**Networking 101**

experimentLabschedule1 hour 15 minutesuniversal\_currency\_alt1 Creditshow\_chartIntroductory

Rate Lab

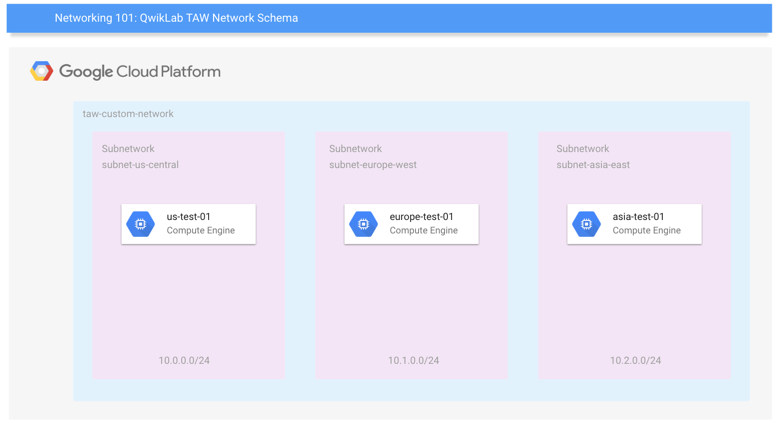
infoThis lab may incorporate AI tools to support your learning.

**GSP016**



**Overview**

In this hands-on lab you will learn how to perform basic networking tasks on Google Cloud (including Compute Engine instances) and how Google Cloud might differ from an on-premises setup. You'll develop a network and 3 subnetworks, resulting in this end-state environment:



The lab exercises you'll do are ordered to reflect a common cloud developer experience:

1. Set up your lab environment and learn how to work with your Google Cloud environment.
2. Deploy a common network with subnets and multiple regions using common open source tools to explore your network around the world.
3. Testing and monitoring your network and instances.

What you'll learn

* Basics concepts and constructs of Google Cloud networking.
* How default and user-created Networks are configured.
* How to measure performance & latency characteristics.
* Basic security configurations of Access, Firewall, and Routing.

Prerequisites

* Basic knowledge of Compute Engine
* Basic networking and TCP/IP knowledge
* Basic Unix/Linux command line knowledge

**Setup and requirements**

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

To complete this lab, you need:

* Access to a standard internet browser (Chrome browser recommended).

**Note:** Use an Incognito or private browser window to run this lab. This prevents any conflicts between your personal account and the Student account, which may cause extra charges incurred to your personal account.

* Time to complete the lab---remember, once you start, you cannot pause a lab.

**Note:** If you already have your own personal Google Cloud account or project, do not use it for this lab to avoid extra charges to your account.

How to start your lab and sign in to the Google Cloud console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:
   * The **Open Google Cloud console** button
   * Time remaining
   * The temporary credentials that you must use for this lab
   * Other information, if needed, to step through this lab
2. Click **Open Google Cloud console** (or right-click and select **Open Link in Incognito Window** if you are running the Chrome browser).

The lab spins up resources, and then opens another tab that shows the **Sign in** page.

***Tip:*** Arrange the tabs in separate windows, side-by-side.

**Note:**If you see the **Choose an account** dialog, click **Use Another Account**.

1. If necessary, copy the **Username** below and paste it into the **Sign in** dialog.

student-00-828c9dd53659@qwiklabs.net

Copied!

content\_copy

You can also find the **Username** in the **Lab Details** panel.

1. Click **Next**.
2. Copy the **Password** below and paste it into the **Welcome** dialog.

jPFXkFwpyCBd

Copied!

content\_copy

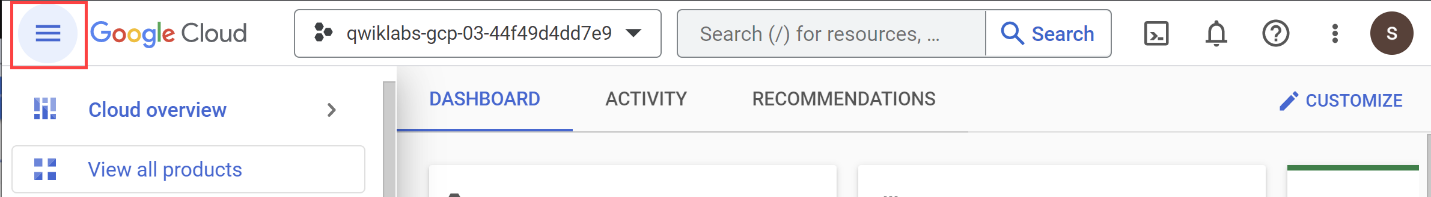
You can also find the **Password** in the **Lab Details** panel.

1. Click **Next**.

**Important:**You must use the credentials the lab provides you. Do not use your Google Cloud account credentials.**Note:**Using your own Google Cloud account for this lab may incur extra charges.

1. Click through the subsequent pages:
   * Accept the terms and conditions.
   * Do not add recovery options or two-factor authentication (because this is a temporary account).
   * Do not sign up for free trials.

After a few moments, the Google Cloud console opens in this tab.

**Note:** To view a menu with a list of Google Cloud products and services, click the **Navigation menu** at the top-left. 

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell** Activate Cloud Shell icon at the top of the Google Cloud console.

When you are connected, you are already authenticated, and the project is set to your **Project\_ID**, qwiklabs-gcp-02-4fd2c777175b. The output contains a line that declares the **Project\_ID** for this session:

Your Cloud Platform project in this session is set to qwiklabs-gcp-02-4fd2c777175b

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

1. (Optional) You can list the active account name with this command:

gcloud auth list

Copied!

content\_copy

1. Click **Authorize**.

**Output:**

ACTIVE: \*

ACCOUNT: student-00-828c9dd53659@qwiklabs.net

To set the active account, run:

$ gcloud config set account `ACCOUNT`

1. (Optional) You can list the project ID with this command:

gcloud config list project

Copied!

content\_copy

**Output:**

[core]

project = qwiklabs-gcp-02-4fd2c777175b

**Note:**For full documentation of gcloud, in Google Cloud, refer to [the gcloud CLI overview guide](https://cloud.google.com/sdk/gcloud).

Understanding Regions and Zones

Certain Compute Engine resources live in regions or zones. A region is a specific geographical location where you can run your resources. Each region has one or more zones. For example, the us-central1 region denotes a region in the Central United States that has zones us-central1-a, us-central1-b, us-central1-c, and us-central1-f.

|  |  |
| --- | --- |
| **Regions** | **Zones** |
| Western US | us-west1-a, us-west1-b |
| Central US | us-central1-a, us-central1-b, us-central1-d, us-central1-f |
| Eastern US | us-east1-b, us-east1-c, us-east1-d |
| Western Europe | europe-west1-b, europe-west1-c, europe-west1-d |
| Eastern Asia | asia-east1-a, asia-east1-b, asia-east1-c |

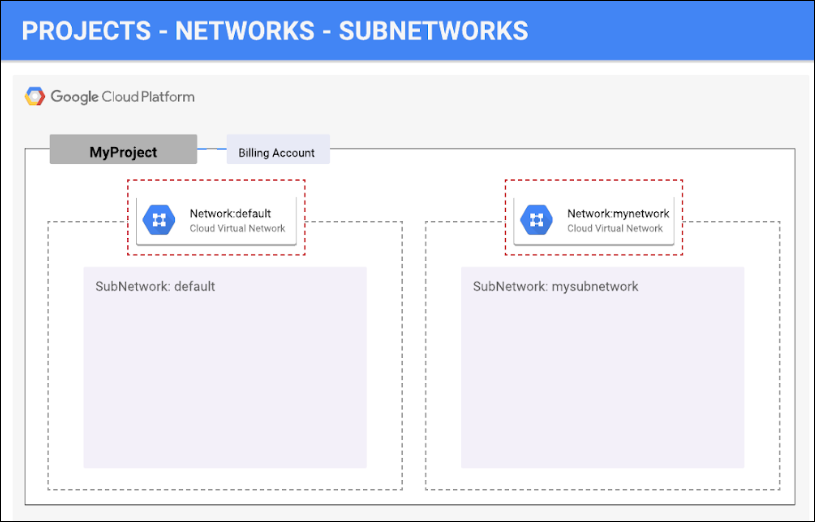
Resources that live in a zone are referred to as zonal resources. Virtual machine Instances and persistent disks live in a zone. To attach a persistent disk to a virtual machine instance, both resources must be in the same zone. Similarly, if you want to assign a static IP address to an instance, the instance must be in the same region as the static IP.

