets:397 errors:0 dropped:0 overruns:0 frame:0
  TX packets:279 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:1000
  RX bytes:66429 (64.8 KiB)  TX bytes:41662 (40.6 KiB)

lo
  Link encap:Local Loopback
  inet addr:127.0.0.1  Mask:255.0.0.0
  inet6 addr: ::1/128 Scope:Host
  UP LOOPBACK RUNNING  MTU:65536  Metric:1
  RX packets:0 errors:0 dropped:0 overruns:0 frame:0
  TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
  collisions:0 txqueuelen:0
  RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Regardless of whether you use an ephemeral or static IP address, the external address is unknown to the OS of the VM. The external IP address is mapped to the VM's internal address transparently by VPC. I am illustrating this here by running `ifconfig` within a VM in GCP, which only returns the internal IP address.

Let's explore this further by looking at DNS resolution for both internal and external addresses.

DNS resolution for internal addresses



Each instance has a hostname that can be resolved to an internal IP address:

- The hostname is the same as the instance name.
- FQDN is [hostname].c.[project-id].internal.

Example: guestbook-test.c.guestbook-151617.internal

Name resolution is handled by internal DNS resolver:

- Provided as part of Compute Engine (169.254.169.254).
- Configured for use on instance via DHCP.
- Provides answer for internal and external addresses.

Let's start with internal addresses.

Each instance has a hostname that can be resolved to an internal IP address. This hostname is the same as the instance name. There is also an internal fully qualified domain name or FQDN for an instance that uses the format shown on the slide.

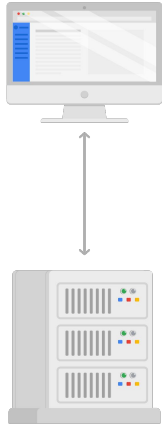
If you delete and recreate an instance, the internal IP address can change. This change can disrupt connections from other Compute Engine resources, which must obtain the new IP address before they can connect again. However, the DNS name always points to a specific instance no matter what the internal IP address is.

Each instance has a metadata server that also acts as a DNS resolver for that instance. The metadata server handles all DNS queries for local network resources, and routes all other queries to Google's public DNS servers for public name resolution. I previously mentioned that an instance is not aware of any external IP address assigned to it. Instead, the network stores a lookup table that matches external IP addresses with the internal IP addresses of the relevant instances.

For more information, including how to set up your own resolver on instances, please refer to link below this video.

[\[https://cloud.google.com/compute/docs/vpc/internal-dns\]](https://cloud.google.com/compute/docs/vpc/internal-dns)

DNS resolution for external addresses



- Instances with external IP addresses can allow connections from hosts outside of the project.
 - Users connect directly using external IP address.
 - Admins can also publish public DNS records pointing to the instance.
 - Public DNS records are not published automatically.
- DNS records for external addresses can be published using existing DNS servers (outside of GCP).
- DNS zones can be hosted using Cloud DNS.

Now, let's look at external addresses.

Instances with external IP addresses can allow connections from hosts outside of the project. Users can do so directly using the external IP address. Public DNS records pointing to instances are not published automatically; however, admins can publish these using existing DNS servers.

Domain name servers can be hosted on GCP, using Cloud DNS. This is a managed service that is definitely worth considering, so let's explore it in more detail.

Host DNS zones using Cloud DNS



- Google's DNS service
- Translate domain names into IP address
- Low latency
- High availability (100% uptime SLA)
- Create and update millions of DNS records
- UI, command line, or API



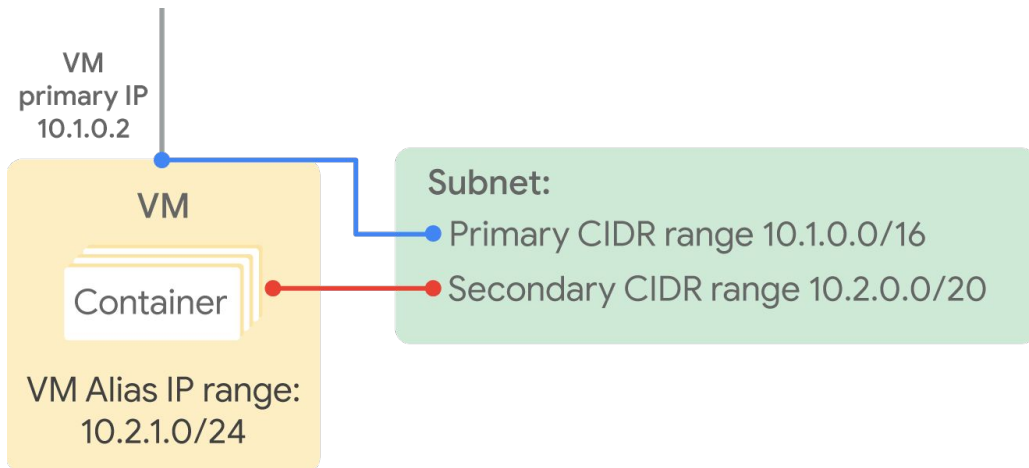
Cloud DNS is a scalable, reliable, and managed authoritative Domain Name System or DNS service running on the same infrastructure as Google. Cloud DNS translates requests for domain names like google.com into IP addresses.

