**Install a NGINX web server**

Once SSH'ed, get root access using sudo:

sudo su -

As the root user, update your OS:

apt-get update

Install NGINX:

apt-get install nginx -y

Check that NGINX is running:

ps auwx | grep nginx

**Create a new instance with gcloud**

gcloud compute instances create gcelab2 --machine-type n1-standard-2 --zone us-central1-c

gcloud compute ssh gcelab2 --zone us-central1-c

Do you want to continue? (Y/n) Y

**Create VM in cloud shell**

gcloud config set compute/zone us-central1-b

gcloud compute instances create "my-vm-2" \

--machine-type "n1-standard-1" \

--image-project "debian-cloud" \

--image "debian-9-stretch-v20190213" \

--subnet "default"

**install the Nginx web server**

sudo apt-get install nginx-light -y

sudo nano /var/www/html/index.nginx-debian.html

**Create the Cloud SQL instance**

Click Create instance.

For Choose a database engine, select MySQL.

For Instance ID, type blog-db, and for Root password type a password of your choice.

Click Create

From the SQL instances details page, copy the Public IP address for your SQL instance to a text editor for use later in this lab.

Click on Users menu on the left-hand side, and then click ADD USER ACCOUNT.

For User name, type blogdbuser

For Password, type a password of your choice. Make a note of it.

Click Create to create the user account in the database.

Click the Connections tab, and then click Add network.

**Getting Started with GKE**

Start a Kubernetes Engine cluster

export MY\_ZONE=us-central1-a

gcloud container clusters create webfrontend --zone $MY\_ZONE --num-nodes 2

Run and deploy a container

kubectl create deploy nginx --image=nginx:1.17.10

kubectl get pods

kubectl expose deployment nginx --port 80 --type LoadBalancer

kubectl get services

kubectl scale deployment nginx --replicas 3

**Getting Started with App Engine**

git clone https://github.com/GoogleCloudPlatform/python-docs-samples

Navigate to the source directory:

cd python-docs-samples/appengine/standard\_python3/hello\_world

sudo apt-get update

sudo apt-get install virtualenv

virtualenv -p python3 venv

source venv/bin/activate

pip install -r requirements.txt

python main.py #run the app

Deploy and run Hello World on App Engine

cd ~/python-docs-samples/appengine/standard\_python3/hello\_world

gcloud app deploy

gcloud app browse

**Getting Started with Deployment Manager**

export MY\_ZONE=us-central1-a

export PROJECT\_ID=

gsutil cp gs://cloud-training/gcpfcoreinfra/mydeploy.yaml mydeploy.yaml

sed -i -e "s/ZONE/$MY\_ZONE/" mydeploy.yaml

cat mydeploy.yaml

gcloud deployment-manager deployments create my-first-depl --config mydeploy.yaml

nano mydeploy.yaml

value: "apt-get update; apt-get install nginx-light -y"

gcloud deployment-manager deployments update my-first-depl --config mydeploy.yaml

**VPC Networking**

Create an auto mode network

On the Navigation menu (Navigation menu), click VPC network > VPC networks.

Click 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.

Create custom mode networks

In the Cloud Console, on the Navigation menu (Navigation menu), click VPC network > VPC networks.

Click 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

Create the privatenet network

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

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

Create the firewall rules for managementnet

In the Cloud Console, on the Navigation menu (Navigation menu), click 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 managementnet-allow-icmp-ssh-rdp

Network managementnet

Targets All instances in the network

Source filter IP Ranges

Source IP ranges 0.0.0.0/0

Protocols and ports Specified protocols and ports

**Implement Private Google Access and Cloud NAT**

Create a VPC network and firewall rules

In the Cloud Console, on the Navigation menu (Navigation menu), click VPC network > VPC networks.

Click Create VPC network.

For Name, type 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

In the left pane, click 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 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.

Create the VM instance with no public IP address

Click Networking.

For Network interfaces, click the pencil icon to edit.