Learn more about regions and zones and see a complete list in the Compute Engine page, [Regions and zones documentation](https://cloud.google.com/compute/docs/regions-zones/)).

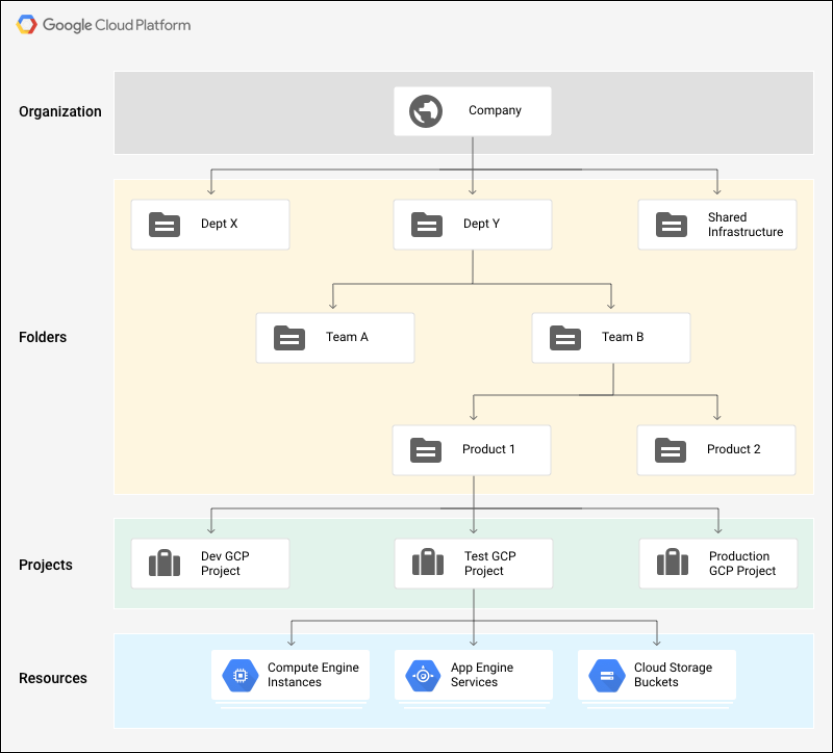
**Google Cloud Network Concepts**

In Google Cloud Platform, networks provide data connections into and out of your cloud resources (mostly Compute Engine instances). Securing your Networks is critical to securing your data and controlling access to your resources.

Google Cloud Platform supports Projects, Networks, and Subnetworks to provide flexible, logical isolation of unrelated resources.



***Projects*** are the outermost container and are used to group resources that share the same trust boundary. Many developers map Projects to teams since each Project has its own access policy (IAM) and member list. Projects also serve as a collector of billing and quota details reflecting resource consumption. Projects contain Networks which contain Subnetworks, Firewall rules, and Routes (see below architecture diagrams for illustration).



***Networks*** directly connect your resources to each other and to the outside world. Networks, using Firewalls, also house the access policies for incoming and outgoing connections. Networks can be Global (offering horizontal scalability across multiple Regions) or Regional (offering low-latency within a single Region).

***Subnetworks*** allow you to group related resources (Compute Engine instances) into RFC1918 private address spaces. Subnetworks can only be Regional. A subnetwork can be in auto mode or custom mode.

* An auto mode network has one subnet per region, each with a predetermined IP range and gateway. These subnets are created automatically when you create the auto mode network, and each subnet has the same name as the overall network.
* A custom mode network has no subnets at creation. In order to create an instance in a custom mode network, you must first create a subnetwork in that region and specify its IP range. A custom mode network can have zero, one, or many subnets per region.

Set your region and zone

Certain Compute Engine resources live in regions and zones. A region is a specific geographical location where you can run your resources. Each region has one or more zones.

Learn more about regions and zones and see a complete list in [Regions & Zones documentation](https://cloud.google.com/compute/docs/regions-zones/).

Run the following gcloud commands in Cloud Shell to set the default region and zone for your lab:

gcloud config set compute/zone "us-west1-c"

export ZONE=$(gcloud config get compute/zone)

gcloud config set compute/region "us-west1"

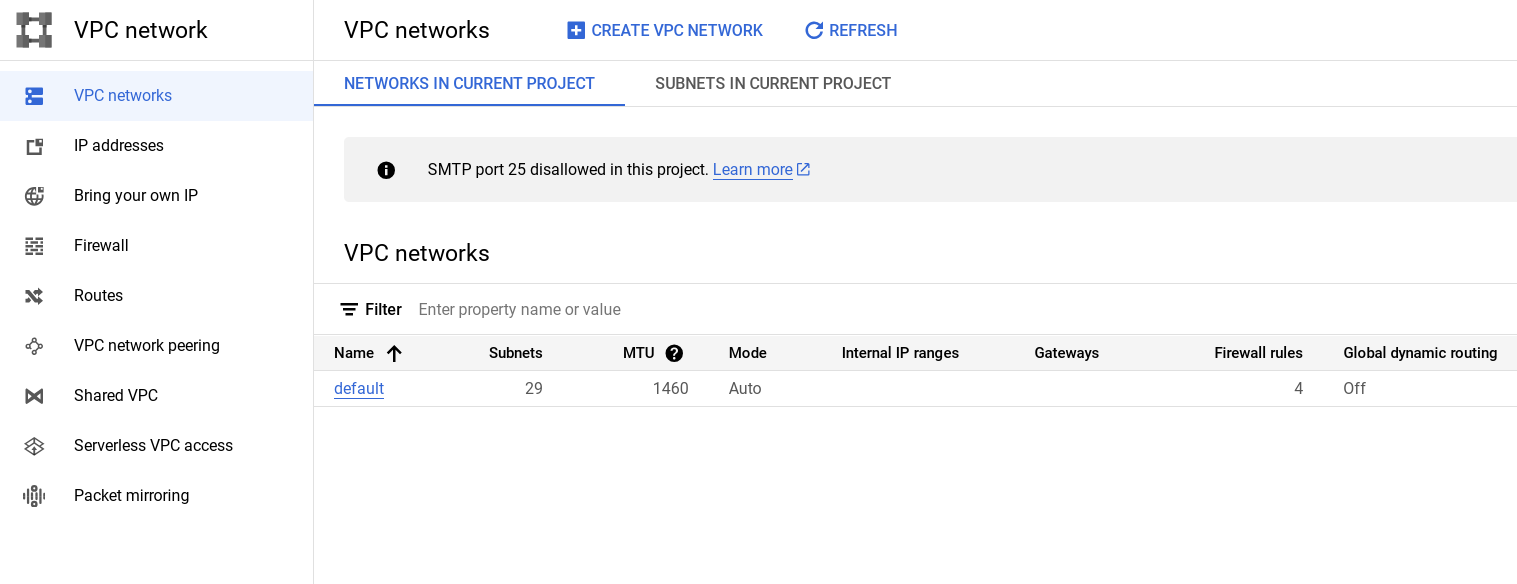
export REGION=$(gcloud config get compute/region)

**Task 1. Review the default network**

When a new project is created, a default network configuration provides each region with an auto subnet network. You can create up to four additional networks in a project. Additional networks can be auto subnet networks, custom subnet networks, or legacy networks.

Each instance created within a subnetwork is assigned an IPv4 address from that subnetwork range.

* Review your network. Click **Navigation menu** > **VPC network**.



