# Elastic Google Cloud Infrastructure: Scaling and Automation

* Interconnecting Networks
  + Cloud VPN 4 minutes - https://youtu.be/E8eyORT9jh0
  + HA VPN
  + Virtual Private Networks (VPN) 34 minutes
    - Overview
      * establish VPN tunnels between two networks in separate regions such that a VM in one network can ping a VM in the other network over its internal IP address.
    - Objectives
      * Create VPN gateways in each network
      * Create VPN tunnels between the gateways
      * Verify VPN connectivity
    - Task 1: Explore the networks and instances
      * Two custom networks with VM instances have been configured for you. Both networks are VPC networks within a GC project. However, in a real-world application, one of these networks might be in a different GC project, on-premises, or in a different cloud.
      * Explore the networks
        + Verify that vpn-network-1 and vpn-network-2 have been created with subnets in separate regions.
        + In the Cloud Console, on the Navigation menu (Navigation menu), click VPC network > VPC networks.

Note the vpn-network-1 network and its subnet-a in us-central1.

Note the vpn-network-2 network and its subnet-b in europe-west1.

* + - * Explore the firewall rules
        + In the navigation pane, click Firewall.

Note the network-1-allow-ssh and network-1-allow-icmp rules for vpn-network-1.

Note the network-2-allow-ssh and network-2-allow-icmp rules for vpn-network-2.

* + - * + These firewall rules allow SSH and ICMP traffic from anywhere.
      * Explore the instances and their connectivity
        + Currently, the VPN connection between the two networks is not established. Explore the connectivity options between the instances in the networks.
        + Navigation menu > Compute Engine > VM instances.
        + From server-1, you should be able to ping the following IP addresses of server-2:
        + For server-1, click SSH to launch a terminal and connect.
        + To test connectivity to server-2's external IP address, run the following command, replacing server-2's external IP address with the value noted earlier:

ping -c 3 <Enter server-2's external IP address here>

ping -c 3 <Enter server-2's internal IP address here>

* + - * + External IP will work, Internal IP won’t ping because you don't have VPN connectivity yet.
        + Let's try the same from server-2.

ping -c 3 <Enter server-1's external IP address here>

ping -c 3 <Enter server-1's internal IP address here>

* + - * Why are we testing both server-1 to server-2 and server-2 to server-1?
        + For the purposes of this lab, the path from subnet-a to subnet-b is not the same as the path from subnet-b to subnet-a. You are using one tunnel to pass traffic in each direction. And if both tunnels are not established, you won't be able to ping the remote server on its internal IP address. The ping might reach the remote server, but the response can't be returned.
        + This makes it much easier to debug the lab during class. In practice, a single tunnel could be used with symmetric configuration. However, it is more common to have multiple tunnels or multiple gateways and VPNs for production work, because a single tunnel could be a single point of failure.
    - Task 2: Create the VPN gateways and tunnels
      * Establish private communication between the two VM instances by creating VPN gateways and tunnels between the two networks.
      * Reserve two static IP addresses
        + Reserve one static IP address for each VPN gateway.
        + Navigation menu > VPC network > External IP addresses > Reserve static address.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name vpn-1-static-ip

IP version IPv4

Region us-central1

* + - * + Click Reserve.
      * Repeat the same for vpn-2-static-ip.
        + Property Value (type value or select option as specified)
        + Name vpn-2-static-ip
        + IP version IPv4
        + Region europe-west1
      * Create the vpn-1 gateway and tunnel1to2
        + Navigation > Hybrid Connectivity > VPN > Create VPN Connection > select Classic VPN > Continue.
        + Specify the following in the VPN gateway section, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name vpn-1

Network vpn-network-1

Region us-central1

IP address vpn-1-static-ip

* + - * + Specify the following in the Tunnels section, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name tunnel1to2

Remote peer IP address [VPN-2-STATIC-IP]

IKE pre-shared key gcprocks

Routing options Route-based

Remote network IP ranges 10.1.3.0/24

* + - * + Click Create.
    - Create the vpn-2 gateway and tunnel2to1
      * + Property Value (type value or select option as specified)
        + Name vpn-2
        + Network vpn-network-2
        + Region europe-west1
        + IP address vpn-2-static-ip
        + Property Value (type value or select option as specified)
        + Name tunnel2to1
        + Remote peer IP address [VPN-1-STATIC-IP]
        + IKE pre-shared key gcprocks
        + Routing options Route-based
        + Remote network IP ranges 10.5.4.0/24
      * Wait for the VPN tunnels status to change to Established for both tunnels before continuing.
    - Task 3: Verify VPN connectivity
      * Verify server-1 to server-2 connectivity
        + Navigation >Compute Engine > VM instances > For server-1 > click SSH
        + To test connectivity to server-2's internal IP address, run the following command:

ping -c 3 <insert server-2's internal IP address here>

* + - * + For server-2 > SSH > to test connectivity to server-1's internal IP address

ping -c 3 <insert server-1's internal IP address here>

* + - * Remove the external IP addresses
        + Now that you verified VPN connectivity, you can remove the instances' external IP addresses. For demonstration purposes, just do this for the server-1 instance.
        + Navigation > Compute Engine > VM instances > server-1 > Stop > Edit

Instances need to be stopped before you can make changes.

For Network interfaces, click the Edit > Change External IP to None.

* + - * + Click Done. Click Save. Click Start.

Notice that External IP is set to None for the server-1 instance.

