# Essential Google Cloud Infrastructure: Foundation

* learn how to use the Google Cloud through the console and Cloud Shell. You'll also learn about the role of a cloud architect, approaches to infrastructure design, and virtual networking configuration with Virtual Private Cloud (VPC), Projects, Networks, Subnetworks, IP addresses, Routes, and Firewall rules.
* Interacting with Google Cloud
  + Using Google Cloud 3 minutes - https://youtu.be/b1s2SSLPzSA
  + Working with GCP Cloud Console and Cloud Shell 20 minutes
    - Task 5: Create a persistent state in Cloud Shell
      * INFRACLASS\_REGION=[YOUR\_REGION]
      * echo $INFRACLASS\_REGION
      * mkdir infraclass
      * touch infraclass/config
      * Append the value of your Region environment variable to the config file:
        + echo INFRACLASS\_REGION=$INFRACLASS\_REGION >> ~/infraclass/config
      * INFRACLASS\_PROJECT\_ID=[YOUR\_PROJECT\_ID]
      * Append the value of your Project ID environment variable to the config file:
        + echo INFRACLASS\_PROJECT\_ID=$INFRACLASS\_PROJECT\_ID >> ~/infraclass/config
      * Use the source command to set the environment variables, and use the echo command to verify that the project variable was set:
        + source infraclass/config
        + echo $INFRACLASS\_PROJECT\_ID
      * This gives you a method to create environment variables and to easily recreate them if the Cloud Shell is recycled or reset. However, you will still need to remember to issue the source command each time Cloud Shell is opened. In the next step, you modify the .profile file so that the source command is issued automatically every time a terminal to Cloud Shell is opened.
      * Close and re-open Cloud Shell. Then issue the echo command again:
        + echo $INFRACLASS\_PROJECT\_ID
      * There will be no output because the environment variable no longer exists.
      * Modify the bash profile and create persistence
        + Edit the shell profile with the following command:

nano .profile

* + - * + Add the following line to the end of the file:

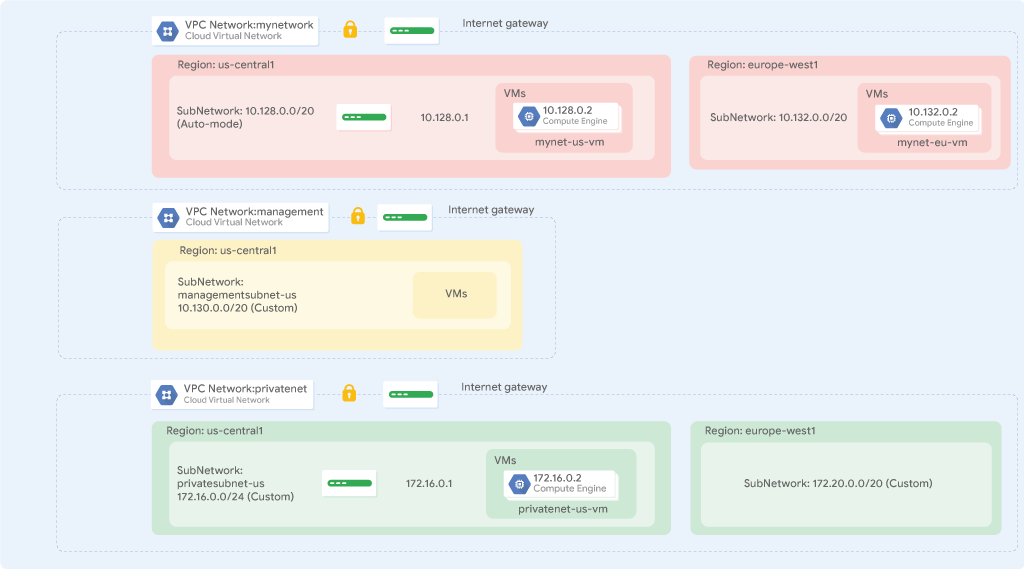
source infraclass/config

* + - * + Press Ctrl+O, ENTER to save the file, and then press Ctrl+X to exit nano.
        + Close and then re-open Cloud Shell to reset the VM.
        + Use the echo command to verify that the variable is still set:

echo $INFRACLASS\_PROJECT\_ID

* + - * + You should now see the expected value that you set in the config file.
  + Lab Review: Working with the Cloud Console and Cloud Shell 8 minutes - https://youtu.be/Lnj7X7Dp1P4
  + Infrastructure Preview 15 minutes
    - Overview
      * build a sophisticated deployment in minutes using Marketplace.
    - Objectives
      * Use Marketplace to build a Jenkins Continuous Integration environment.
      * Verify that you can manage the service from the Jenkins UI.
      * Administer the service from the Virtual Machine host through SSH.
    - Task 1: Use Marketplace to build a deployment
      * Navigation > Marketplace > searching for Jenkins Certified by Bitnami.
        + Jenkins is an open-source continuous integration environment. You can define jobs in Jenkins that can perform tasks such as running a scheduled build of software and backing up data. Notice the software that is installed as part of Jenkins shown in the left side of the description.
        + field that says "Last updated." How recently was this template updated?
        + Google Cloud Marketplace lets you quickly deploy functional software packages by providing pre-defined templates with which Google Cloud service?