Firewalls

for more information on how you can use firewall rules to isolate subnetworks, refer to [subnetworks](https://cloud.google.com/vpc/docs/vpc#vpc_networks_and_subnets) and [firewall rules](https://cloud.google.com/compute/docs/vpc/firewalls).

Each network has a default firewall that blocks all inbound traffic to instances. To allow traffic to come into an instance, you must create "allow" rules for the firewall. Additionally, the default firewall allows traffic from instances unless you configure it to block outbound connections using an "egress" firewall configuration. Therefore, by default you can create "allow" rules for traffic you wish to pass ingress, and "deny" rules for traffic you wish to restrict egress. You may also create a default-deny policy for egress and prohibit external connections entirely.

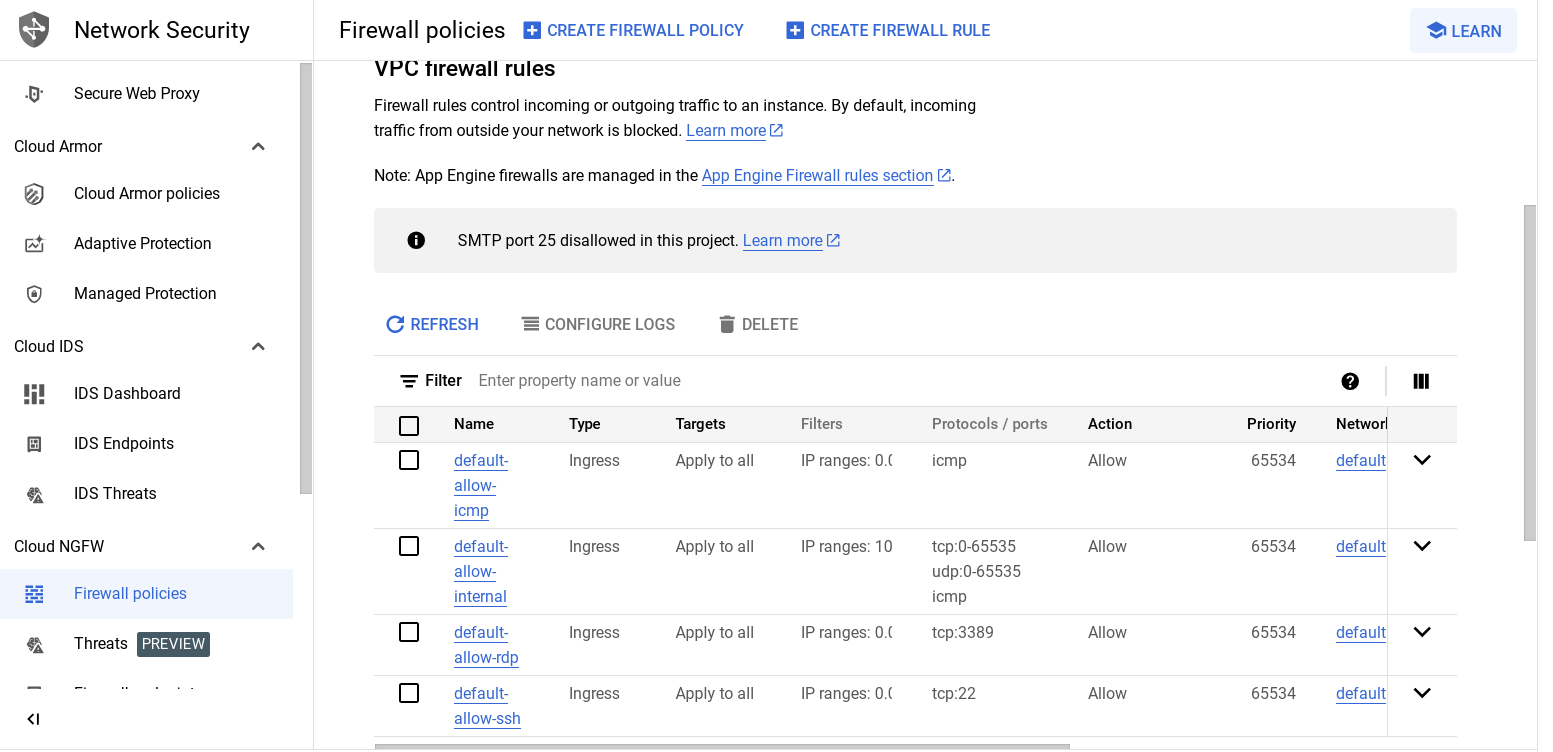
In general, it is recommended to configure the least permissive firewall rule that will support the kind of traffic you are trying to pass. For example, if you need to allow traffic to reach some instances, but restrict traffic from reaching others, create rules that allow traffic to the intended instances only. This more restrictive configuration is more predictable than a large firewall rule that allows traffic to all of the instances. If you want to have "deny" rules to override certain "allow" rules, you can set priority levels on each rule and the rule with the lowest numbered priority will be evaluated first. Creating large and complex sets of override rules can lead to allowing or blocking traffic that is not intended.

The default network has automatically created firewall rules, which are shown below. No manually created network of any type has automatically created firewall rules. For all networks except the default network, you must create any firewall rules you need.

The ingress firewall rules automatically created for the default network are as follows:

|  |  |
| --- | --- |
| default-allow-internal | Allows network connections of any protocol and port between instances on the network. |
| default-allow-ssh | Allows SSH connections from any source to any instance on the network over TCP port 22. |
| default-allow-rdp | Allows RDP connections from any source to any instance on the network over TCP port 3389. |
| default-allow-icmp | Allows ICMP traffic from any source to any instance on the network. |

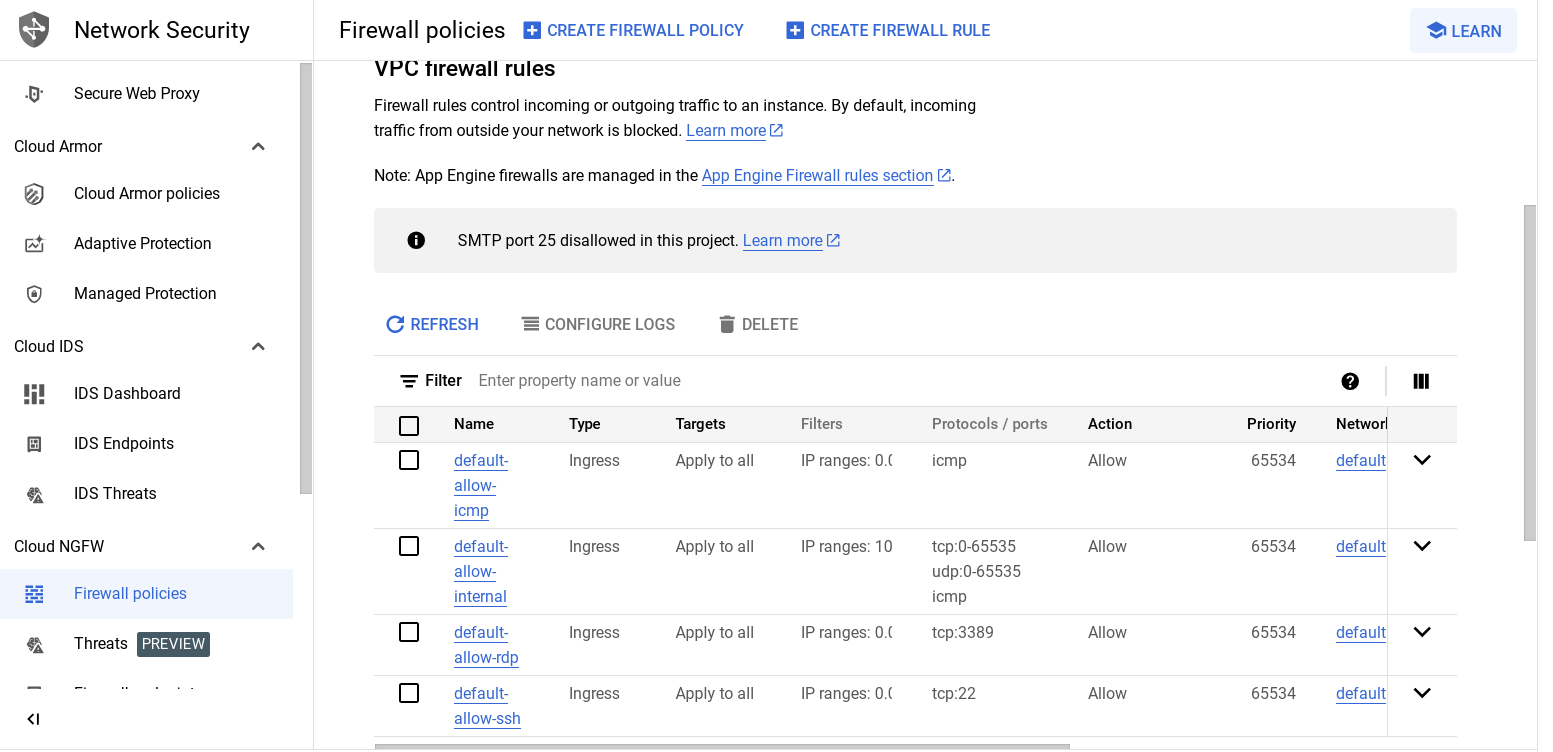
* To review the default Firewall rules, in the Cloud console click **Navigation menu** > **VPC network** > **Firewall**.



