Google Cloud Fundamentals: Getting Started with GKE

Overview

In this lab, you create a Google Kubernetes Engine cluster containing several containers, each containing a web server. You place a load balancer in front of the cluster and view its contents.

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.

Task 1: Sign in to the Google Cloud Platform (GCP) Console

1. Login with your credentials.

Task 2: Confirm that needed APIs are enabled

1. 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.

```
gcloud projects list
```

2. Using the applicable project ID from the previous step, set the default project to the one in which you want to enable the API:

```
gcloud config set project YOUR_PROJECT_ID
```

3. Get a list of services that you can enable in your project

```
gcloud services list --available
```

If you don't see the API listed, that means you haven't been granted access to enable the API

4. Enable API service if the listed two below are not enabled: Check for :
    - Kubernetes Engine API
    - Container Registry API

```
```

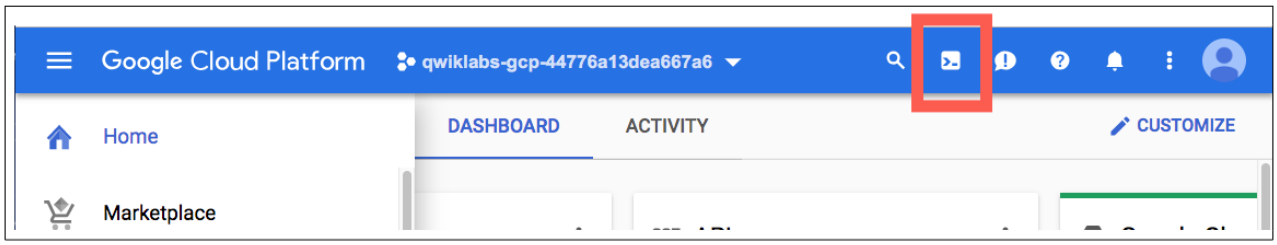
```
gcloud services enable <SERVICE_NAME>
```

see example:

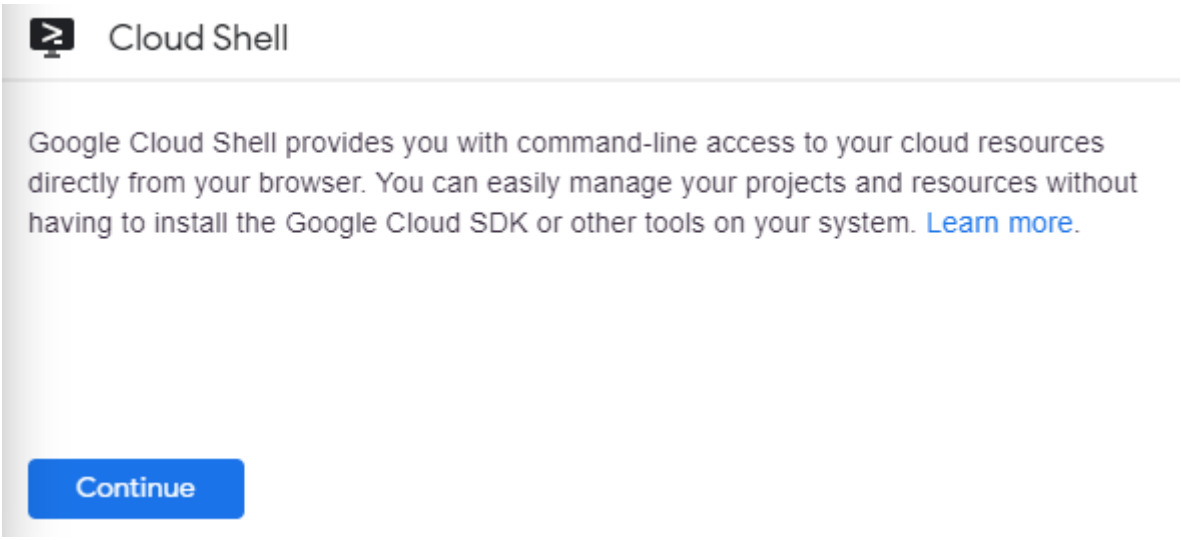
```
gcloud services enable containerregistry.googleapis.com
```

Task 3: Start a Kubernetes Engine cluster

1. In GCP console, on the top right toolbar, click Activate Cloud Shell button



2. Click Continue.



3. 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
```

4. 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.

5. 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.

6. View your running nodes

```
kubectl get node
```

Task 4: Run and deploy a container

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.

2. View the pod running the nginx container:

```
kubectl get pods
```

3. Expose the nginx container to the Internet:

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

4. 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.

5. Open a new web browser tab and paste your cluster's external IP address into the address bar. The default home page of the Nginx browser is displayed.

6. 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.

7. Confirm that Kubernetes has updated the number of pods:

```
kubectl get pods
```

8. Confirm that your external IP address has not changed:

```
kubectl get services
```

9. Return to the web browser tab in which you viewed your cluster's external IP address. Refresh the page to confirm that the nginx web server is still responding.

Congratulations!

---

compiled by

```
joyceMimi
```