SSH to vm-internal to test the IAP tunnel

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

Enable Private Google Access

In the Cloud Console, on the Navigation menu (Navigation menu), click Storage > Browser.

Click Create bucket.

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

Property Value (type value or select option as specified)

Name Enter a globally unique name

Location type Multi-region

Configure a Cloud NAT gateway

In the Cloud Console, on the Navigation menu (Navigation menu), click Network services > Cloud NAT.

Click Get started to configure a NAT gateway.

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, select Create new router.

For Name, type nat-router

**Working with Virtual Machines**

In the Cloud Console, on the Navigation menu (Navigation menu), click Compute Engine > VM instances.

Click Create.

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

Click Disks. You will add a disk to be used for game storage.

Click Add new disk.

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 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 Create.

Create a directory and format and mount the disk

For mc-server, click SSH to open a terminal and connect.

sudo mkdir -p /home/minecraft

sudo mkfs.ext4 -F -E lazy\_itable\_init=0,\

lazy\_journal\_init=0,discard \

/dev/disk/by-id/google-minecraft-disk

sudo mount -o discard,defaults /dev/disk/by-id/google-minecraft-disk /home/minecraft

Install and run the application

sudo apt-get update

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

cd /home/minecraft

sudo apt-get install wget

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

Initialize the Minecraft server

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

sudo ls -l

sudo nano eula.txt

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

sudo apt-get install -y screen

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

sudo screen -r mcs

Allow client traffic

In the Cloud Console, on the Navigation menu (Navigation menu), click 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.

**Implementing Cloud SQL**

On the Navigation menu (Navigation menu), click SQL.

Click Create instance.

Click Choose MySQL.

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

Property Value

Instance ID wordpress-db

Root password type a password

Region us-central1

Zone Any

Database Version MySQL 5.7

Expand Show configuration options.

Expand the Connectivity section.

Select Private IP.

In the dialog box, click Enable API, click Allocate and connect, and then click Close. This enables Private Services Access and attaches a Private IP address to your Cloud SQL server.

Configure a proxy on a virtual machine

On the Navigation menu (Navigation menu) click Compute Engine.

Click SSH next to wordpress-europe-proxy.

Download the Cloud SQL Proxy and make it executable:

wget https://dl.google.com/cloudsql/cloud\_sql\_proxy.linux.amd64 -O cloud\_sql\_proxy && chmod +x cloud\_sql\_proxy

On the Navigation menu (Navigation menu), click SQL.

Click on the wordpress-db instance and wait for a green checkmark next to its name, which indicates that it is operational (this could take a couple of minutes).

Note the Instance connection name; it will be used later and referred to as [SQL\_CONNECTION\_NAME].

In addition, for the application to work, you need to create a table. Click Databases.

Click Create database, type wordpress, which is the name the application expects, and then click Create.

Return to the SSH window and save the connection name in an environment variable, replacing [SQL\_CONNECTION\_NAME] with the unique name you copied in a previous step.

export SQL\_CONNECTION=[SQL\_CONNECTION\_NAME]

To activate the proxy connection to your Cloud SQL database and send the process to the background, run the following command:

./cloud\_sql\_proxy -instances=$SQL\_CONNECTION=tcp:3306 &

Press ENTER

Connect an application to the Cloud SQL instance

curl -H "Metadata-Flavor: Google" http://169.254.169.254/computeMetadata/v1/instance/network-interfaces/0/access-configs/0/external-ip && echo

Go to the wordpress-europe-proxy external IP address in your browser and configure the Wordpress application.

Click Let's Go.

Specify the following, replacing [ROOT\_PASSWORD] with the password you configured upon machine creation, and leave the remaining settings as their defaults:

Property Value

Username root

Password [ROOT\_PASSWORD]

Database Host 127.0.0.1

Click Submit.

When a connection has been made, click Run the installation

**Examining Billing data with BigQuery**

Sign in to BigQuery and create a dataset

In the Cloud Console, on the Navigation menu ( 7a91d354499ac9f1.png), click BigQuery.

click on to your Project ID (starts with qwiklabs-gcp) and click Create Dataset.