* + - * + server-2 SSH, verify that can still ping the server-1 instance's internal IP address.
    - Task 4: Review
      * configured a VPN connection between two networks with subnets in different regions. Then you verified the VPN connection by pinging VMs in different networks using their internal IP addresses.
      * configured the VPN gateways and tunnels using the Cloud Console. However, this approach obfuscated the creation of forwarding rules, which you explored with the command line button in the Console. This can help in troubleshooting a configuration if needed.
  + Lab Review: Virtual Private Networks (VPN) 11 minutes - https://youtu.be/mufuUHz54\_w
  + Cloud Interconnect and Peering 1 minute - https://youtu.be/AQ7QntDaM-I
  + Cloud Interconnect 4 minutes - https://youtu.be/HKWA5M2-Mc4
  + Peering 2 minutes - https://youtu.be/ZPN88W\_07EQ
  + Choosing a connection 2 minutes - https://youtu.be/7kyAFPS3UpA
  + Shared VPC and VPC Peering 5 minutes - https://youtu.be/LKhkyYvOsJ4
  + Quiz: Interconnecting Networks
    - Which Google Cloud Interconnect service requires a connection in a Google Cloud colocation facility and provides 10 Gbps per link?
      * Dedicated Interconnect
      * Correct! Dedicated Interconnect requires a connection in a Google Cloud colocation facility and provides 10 Gbps per link.
    - What is the purpose of Virtual Private Networking (VPN)?
      * To enable a secure communication method (a tunnel) to connect two trusted environments through an untrusted environment, such as the Internet.
    - Which of the following approaches to multi-project networking, uses a centralized network administration model?
      * Shared VPC
      * Correct! Shared VPC is a centralized approach to multi-project networking, because security and network policy occurs in a single designated VPC network.
    - If you cannot meet Google’s peering requirements, which network connection service should you choose to connect to Google Workspace and YouTube?
      * Carrier Peering
      * That’s correct! Carrier Peering allows you to connect to Google Workspace and YouTube without meeting Google’s peering requirements.
* Load Balancing and Autoscaling
  + Managed instance groups 3 minutes - https://youtu.be/VHO7LQLswbE
  + Autoscaling and health checks 3 minutes - https://youtu.be/cU8NREqaZ9A
  + Overview of HTTP(S) load balancing 4 minutes - https://youtu.be/hrXodtCIkoM
  + Example: HTTP load balancer 2 minutes - https://youtu.be/ioqrgrErdoU
  + HTTP(S) load balancing 3 minutes - https://youtu.be/xOxmTf44kaI
  + Configuring an HTTP Load Balancer with Autoscaling 1 hour - <https://www.cloudskillsboost.google/course_sessions/819373/labs/111515>
    - Overview
      * Google Cloud HTTP(S) load balancing is implemented at the edge of Google's network in Google's points of presence (POP) around the world. User traffic directed to an HTTP(S) 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 available capacity.
      * In this lab, you configure an HTTP load balancer as shown in the diagram below. Then, you stress test the load balancer to demonstrate global load balancing and autoscaling.
      * Diagram

        Description automatically generated
    - Objectives
      * Create a health check firewall rule
      * Create a NAT configuration using Cloud Router
      * Create a custom image for a web server
      * Create an instance template based on the custom image
      * Create two managed instance groups
      * Configure an HTTP load balancer with IPv4 and IPv6
      * Stress test an HTTP load balancer
    - Task 1. Configure a health check firewall rule
      * Health checks determine which instances of a load balancer can receive new connections. For HTTP load balancing, the health check probes to your load-balanced instances come from addresses in the ranges 130.211.0.0/22 and 35.191.0.0/16. Your firewall rules must allow these connections.
      * Create the health check rule
        + Create a firewall rule to allow health checks.
        + Navigation > VPC network > Firewall. Notice the existing ICMP, internal, RDP, and SSH firewall rules.

Each GC project starts with the default network and these firewall rules.

* + - * + 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 fw-allow-health-checks

Network default

Targets Specified target tags

Target tags allow-health-checks

Source filter IPv4 ranges

Source IPv4 ranges 130.211.0.0/22 and 35.191.0.0/16

Protocols and ports Specified protocols and ports

Make sure to include the /22 and /16 in the Source IP ranges.

* + - * + Select tcp and specify port 80. Click Create.
    - Task 2: Create a NAT configuration using Cloud Router
      * The GC VM backend instances that you setup in Task 3 will not be configured with external IP addresses.
      * Instead, you will setup the Cloud NAT service to allow these VM instances to send outbound traffic only through the Cloud NAT, and receive inbound traffic through the load balancer.
      * Create the Cloud Router instance
        + Navigation > click Network services > Cloud NAT > Click Get started.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Gateway name nat-config

Network default

Region us-central1

* + - * + Click Cloud Router, and select Create new router.

For Name, type nat-router-us-central1.

* + - * + Click Create.
        + In Create a NAT gateway, click Create.
        + Wait until the NAT Gateway Status changes to Running before moving onto the next task.
    - Task 3: Create a custom image for a web server
      * Create a custom web server image for the backend of the load balancer.
      * Create a VM
        + 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 webserver

Region us-central1

Zone us-central1-a

Boot disk Debian GNU/Linux 10 (buster)

* + - * + Under Boot disk, click Change > Show Advanced Configuration.
        + For Deletion rule, select Keep boot disk.
        + Click Select. Click Networking, Disks, Security, Management, Sole-tenancy.
        + Click Networking.

For Network tags, type allow-health-checks.

Under Network interfaces , click default.

Under External IP dropdown, select None.

* + - * + Click Done. Click Create.
      * Customize the VM
        + For webserver, click SSH to launch a terminal and connect.
        + If you see the Connection via Cloud Identity-Aware Proxy Failed popup, click Retry.
        + To install Apache2, run the following commands:

sudo apt-get update

sudo apt-get install -y apache2

* + - * + To start the Apache server, run the following command:

sudo service apache2 start

* + - * + To test the default page for the Apache2 server, run the following command:

curl localhost

* + - * + The default page for the Apache2 server should be displayed.
      * Set the Apache service to start at boot
        + The software installation was successful. However, when a new VM is created using this image, the freshly booted VM does not have the Apache web server running. Use the following command to set the Apache service to automatically start on boot. Then test it to make sure it works.
        + In the webserver SSH terminal, set the service to start on boot:

sudo update-rc.d apache2 enable

* + - * + select webserver, Click Reset.

Reset will stop and reboot the machine. It keeps the same IPs and the same persistent boot disk, but memory is wiped. Therefore, if the Apache service is available after the reset, the update-rc command was successful.