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 GCP offers a 100% uptime Service Level Agreement or SLA for domains configured in Cloud DNS. For more information on this SLA, please refer to link below this video.

[\[https://cloud.google.com/dns/sla\]](https://cloud.google.com/dns/sla)

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 on Cloud DNS, please refer to link below this video. [\[https://cloud.google.com/dns/docs/\]](https://cloud.google.com/dns/docs/)

Assign a range of IP addresses as aliases to a VM's network interface using alias IP ranges



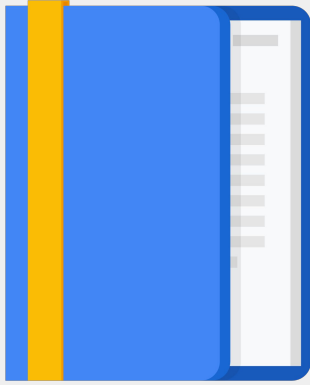
Another networking feature of GCP is Alias IP Ranges.

Alias IP Ranges lets you assign a range of internal IP addresses as an alias to a virtual machine's network interface. This is useful if you have multiple services running on a VM and you want to assign a different IP address to each service. In essence, you can configure multiple IP addresses, representing containers or applications hosted in a VM, without having to define a separate network interface. You just draw the alias IP range from the local subnet's primary or secondary CIDR ranges. This diagram provides a basic illustration of primary and secondary CIDR ranges and VM alias IP ranges.

For more information on Alias IP Ranges, please refer to link below this video.

[\[https://cloud.google.com/compute/docs/alias-ip/\]](https://cloud.google.com/compute/docs/alias-ip/)

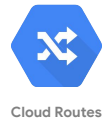
Agenda



- Virtual Private Cloud (VPC)
- Projects, networks, and subnetworks
- IP addresses
- **Routes and firewall rules**
- Lab
- Multiple network interfaces
- Lab

So far you've learned about projects, networks, subnetworks, and IP addresses. Let's use what you learned to understand how GCP routes traffic and establishes firewall rules.

A route is a mapping of an IP range to a destination



Every network has:

- Routes that let instances in a network send traffic directly to each other.
- A default route that directs packets to destinations that are outside the network.

Firewall rules must also allow the packet

By default, every network has routes that let instances in a network send traffic directly to each other, even across subnets. In addition, every network has a default route that directs packets to destinations that are outside the network.

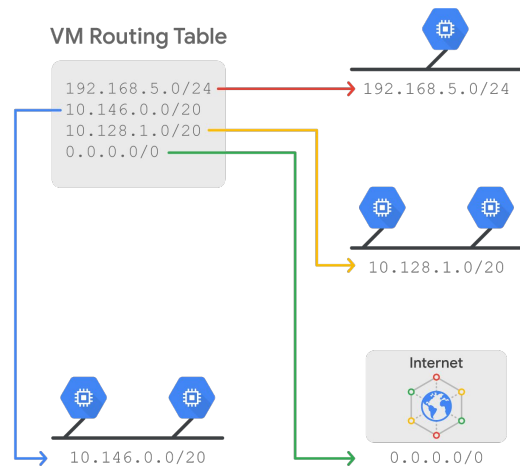
Although these routes cover most of your normal routing needs, you can also create special routes that override these routes.

Just creating a route does not ensure that your packets will be received by the specified next hop. Firewall rules must also allow the packet.

The default network has pre-configured firewall rules that allow all instances in the network to talk with each other. Manually created networks do not have such rules, so you must create them, as you will experience in the first lab.