Network route

All networks have routes created automatically to the Internet (default route) and to the IP ranges in the network. The route names are automatically generated and will look different for each project.

* To review default routes, click **Navigation menu** > **VPC network** > **Routes** > Select **Network** and **Region** to view **Routes**.



**Task 2. Creating a custom network**

Creating a new network with custom subnet ranges

When manually assigning subnetwork ranges, you first create a custom subnet network, then create the subnetworks that you want within a region. You do not have to specify subnetworks for all regions right away, or even at all, but you cannot create instances in regions that have no subnetwork defined.

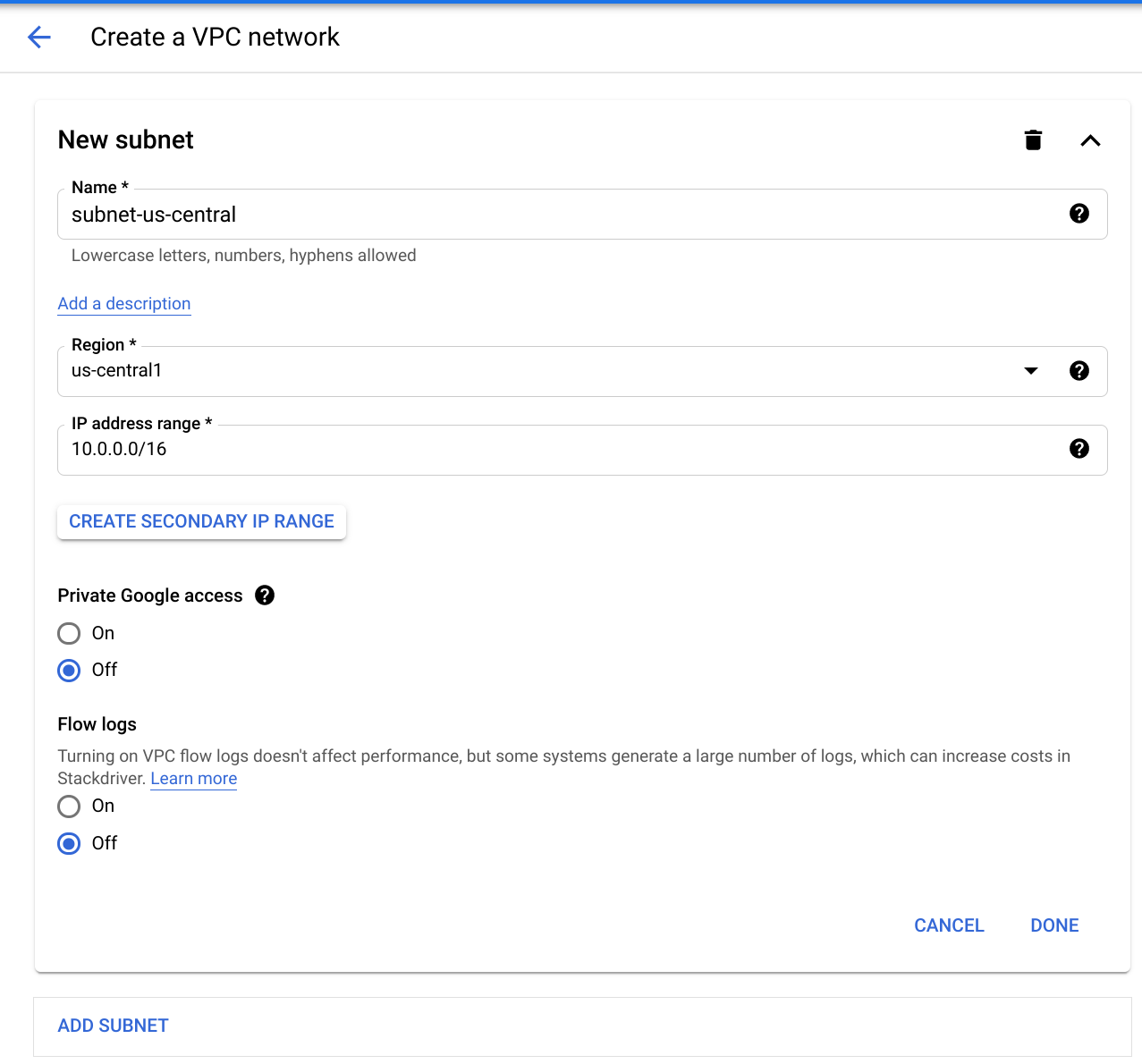
When you create a new subnetwork, its name must be unique in that project for that region, even across networks. The same name can appear twice in a project as long as each one is in a different region. Because this is a subnetwork, there is no network-level IPv4 range or gateway IP, so none will be displayed.

You can either create your custom network with the console or with Cloud Shell. You'll be shown you both, but you have to decide which method to use while taking the lab. For example, you cannot go through a section using the instructions for the console, then go through the same section using gcloud command line.

**Task 3. Create custom network with the console**

**Note:**If you prefer to use the command line, skip this section and continue with **Create custom network with Cloud Shell**.

1. To create a custom network, click **Navigation menu** > **VPC network**.
2. Click **Create VPC Network** and name it taw-custom-network.
3. On the **Custom** tab create:
   * Subnet name: subnet-us-west1
   * Region: us-west1
   * IP address range: 10.0.0.0/16
4. Click **Done**.



1. Now click **Add Subnet** and add 2 more subnets in their respective regions:
   * subnet-us-central1, us-central1, 10.1.0.0/16
   * subnet-us-east4, us-east4, 10.2.0.0/16
2. Click **Create** to finish.

At this point, the network has routes to the Internet and to any instances that you might create. But it has no firewall rules allowing access to instances, even from other instances. To allow access, you must create [firewall rules](https://cloud.google.com/compute/docs/vpc/firewalls).

Continue to the **Adding firewall rules** section.

**Task 4. Create custom network with Cloud Shell**

**Note:** If you just created a network using the Console, do not repeat this exercise using Cloud Shell. Re-take the lab instead to practice using the gcloud command line.

* Enter the following command in cloud shell to create the custom network:

gcloud compute networks create taw-custom-network --subnet-mode custom

Copied!

content\_copy

**Output:**

NAME MODE IPV4\_RANGE GATEWAY\_IPV4

taw-custom-network custom

Instances on this network will not be reachable until firewall rules

are created. As an example, you can allow all internal traffic between

instances as well as SSH, RDP, and ICMP by running:

$ gcloud compute firewall-rules create --network taw-custom-network --allow tcp,udp,icmp --source-ranges

$ gcloud compute firewall-rules create --network taw-custom-network --allow tcp:22,tcp:3389,icmp

Now create subnets for your new custom network. You'll create three of them.