* + - * + Check the server by connecting via SSH:

sudo service apache2 status

* + - * + NOTE: If you see the Connection via Cloud Identity-Aware Proxy Failed popup, click Retry.
        + The result should show Started The Apache HTTP Server.
      * Prepare the disk to create a custom image
    - Verify that the boot disk will not be deleted when the instance is deleted.
      * On the VM instances page, click webserver to view the VM instance details.
      * Under Storage > Boot disk, verify that When deleting instance is set to Keep disk.
      * select webserver, Click Delete.
      * In the left pane, click Disks and verify that the webserver disk exists.
    - Create the custom image
      * In the left pane, click Images.
      * Click Create image.
      * Specify the following, and leave the remaining settings as their defaults:
        + Property Value (type value or select option as specified)
        + Name mywebserver
        + Source Disk
        + Source disk webserver
      * Click Create.
        + You have created a custom image that multiple identical webservers can be started from. At this point, you could delete the webserver disk.
      * The next step is to use that image to define an instance template that can be used in the managed instance groups.
    - Task 4. Configure an instance template and create instance groups
      * A managed instance group uses an instance template to create a group of identical instances. Use these to create the backends of the HTTP load balancer.
      * Configure the instance template
        + An instance template is an API resource that you can use to create VM instances and managed instance groups. Instance templates define the machine type, boot disk image, subnet, labels, and other instance properties.
        + Navigation > Compute Engine > Instance templates > Create instance template.
        + For Name, type mywebserver-template.
        + For Series, select N1. For Machine type, select f1-micro (1 vCPU).
        + For Boot disk, click Change. Click Custom images > Select a Project.
        + In Select a Project dialog, click Project\_ID.
        + For Image, Select mywebserver. Click Select.
        + Click Networking, Disks, Security, Management, Sole-tenancy.

Click Networking.

For Network tags, type allow-health-checks.

* + - * + Under Network interfaces , click default.
        + Under External IP dropdown, select None.
        + Click Done. Click Create.
      * Create the managed instance groups
        + Create a managed instance group in us-central1 and one in europe-west1.
        + Navigation > Compute Engine > Instance groups > Create Instance group.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name us-central1-mig

Instance template mywebserver-template

Location Multiple zones

Region us-central1

* + - * + Under Autoscaling, enter Minimum number of instances 1 and Maximum number of instances 2.
        + Under Autoscaling metrics, click on the CPU utilization.
        + Under Metric type, select HTTP load balancing utilization.
        + Enter Target HTTP load balancing utilization to 80. Click Done.
        + Click Cool down period and set to 60 seconds.

Managed instance groups offer autoscaling capabilities that allow you to automatically add or remove instances from a managed instance group based on increases or decreases in load. Autoscaling helps your applications gracefully handle increases in traffic and reduces cost when the need for resources is lower. You just define the autoscaling policy, and the autoscaler performs automatic scaling based on the measured load.

* + - * + Under Autohealing for Health check, select Create a health check.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (select option as specified)

Name http-health-check

Protocol TCP

Port 80

Managed instance group health checks proactively signal to delete and recreate instances that become unhealthy.

* + - * + Click Save.
        + For Initial delay, type 60. This is how long the Instance Group waits after initializing the boot-up of a VM before it tries a health check. You don't want to wait 5 minutes for this during the lab, so you set it to 1 minute.
        + Click Create.
        + NOTE: If a warning window will appear stating that There is no backend service attached to the instance group. Ignore this; you will configure the load balancer with a backend service in the next section of the lab.
      * Repeat the same procedure for europe-west1-mig in europe-west1:
        + Click Create Instance group.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name europe-west1-mig

Instance template mywebserver-template

Location Multiple zones

Region europe-west1

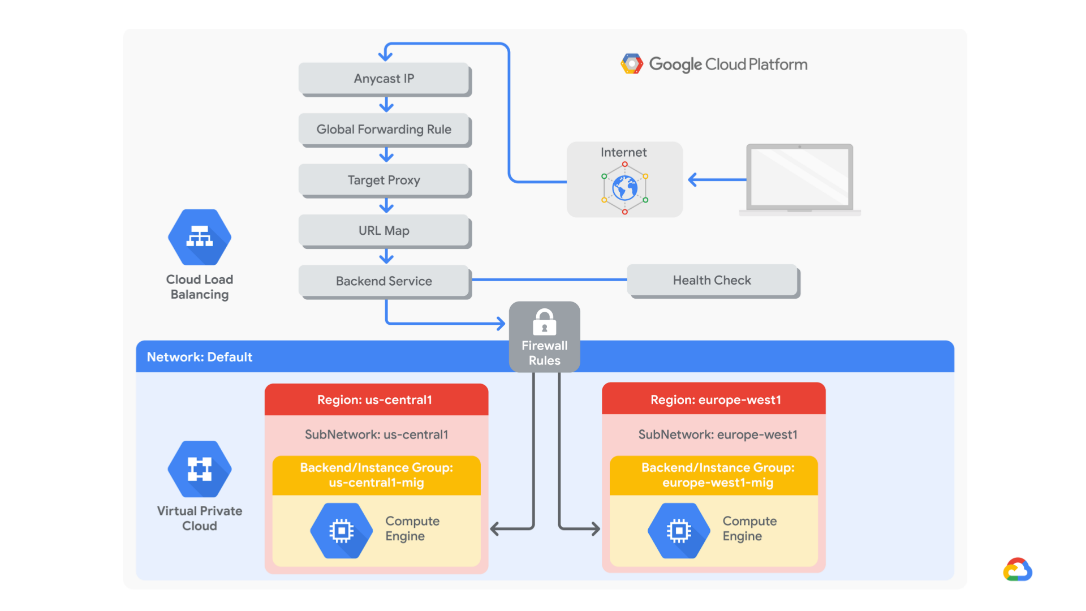
Autoscaling > Minimum number of instances 1