Deployment Manager

* + - * Launch > Deploy > Take a 1 or 2 min for Deployment Manager to set up the deployment.
        + Deployment Manager is a Google Cloud service that uses templates written in a combination of YAML, python, and Jinja2 to automate the allocation of Google Cloud resources and perform setup tasks. Behind the scenes a virtual machine has been created. A startup script was used to install and configure software, and network Firewall Rules were created to allow traffic to the service.
      * In the right pane, click More about the software to view additional software details.
      * Copy the Admin user and Admin password values to a text editor.
      * Click Visit the site to view the site, you might have to reload the page a couple of times.
      * Log in with the Admin user and Admin password values.
      * Navigation > Deployment Manager > jenkins-1 > SSH to connect to the Jenkins server.
      * In the SSH window, enter the following command to shut down all the running services:
        + sudo /opt/bitnami/ctlscript.sh stop
      * In the SSH window, enter the following command to restart the services:
        + sudo /opt/bitnami/ctlscript.sh restart
  + Lab Review: Infrastructure Preview 7 minutes - https://youtu.be/HAuJYszD4E0
  + Demo: Projects 6 minutes - https://youtu.be/1IVgnW2Z6ZA
  + Quiz: Interacting with Google Cloud
    - Which of the following does not allow you to interact with Google Cloud?
      * Cloud Explorer
      * That's correct! There are four ways you can interact with Google Cloud: There’s the Cloud Console, Cloud Shell and the Cloud SDK, the APIs, and the Cloud Mobile App. The Cloud Explorer is not a Google Cloud tool.
    - What is the difference between the Google Cloud Console and Cloud Shell?
      * Cloud Shell is a command-line tool, while the Cloud Console is a graphical user interface
      * The Cloud Console is a graphical user interface and Cloud Shell is a command-line tool. Both tools allow you to interact with Google Cloud. Even though the Cloud Console can do things Cloud Shell can't do and vice-versa, don’t think of them as alternatives, but think of them as one extremely flexible and powerful interface.
* Virtual Networks
  + Virtual Private Cloud 1 minute - https://youtu.be/ECb20jnlaTE
  + Projects, networks, and subnetworks 7 minutes - https://youtu.be/ECb20jnlaTE
  + Demo: Expand a Subnet 3 minutes - https://youtu.be/bGZODriew7k
  + IP addresses 1 minute - https://youtu.be/12k04e\_aXyc
  + Demo: Internal and external IP 4 minutes - https://youtu.be/5FxZoZg1sGE
  + Mapping IP addresses 4 minutes - https://youtu.be/1Sf-j\_xWL48
  + IP addresses for default domains - Document
  + Routes and firewall rules 6 minutes - https://youtu.be/emL7gNwWWnM
  + Pricing 3 minutes - https://youtu.be/WWNtZKrgGuY
  + VPC Networking 45 minutes - <https://www.cloudskillsboost.google/course_sessions/834645/labs/112421>
    - Overview
      * Google Cloud Virtual Private Cloud (VPC) provides networking functionality to Compute Engine virtual machine (VM) instances, Kubernetes Engine containers, and the App Engine flexible environment. In other words, without a VPC network, you cannot create VM instances, containers, or App Engine applications. Therefore, each Google Cloud project has a default network to get you started.
      * You can think of a VPC network as similar to a physical network, except that it is virtualized within Google Cloud. A VPC network is a global resource that consists of a list of regional virtual subnetworks (subnets) in data centers, all connected by a global wide area network (WAN). VPC networks are logically isolated from each other in Google Cloud.
      * you create an auto mode VPC network with firewall rules and two VM instances. Then, you convert the auto mode network to a custom mode network and create other custom mode networks as shown in the network diagram below. You also test connectivity across networks.
        + 
    - Objectives
      * Explore the default VPC network
      * Create an auto mode network with firewall rules
      * Convert an auto mode network to a custom mode network
      * Create custom mode VPC networks with firewall rules
      * Create VM instances using Compute Engine
      * Explore the connectivity for VM instances across VPC networks
    - Task 1. Explore the default network
      * Each Google Cloud project has a default network with subnets, routes, and firewall rules.
      * View the subnets
        + The default network has a subnet in each Google Cloud region.
        + Navigation menu , click VPC network > VPC networks.
        + Notice the default network with its subnets. Each subnet is associated with a Google Cloud region and a private RFC 1918 CIDR block for its internal IP addresses range and a gateway.
      * View the routes
        + Routes tell VM instances and the VPC network how to send traffic from an instance to a destination, either inside the network or outside Google Cloud. Each VPC network comes with some default routes to route traffic among its subnets and send traffic from eligible instances to the internet.
        + click Routes. Notice that there is a route for each subnet and one for the Default internet gateway (0.0.0.0/0). These routes are managed for you, but you can create custom static routes to direct some packets to specific destinations. For example, you can create a route that sends all outbound traffic to an instance configured as a NAT gateway.
      * View the firewall rules
        + Each VPC network implements a distributed virtual firewall that you can configure. Firewall rules allow you to control which packets are allowed to travel to which destinations. Every VPC network has two implied firewall rules that block all incoming connections and allow all outgoing connections.
        + click Firewall. Notice that there are 4 Ingress firewall rules for the default network:

default-allow-icmp

default-allow-rdp

default-allow-ssh

default-allow-internal

* + - * + These firewall rules allow ICMP, RDP, and SSH ingress traffic from anywhere (0.0.0.0/0) and all TCP, UDP, and ICMP traffic within the network (10.128.0.0/9). The Targets, Filters, Protocols/ports, and Action columns explain these rules.
        + Delete all the default Firewall rules
        + Delete the default VPC network
        + Note: Without a VPC network, there are no routes!
        + Without a VPC network, you cannot create VM instances, containers, or App Engine applications. True
      * Try to create a VM instance
        + Verify that you cannot create a VM instance without a VPC network.
        + Navigation > Compute Engine > VM instances > Create Instance > Create.

Click Management, security, disks, networking, sole tenancy.

Click Networking. Notice the No local network available error under Network interfaces. Click Cancel.

Note: As expected, you cannot create a VM instance without a VPC network!

* + - Task 2. Create an auto mode network
      * You have been tasked to create an auto mode network with two VM instances. Auto mode networks are easy to set up and use because they automatically create subnets in each region. However, you don't have complete control over the subnets created in your VPC network, including regions and IP address ranges used.
      * Feel free to explore more considerations for choosing an auto mode network, but for now, assume that you are using the auto mode network for prototyping purposes.
      * Create an auto mode VPC network with firewall rules
        + Navigation > click VPC network > VPC networks > Create VPC network.
        + For Name, type mynetwork For Subnet creation mode, click Automatic.

Auto mode networks create subnets in each region automatically.

* + - * + For Firewall rules, select all available rules.

These are the same standard firewall rules that the default network had. The deny-all-ingress and allow-all-egress rules are also displayed, but you cannot select or disable them because they are implied. These two rules have a lower Priority (higher integers indicate lower priorities) so that the allow ICMP, internal, RDP, and SSH rules are considered first.