1. Create subnet-us-west1 with an IP prefix:

gcloud compute networks subnets create subnet-us-west1 \

--network taw-custom-network \

--region us-west1 \

--range 10.0.0.0/16

Copied!

content\_copy

**Output:**

Created [https://www.googleapis.com/compute/v1/projects/cloud-network-module-101/regions/us-west1/subnetworks/subnet-us-west1].

NAME REGION NETWORK RANGE

subnet-us-west1 us-west1 taw-custom-network 10.0.0.0/16

1. Create subnet-us-central1 with an IP prefix:

gcloud compute networks subnets create subnet-us-central1 \

--network taw-custom-network \

--region us-central1 \

--range 10.1.0.0/16

Copied!

content\_copy

**Output:**

Created [https://www.googleapis.com/compute/v1/projects/cloud-network-module-101/regions/us-central1/subnetworks/subnet-us-central1].

NAME REGION NETWORK RANGE

subnet-us-central1 us-central1 taw-custom-network 10.1.0.0/16

1. Create subnet-us-east4 with an IP prefix:

gcloud compute networks subnets create subnet-us-east4 \

--network taw-custom-network \

--region us-east4 \

--range 10.2.0.0/16

Copied!

content\_copy

**Output:**

Created [https://www.googleapis.com/compute/v1/projects/cloud-network-module-101/regions/us-east4/subnetworks/subnet-us-east4].

NAME REGION NETWORK RANGE

subnet-us-central1 us-central1 taw-custom-network 10.2.0.0/16

1. List your networks:

gcloud compute networks subnets list \

--network taw-custom-network

Copied!

content\_copy

If you created an auto subnet network in the prior section, those subnets will be listed as well.

**Output:**

NAME REGION NETWORK RANGE

subnet-us-east4 us-east4 taw-custom-network 10.1.0.0/16

subnet-us-central1 us-central1 taw-custom-network 10.2.0.0/16

subnet-us-west1 us-west1 taw-custom-network 10.0.0.0/16

At this point, the network has routes to the Internet and to any instances that you might create. But, it has no firewall rules allowing access to instances, even from other instances. To allow access, you must create [firewall rules](https://cloud.google.com/vpc/docs/firewalls).

**Task 5. Adding firewall rules**

To allow access to VM instances, you must apply firewall rules. You will use an instance tag to apply the firewall rule to your VM instances. The firewall rule will apply to any VM using the same instance tag.

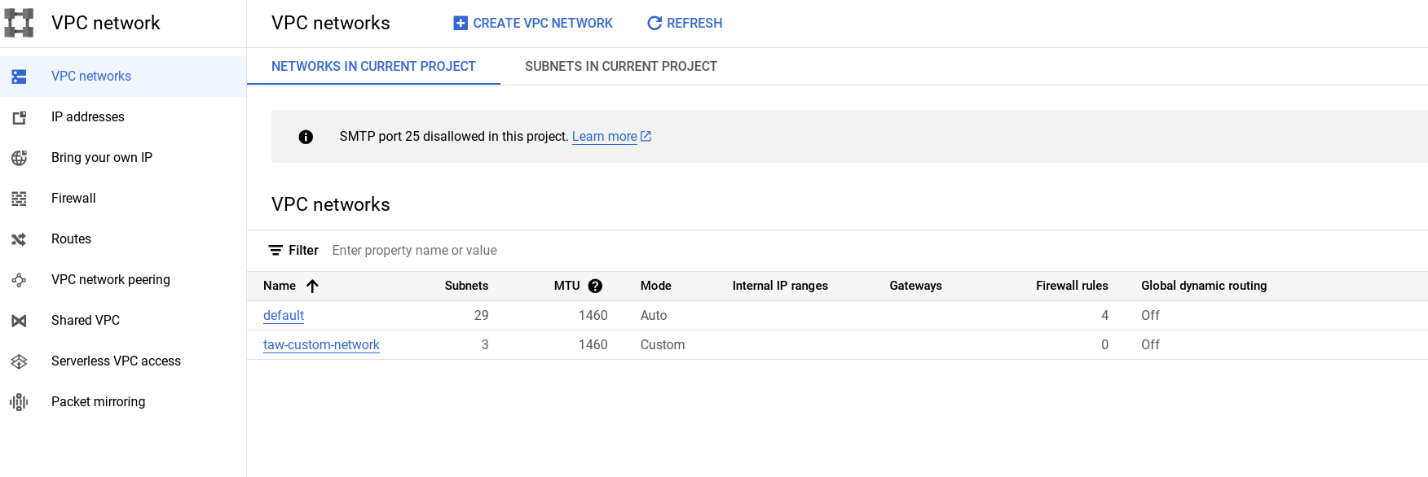
**Note:**Instance Tags are used by networks and firewalls to apply certain firewall rules to tagged VM instances. For example, if there are several instances that perform the same task, such as serving a large website, you can tag these instances with a shared word or term and then use that tag to allow HTTP access to those instances with a firewall rule.

Tags are also reflected in the metadata server, so you can use them for applications running on your instances.

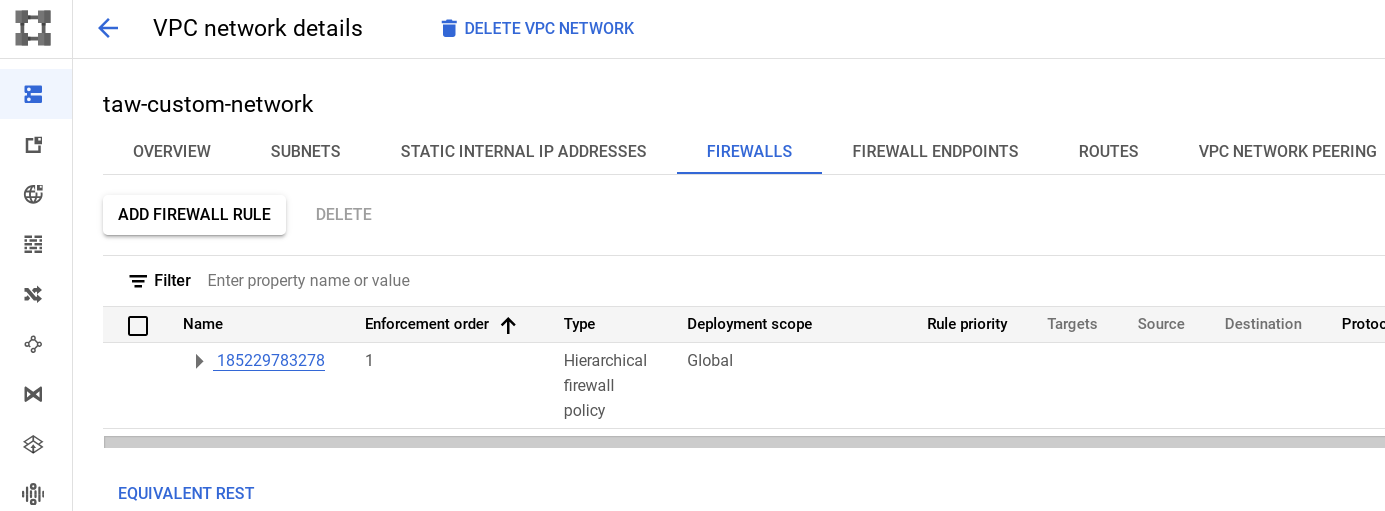
* Start by opening the firewall to allow HTTP Internet requests, then you'll add more firewall rules. Firewall rules can be added using the Console or Cloud Shell.