Routes map traffic to destination networks

- Destination in CIDR notation
- Applies to traffic egressing a VM
- Forwards traffic to most specific route
- Traffic is delivered only if it also matches a firewall rule
- Created when a subnet is created
- Enables VMs on same network to communicate

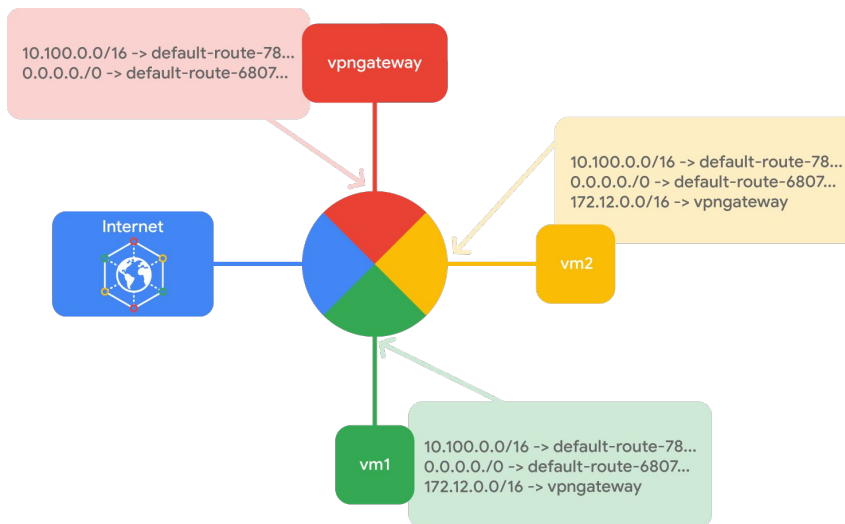


Routes match packets by destination IP address. However, no traffic will flow without also matching a firewall rule.

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. 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.

This diagram shows a massively scalable virtual router at the core of each network. Every virtual machine instance in the network is directly connected to this router, and 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.

Firewall rules protect your VM instances from unapproved connections



- VPC network functions as a distributed firewall.
- Firewall rules are applied to the network as a whole.
- Connections are *allowed* or *denied* at the instance level.
- Firewall rules are stateful.
- Implied *deny all* ingress and *allow all* egress.

GCP 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.

GCP 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 is composed of different parameters

Parameter	Details
direction	Inbound connections are matched against <code>ingress</code> rules only
	Outbound connections are matched against <code>egress</code> rules only
source or destination	For the <code>ingress</code> direction, <code>sources</code> can be specified as part of the rule with IP addresses, source tags, or a source service account
	For the <code>egress</code> direction, <code>destinations</code> can be specified as part of the rule with one or more ranges of IP addresses
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 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 assignment	All rules are assigned to all instances, but you can assign certain rules to certain instances only

You can express your desired firewall configuration as a set of firewall rules.

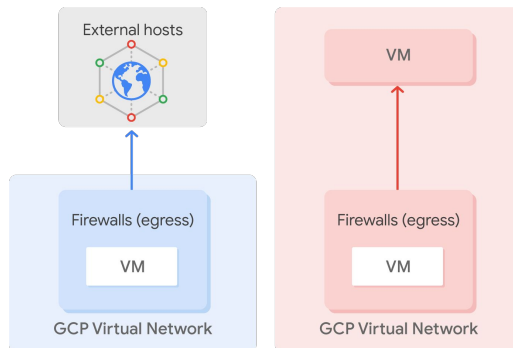
Conceptually, a firewall rule is composed of the following parameters:

- 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, as we will explore in more depth in the next module.
- Let's look at some Google Cloud firewall use cases for both egress and ingress, as well as hierarchical firewall policies.

For more information, please refer to link below this video.

[\[https://cloud.google.com/compute/docs/vpc/firewalls#firewall_rule_components\]](https://cloud.google.com/compute/docs/vpc/firewalls#firewall_rule_components)

GCP firewall use case: Egress



Conditions:

- Destination CIDR ranges
- Protocols
- Ports

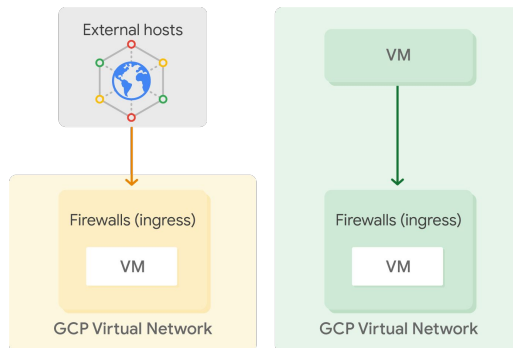
Action:

- Allow: permit the matching egress connection
- Deny: block the matching egress connection

Egress firewall rules control outgoing connections originated inside your GCP network. Egress **allow** rules allow outbound connections that match specific protocol, ports, and IP addresses. Egress **deny** rules prevent instances from initiating connections that match non-permitted port, protocol, and IP range combinations.