* + - * + Click Create.

When the new network is ready, notice that a subnet was created for each region.

* + - * + Note: If you ever delete the default network, you can quickly re-create it by creating an auto mode network as you just did.
      * Create a VM instance in us-central1
        + Create a VM instance in the us-central1 region. Selecting a region and zone determines the subnet and assigns the internal IP address from the subnet's IP address range.
        + Navigation > Compute Engine > VM instances > Create Instance.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name mynet-us-vm

Region us-central1

Zone us-central1-c

Series N1

Machine type n1-standard-1 (1 vCPU, 3.75 GB memory)

Boot disk Debian GNU/Linux 10 (buster)

* + - * + Click Create.
        + Verify that the Internal IP for the new instance was assigned from the IP address range for the subnet in us-central1 (10.128.0.0/20).
        + The Internal IP should be 10.128.0.2 because 10.128.0.1 is reserved for the gateway, and you have not configured any other instances in that subnet.
      * Create a VM instance in europe-west1
        + Create a VM instance in the europe-west1 region.
        + Click Create instance.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name mynet-eu-vm

Region europe-west1

Zone europe-west1-c

Series N1

Machine type n1-standard-1 (1 vCPU, 3.75 GB memory)

Boot disk Debian GNU/Linux 10 (buster)

* + - * + Click Create.
        + Verify that the Internal IP for the new instance was assigned from the IP address range for the subnet in europe-west1 (10.132.0.0/20).
        + The Internal IP should be 10.132.0.2 because 10.132.0.1 is reserved for the gateway, and you have not configured any other instances in that subnet.
        + Note: The External IP addresses for both VM instances are ephemeral. If an instance is stopped, any ephemeral external IP addresses assigned to the instance are released back into the general Compute Engine pool and become available for use by other projects.
        + When a stopped instance is started again, a new ephemeral external IP address is assigned to the instance. Alternatively, you can reserve a static external IP address, which assigns the address to your project indefinitely until you explicitly release it.
      * Verify connectivity for the VM instances
        + The firewall rules that you created with mynetwork allow ingress SSH and ICMP traffic from within mynetwork (internal IP) and outside that network (external IP).
        + Navigation > Compute Engine > VM instances.

Note the external and internal IP addresses for mynet-eu-vm.

* + - * + For mynet-us-vm, click SSH to launch a terminal and connect.

Note: You can SSH because of the allow-ssh firewall rule, which allows incoming traffic from anywhere (0.0.0.0/0) for tcp:22. The SSH connection works seamlessly because Compute Engine generates an SSH key for you and stores it in one of the following locations:

By default, Compute Engine adds the generated key to project or instance metadata.

If your account is configured to use OS Login, Compute Engine stores the generated key with your user account.

Alternatively, you can control access to Linux instances by creating SSH keys and editing public SSH key metadata.

* + - * + To test connectivity to mynet-eu-vm's internal IP, run the following command, replacing mynet-eu-vm's internal IP:

ping -c 3 <Enter mynet-eu-vm's internal IP here>

* + - * + You can ping mynet-eu-vm's internal IP because of the allow-internal firewall rule.
        + To test connectivity to mynet-eu-vm's external IP, run the following command, replacing mynet-eu-vm's external IP:

ping -c 3 <Enter mynet-eu-vm's external IP here>

* + - * + Which firewall rule allows the ping to mynet-eu-vm's external IP address?

mynetwork-allow-icmp

* + - * + Note: You can SSH to mynet-us-vm and ping mynet-eu-vm's internal and external IP addresses as expected. Alternatively, you can SSH to mynet-eu-vm and ping mynet-us-vm's internal and external IP addresses, which also works.
      * Convert the network to a custom mode network
        + The auto mode network worked great so far, but you have been asked to convert it to a custom mode network so that new subnets aren't automatically created as new regions become available. This could result in overlap with IP addresses used by manually created subnets or static routes, or could interfere with your overall network planning.
        + Navigation > VPC network > VPC networks > Click mynetwork > Click Edit.
        + Select Custom for the Subnet creation mode. Click Save.
        + Return to the VPC networks page.
        + Note: Converting an auto mode network to a custom mode network is an easy task, and it provides you with more flexibility. We recommend that you use custom mode networks in production.
    - Task 3. Create custom mode networks
      * You have been tasked to create two additional custom networks, managementnet and privatenet, along with firewall rules to allow SSH, ICMP, and RDP ingress traffic and VM instances as shown in this diagram (with the exception of vm-appliance):
      * VPC Networking Architecture [image)
      * Note that the IP CIDR ranges of these networks do not overlap. This allows you to set up mechanisms such as VPC peering between the networks. If you specify IP CIDR ranges that are different from your on-premises network, you could even configure hybrid connectivity using VPN or Cloud Interconnect.
      * Create the managementnet network using the Cloud Console.
        + Navigation > VPC network > VPC networks > Create VPC Network.
        + For Name, type managementnet
        + For Subnet creation mode, click Custom.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name managementsubnet-us

Region us-central1

IP address range 10.130.0.0/20

* + - * + Click Done. Click Create.
      * Create the privatenet network
        + To create the privatenet network, run the following command:

gcloud compute networks create privatenet --subnet-mode=custom

* + - * + To create the privatesubnet-us subnet, run the following command:

gcloud compute networks subnets create privatesubnet-us --network=privatenet --region=us-central1 --range=172.16.0.0/24

* + - * + To create the privatesubnet-eu subnet, run the following command:

gcloud compute networks subnets create privatesubnet-eu --network=privatenet --region=europe-west1 --range=172.20.0.0/20

* + - * + To list the available VPC networks, run the following command:

gcloud compute networks list

* + - * + To list the available VPC subnets (sorted by VPC network), run the following cmd:

gcloud compute networks subnets list --sort-by=NETWORK

* + - * + Note: The managementnet and privatenet networks only have the subnets that you created because they are custom mode networks. mynetwork is also a custom mode network, but it started out as an auto mode network, resulting in subnets in each region.
      * Create the firewall rules for managementnet
        + Create firewall rules to allow SSH, ICMP, and RDP ingress traffic to VM instances on the managementnet network.
        + Navigation > click VPC network > Firewall > Create Firewall Rule.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name managementnet-allow-icmp-ssh-rdp