Autoscaling > Maximum number of instances 2

Autoscaling metrics > Metric Type HTTP load balancing utilization

Target HTTP load balancing utilization 80

Cool down period 60

* + - * + For Health check, select http-health-check (TCP).
        + For Initial delay, type 60.
        + Click Create.
      * Verify the backends
        + Verify that VM instances are being created in both regions.
        + On the Navigation menu, click Compute Engine > VM instances. Notice the instances that start with us-central1-mig and europe-west1-mig. These instances are part of the managed instance groups.
    - Task 5. Configure the HTTP load balancer
      * Configure the HTTP load balancer to balance traffic between the two backends (us-central1-mig in us-central1 and europe-west1-mig in europe-west1) as illustrated in the network diagram:
        + 
      * Start the configuration
        + On the Navigation menu, click Network Services > Load balancing.
        + Click Create load balancer.
        + Under HTTP(S) Load Balancing, click Start configuration.
        + Under Internet facing or internal only, select From Internet to my VMs or serverless services.
        + Under Advanced traffic management, select Classic HTTP(S) Load Balancer.
        + Click Continue. For Name, type http-lb.
      * Configure the backend
        + Backend services direct incoming traffic to one or more attached backends. Each backend is composed of an instance group and additional serving capacity metadata.
        + Click Backend configuration.
        + Click Backend services & backend buckets > Create a backend service.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (select option as specified)

Name http-backend

Backend type Instance group

Instance group us-central1-mig

Port numbers 80

Balancing mode Rate

Maximum RPS 50

Capacity 100

* + - * + This configuration means that the load balancer attempts to keep each instance of us-central1-mig at or below 50 requests per second (RPS).
        + Click Done. Click Add backend.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (select option as specified)

Instance group europe-west1-mig

Port numbers 80

Balancing mode Utilization

Maximum backend utilization 80

Capacity 100

* + - * + This configuration means that the load balancer attempts to keep each instance of europe-west1-mig at or below 80% CPU utilization.
        + Click Done.
        + For Health Check, select http-health-check.
        + Click check for the Enable logging checkbox.
        + Specify Sample rate as 1.
        + Click Create. Click OK.
      * Configure the frontend
        + The host and path rules determine how your traffic will be directed. For example, you could direct video traffic to one backend and direct static traffic to another backend. However, you are not configuring the host and path rules in this lab.
        + Click Frontend configuration.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Protocol HTTP

IP version IPv4

IP address Ephemeral

Port 80

* + - * + Click Done. Click Add Frontend IP and port.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Protocol HTTP

IP version IPv6

IP address Ephemeral

Port 80

* + - * + Click Done.
        + HTTP(S) load balancing supports both IPv4 and IPv6 addresses for client traffic. Client IPv6 requests are terminated at the global load balancing layer and then proxied over IPv4 to your backends.
      * Review and create the HTTP load balancer
        + Click Review and finalize.
        + Review the Backend services and Frontend.
        + Click Create. Wait for the load balancer to be created.
        + Click on the name of the load balancer (http-lb).
        + Note the IPv4 and IPv6 addresses of the load balancer for the next task. They will be referred to as [LB\_IP\_v4] and [LB\_IP\_v6], respectively.
    - Task 6. Stress test the HTTP load balancer
      * Now that you have created the HTTP load balancer for your backends, it is time to verify that traffic is forwarded to the backend service.
      * Access the HTTP load balancer
        + Cloud Shell, To check the status of the load balancer, run the following command, replace [LB\_IP\_v4] with the IPv4 address of the load balancer:

LB\_IP=[LB\_IP\_v4]

while [ -z "$RESULT" ] ;

do

echo "Waiting for Load Balancer";

sleep 5;

RESULT=$(curl -m1 -s $LB\_IP | grep Apache);

done

* + - * Once the load balancer is ready, the command will exit.
      * Open a new tab in your browser and navigate to http://[LB\_IP\_v4]. Make sure to replace [LB\_IP\_v4] with the IPv4 address of the load balancer.
    - Stress test the HTTP load balancer
      * Create a new VM to simulate a load on the HTTP load balancer. Then determine whether traffic is balanced across both backends when the load is high.
      * Navigation menu > 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 stress-test
        + Region us-west1
        + Zone us-west1-c
        + Series N1
        + Machine type f1-micro (1 vCPU)
      * Because us-west1 is closer to us-central1 than to europe-west1, traffic should be forwarded only to us-central1-mig (unless the load is too high).
      * For Boot Disk, click Change.
      * Click Custom images > Select a Project.
      * In Select a Project dialog, click Project\_ID.
      * For Image, select mywebserver. Click Select.
      * Click Create. Wait for the stress-test instance to be created.
      * For stress-test, click SSH to launch a terminal and connect.
      * To place a load on the load balancer, run the following command:
        + ab -n 500000 -c 1000 http://$LB\_IP/
      * Navigation > Network Services > Load balancing > Click http-lb > Monitoring.
        + Monitor the Frontend Location (Total inbound traffic) between North America and the two backends for a couple of minutes.

At first, traffic should just be directed to us-central1-mig, but as the RPS increases, traffic is also directed to europe-west1-mig. This demonstrates that by default traffic is forwarded to the closest backend, but if the load is very high, traffic can be distributed across the backends.