For egress firewall rules, destinations to which a rule applies may be specified using IP CIDR ranges. Specifically, you can use destination ranges to protect from undesired connections initiated by a VM instance towards an external host, as shown on the left. You can also use destination ranges to prevent undesired connections from an internal VM instance to a specific GCP CIDR range. This is illustrated in the middle, where a VM in a specific subnet is shown attempting to connect inappropriately to another VM within the same network.

GCP firewall use case: Ingress



Conditions:

- Source CIDR ranges
- Protocols
- Ports

Action:

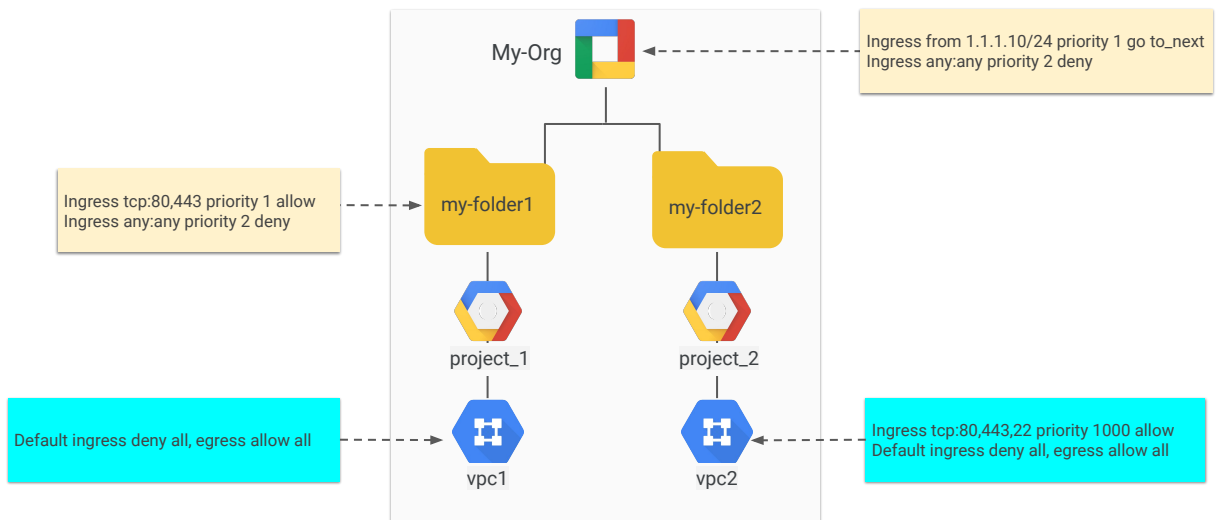
- Allow: permit the matching ingress connection
- Deny: block the matching ingress connection

Ingress firewall rules protect against incoming connections to the instance from any source. Ingress **allow** rules allow specific protocol, ports, and IP addresses to connect in. The firewall prevents instances from receiving connections on non-permitted ports or protocols. Rules can be restricted to only affect particular sources.

Source CIDR ranges can be used to protect an instance from undesired connections coming either from external networks or from GCP IP ranges.

This diagram illustrates a VM receiving a connection from an external address, and another VM receiving a connection from a VM in the same network. You can control ingress connections from a VM instance by constructing inbound connection conditions using source CIDR ranges, protocols, or ports.

Hierarchical firewall policies



Hierarchical firewall policies let you create and enforce a consistent firewall policy across your organization. You can assign hierarchical firewall policies to the organization as a whole or to individual folders. These policies contain rules that can explicitly deny or allow connections, as do Virtual Private Cloud (VPC) firewall rules. In addition, hierarchical firewall policy rules can delegate evaluation to lower-level policies or VPC network firewall rules with a `goto_next` action. Lower-level rules cannot override a rule from a higher place in the resource hierarchy. This lets organization-wide admins manage critical firewall rules in one place.