Network managementnet

Targets All instances in the network

Source filter IPv4 Ranges

Source IPv4 ranges 0.0.0.0/0

Protocols and ports Specified protocols and ports

* + - * + Note: Make sure to include the /0 in the Source IPv4 ranges to specify all networks.
        + Select tcp and specify ports 22 and 3389.
        + Select Other protocols and specify icmp protocol.
        + Click Create.
      * Create the firewall rules for privatenet
        + Create the firewall rules for privatenet network using the gcloud command line.
        + To create the privatenet-allow-icmp-ssh-rdp firewall rule, run the following cmd:

gcloud compute firewall-rules create privatenet-allow-icmp-ssh-rdp --direction=INGRESS --priority=1000 --network=privatenet --action=ALLOW --rules=icmp,tcp:22,tcp:3389 --source-ranges=0.0.0.0/0

* + - * + To list all the firewall rules (sorted by VPC network), run the following command:

gcloud compute firewall-rules list --sort-by=NETWORK

* + - * + The firewall rules for mynetwork network have been created for you. You can define multiple protocols and ports in one firewall rule (privatenet and managementnet) or spread them across multiple rules (default and mynetwork).
      * Next, create two VM instances:
        + managementnet-us-vm in managementsubnet-us
        + privatenet-us-vm in privatesubnet-us
      * Create the managementnet-us-vm instance
        + Create the managementnet-us-vm instance using the Cloud Console.
        + Navigation > Compute Engine > VM instances > Create instance.

Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name managementnet-us-vm

Region us-central1

Zone us-central1-c

Series N1

Machine type f1-micro (1 vCPU, 614 MB memory)

Boot disk Debian GNU/Linux 10 (buster)

* + - * + Click Management, security, disks, networking, sole tenancy.
        + Click Networking. For Network interfaces, click the dropdown arrow to edit.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Network managementnet

Subnetwork managementsubnet-us

Note: The subnets available for selection are restricted to those in the selected region (us-central1).

Click Done.

* + - * + Click Create.
      * Create the privatenet-us-vm instance
        + Create the privatenet-us-vm instance using the gcloud command line.
      * To create the privatenet-us-vm instance, run the following command:
        + gcloud compute instances create privatenet-us-vm --zone=us-central1-c --machine-type=f1-micro --subnet=privatesubnet-us --image-family=debian-10 --image-project=debian-cloud --boot-disk-size=10GB --boot-disk-type=pd-standard --boot-disk-device-name=privatenet-us-vm
        + To list all the VM instances (sorted by zone), run the following command:

gcloud compute instances list --sort-by=ZONE

* + - * Check VM Zone.
      * There are three instances in us-central1-c and one instance in europe-west1-c. However, these instances are spread across three VPC networks (managementnet, mynetwork, and privatenet), with no instance in the same zone and network as another. In the next task, you explore the effect this has on internal connectivity.
    - Task 4. Explore the connectivity across networks
      * Explore the connectivity between the VM instances. Specifically, determine the effect of having VM instances in the same zone versus having instances in the same VPC network.
      * Ping the external IP addresses
        + Ping the external IP addresses of the VM instances to determine whether you can reach the instances from the public internet.
        + Compute Engine > VM instances. Note the external IP addresses for mynet-eu-vm, managementnet-us-vm, and privatenet-us-vm.
        + For mynet-us-vm, click SSH to launch a terminal and connect.
        + To test connectivity to mynet-eu-vm's external IP, run the following command, replacing mynet-eu-vm's external IP:

ping -c 3 <Enter mynet-eu-vm's external IP here>

* + - * + To test connectivity to managementnet-us-vm's external IP, run the following command, replacing managementnet-us-vm's external IP:

ping -c 3 <Enter managementnet-us-vm's external IP here>

* + - * + To test connectivity to privatenet-us-vm's external IP, run the following command, replacing privatenet-us-vm's external IP:

ping -c 3 <Enter privatenet-us-vm's external IP here>

* + - * + All Work
        + Note: You can ping the external IP address of all VM instances, even though they are in either a different zone or VPC network. This confirms that public access to those instances is only controlled by the ICMP firewall rules that you established earlier.
      * Ping the internal IP addresses
        + Ping the internal IP addresses of the VM instances to determine whether you can reach the instances from within a VPC network.
        + Which instances should you be able to ping from mynet-us-vm using internal IP addresses?

mynet-eu-vm

* + - * + Navigation >Compute Engine > VM instances. Note the internal IP addresses for mynet-eu-vm, managementnet-us-vm, and privatenet-us-vm.
        + Return to the SSH terminal for mynet-us-vm.
        + To test connectivity to mynet-eu-vm's internal IP, run the following command, replacing mynet-eu-vm's internal IP:

ping -c 3 <Enter mynet-eu-vm's internal IP here>

* + - * + Note: You can ping the internal IP address of mynet-eu-vm because it is on the same VPC network as the source of the ping (mynet-us-vm), even though both VM instances are in separate zones, regions, and continents!
        + To test connectivity to managementnet-us-vm's internal IP, run the following command, replacing managementnet-us-vm's internal IP:

ping -c 3 <Enter managementnet-us-vm's internal IP here>

* + - * + Note: This should not work, as indicated by a 100% packet loss!
        + To test connectivity to privatenet-us-vm's internal IP, run the following command, replacing privatenet-us-vm's internal IP:

ping -c 3 <Enter privatenet-us-vm's internal IP here>

* + - * + Note: This should not work either, as indicated by a 100% packet loss! You cannot ping the internal IP address of managementnet-us-vm and privatenet-us-vm because they are in separate VPC networks from the source of the ping (mynet-us-vm), even though they are all in the same zone, us-central1-c.
  + Lab Review: VPC Networking 22 minutes - https://youtu.be/CmSBvw2SXUY
  + Common network designs 3 minutes - https://youtu.be/PmBxhEHmKVc
  + Implement Private Google Access and Cloud NAT 1 hour 20 minutes
    - Overview
      * implement Private Google Access and Cloud NAT for a VM instance that doesn't have an external IP address. Then, you verify access to public IP addresses of Google APIs and services and other connections to the internet.
      * VM instances without external IP addresses are isolated from external networks. Using Cloud NAT, these instances can access the internet for updates and patches, and in some cases, for bootstrapping. As a managed service, Cloud NAT provides high availability without user management and intervention.
    - Objectives
      * Configure a VM instance that doesn't have an external IP address
      * Connect to a VM instance using an Identity-Aware Proxy (IAP) tunnel
      * Enable Private Google Access on a subnet
      * Configure a Cloud NAT gateway
      * Verify access to public IP addresses of Google APIs and services and other connections to the internet
    - Task 1. Create the VM instance
      * Create a VPC network with some firewall rules and a VM instance that has no external IP address, and connect to the instance using an IAP tunnel.
      * Create a VPC network and firewall rules
        + create a VPC network for the VM instance and a firewall rule to allow SSH access.
        + Navigation > VPC network > Create VPC network > For Name - privatenet.

For Subnet creation mode, click Custom.

Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name privatenet-us

Region us-central1

IP address range 10.130.0.0/20

Don't enable Private Google access yet!

Click Done. Click Create and wait for the network to be created.

* + - * + Firewall > Create firewall rule.

Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name privatenet-allow-ssh

Network privatenet

Targets All instances in the network

Source filter IP ranges

Source IP ranges 35.235.240.0/20

Protocols and ports Specified protocols and ports

For tcp, click the checkbox and specify port 22.

Click Create.

* + - * + In order to connect to your private instance using SSH, you need to open an appropriate port on the firewall. IAP connections come from a specific set of IP addresses (35.235.240.0/20). Therefore, you can limit the rule to this CIDR range.
      * Create the VM instance with no public IP address
        + Navigation > Compute Engine > VM instances > Create.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name vm-internal

Region us-central1

Zone us-central1-c

Series N1

Machine type n1-standard-1 (1vCPU, 3.75 GB memory)

Boot disk Debian GNU/Linux 10 (buster)

* + - * + Click Management, security, disks, networking, sole tenancy.
        + Click Networking. For Network interfaces, click the pencil icon to edit.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Network privatenet

Subnetwork privatenet-us

External IP None

* + - * + The default setting for a VM instance is to have an ephemeral external IP address. This behavior can be changed with a policy constraint at the organization or project level. To learn more about controlling external IP addresses on VM instances, refer to the external IP address documentation.
        + Click Done. Click Create, and wait for the VM instance to be created.
        + VM instances page, verify that the External IP of vm-internal is None.
      * SSH to vm-internal to test the IAP tunnel
        + Activate Cloud Shell. To connect to vm-internal, run the following command:

gcloud compute ssh vm-internal --zone us-central1-c --tunnel-through-iap

If prompted about continuing, type Y.

When prompted for a passphrase, press ENTER.

Did the command prompt change to @vm-internal?

True

* + - * + To test the external connectivity of vm-internal, run the following command:

ping -c 2 www.google.com

* + - * + This should not work because vm-internal has no external IP address!
        + When instances do not have external IP addresses, they can only be reached by other instances on the network via a managed VPN gateway or via a Cloud IAP tunnel. Cloud IAP enables context-aware access to VMs via SSH and RDP without bastion hosts. For more information about this, see this blog post.
    - Task 2. Enable Private Google Access
      * VM instances that have no external IP addresses can use Private Google Access to reach external IP addresses of Google APIs and services. By default, Private Google Access is disabled on a VPC network.
      * Create a Cloud Storage bucket
        + Create a Cloud Storage bucket to test access to Google APIs and services.
        + Navigation > Storage > bucket > Name, Location type Multi-region > Create.
        + Copy an image from a public Cloud Storage bucket to your own bucket.

gsutil cp gs://cloud-training/gcpnet/private/access.svg gs://[my\_bucket]

* + - * + Currently, which of your VM instances can access the image from your bucket?

Cloud Shell

* + - * + In Cloud Shell, to try to copy the image from your bucket

gsutil cp gs://[my\_bucket]/\*.svg .

* + - * + This should work because Cloud Shell has an external IP address!
        + To connect to vm-internal, run the following command:

gcloud compute ssh vm-internal --zone us-central1-c --tunnel-through-iap

* + - * + To try to copy the image to vm-internal:

gsutil cp gs://[my\_bucket]/\*.svg .

* + - * + This should not work: vm-internal can only send traffic within the VPC network because Private Google Access is disabled (by default).
      * Enable Private Google Access
        + Private Google Access is enabled at the subnet level. When it is enabled, instances in the subnet that only have private IP addresses can send traffic to Google APIs and services through the default route (0.0.0.0/0) with a next hop to the default internet gateway.
        + Navigation > VPC network > VPC networks > Click privatenet > privatenet-us
        + Click Edit. For Private Google access, select On. Click Save.
        + Yes, enabling Private Google Access is as simple as selecting On within the subnet!
        + In Cloud Shell for vm-internal:

gsutil cp gs://[my\_bucket]/\*.svg .

* + - * + This should work because vm-internal's subnet has Private Google Access enabled!
      * To view the eligible APIs and services that you can use with Private Google Access, see supported services in the Private Google Access overview.
    - Task 3. Configure a Cloud NAT gateway
      * Although vm-internal can now access certain Google APIs and services without an external IP address, the instance cannot access the internet for updates and patches. Configure a Cloud NAT gateway, which allows vm-internal to reach the internet.
      * Try to update the VM instances
        + In Cloud Shell, to try to re-synchronize the package index, run the following:

sudo apt-get update

* + - * + This should work because Cloud Shell has an external IP address!
        + To connect to vm-internal, run the following command:

gcloud compute ssh vm-internal --zone us-central1-c --tunnel-through-iap

* + - * + To try to re-synchronize the package index of vm-internal:

sudo apt-get update

* + - * + This should only work for Google Cloud packages because vm-internal only has access to Google APIs and services!
      * Configure a Cloud NAT gateway
        + Cloud NAT is a regional resource. You can configure it to allow traffic from all ranges of all subnets in a region, from specific subnets in the region only, or from specific primary and secondary CIDR ranges only.
        + Navigation > Network services > Cloud NAT > Click Get started
        + Specify the following:

Property Value (type value or select option as specified)

Gateway name nat-config

VPC network privatenet

Region us-central1

* + - * + For Cloud Router > Create new router. For Name, type nat-router > Create.
        + The NAT mapping section allows you to choose the subnets to map to the NAT gateway. You can also manually assign static IP addresses that should be used when performing NAT. Do not change the NAT mapping configuration in this lab.
        + Click Create.
      * Verify the Cloud NAT gateway
        + It may take up to 3 minutes for the NAT configuration to propagate to the VM, so wait at least a minute before trying to access the internet again.
        + In Cloud Shell for vm-internal, to try to re-synchronize the package index of vm-internal, run the following command:

sudo apt-get update

* + - * + This should work because vm-internal is using the NAT gateway!
        + The Cloud NAT gateway implements outbound NAT, but not inbound NAT. In other words, hosts outside of your VPC network can only respond to connections initiated by your instances; they cannot initiate their own, new connections to your instances via NAT.
    - Task 4. Configure and view logs with Cloud NAT Logging
      * Cloud NAT logging allows you to log NAT connections and errors. When Cloud NAT logging is enabled, one log entry can be generated for each of the following scenarios:
        + When a network connection using NAT is created.
        + When a packet is dropped because no port was available for NAT.
      * You can opt to log both kinds of events, or just one or the other. Created logs are sent to Cloud Logging.
      * Enabling logging
        + If logging is enabled, all collected logs are sent to Cloud Logging by default. You can filter these so that only certain logs are sent.
        + You can also specify these values when you create a NAT gateway or by editing one after it has been created. The following directions show how to enable logging for an existing NAT gateway.
        + Navigation > Network services > Cloud NAT.
        + Click on the nat-config gateway and then click Edit.
        + Click the Advanced configurations dropdown to open that section.
        + Under Stackdriver logging, select Translation and errors and then click Save.
      * NAT logging in Cloud Operations
        + Now that you have set up Cloud NAT logging for the nat-config gateway, let's find out where we can view our logs.

Click on nat-config to expose its details. Then click on the Logs tab. Then click the link to Stackdriver Logging.

This will open a new tab with Operations logging.

* + - * + You will see that there aren't any logs yet—that's because we just enabled this feature for the gateway. Keep this tab open and return to your other GCP Console tab.
      * Generating logs
        + As a reminder, Cloud NAT logs are generated for the following sequences:

When a network connection using NAT is created.

When a packet is dropped because no port was available for NAT.

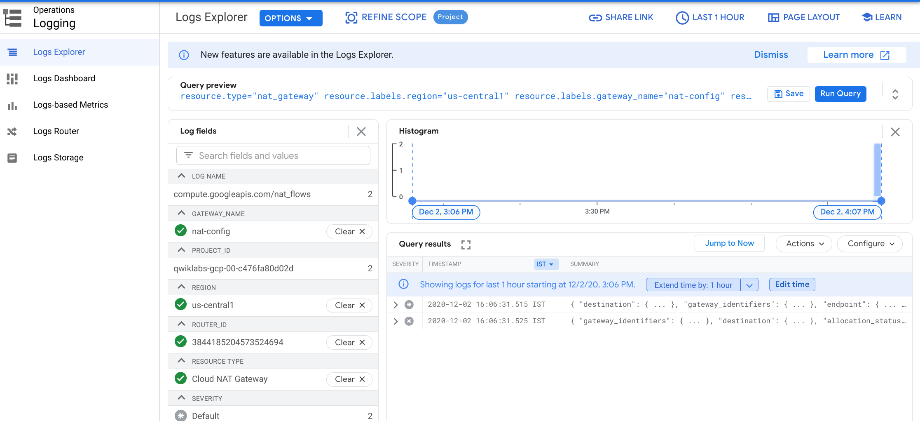
* + - * + Let's connect the host to the internal VM again to see if any logs are generated.
        + In Cloud Shell for vm-internal, to try to re-synchronize the package index of vm-internal, run the following command:

gcloud compute ssh vm-internal --zone us-central1-c --tunnel-through-iap

* + - * + Try to re-synchronize the package index of vm-internal by running the following:

sudo apt-get update

* + - * Viewing Logs
        + Return to your NAT Logging tab and under Configure dropdown, select Show newest logs first.
        + You should see two new logs that were generated after connecting to the internal VM.



