**Lab 1: Google Cloud Fundamentals: Getting Started with GKE**

**Objectives:**

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

* Provision a Kubernetes cluster using Kubernetes Engine.
* Deploy and manage Docker containers using kubectl.

**Steps:**

1. Confirm that needed APIs are enabled.
   1. Use the gcloud services command to confirm that both the Kubernetes Engine API and the Containers Registry API are enabled:

gcloud services list --enabled

1. Result: Both API's are enabled.
2. Start a Kubernetes Engine cluster.
   1. Assign your Qwiklabs zone to an environment variable called MY\_ZONE:

export MY\_ZONE=us-central1-a

* 1. Start a Kubernetes cluster managed by Kubernetes Engine. Name the cluster webfrontend and configure it to run 2 nodes:

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

* 1. After the cluster is created, check your installed version of Kubernetes using the kubectl version command:

kubectl version

1. Run and deploy a container.
   1. Launch a single instance of the nginx container:

kubectl create deploy nginx --image=nginx:1.17.10

* 1. View the pod running the nginx container:

kubectl get pods

* 1. Expose the nginx container to the Internet:

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

* 1. View the new service:

kubectl get services

* + - Result: The external IP address to view your webserver's home page.
  1. Scale up the number of pods running on your service:

kubectl scale deployment nginx --replicas 3

* 1. Confirm that Kubernetes has updated the number of pods:

kubectl get pods

* 1. Confirm that your external IP address has not changed:

kubectl get services

* + - Result: The IP address did not change.

**LAB 2: Google Cloud Fundamentals: Getting Started with App Engine**

**Objectives:**

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

* Initialize App Engine.
* Preview an App Engine application running locally in Cloud Shell.
* Deploy an App Engine application, so that others can reach it.
* Disable an App Engine application, when you no longer want it to be visible.

**Steps:**

1. Initialize App Engine:
   1. Initialize App Engine app with a project and choose its region:

gcloud app create --project=$DEVSHELL\_PROJECT\_ID --region=us-central

* 1. Clone the source code repository for a sample application in the hello\_world directory:

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

* 1. Navigate to the source directory:

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

1. Run Hello World application locally:
   1. Download and update the packages list:

sudo apt-get update

* 1. Set up a python virtual environment in which the application will run:

sudo apt-get install virtualenv -y

virtualenv -p python3 venv

* 1. Activate the virtual environment

source venv/bin/activate

* 1. Navigate to the project directory and install dependencies

pip install -r requirements.txt

* 1. To run the application:

python main.py

1. Deploy and run Hello World on App Engine:
   1. Navigate to the source directory:

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

* 1. Deploy your Hello World application:

gcloud app deploy

* + - If prompted enter y
  1. Launch your browser to view the app:

gcloud app browse

1. Disable the application:

gcloud app versions stop 20200910t135648

* Note: Auto-scaling has to be disabled

**Lab 3: Google Cloud Fundamentals: Getting Started with Compute Engine**

**Objectives:**

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

* Create a Compute Engine virtual machine using the Google Cloud Platform (GCP) Console.
* Create a Compute Engine virtual machine using the gcloud command-line interface.
* Connect between the two instances.

**Steps:**

1. Create a Compute Engine virtual machine using the Google Cloud Platform (GCP) Console.

gcloud compute instances create "my-vm-1" --machine-type "n1-standard-1" --image-project "debian-cloud" --image "debian-9-stretch-v20190213" --subnet "default" --tags http

gcloud compute firewall-rules create allow-http --action=ALLOW --destination=INGRESS --rules=http:80 --target-tags=http

1. Create a Compute Engine virtual machine using the gcloud command-line interface.

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"

1. Connect between the two instances.
   1. Use the ping command to confirm that my-vm-2 can reach my-vm-1 over the network:
      * Connect to my-vm-2:

gcloud compute ssh my-vm-2

* + - Ping my-vm-1 from my-vm-2:

ping -c 4 my-vm-1

* + - Use the ssh command to open a command prompt on my-vm-1 from my-vm-2:

ssh my-vm-1

* + - At the command prompt on my-vm-1, install Nginx web server:

sudo apt-get install nginx-light -y

* + - Use the nano text editor to add a custom message to the home page of the web server:

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

* + - Add text like this, and replace YOUR\_NAME with your name:

Hi from Mary

* + - Exit the editor and confirm that the web server is serving your new page. At the command prompt on my-vm-1, execute this command:

curl <http://localhost/>

* + - Result: The response will be the HTML source of the web server's home page, including your line of custom text.
    - To exit the command prompt on my-vm-1, execute this command:

exit

* + - To confirm that my-vm-2 can reach the web server on my-vm-1, at the command prompt on my-vm-2, execute this command:

curl <http://my-vm-1/>

* + - Result: The response will again be the HTML source of the web server's home page, including your line of custom text.
  1. Now get the external IP of the my-vm-1 instance from this command:

gcloud compute instances list --zone us-central1-a

* 1. Paste the copied IP address of my-vm-1 into a new browser tab and hit enter:
     + Result: You will see your web server's home page, including your custom text.