By default, all hierarchical firewall policy rules apply to all VMs in all projects under the organization or folder where the policy is associated. However, you can restrict which VMs get a given rule by specifying a target network or target service account. The levels of the hierarchy at which firewall rules can now be applied are represented in the diagram, shown here. The yellow boxes near the top represent hierarchical firewall policies, while the blue boxes at the bottom represent VPC firewall rules.

Lab

Getting Started with VPC Networking

Philipp Maier

Let's apply some of the network features we just discussed in a lab.

In this lab, you create an auto mode VPC network with firewall rules and two VM instances. Then, you explore the connectivity for the VM instances.

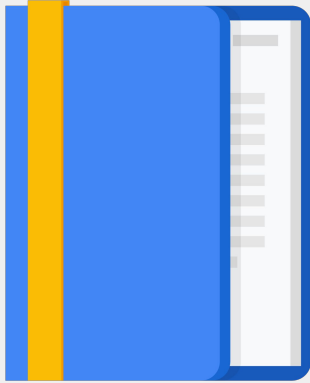
Lab Review

Getting Started with VPC Networking

Philipp Maier

In this lab, you explored the default network along with its subnets, routes, and firewall rules. You deleted the default network and determined that you cannot create any VM instances without a VPC network. So, you created a new auto mode VPC network with subnets, routes, firewall rules, and two VM instances. Then, you tested the connectivity for the VM instances and explored the effects of the firewall rules on connectivity.

Agenda



- Virtual Private Cloud (VPC)
- Projects, networks, and subnetworks
- IP addresses
- Routes and firewall rules
- Lab
- **Multiple network interfaces**
- Lab

Next, let's look at Multiple Network Interfaces.

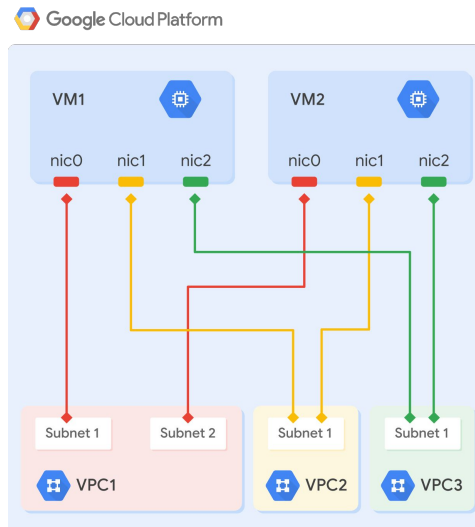
Multiple network interfaces

VPC networks are isolated (by default)

- Communicate within networks using **internal IP**
- Communicate across networks using **external IP**

Multiple Network Interfaces

- Network interface controllers (NICs)
- Each NIC is attached to a VPC network
- Communicate across networks using **internal IP**



VPC networks are by default isolated private networking domains. As I mentioned earlier, VM instances within a VPC network can communicate among themselves via internal IP addresses as long as firewall rules permit. However, no internal IP address communication is allowed between networks unless you set up mechanisms such as VPC peering or VPN.

Every instance in a VPC network has a default network interface. You can create additional network interfaces attached to your VMs through network interface controllers (NICs). Multiple network interfaces enable you to create configurations in which an instance connects directly to several VPC networks. Each of the interfaces must have an internal IP address, and each interface can also have an external IP address.

For example, in this diagram you have two VM instances. Each instance has network interfaces to a subnet within VPC1, VPC2, and VPC3.

Typically, you might require multiple interfaces if you want to configure an instance as a network appliance that does load balancing, Intrusion Detection and Prevention, Web Application Firewall, or WAN optimization between networks. Multiple network interfaces are also useful when applications running in an instance require traffic separation, such as separation of data plane traffic from management plane traffic.

Multiple network interfaces limitations

- Configure when you create instance
- Each interface in different network
- Networks' IP range cannot overlap
- Networks must exist to create VM
- Cannot delete interface without deleting VM
- Internal DNS only associated to nic0
- Up to 8 NICs, depends on VM

Type of instance	# of virtual NICs
VM <= 2 vCPU	2 NICs
VM >2vCPU	1 NIC per vCPU (Max: 8)

Now, there are some limitations that I want you to remember when creating VM instances with multiple network interfaces:

- First, you can only configure a network interface when you create an instance.
- Each network interface configured in a single instance must be attached to a different VPC network, and each interface must belong to a subnet whose IP range does not overlap with the subnets of any other interfaces.
- The additional VPC networks that the multiple interfaces will attach to must exist before you create the instance.
- You cannot delete a network interface without deleting the instance.
- When an internal DNS query is made with the instance hostname, it resolves to the primary interface (nic0) of the instance. If the nic0 interface of the instance belongs to a VPC network different from the VPC network of the instance issuing the internal DNS query, the query will fail. You will explore this in the upcoming lab.