* + - * + You may need to wait for few minutes. If still you are unable to see the logs then repeat the step 1 to step 4 from Generating logs section and refresh the logging page.
        + As we see, the logs give us details on the VPC network we connected to and the connection method we used. Feel free to expand different labels and details.
  + Lab Review: Implement Private Google Access and Cloud NAT 15 minutes - https://youtu.be/4YjCGQKvc2k
  + Quiz: Virtual Networks
    - What are the three types of networks offered in Google Cloud?
      * Default network, auto network, and custom network.
      * The three network types offered by Google Cloud are: default, auto and custom. Each project starts with a default network. The auto-type network uses the same subnet IP ranges as the default-type, with a network name other than default. A custom-type allows you to specify the IP ranges of subnets.
    - In Google Cloud, what is the minimum number of IP addresses that a VM instance needs?
      * One: Only an internal IP address
      * That's correct! In Google Cloud, each virtual machine needs to have an internal IP address. The external IP address is optional; therefore, a VM instance only needs one IP address.
    - What is one benefit of applying firewall rules by tag rather than by address?
      * When a VM is created with a matching tag, the firewall rules apply irrespective of the IP address it is assigned.
      * When a VM is created the ephemeral external IP address is assigned from a pool. There is no way to predict which address will be assigned, so there is no way to write a rule that will match that VM's IP address before it is assigned. Tags allow a symbolic assignment that does not depend on order in the IP addresses. It makes for simpler, more general, and easier to maintain, firewall rules.
* Virtual Machines
  + Compute Engine 5 minutes - https://youtu.be/VqvlgQZEh-g
  + Demo: Create a VM 6 minutes - https://youtu.be/hgZIjuAr0IE
  + VM access and lifecycle 6 minutes - https://youtu.be/OIETs4jevUQ
  + Creating Virtual Machines 30 minutes - https://www.cloudskillsboost.google/course\_sessions/834645/labs/112434
  + Lab Review: Creating Virtual Machines 12 minutes - https://youtu.be/-xGtK7mW8Ng
  + Compute options 7 minutes - https://youtu.be/l-XqlOoNKF8
  + Compute pricing 5 minutes - https://youtu.be/AR-ZDU29zQk
  + Special compute configurations 5 minutes - https://youtu.be/6Hu-6dR1ag8
  + Images 1 minute - https://youtu.be/uWLezYdQ6oM
  + Disk options 7 minutes - https://youtu.be/8X4pO75q0W4
  + Common Compute Engine actions 4 minutes - https://youtu.be/wJ-3DVmukSw
  + Working with Virtual Machines 45 minutes - <https://www.cloudskillsboost.google/course_sessions/834645/labs/112443>
    - Overview
      * In this lab, you set up a game application—a Minecraft server.
      * The Minecraft server software will run on a Compute Engine instance.
      * You use an e2-medium machine type that includes a 10-GB boot disk, 2 virtual CPU (vCPU), and 4 GB of RAM. This machine type runs Debian Linux by default.
      * To make sure there is plenty of room for the Minecraft server's world data, you also attach a high-performance 50-GB persistent solid-state drive (SSD) to the instance. This dedicated Minecraft server can support up to 50 players.
    - Objectives
      * Customize an application server
      * Install and configure necessary software
      * Configure network access
      * Schedule regular backups
    - Task 1: Create the VM
      * Define a VM using advanced options
      * Navigation > Compute Engine > VM instances > Create.
      * Specify the following and leave the remaining settings as their defaults:
        + Property Value (type value or select option as specified)
        + Name mc-server
        + Region us-central1
        + Zone us-central1-a
        + Boot disk Debian GNU/Linux 9 (stretch)
      * Identity and API access > Access scopes Set access for each API > Storage Read Write
      * Click Management, security, disks, networking, sole tenancy.
      * Click Disks. You will add a disk to be used for game storage. Click Add new disk.
      * Specify the following and leave the remaining settings as their defaults:
        + Property Value (type value or select option as specified)
        + Name minecraft-disk
        + Disk type SSD Persistent Disk
        + Source type None (blank disk)
        + Size (GB) 50
        + Encryption Google-managed key
      * Click Done. This creates the disk and automatically attaches it to the VM when the VM is created.
      * Click Networking.
        + Specify the following and leave the remaining settings as their defaults:
        + Property Value (type value or select option as specified)
        + Network tags minecraft-server
        + Network interfaces Click default to edit the interface
        + External IP Create IP Address
        + Name mc-server-ip
      * Click Reserve. Click Done. Click Create.
    - Task 2: Prepare the data disk
      * Create a directory and format and mount the disk
      * The disk is attached to the instance, but it is not yet mounted or formatted.
      * For mc-server, click SSH to open a terminal and connect.
      * To create a directory that serves as the mount point for the data disk:
        + sudo mkdir -p /home/minecraft
      * To format the disk, run the following command:
        + sudo mkfs.ext4 -F -E lazy\_itable\_init=0, lazy\_journal\_init=0,discard /dev/disk/by-id/google-minecraft-disk
      * To mount the disk, run the following command:
        + sudo mount -o discard,defaults /dev/disk/by-id/google-minecraft-disk /home/minecraft
    - Task 3: Install and run the application
      * The Minecraft server runs on top of the Java Virtual Machine (JVM), so it requires the Java Runtime Environment (JRE) to run. Because the server doesn't need a graphical user interface, you use the headless version of the JRE. This reduces the JRE's resource usage on the machine, which helps ensure that the Minecraft server has enough room to expand its own resource usage if needed.
      * Install the Java Runtime Environment (JRE) and the Minecraft server
        + In the SSH terminal for mc-server, to update the Debian repositories on the VM:

sudo apt-get update

* + - * + After the repositories are updated, to install the headless JRE, run the following command:

sudo apt-get install -y default-jre-headless

* + - * + To navigate to the directory where the persistent disk is mounted:

cd /home/minecraft

* + - * + To install wget, run the following command:

sudo apt-get install wget

* + - * + If prompted to continue, type Y
        + To download the current Minecraft server JAR file (1.11.2 JAR):

sudo wget https://launcher.mojang.com/v1/objects/d0d0fe2b1dc6ab4c65554cb734270872b72dadd6/server.jar

* + - * + Initialize the Minecraft server, To initialize the Minecraft server:

sudo java -Xmx1024M -Xms1024M -jar server.jar nogui

* + - * + The Minecraft server won't run unless you accept the terms of the End User Licensing Agreement (EULA).
        + To see the files that were created in the first initialization of the Minecraft server:

sudo ls -l

* + - * + You could edit the server.properties file to change the default behavior of the Minecraft server.
        + To edit the EULA, run the following command:

sudo nano eula.txt

Change the last line of the file from eula=false to eula=true

Press Ctrl+O, ENTER to save the file and then press Ctrl+X to exit nano.

* + - * + Don't try to restart the Minecraft server yet. You use a different technique in the next procedure.
      * Create a virtual terminal screen to start the Minecraft server
        + If you start the Minecraft server again now, it is tied to the life of your SSH session: that is, if you close your SSH terminal, the server is also terminated. To avoid this issue, you can use screen, an application that allows you to create a virtual terminal that can be "detached," becoming a background process, or "reattached," becoming a foreground process. When a virtual terminal is detached to the background, it will run whether you are logged in or not.
        + To install screen, run the following command:

sudo apt-get install -y screen

* + - * + To start your Minecraft server in a screen virtual terminal, run the following command: (Use the -S flag to name your terminal mcs)

sudo screen -S mcs java -Xmx1024M -Xms1024M -jar server.jar nogui

* + - * Detach from the screen and close your SSH session
        + To detach the screen terminal, press Ctrl+A, Ctrl+D. The terminal continues to run in the background. To reattach the terminal, run the following command:

sudo screen -r mcs

* + - * + If necessary, exit the screen terminal by pressing Ctrl+A, Ctrl+D.
        + To exit the SSH terminal, run the following command:

exit

* + - Task 4: Allow client traffic
      * Up to this point, the server has an external static IP address, but it cannot receive traffic because there is no firewall rule in place. Minecraft server uses TCP port 25565 by default. So you need to configure a firewall rule to allow these connections.
      * Create a firewall rule
        + Navigation > VPC network > Firewall.