Diagram

Description automatically generated

* + - * Navigation > Compute Engine > Instance groups.
        + Click on us-central1-mig to open the instance group page.
        + Click Monitoring to monitor the number of instances and LB capacity.
        + Repeat the same for the europe-west1-mig instance group.
      * Depending on the load, you might see the backends scale to accommodate the load.
  + Lab Review: Configuring an HTTP Load Balancer with Autoscaling 25 minutes - https://youtu.be/gjw1eaRn9U0
  + SSL proxy load balancing 2 minutes -https://youtu.be/mU8ZG8p88zU
  + TCP proxy load balancing 1 minute - https://youtu.be/l4eYxos5wus
  + Network load balancing 2 minutes - https://youtu.be/blkuXqvlKC8
  + Internal load balancing 3 minutes - https://youtu.be/P9cGAsEZ-Uo
  + Configuring an Internal Load Balancer 50 minutes - <https://www.cloudskillsboost.google/course_sessions/819373/labs/111522>
    - Overview
      * Google Cloud offers Internal Load Balancing for your TCP/UDP-based traffic. Internal Load Balancing enables you to run and scale your services behind a private load balancing IP address that is accessible only to your internal virtual machine instances.
      * create two managed instance groups in the same region. Then you configure and test an internal load balancer with the instances groups as the backends, as shown in this network diagram:
    - Objectives
      * Create internal traffic and health check firewall rules
      * Create a NAT configuration using Cloud Router
      * Configure two instance templates
      * Create two managed instance groups
      * Configure and test an internal load balancer
    - Task 1. Configure internal traffic and health check firewall rules.
      * Configure firewall rules to allow internal traffic connectivity from sources in the 10.10.0.0/16 range. This rule allows incoming traffic from any client located in the subnet.
      * Health checks determine which instances of a load balancer can receive new connections. For HTTP load balancing, the health check probes to your load-balanced instances come from addresses in the ranges 130.211.0.0/22 and 35.191.0.0/16. Your firewall rules must allow these connections.
      * Explore the my-internal-app network
        + The network my-internal-app with subnet-a and subnet-b and firewall rules for RDP, SSH, and ICMP traffic have been configured for you.
        + Navigation > VPC network > VPC networks. Notice the my-internal-app network with its subnets: subnet-a and subnet-b.
        + Each Google Cloud project starts with the default network. In addition, the my-internal-app network has been created for you as part of your network diagram.
        + You will create the managed instance groups in subnet-a and subnet-b. Both subnets are in the us-central1 region because an internal load balancer is a regional service. The managed instance groups will be in different zones, making your service immune to zonal failures.
      * Create the firewall rule to allow traffic from any sources in the 10.10.0.0/16 range
      * Create a firewall rule to allow traffic in the 10.10.0.0/16 subnet.
      * Navigation > VPC network > Firewall > Click Create Firewall Rule.
        + Notice the app-allow-icmp and app-allow-ssh-rdp firewall rules, have been created for you.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name fw-allow-lb-access

Network my-internal-app

Targets Specified target tags

Target tags backend-service

Source filter IP ranges

Source IP ranges 10.10.0.0/16

Protocols and ports Allow all

* + - * + Make sure to include the /16 in the Source IP ranges.
        + Click Create.
      * Create the health check rule
      * Create a firewall rule to allow health checks.
        + Navigation > 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 fw-allow-health-checks

Network my-internal-app

Targets Specified target tags

Target tags backend-service

Source filter IP Ranges

Source IP ranges 130.211.0.0/22 and 35.191.0.0/16

Protocols and ports Specified protocols and ports

* + - * + Make sure to include the /22 and /16 in the Source IP ranges.
        + For tcp, specify port 80. Click Create.
    - Task 2: Create a NAT configuration using Cloud Router
      * The Google Cloud VM backend instances that you setup in Task 3 will not be configured with external IP addresses.
      * Instead, you will setup the Cloud NAT service to allow these VM instances to send outbound traffic only through the Cloud NAT, and receive inbound traffic through the load balancer.
      * Create the Cloud Router instance
        + Navigation > click Network services > Cloud NAT > Click Get started.

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

Property Value (type value or select option as specified)

Gateway name nat-config

VPC network my-internal-app

Region us-central1

* + - * + Cloud Router > Create new router. For Name nat-router-us-central1.> Create.
        + In Create a NAT gateway, click Create.
    - Task 3. Configure instance templates and create instance groups
      * A managed instance group uses an instance template to create a group of identical instances. Use these to create the backends of the internal load balancer.
      * Configure the instance templates
        + An instance template is an API resource that you can use to create VM instances and managed instance groups. Instance templates define the machine type, boot disk image, subnet, labels, and other instance properties. Create an instance template for both subnets of the my-internal-app network.
        + Navigation > Compute Engine > Instance templates > Create instance template.
        + For Name, type instance-template-1
        + Under Machine configuration, For Series, Select N1.
        + Machine type f1-micro(1 vCPU).
        + Click Management, security, disks, networking, sole … > Management.
        + Under Metadata, specify the following:

Key Value

startup-script-url gs://cloud-training/gcpnet/ilb/startup.sh

* + - * + The startup-script-url specifies a script that is executed when instances are started. This script installs Apache and changes the welcome page to include the client IP and the name, region, and zone of the VM instance. You can explore this script here.
        + Click Networking.
        + For Network interfaces, specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Network my-internal-app

Subnet subnet-a

Network tags backend-service

External IP None

* + - * + The network tag backend-service ensures that the firewall rule to allow traffic from any sources in the 10.10.0.0/16 subnet and the Health Check firewall rule applies to these instances.
        + Click Create. Wait for the instance template to be created.
        + Create another instance template for subnet-b by copying instance-template-1:
        + Select the instance-template-1 and click Copy.
        + Click Management, security, disks, networking, sole tenancy. > Networking.
        + For Network interfaces, select subnet-b as the Subnet.
        + Click Create.
        + Create the managed instance groups
      * Create a managed instance group in subnet-a (us-central1-a) and subnet-b (us-central1-b).
        + Navigation > Compute Engine > Instance groups > Create Instance group.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name instance-group-1

Location Single zone

Region us-central1

Zone us-central1-a

Instance template instance-template-1

Autoscaling > metrics type (Click the pencil edit icon) CPU utilization

Target CPU utilization 80, click Done.

Cool-down period 45

Minimum number of instances 1

Maximum number of instances 5

* + - * + Managed instance groups offer autoscaling capabilities that allow you to automatically add or remove instances from a managed instance group based on increases or decreases in load. Autoscaling helps your applications gracefully handle increases in traffic and reduces cost when the need for resources is lower. Just define the autoscaling policy, and the autoscaler performs automatic scaling based on the measured load.
        + Click Create.
        + Repeat the same procedure for instance-group-2 in us-central1-b:
        + Click Create Instance group.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name instance-group-2

Location Single zone

Region us-central1

Zone us-central1-b

Instance template instance-template-2

Autoscaling > metric type (Click the pencil edit icon) CPU utilization

Target CPU utilization 80, click Done.