Add firewall rules through the Console

1. In the Cloud console, navigate to **VPC networks** and click on the **taw-custom-network**:



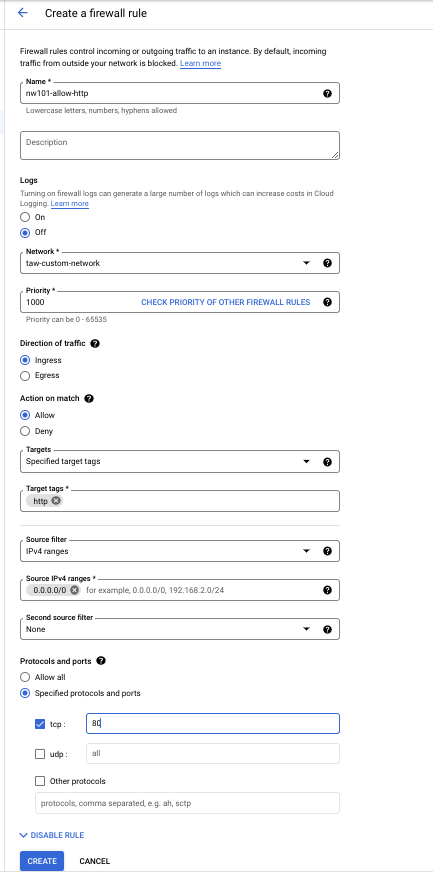
1. Click the **Firewalls** tab, then **Add Firewall rule**.



1. Enter the following info:

|  |  |  |
| --- | --- | --- |
| **Field** | **Value** | **Comments** |
| Name | nw101-allow-http | New rule name |
| Targets | Specified target tags | Which instances to which the firewall rule applies. |
| Target tags | http | The tag we created |
| Source filter | IPv4 ranges | We will open the firewall for any IP address from the Internet. |
| Source IPv4 ranges | 0.0.0.0/0 | You will open the firewall for any IP address from the Internet. |
| Protocols and ports | Select **Specified protocols and ports**, then check the **tcp** box, and type *80* | Only HTTP |

Your screen will look like this:



1. Click **Create** and wait until the command succeeds. Next you'll create the additional firewall rules you'll need.

Add firewall rules using Cloud Shell

**Note:**If you just created a network using the Cloud console, do not do the same exercise using Cloud Shell. Re-take the lab instead to practice using the gcloud command line.

* To create firewall rules in Cloud Shell, run the following:

gcloud compute firewall-rules create nw101-allow-http \

--allow tcp:80 --network taw-custom-network --source-ranges 0.0.0.0/0 \

--target-tags http

Copied!

content\_copy

**Output:**



Create additional firewall rules

These additional firewall rules will allow ICMP, internal communication, SSH, and RDP. You can create them using the Console (or Cloud Shell). Remember to use one method or the other for each type of firewall rule, you can't do both.

* **ICMP**

|  |  |  |
| --- | --- | --- |
| Field | Value | Comments |
| Name | nw101-allow-icmp | New rule name |
| Targets | Specified target tags | Select from the Targets dropdown |
| Target tags | rules | tag |
| Source filter | IPv4 ranges | We will open the firewall for any IP address on this list. |
| Source IPv4 ranges | 0.0.0.0/0 | We will open the firewall for any IP address from the Internet. |
| Protocols and ports | Select **Specified protocols and ports**, **other protocols**, then type *icmp* | The protocols and ports the firewall applies to |

(Cloud Shell commands)

gcloud compute firewall-rules create "nw101-allow-icmp" --allow icmp --network "taw-custom-network" --target-tags rules

Copied!

content\_copy

* **Internal Communication**

|  |  |  |
| --- | --- | --- |
| Field | Value | Comments |
| Name | nw101-allow-internal | New rule name |
| Targets | All instances in the network | Select from the Targets dropdown |
| Source filter | IPv4 ranges | The filter used to apply the rule to specific traffic sources |
| Source IPv4 ranges | 10.0.0.0/16,  10.1.0.0/16,  10.2.0.0/16 | We will open the firewall for any IP address from the Internet. |
| Protocols and ports | Select **Specified protocols and ports**, then **tcp** and type *0-65535*; check **udp** type *0-65535*; check **Other protocols** and type *icmp* | Allows Tcp:0-65535, udp:0-65535,icmp |

(Cloud Shell commands)

gcloud compute firewall-rules create "nw101-allow-internal" --allow tcp:0-65535,udp:0-65535,icmp --network "taw-custom-network" --source-ranges "10.0.0.0/16","10.2.0.0/16","10.1.0.0/16"

Copied!

content\_copy

* **SSH**

|  |  |  |
| --- | --- | --- |
| Field | Value | Comments |
| Name | nw101-allow-ssh | New rule name |
| Targets | Specified target tags | ssh |
| Target tags | ssh | The instances to which you apply the firewall rule |
| Source filter | IPv4 ranges | The filter used to apply the rule to specific traffic sources |
| Source IPv4 ranges | 0.0.0.0/0 | We will open the firewall for any IP address from the Internet. |
| Protocols and ports | Select **Specified protocols and ports**, check the **tcp** box, then type *22* | Allows tcp:22 |

(Cloud Shell commands)

gcloud compute firewall-rules create "nw101-allow-ssh" --allow tcp:22 --network "taw-custom-network" --target-tags "ssh"

Copied!

content\_copy

* **RDP**

|  |  |  |
| --- | --- | --- |
| Field | Value | Comments |
| Name | nw101-allow-rdp | New rule name |
| Targets | All instances in the network | Select from the Targets dropdown |
| Source filter | IPv4 ranges | Filter IP addresses |
| Source IPv4 ranges | 0.0.0.0/0 | We will open the firewall for any IP address from the Internet. |
| Protocols and ports | Select **Specified protocols and ports**, check the **tcp**, then type *3389* | Allows tcp:3389 |

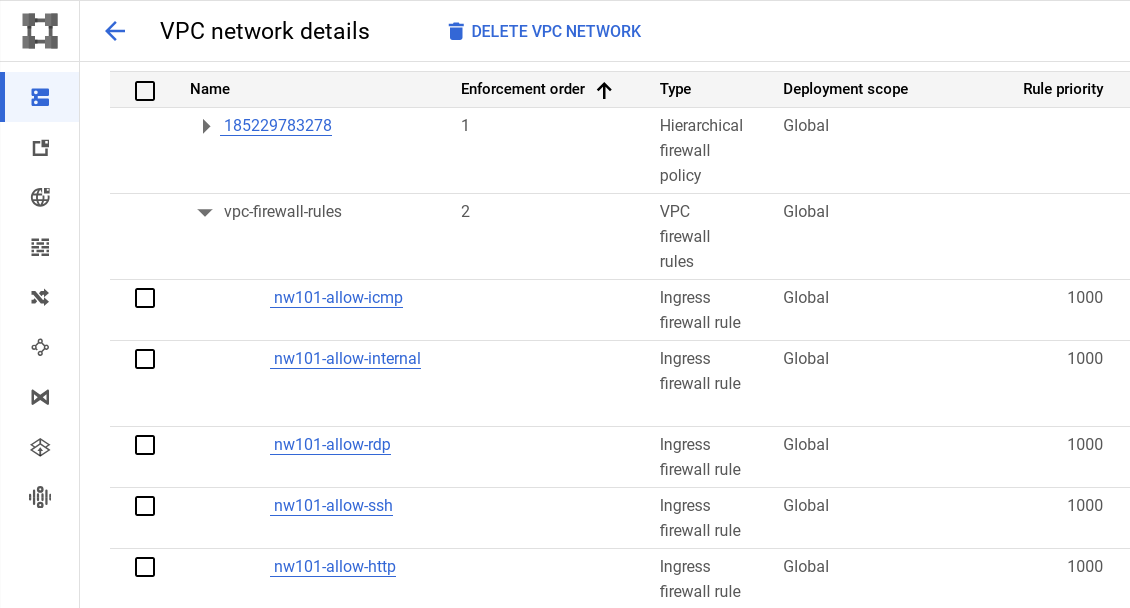
(Cloud Shell commands)

gcloud compute firewall-rules create "nw101-allow-rdp" --allow tcp:3389 --network "taw-custom-network"

Copied!

content\_copy

* Use the Console to review the firewall rules in your network. It should look like this:



**Note:**What about those Routes I see in the Network console?

Google Cloud Networking uses Routes to direct packets between subnetworks and to the Internet. Whenever a subnetwork is created (or pre-created) in your Network, routes are automatically created in each region to allow packets to route between subnetworks. These cannot be modified.