Click Create firewall rule.

Specify the following and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name minecraft-rule

Target Specified target tags

Target tags minecraft-server

Source filter IP ranges

Source IP ranges 0.0.0.0/0

Protocols and ports Specified protocols and ports

For tcp, specify port 25565.

* + - * + Click Create. Users can now access your server from their Minecraft clients.
      * Verify server availability
        + In the left pane, click External IP addresses.
        + Locate and copy the External IP address for the mc-server VM.
        + Use the following website to test your Minecraft server: https://mcsrvstat.us/
        + If the above website is not working, you can use a different site or the Chrome extension:

https://dinnerbone.com/minecraft/tools/status/

* + - Task 5: Schedule regular backups
      * Backing up your application data is a common activity. In this case, you configure the system to back up Minecraft world data to Cloud Storage.
      * Create a Cloud Storage bucket
        + Navigation > Compute Engine > VM instances > For mc-server, click SSH.
        + Create a globally unique bucket name

gsutil mb gs://$YOUR\_BUCKET\_NAME-minecraft-backup

* + - * Create a backup script
        + In the mc-server SSH terminal, navigate to your home directory:

cd /home/minecraft

* + - * + To create the script, run the following command:

sudo nano /home/minecraft/backup.sh

* + - * + Copy and paste the following script into the file:

#!/bin/bash

screen -r mcs -X stuff '/save-all\n/save-off\n'

/usr/bin/gsutil cp -R ${BASH\_SOURCE%/\*}/world gs://${YOUR\_BUCKET\_NAME}-minecraft-backup/$(date "+%Y%m%d-%H%M%S")-world

screen -r mcs -X stuff '/save-on\n'

* + - * + Press Ctrl+O, ENTER to save the file, and press Ctrl+X to exit nano.
        + The script saves the current state of the server's world data and pauses the server's auto-save functionality. Next, it backs up the server's world data directory (world) and places its contents in a timestamped directory (<timestamp>-world) in the Cloud Storage bucket. After the script finishes backing up the data, it resumes auto-saving on the Minecraft server.
        + To make the script executable, run the following command:

sudo chmod 755 /home/minecraft/backup.sh

* + - * + Test the backup script and schedule a cron job
        + In the mc-server SSH terminal, run the backup script:

. /home/minecraft/backup.sh

* + - * + After the script finishes, return to the Cloud Console.
        + To verify that the backup file was written.
        + In the mc-server SSH terminal, open the cron table for editing:
        + sudo crontab -e
        + When you are prompted to select an editor, type the number corresponding to nano, and press ENTER.
        + At the bottom of the cron table, paste the following line:

0 \*/4 \* \* \* /home/minecraft/backup.sh

* + - * + That line instructs cron to run backups every 4 hours.
        + Press Ctrl+O, ENTER to save the cron table, and press Ctrl+X to exit nano.
        + This creates about 300 backups a month in Cloud Storage, so you will want to regularly delete them to avoid charges. Cloud Storage offers the Object Lifecycle Management feature to set a Time to Live (TTL) for objects, archive older versions of objects, or "downgrade" storage classes of objects to help manage costs.
    - Task 6: Server maintenance
      * To perform server maintenance, you need to shut down the server.
      * Connect via SSH to the server, stop it and shut down the VM
      * In the mc-server SSH terminal, run the following command:
        + sudo screen -r -X stuff '/stop\n'
      * Navigation > Compute Engine > VM instances > mc-server > Stop.
        + In the confirmation dialog, click Stop to confirm. You will be logged out of your SSH session.
        + To start up your instance again, visit the instance page and then click Start. To start the Minecraft server again, you can establish an SSH connection with the instance, remount your persistent disk, and start your Minecraft server in a new screen terminal, just as you did previously.
      * Automate server maintenance with startup and shutdown scripts
      * Instead of following the manual process to mount the persistent disk and launch the server application in a screen, you can use metadata scripts to create a startup script and a shutdown script to do this for you.
      * Click mc-server. Click Edit.
        + For Custom metadata, specify the following:
        + Key Value
        + startup-script-url https://storage.googleapis.com/cloud-training/archinfra/mcserver/startup.sh
        + shutdown-script-url https://storage.googleapis.com/cloud-training/archinfra/mcserver/shutdown.sh
      * You'll have to click Add item to add the shutdown-script-url. When you restart your instance, the startup script automatically mounts the Minecraft disk to the appropriate directory, starts your Minecraft server in a screen session, and detaches the session. When you stop the instance, the shutdown script shuts down your Minecraft server before the instance shuts down. It's a best practice to store these scripts in Cloud Storage.
    - Click Save.
  + Lab Review: Working with Virtual Machines 19 minutes - https://youtu.be/xiOUGdUdUKM
  + Quiz: Virtual Machines
    - Which statement is true of persistent disks?
      * Persistent disks are encrypted by default.
      * Persistent Disks are not physical disks, they are a virtual-networked service. Each persistent disk remains encrypted either with system-defined keys or with customer-supplied keys.
    - What are sustained use discounts?
      * Automatic discounts that you get for running specific Compute Engine resources for a significant portion of the billing month
      * That's correct! Sustained use discounts are automatic discounts that you get for running specific Compute Engine resources (vCPUs, memory, GPU devices) for a significant portion of the billing month. To take advantage of the full 30% discount, create your VM instances on the first day of the month, because discounts reset at the beginning of each month.
    - Which statement is true of Virtual Machine Instances in Compute Engine?
      * In Compute Engine, a VM is a networked service that simulates the features of a computer.
      * VMs in Compute Engine are a collection of networked services which includes persistent disks that are network-attached. In some cases the Google Cloud VM behaves unlike hardware or other kinds of virtual machines, for example, when a multi-tenant virtual CPU "bursts", using excess capacity beyond the VM spec.