Cool-down period 45

Minimum number of instances 1

Maximum number of instances 5

* + - * + Click Create.
      * Verify the backends
        + Verify that VM instances are being created in both subnets and create a utility VM to access the backends' HTTP sites.
        + Navigation > Compute Engine > VM instances. Notice two instances that start with instance-group-1 and instance-group-2.
        + These instances are in separate zones, and their internal IP addresses are part of the subnet-a and subnet-b CIDR blocks.
        + Click Create Instance.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Name utility-vm

Region us-central1

Zone us-central1-f

Series N1

Machine type f1-micro (1 vCPU)

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 my-internal-app

Subnetwork subnet-a

Primary internal IP Ephemeral (Custom)

Custom ephemeral IP address 10.10.20.50

External IP None

* + - * + Click Done. Click Create.
        + Note that the internal IP addresses for the backends are 10.10.20.2 and 10.10.30.2.
        + If these IP addresses are different, replace them in the two curl commands below.
        + For utility-vm, click SSH to launch a terminal and connect. If you see the Connection via Cloud Identity-Aware Proxy Failed popup, click Retry.
        + To verify the welcome page for instance-group-1-xxxx, run the following cmd:

curl 10.10.20.2

* + - * + The output should look like this (do not copy; this is example output):

<h1>Internal Load Balancing Lab</h1><h2>Client IP</h2>Your IP address : 10.10.20.50<h2>Hostname</h2>Server Hostname:

instance-group-1-1zn8<h2>Server Location</h2>Region and Zone: us-central1-a

* + - * + To verify the welcome page for instance-group-2-xxxx, run the following cmd:

curl 10.10.30.2

* + - * + The output should look like this (do not copy; this is example output):

<h1>Internal Load Balancing Lab</h1><h2>Client IP</h2>Your IP address : 10.10.20.50<h2>Hostname</h2>Server Hostname:

instance-group-2-q5wp<h2>Server Location</h2>Region and Zone: us-central1-b

* + - * + Which of these fields identify the location of the backend?

Server Location

Server Hostname

* + - * + This will be useful when verifying that the internal load balancer sends traffic to both backends.
    - Task 4. Configure the internal load balancer
      * Configure the internal load balancer to balance traffic between the two backends (instance-group-1 in us-central1-a and instance-group-2 in us-central1-b), as illustrated in the network diagram:
        + Timeline

          Description automatically generated with medium confidence
      * Start the configuration
        + Navigation > click Network Services > Load balancing > Create load balancer.
        + Under TCP Load Balancing, click Start configuration.
        + For Internet facing or internal only, select Only between my VMs.

Choosing Only between my VMs makes this load balancer internal. This choice requires the backends to be in a single region (us-central1) and does not allow offloading TCP processing to the load balancer.

* + - * + Click Continue. For Name, type my-ilb.
      * Configure the regional backend service
        + The backend service monitors instance groups and prevents them from exceeding configured usage.
        + Click Backend configuration.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (select option as specified)

Region us-central1

Network my-internal-app

Instance group instance-group-1 (us-central1-a)

* + - * + Click Done. Click Add backend.
        + For Instance group, select instance-group-2 (us-central1-b). Click Done.
        + For Health Check, select Create a health check.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (select option as specified)

Name my-ilb-health-check

Protocol TCP

Port 80

Check interval 10 sec

Timeout 5 sec

Healthy threshold 2

Unhealthy threshold 3

* + - * + Health checks determine which instances can receive new connections. This HTTP health check polls instances every 10 seconds, waits up to 5 seconds for a response, and treats 2 successful or 3 failed attempts as healthy threshold or unhealthy threshold, respectively.
        + Click Save and continue.
        + Verify that there is a blue check mark next to Backend configuration in the Cloud Console. If there isn't, double-check that you have completed all the steps above.
      * Configure the frontend
        + The frontend forwards traffic to the backend.
        + Click Frontend configuration.
        + Specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Subnetwork subnet-b

Internal IP > IP address Reserve static internal IP address

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

Property Value (type value or select option as specified)

Name my-ilb-ip

Static IP address Let me choose

Custom IP address 10.10.30.5

* + - * + Click Reserve. For Ports, type 80. Click Done.
      * Review and create the internal load balancer
        + Click Review and finalize.
        + Review the Backend and Frontend.
        + Click Create. Wait for the load balancer to be created before moving to the next task.
    - Task 5. Test the internal load balancer
      * Verify that the my-ilb IP address forwards traffic to instance-group-1 in us-central1-a and instance-group-2 in us-central1-b.
      * Access the internal load balancer
        + On the Navigation menu, click Compute Engine > VM instances.
        + For utility-vm, click SSH to launch a terminal and connect.
        + To verify that the internal load balancer forwards traffic, run the following command:

curl 10.10.30.5

* + - * + The output should look like this (do not copy; this is example output):

<h1>Internal Load Balancing Lab</h1><h2>Client IP</h2>Your IP address : 10.10.20.50<h2>Hostname</h2>Server Hostname:

instance-group-2-1zn8<h2>Server Location</h2>Region and Zone: us-central1-b

* + - * + As expected, traffic is forwarded from the internal load balancer (10.10.30.5) to the backend.
        + Run the same command a couple of times:

curl 10.10.30.5

curl 10.10.30.5

curl 10.10.30.5

curl 10.10.30.5

curl 10.10.30.5

curl 10.10.30.5

curl 10.10.30.5

curl 10.10.30.5

curl 10.10.30.5

curl 10.10.30.5

* + - * + You should be able to see responses from instance-group-1 in us-central1-a and instance-group-2 in us-central1-b. If not, run the command again.
  + Lab Review: Configuring an Internal Load Balancer 13 minutes - https://youtu.be/a890xluKfwI
  + Choosing a load balancer 3 minutes - https://youtu.be/7pMJfko9zLM
  + Quiz: Load Balancing and Autoscaling
    - Which of the following is not a Google Cloud load balancing service?
      * Hardware-defined load balancing
      * Correct! Cloud Load Balancing is a fully distributed, software-defined, managed service for all your traffic. It is not an instance or device based solution, so you won’t be locked into physical load balancing infrastructure.
    - Which three Google Cloud load balancing services support IPv6 clients?
      * TCP proxy load balancing
        + Correct! TCP proxy load balancing supports IPv6 clients in addition to IPv4 clients.
      * SSL proxy load balancing
        + Correct! SSL proxy load balancing supports IPv6 clients in addition to IPv4 clients.
      * HTTP(S) load balancing
        + Correct! HTTP(S) load balancing supports IPv6 clients in addition to IPv4 clients.
    - Which of the following are applicable autoscaling policies for managed instance groups?
      * Monitoring metrics
      * Load balancing capacity
      * CPU utilization
      * Queue-based workload
* Infrastructure Automation
  + Terraform 6 minutes - https://youtu.be/YeiqdqV6Z9w
  + Automating the Deployment of Infrastructure Using Terraform 30 minutes - <https://www.cloudskillsboost.google/course_sessions/819373/labs/111530>
    - Overview
      * Terraform enables you to safely and predictably create, change, and improve infrastructure. It is an open-source tool that codifies APIs into declarative configuration files that can be shared among team members, treated as code, edited, reviewed, and versioned.
      * In this lab, you create a Terraform configuration with a module to automate the deployment of Google Cloud infrastructure. Specifically, you deploy one auto mode network with a firewall rule and two VM instances, as shown in this diagram:
        + Graphical user interface