Additional Routes can be created to send traffic to an instance, a VPN gateway, or default internet gateway. These Routes can be modified to tailor the desired network architecture. Routes and Firewalls work together to ensure your traffic gets where it needs to go.

Click **Check my progress** to verify the objective.

Assessment Completed!

Create a custom network, subnetworks and firewall rules.

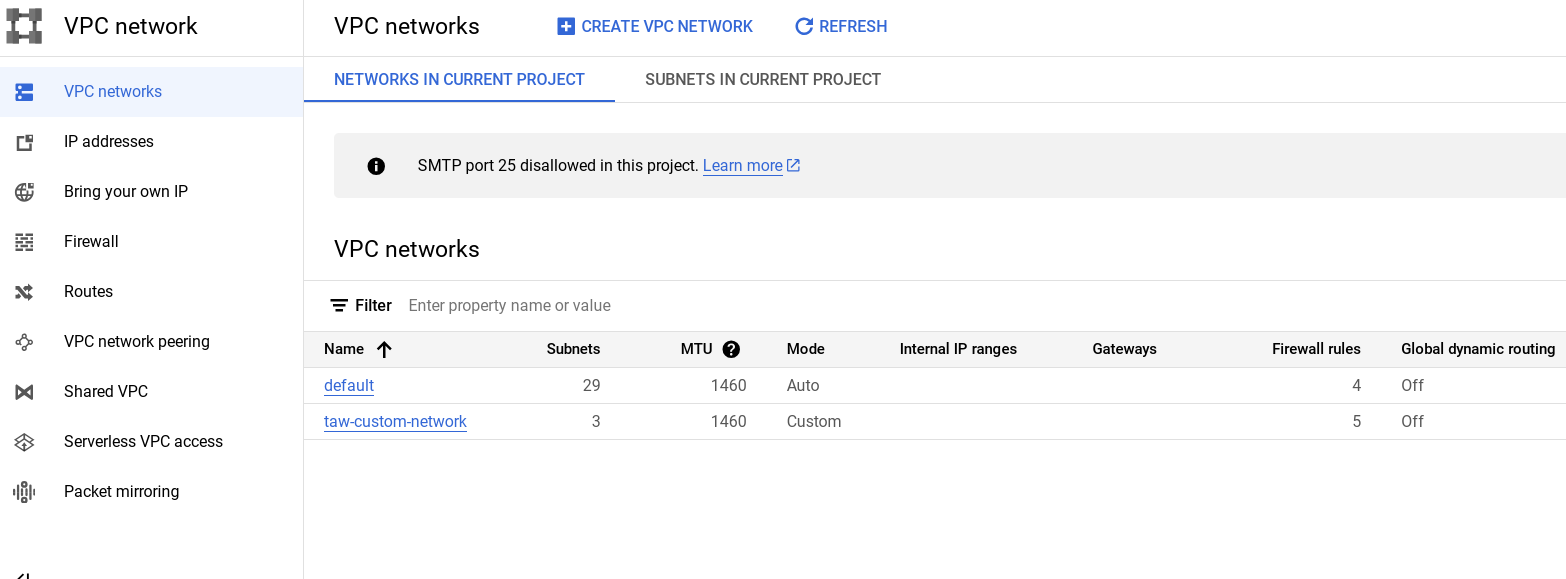
Check my progress

*Assessment Completed!*

**Task 6. Connecting to your lab VMs and checking latency**

* Click on **VPC networks** in the left menu to see your entire network. The taw-custom-network has three subnetworks and the firewalls rules applied.

You should see this:



The next steps are to create a VM in each subnet and make sure you can connect to them.

Creating a VM in each zone

For this section of the lab you will work in Cloud Shell.

1. Run this commands to create an instance named us-test-01 in the subnet-us-west1 subnet:

gcloud compute instances create us-test-01 \

--subnet subnet-us-west1 \

--zone us-west1-c \

--machine-type e2-standard-2 \

--tags ssh,http,rules

Copied!

content\_copy

Be sure to note the external IP for later use in this lab.

**Output:**

Created [https://www.googleapis.com/compute/v1/projects/cloud-network-module-101/zones/us-west1-c/instances/us-test-01].

NAME ZONE MACHINE\_TYPE PREEMPTIBLE INTERNAL\_IP EXTERNAL\_IP STATUS

us-test-01 us-west1-c e2-standard-2 10.0.0.2 104.198.230.22 RUNNING

1. Now make the us-test-02 and us-test-03 VMs in their correlated subnets:

gcloud compute instances create us-test-02 \

--subnet subnet-us-central1 \

--zone us-central1-b \

--machine-type e2-standard-2 \

--tags ssh,http,rules

Copied!

content\_copy

gcloud compute instances create us-test-03 \

--subnet subnet-us-east4 \

--zone us-east4-a \

--machine-type e2-standard-2 \

--tags ssh,http,rules

Copied!

content\_copy

Click **Check my progress** to verify the objective.

Assessment Completed!

Create three instances in specified zones for Traceroute and performance testing.

Check my progress

*Assessment Completed!*

Verify you can connect your VM

Now do a few exercises to test the connection to your VMs.

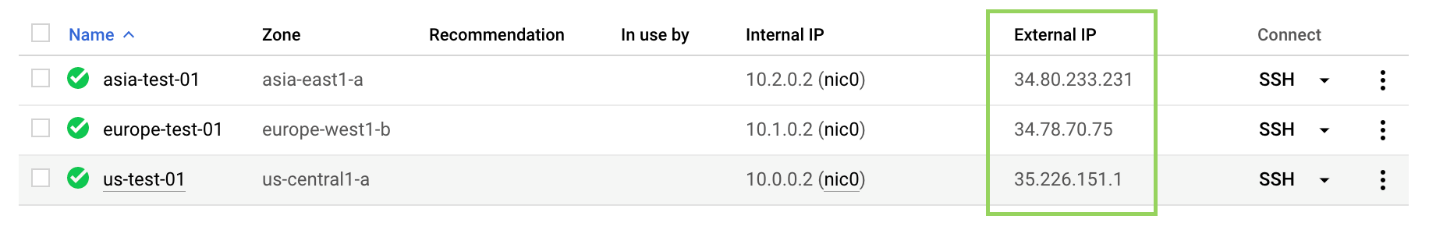
1. Switch back to the Console and navigate to **Compute Engine**.
2. Click the **SSH** button corresponding to the us-test-01 instance. This opens an SSH connection to the instance in a new window.
3. In the SSH window of us-test-01, type the following command to use an ICMP echo against us-test-02, adding the external IP address for the VM in-line:

ping -c 3 <us-test-02-external-ip-address>

Copied!

content\_copy

**Note:**You can locate the external IP of your virtual machines in the **Compute Engine** browser tab under the **External IP** field.



**Your IP addresses will differ from the picture**.

1. Run this command to use an ICMP echo against us-test-03, adding the external IP address for the VM in-line:

ping -c 3 <us-test-03-external-ip-address>

Copied!

content\_copy

**Example output:**

PING 35.187.149.67 (35.187.149.67) 56(84) bytes of data.

64 bytes from 35.187.149.67: icmp\_seq=1 ttl=76 time=152 ms

64 bytes from 35.187.149.67: icmp\_seq=2 ttl=76 time=152 ms

64 bytes from 35.187.149.67: icmp\_seq=3 ttl=76 time=152 ms

1. Now check that SSH also works for instances us-test-02 and us-test-03. Try an ICMP echo against us-test-01.

Use ping to measure latency

Use ping to measure the latency between instances between all the regions.

* To observe the latency from the US Central region to the Europe West region, run the following command after opening an SSH window on the us-test-01:

ping -c 3 us-test-02.us-central1-b

Copied!

content\_copy

**Command output:**

PING us-test-02.us-central1-b.c.cloud-network-module-101.internal (10.2.0.2) 56(84) bytes of data.