The maximum number of network interfaces per instance is 8, but this depends on the instance's machine type, as shown in this table:

- Instances with less than or equal to 2 vCPU can have up to 2 virtual NICs. This includes the f1-micro, g1-small, n1-standard-1, and any other custom VMs with 1 or 2 vCPUs.
- Instances with more than 2 vCPU can have 1 NIC per vCPU, with a maximum

- of 8 virtual NICs.

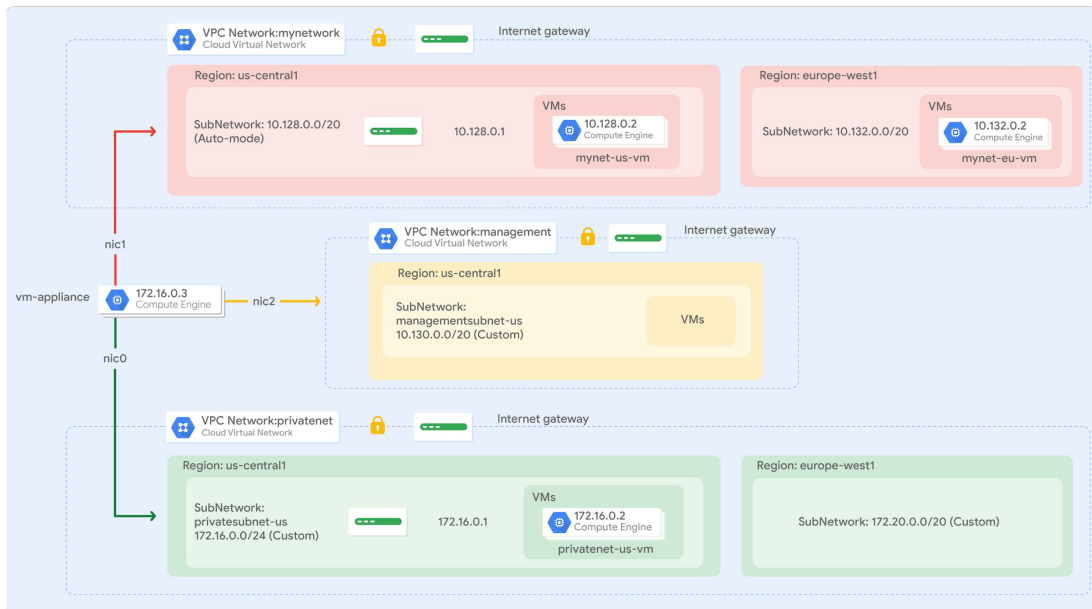
Lab

Working with Multiple VPC Networks

Philipp Maier

Let's apply the multiple network interface concepts that we just discussed.

In this lab, you create several VPC networks and VM instances and test connectivity across networks.



Specifically, you create two custom mode networks, managementnet and privatenet, with firewall rules and VM instances as shown in this network diagram.

The mynetwork network with its firewall rules and two VM instances, mynet-eu-vm and mynet-us-vm, have already been created for you.

Lab Review

Working with Multiple VPC Networks

Philipp Maier

In this lab, you created several custom mode VPC networks, firewall rules, and VM instances using the GCP Console and the `gcloud` command line. Then, you tested the connectivity across VPC networks, which worked when pinging external IP addresses but not when pinging internal IP addresses. Thus, you created a VM instance with three network interfaces and verified internal connectivity for VM instances that are on the subnets that are attached to the multiple interface VM.

Review

Google Cloud VPC Networking Fundamentals

Philipp Maier

In this module, I gave you an overview of Google's Virtual Private Cloud. We looked at the different objects within VPC, like projects, networks, IP addresses, routes, and firewall rules. Then, you explored these objects in a short lab.

Next, we looked at VM instances with multiple network interfaces, which you implemented in a more thorough lab.

Now that we have covered some of the fundamentals of VPC networks, let's move on to Controlling Access to VPC networks.