          Description automatically generated with medium confidence
    - Objectives
      * Create a configuration for an auto mode network
      * Create a configuration for a firewall rule
      * Create a module for VM instances
      * Create and deploy a configuration
      * Verify the deployment of a configuration
    - Task 1. Set up Terraform and Cloud Shell
      * Configure your Cloud Shell environment to use Terraform.
      * Install Terraform
        + Terraform is now integrated into Cloud Shell. Verify which version is installed.
        + In the Cloud Console, click Activate Cloud Shell (Cloud Shell).
        + To confirm that Terraform is installed, run the following command:

terraform --version

* + - * + To create a directory for your Terraform configuration:

mkdir tfinfra

* + - * + In Cloud Shell, click Open Editor (Cloud Shell Editor).
      * Initialize Terraform
        + Terraform uses a plugin-based architecture to support the numerous infrastructure and service providers available. Each "provider" is its own encapsulated binary distributed separately from Terraform itself. Initialize Terraform by setting Google as the provider.
        + To create a new file, provider.tf, and then open it.

provider "google" {}

* + - * + To initialize Terraform, run the following command:

cd tfinfra

terraform init

* + - Task 2. Create mynetwork and its resources
      * Create the auto mode network mynetwork along with its firewall rule and two VM instances (mynet\_us\_vm and mynet\_eu\_vm).
      * Configure mynetwork
    - Create a new configuration, and define mynetwork.
      * To create a new file mynetwork.tf, and then open it.
      * Copy the following base code into mynetwork.tf:

# Create the mynetwork network

resource [RESOURCE\_TYPE] "mynetwork" {

name = [RESOURCE\_NAME]

#RESOURCE properties go here

}

* + - * The name field allows you to name the resource, and the type field allows you to specify the Google Cloud resource that you want to create. You can also define properties, but these are optional for some resources.
      * In mynetwork.tf, replace [RESOURCE\_TYPE] with "google\_compute\_network"
      * The google\_compute\_network resource is a VPC network.
      * In mynetwork.tf, replace [RESOURCE\_NAME] with "mynetwork" (with the quotes).
      * Add the following property to mynetwork.tf:
        + auto\_create\_subnetworks = "true"
      * By definition, an auto mode network automatically creates a subnetwork in each region. Therefore, you are setting auto\_create\_subnetworks to true.
    - Configure the firewall rule
      * Define a firewall rule to allow HTTP, SSH, RDP, and ICMP traffic on mynetwork.
      * Add the following base code to mynetwork.tf:

# Add a firewall rule to allow HTTP, SSH, RDP and ICMP traffic on mynetwork

resource [RESOURCE\_TYPE] "mynetwork-allow-http-ssh-rdp-icmp" {

name = [RESOURCE\_NAME]

#RESOURCE properties go here

}

* + - * In mynetwork.tf, replace [RESOURCE\_TYPE] with "google\_compute\_firewall"
      * The google\_compute\_firewall resource is a firewall rule.
      * In mynetwork.tf, replace [RESOURCE\_NAME] with "mynetwork-allow-http-ssh-rdp-icmp"
      * Add the following property to mynetwork.tf:
        + network = google\_compute\_network.mynetwork.self\_link
      * Because this firewall rule depends on its network, you are using the google\_compute\_network.mynetwork.self\_link reference to instruct Terraform to resolve these resources in a dependent order. In this case, the network is created before the firewall rule.
      * Add the following properties to mynetwork.tf:

allow {

protocol = "tcp"

ports = ["22", "80", "3389"]

}

allow {

protocol = "icmp"

}

source\_ranges = ["0.0.0.0/0"]

* + - * The list of allow rules specifies which protocols and ports are permitted.
      * Verify that your additions to mynetwork.tf look like this:

# Add a firewall rule to allow HTTP, SSH, RDP, and ICMP traffic on mynetwork

resource "google\_compute\_firewall" "mynetwork-allow-http-ssh-rdp-icmp" {

name = "mynetwork-allow-http-ssh-rdp-icmp"

network = google\_compute\_network.mynetwork.self\_link

allow {

protocol = "tcp"

ports = ["22", "80", "3389"]

}

allow {

protocol = "icmp"

}

source\_ranges = ["0.0.0.0/0"]