64 bytes from us-test-02.us-central1-b.c.cloud-network-module-101.internal (10.2.0.2): icmp\_seq=1 ttl=64 time=105 ms

64 bytes from us-test-02.us-central1-b.c.cloud-network-module-101.internal (10.2.0.2): icmp\_seq=2 ttl=64 time=104 ms

64 bytes from us-test-02.us-central1-b.c.cloud-network-module-101.internal (10.2.0.2): icmp\_seq=3 ttl=64 time=104 ms

The latency you get back is the "Round Trip Time" (RTT) - the time the packet takes to get from us-test-01 to us-test-02.

To test connectivity, Ping uses the ICMP [Echo Request and Echo Reply Messages](http://www.tcpipguide.com/free/t_ICMPv4EchoRequestandEchoReplyMessages.htm).

**Note: Things to think about**

What is the latency you see between regions? What would you expect under ideal conditions? What is special about the connection from `us-test-02` to `us-test-03`?**Note:**Internal DNS: How is DNS provided for VM instances?

Each instance has a metadata server that also acts as a DNS resolver for that instance. DNS lookups are performed for instance names. The metadata server itself stores all DNS information for the local network and queries Google's public DNS servers for any addresses outside of the local network.

An internal fully qualified domain name (FQDN) for an instance looks like this: hostName.[ZONE].c.[PROJECT\_ID].internal .

You can always connect from one instance to another using this FQDN. If you want to connect to an instance using, for example, just hostName, you need information from the internal DNS resolver that is provided as part of Compute Engine.

**Task 7. (Optional) Traceroute and Performance testing**

Optional exercise

Traceroute is a tool to trace the path between two hosts. A traceroute can be a helpful first step to uncovering many different types of network problems. Support or network engineers often ask for a traceroute when diagnosing network issues.

**Note: Functionality**

Traceroute shows all Layer 3 (routing layer) hops between the hosts. This is achieved by sending packets to the remote destination with increasing TTL (Time To Live) value (starting at 1). The TTL field is a field in the IP packet which gets decreased by one at every router. Once the TTL hits zero, the packet gets discarded and a "TTL exceeded" ICMP message is returned to the sender. This approach is used to avoid routing loops; packets cannot loop continuously because the TTL field will eventually decrement to 0. By default the OS sets the TTL value to a high value (64, 128, 255 or similar), so this should only ever be reached in abnormal situations.

So traceroute sends packets first with TTL value of 1, then TTL value of 2, etc., causing these packets to expire at the first/second/etc. router in the path. It then takes the source IP/host of the ICMP TTL exceeded message returned to show the name/IP of the intermediate hop. Once the TTL is high enough, the packet reaches the destination, and the destination responds.

The type of packet sent varies by implementation. Under Linux, UDP packets are sent to a high, unused port. So the final destination responds with an ICMP Port Unreachable. Windows and the mtr tool by default use ICMP echo requests (like ping), so the final destination answers with an ICMP echo reply.

Now try it out by setting up a traceroute on one of your virtual machines.

1. For this step go back to using the us-test-01 VM and us-test-02 VM and SSH into both of them.
2. Install these performance tools in the SSH window for us-test-01:

sudo apt-get update

sudo apt-get -y install traceroute mtr tcpdump iperf whois host dnsutils siege

Copied!

content\_copy

traceroute www.icann.org

Copied!

content\_copy

1. Now try a few other destinations and also from other sources:
   * VMs in the same region or another region (eu1-vm, asia1-vm, w2-vm)
   * [www.wikipedia.org](http://www.wikipedia.org/)
   * [www.adcash.com](http://www.adcash.com/)
   * bad.horse (works best if you increase max TTL, so traceroute -m 255 bad.horse)
   * Anything else you can think of
2. To stop traceroute, **Ctrl**-**c** in the SSH window and return to the command line.

**Note:**Things to think about

What do you notice with the different traceroutes?

**Task 8. (Optional) Use iperf to test performance**

Between two hosts

**Optional exercise**

When you use iperf to test the performance between two hosts, one side needs to be set up as the iperf server to accept connections.

**Note:**The following commands transfer Gigabytes of traffic between regions, which is charged at [Internet egress](https://cloud.google.com/compute/pricing#internet_egress) rates. Be mindful of this when using them. If you are not on a allowlisted project, or in the free trial, you might want to skip, or only skim, this section. (Costs should be less than $1 USD.)

Try a very simple test:

1. SSH into us-test-02 and install the performance tools:

sudo apt-get update

sudo apt-get -y install traceroute mtr tcpdump iperf whois host dnsutils siege

Copied!

content\_copy

1. SSH into us-test-01 and run:

iperf -s #run in server mode

Copied!

content\_copy

1. On us-test-02 SSH run this iperf:

iperf -c us-test-01.us-west1-c #run in client mode

Copied!

content\_copy

You will see some output like this:

Client connecting to eu-vm, TCP port 5001

TCP window size: 45.0 KByte (default)

[ 3] local 10.20.0.2 port 35923 connected with 10.30.0.2 port 5001

[ ID] Interval Transfer Bandwidth

[ 3] 0.0-10.0 sec 298 MBytes 249 Mbits/sec

1. On us-test-01 use **Ctrl + c** to exit the server side when you're done.

Between VMs within a region

**Optional exercise**

Now you'll deploy another instance (us-test-04)in a different zone than us-test-01. You will see that within a region, the bandwidth is limited by the [2 Gbit/s per core](https://cloud.google.com/compute/docs/networks-and-firewalls#egress_throughput_caps) egress cap.

1. In Cloud Shell, create us-test-04:

gcloud compute instances create us-test-04 \

--subnet subnet-us-west1 \

--zone us-west1-c \

--tags ssh,http

Copied!

content\_copy

1. SSH to us-test-04 and install performance tools:

sudo apt-get update

sudo apt-get -y install traceroute mtr tcpdump iperf whois host dnsutils siege

Copied!

content\_copy

Between regions you reach much lower limits, mostly due to limits on TCP window size and single stream performance. You can increase bandwidth between hosts by using other parameters, like UDP.

1. On us-test-02 SSH run:

iperf -s -u #iperf server side

Copied!

content\_copy

1. On us-test-01 SSH run:

iperf -c us-test-02.us-central1-b -u -b 2G #iperf client side - send 2 Gbits/s

Copied!

content\_copy

This should be able to achieve a higher speed between EU and US. Even higher speeds can be achieved by running a bunch of TCP iperfs in parallel. Let's test this.

1. In the SSH window for us-test-01 run:

iperf -s

Copied!

content\_copy

1. In the SSH window for us-test-02 run:

iperf -c us-test-01.us-west1-c -P 20

Copied!

content\_copy

The combined bandwidth should be really close to the maximum achievable bandwidth.

1. Test a few more combinations. If you use Linux on your laptop you can test against your laptop as well. (You can also try [iperf3](https://github.com/esnet/iperf) which is available for [many OSes](https://iperf.fr/iperf-download.php), but this is not part of the lab.)

As you can see, to reach the maximum bandwidth, just running a single TCP stream (for example, file copy) is not sufficient; you need to have several TCP sessions in parallel. Reasons are: TCP parameters such as Window Size; and functions such as Slow Start.

For more information on this and all other TCP/IP topics, refer to [TCP/IP Illustrated](https://www.amazon.com/TCP-Ip-Illustrated-Volume-Protocols/dp/9332535957/ref=dp_ob_title_bk).

Tools like [bbcp](https://github.com/eeertekin/bbcp" \t "_blank) can help to copy files as fast as possible by parallelizing transfers and using configurable window size.

**Task 9. Test your knowledge**

Which of the following subnet creation modes are supported by Google Cloud?

Legacy Mode

checkCustom mode

All of these

checkAuto mode

Submit

**End your lab**

When you have completed your lab, click **End Lab**. Your account and the resources you've used are removed from the lab platform.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

* 1 star = Very dissatisfied
* 2 stars = Dissatisfied
* 3 stars = Neutral
* 4 stars = Satisfied
* 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.