Specify the following:

Property Value (type value or select option as specified)

Dataset ID: imported\_billing\_data

Data location: US

Default table expiration > Number of days after table creation: In 1 days

Click Create Dataset. You should see imported\_billing\_data in the left pane.

Create a table and import

Point to imported\_billing\_data, and then click Create Table to create a new table.

For Source, specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Create table from: Google Cloud Storage

Select file from GCS bucket gs://cloud-training/archinfra/export-billing-example.csv

File format CSV

For Destination, specify the following, and leave the remaining settings as their defaults:

Property Value (type value or select option as specified)

Table name sampleinfotable

Table type Native table

Under Schema for Auto detect click Schema and input parameters .

Open Advanced options

Under Header rows to skip specify 1

Click Create Table

Examine the table

Click sampleinfotable.

Click Details. As you can see in Number of Rows, this is a relatively small table with 44 rows.

Click Preview.

**Virtual Private Networks (VPN)**

Create the VPN gateways and tunnels

Reserve two static IP addresses

Reserve one static IP address for each VPN gateway.

In the Cloud Console, on the Navigation menu (Navigation menu), click VPC network > External IP addresses.

Click 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.

Click 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-2-static-ip

IP version IPv4

Region europe-west1

Click Reserve.

Create the vpn-1 gateway and tunnel1to2

In the Cloud Console, on the Navigation menu (Navigation menu), click Hybrid Connectivity > VPN.

Click Create VPN Connection.

If asked, select Classic VPN, and then click 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

Make sure to replace [VPN-2-STATIC-IP] with your reserved IP address for europe-west1.

Click Create.

Create the vpn-2 gateway and tunnel2to1

Click VPN setup wizard.

If asked, select Classic VPN, and then click 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-2

Network vpn-network-2

Region europe-west1

IP address vpn-2-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 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

Make sure to replace [VPN-1-STATIC-IP] with your reserved IP address for us-central1.

Click Create.

Verify server-1 to server-2 connectivity

**Configuring an HTTP Load Balancer with Autoscaling**

Create the health check rule

Create a firewall rule to allow health checks.

In the Cloud Console, on the Navigation menu (Navigation menu), click VPC network > Firewall. Notice the existing ICMP, internal, RDP, and SSH firewall rules.

Each Google Cloud 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 IP Ranges

Source IP ranges 130.211.0.0/22 and 35.191.0.0/16

Protocols and ports Specified protocols and ports

Select tcp and specify port 80.

Click Create.

Create a NAT configuration using Cloud Router

In the Cloud Console, on the Navigation menu (Navigation menu), 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

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.

Create a custom image for a web server

In the Cloud Console, on the Navigation menu (Navigation menu), click Compute Engine > VM instances.

Click Create.

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

Series N1

Machine type f1-micro (1 vCPU)

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

Click Disks, and clear Delete boot disk when instance is deleted.

Click Networking.

For Network tags, type allow-health-checks.

Under Network interfaces , click default.

Under External IP dropdown, select None.

Click Done.

Click Create.

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

In the webserver SSH terminal, set the service to start on boot:

sudo update-rc.d apache2 enable

In the Cloud Console, select webserver, and then click Reset.

In the confirmation dialog, click Reset.

Configure an instance template and create instance groups

In the Cloud Console, on the Navigation menu (Navigation menu), click Compute Engine > Instance templates.

Click Create instance template.

For Name, type mywebserver-template.

For Machine type, select f1-micro (1 vCPU).

For Boot disk, click Change.

Click Custom images.

For Image, Select mywebserver.

Click Select.

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

Click Networking.

For Network tags, type allow-health-checks.

Under External IP dropdown, select None.

Click Create.

Configure the HTTP load balancer

Start the configuration

On the Navigation menu, click Network Services > Load balancing.

Click Create load balancer.

Under HTTP(S) Load Balancing, click Start configuration.

Select From Internet to my VMs, then click Continue.