}

* + - Configure the VM instance
      * Define the VM instances by creating a VM instance module. A module is a reusable configuration inside a folder. You will use this module for both VM instances of this lab.
      * To create a new folder inside tfinfra, select the tfinfra folder, and then click File > New Folder.
      * Name the new folder instance/main.tf
      * Copy the following base code into main.tf:
        + resource [RESOURCE\_TYPE] "vm\_instance" {
        + name = [RESOURCE\_NAME]
        + #RESOURCE properties go here
        + }
      * In main.tf, replace [RESOURCE\_TYPE] with "google\_compute\_instance" (with the quotes).
      * The google\_compute\_instance resource is a Compute Engine instance. For more information about this specific resource, see the Terraform documentation.
      * In main.tf, replace [RESOURCE\_NAME] with "${var.instance\_name}" (with the quotes).
      * Because you will be using this module for both VM instances, you are defining the instance name as an input variable. This allows you to control the name of the variable from mynetwork.tf. For more information about input variables, see this guide.
      * Add the following properties to main.tf:
        + zone = "${var.instance\_zone}"
        + machine\_type = "${var.instance\_type}"
      * These properties define the zone and machine type of the instance as input variables.
      * Add the following properties to main.tf:
        + boot\_disk {
        + initialize\_params {
        + image = "debian-cloud/debian-9"
        + }
        + }
      * This property defines the boot disk to use the Debian 9 OS image. Because both VM instances will use the same image, you can hard-code this property in the module.
      * Add the following properties to main.tf:
        + network\_interface {
        + network = "${var.instance\_network}"
        + access\_config {
        + # Allocate a one-to-one NAT IP to the instance
        + }
        + }
      * This property defines the network interface by providing the network name as an input variable and the access configuration. Leaving the access configuration empty results in an ephemeral external IP address (required in this lab). To create instances with only an internal IP address, remove the access\_config section. For more information, see the Terraform documentation.
      * Define the 4 input variables at the top of main.tf, and verify that main.tf looks like this, including brackets {}:
        + variable "instance\_name" {}
        + variable "instance\_zone" {}
        + variable "instance\_type" {
        + default = "n1-standard-1"
        + }
        + variable "instance\_network" {}
        + resource "google\_compute\_instance" "vm\_instance" {
        + name = "${var.instance\_name}"
        + zone = "${var.instance\_zone}"
        + machine\_type = "${var.instance\_type}"
        + boot\_disk {
        + initialize\_params {
        + image = "debian-cloud/debian-9"
        + }
        + }
        + network\_interface {
        + network = "${var.instance\_network}"
        + access\_config {
        + # Allocate a one-to-one NAT IP to the instance
        + }
        + }
        + }
      * By giving instance\_type a default value, you make the variable optional. The instance\_name, instance\_zone, and instance\_network are required, and you will define them in mynetwork.tf.
      * Add the following VM instances to mynetwork.tf:
        + # Create the mynet-us-vm instance
        + module "mynet-us-vm" {
        + source = "./instance"
        + instance\_name = "mynet-us-vm"
        + instance\_zone = "us-central1-a"
        + instance\_network = google\_compute\_network.mynetwork.self\_link
        + }
        + # Create the mynet-eu-vm" instance
        + module "mynet-eu-vm" {
        + source = "./instance"
        + instance\_name = "mynet-eu-vm"
        + instance\_zone = "europe-west1-d"
        + instance\_network = google\_compute\_network.mynetwork.self\_link
        + }
      * These resources are leveraging the module in the instance folder and provide the name, zone, and network as inputs. Because these instances depend on a VPC network, you are using the google\_compute\_network.mynetwork.self\_link reference to instruct Terraform to resolve these resources in a dependent order. In this case, the network is created before the instance.
      * The benefit of writing a Terraform module is that it can be reused across many configurations. Instead of writing your own module, you can also leverage existing modules from the Terraform Module registry.
    - Create mynetwork and its resources
      * It's time to apply the mynetwork configuration.
      * To rewrite the Terraform configuration files to a canonical format and style, run the following command:
        + terraform fmt
      * The output should look like this (do not copy; this is example output):
        + mynetwork.tf
      * To initialize Terraform, run the following command:
        + terraform init
      * To create an execution plan, run the following command:
        + terraform plan
      * Terraform determined that the following 4 resources need to be added:
        + Name Description
        + mynetwork VPC network
        + mynetwork-allow-http-ssh-rdp-icmp Firewall rule to allow HTTP, SSH, RDP and ICMP
        + mynet-us-vm VM instance in us-central1-a
        + mynet-eu-vm VM instance in europe-west1-d
      * To apply the desired changes, run the following command:
        + terraform apply
      * To confirm the planned actions, type:
        + yes
    - Task 3. Verify your deployment
      * Navigation menu (Navigation menu), click VPC network > VPC networks.
        + View the mynetwork VPC network with a subnetwork in every region.
      * On the Navigation menu, click VPC network > Firewall.
        + Sort the firewall rules by Network.
        + View the mynetwork-allow-http-ssh-rdp-icmp firewall rule for mynetwork.
      * Navigation menu (Navigation menu), click Compute Engine > VM instances.
        + View the mynet-us-vm and mynet-eu-vm instances.
        + Note the internal IP address for mynet-eu-vm.
        + For mynet-us-vm, click SSH to launch a terminal and connect.
      * To test connectivity to mynet-eu-vm's internal IP address, run the following command in the SSH terminal (replacing mynet-eu-vm's internal IP address with the value noted earlier):
        + ping -c 3 <Enter mynet-eu-vm's internal IP here>
  + Lab Review: Automating the Infrastructure of networks using Terraform 14 minutes - https://youtu.be/UujS4Wv1JW4
  + Google Cloud Marketplace 1 minute - https://youtu.be/sWZHI3rzXkw
  + Demo: Launch on Google Cloud Marketplace 4 minutes
  + Quiz: Infrastructure Automation
    - What does Google Cloud Marketplace offer?
      * Production-grade solutions from third-party vendors who have already created their own deployment configurations based on Terraform
    - What’s the benefit of writing templates for your Terraform configuration?
      * Allows you to abstract part of your configuration into individual building blocks that you can reuse
      * Correct! After you create a template, you can reuse them across deployments as necessary. Similarly, if you find yourself rewriting configurations that share very similar properties, you can abstract the shared parts into templates.
* Managed Services
  + BigQuery 1 minute - https://youtu.be/\_SKTrcxUD5s
  + Dataflow 1 minute - https://youtu.be/\_FmXPBGYMzc
  + Dataprep 1 minute - https://youtu.be/4YuQnvuBgJc
  + Dataproc 2 minutes - https://youtu.be/7q8jjBzkyc8
  + Demo: Dataproc 6 minutes - https://youtu.be/ENeQmfpnU6c
  + Quiz: Managed Services
    - Which of the following is a feature of Dataproc?
      * It typically takes less than 90 seconds to start a cluster.
      * That's correct! Fast to start a cluster.
    - How are Managed Services useful?
      * Managed Services may be an alternative to creating and managing infrastructure solutions.
      * That's correct! Managed Services are presented as a possible alternative to building your own infrastructure data processing solution.
  + What’s Next? Get Certified