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

* **Provision a**[**Kubernetes**](http://kubernetes.io/)**cluster using**[**Kubernetes Engine.**](https://cloud.google.com/container-engine)
* **Deploy and manage Docker containers using kubectl**

**Task 1: Confirm that the required API’s are**

**Enabled**

Make a note of the name of your GCP project. This value is shown in the top bar of the Google Cloud Platform Console. It will be of the form **qwiklabs-gcp-** followed by hexadecimal numbers.

1. In the GCP cloud shell prompt execute the following command to list the enabled APIs and services in your current project:

gcloud services list

1. Scroll down in the list of enabled APIs, and confirm that both of these APIs are appearing in the list:

Kubernetes Engine API

Container Registry API

1. If either API is missing, In the GCP command line execute the following command:

gcloud services enable container.googleapis.com (For Kubernetes engine API)

gcloud services enable containerregistry.googleapis.com(For Container registry API)

1. Run the following command to confirm that the API’s are now enabled:

gcloud services list

**Task 2: Start a Kubernetes Engine cluster**

1. For convenience, place the zone that Qwiklabs assigned you to into an environment variable called MY\_ZONE. At the Cloud Shell prompt, type this partial command:

export MY\_ZONE=

followed by the zone that Qwiklabs assigned to you. Your complete command will look similar to this:

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

It takes several minutes to create a cluster as Kubernetes Engine provisions virtual machines for you.

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

kubectl version

The gcloud container clusters create command automatically authenticated kubectl for you.

1. View your running nodes by running the following command:

gcloud compute instances list

Your Kubernetes cluster is now ready for use.

**Task 3: Start a Kubernetes Engine cluster**

1. From your Cloud Shell prompt, launch a single instance of the nginx container. (Nginx is a popular web server.)

kubectl create deploy nginx --image=nginx:1.17.10

In Kubernetes, all containers run in pods. This use of the **kubectl create** command caused Kubernetes to create a deployment consisting of a single pod containing the nginx container. A Kubernetes deployment keeps a given number of pods up and running even in the event of failures among the nodes on which they run. In this command, you launched the default number of pods, which is 1.

**Note**: If you see any deprecation warning about future version you can simply ignore it for now and can proceed further.

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

Kubernetes created a service and an external load balancer with a public IP address attached to it. The IP address remains the same for the life of the service. Any network traffic to that public IP address is routed to pods behind the service: in this case, the nginx pod.

1. View the new service:

kubectl get services

You can use the displayed external IP address to test and contact the nginx container remotely.

It may take a few seconds before the **External-IP** field is populated for your service. This is normal. Just re-run the kubectl get services command every few seconds until the field is populated.

1. Curl to default homepage of the nginx browser by running the following curl command using your cluster's external IP address in the url:

Curl http://(insert your cluster's external IP address in this space).

1. The default home page of the Nginx browser is displayed
2. Scale up the number of pods running on your service:

kubectl scale deployment nginx --replicas 3

Scaling up a deployment is useful when you want to increase available resources for an application that is becoming more popular.

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

1. Run the following command again to view the nginx homepage to

confirm that the nginx web server is still responding.

Curl http://(insert your cluster’s external IP address in

this space

**Pat yourself on the back. Great Job.**

**This is the end of the lab!!!**