For Name, type http-lb.

Click Backend configuration.

For Backend services & backend buckets, click Create or select backend services & backend buckets > Backend services > 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 (TCP).

Expand Advanced configurations (Session affinity, connection draining timeout, security policies) option and check the Enable logging checkbox.

Specify Sample rate as 1.

Click Create.

Stress test the HTTP load balancer

In the Cloud Console, on the Navigation menu (Navigation menu), click Compute Engine > VM instances.

Click Create instance.

To place a load on the load balancer, run the following command:

ab -n 500000 -c 1000 <http://$LB_IP/>

**Automating the Deployment of Infrastructure Using Deployment Manager**

In the Cloud Console, click Activate Cloud Shell (Cloud Shell).

Run the following commands:

mkdir dminfra

cd dminfra

In Cloud Shell, click Launch code editor (Cloud Shell Editor).

In the left pane of the code editor, expand the dminfra folder.

To create a new file, click File > New File.

Name the new file config.yaml, and then open it.

imports:

- path: instance-template.jinja

resources:

# Create the auto-mode network

- name: mynetwork

type: compute.v1.network

properties:

autoCreateSubnetworks: true

# Create the firewall rule

- name: mynetwork-allow-http-ssh-rdp-icmp

type: compute.v1.firewall

properties:

network: $(ref.mynetwork.selfLink)

sourceRanges: ["0.0.0.0/0"]

allowed:

- IPProtocol: TCP

ports: [22, 80, 3389]

- IPProtocol: ICMP

# Create the mynet-us-vm instance

- name: mynet-us-vm

type: instance-template.jinja

properties:

zone: us-central1-a

machineType: n1-standard-1

network: $(ref.mynetwork.selfLink)

subnetwork: regions/us-central1/subnetworks/mynetwork

Create the VM instance template

To create a new file, click File > New File.

Name the new file instance-template.jinja, and then open it

resources:

- name: {{ env["name"] }}

type: compute.v1.instance

properties:

machineType: zones/{{ properties["zone"] }}/machineTypes/{{ properties["machineType"] }}

zone: {{ properties["zone"] }}

networkInterfaces:

- network: {{ properties["network"] }}

subnetwork: {{ properties["subnetwork"] }}

accessConfigs:

- name: External NAT

type: ONE\_TO\_ONE\_NAT

disks:

- deviceName: {{ env["name"] }}

type: PERSISTENT

boot: true

autoDelete: true

initializeParams:

sourceImage: https://www.googleapis.com/compute/v1/projects/debian-cloud/global/images/family/debian-9

In Cloud Shell, run the following command:

gcloud deployment-manager deployments create dminfra --config=config.yaml --preview

Run the following command to create the deployment:

gcloud deployment-manager deployments update dminfra

**Automating the Deployment of Infrastructure Using Terraform**

**gcloud config set project qwiklabs-gcp-04-d17ae3a00145**

mkdir tfinfra

In Cloud Shell, click Open Editor (Cloud Shell Editor).

To create a new file, click File > New File.

Name the new file provider.tf, and then open it.

Copy the code into provider.tf:

provider "google" {}

To initialize Terraform, run the following command:

cd tfinfra

terraform init

To create a new file, click File > New File.

Name the new file mynetwork.tf, and then open it.

# Create the mynetwork network

resource "google\_compute\_network" "mynetwork" {

name = "mynetwork"

auto\_create\_subnetworks = true

}

# 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"

}

}

# 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

}

To create a new folder inside tfinfra, select the tfinfra folder, and then click File > New Folder.

Name the new folder instance.

To create a new file inside instance, select the instance folder, and then click File > New File.

Name the new file main.tf, and then open it.

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

}

}

}

To rewrite the Terraform configuration files to a canonical format and style, run the following command:

terraform fmt

To initialize Terraform, run the following command:

terraform init

To create an execution plan, run the following command:

terraform plan

To apply the desired changes, run the following command:

terraform apply

To confirm the planned actions, type:

yes