# Google Cloud Fundamentals: Getting Started with GKE

https://googlepluralsight.qwiklabs.com/focuses/10995496?parent=lti_session

35 minutes Free

## 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)

For each lab, you get a new GCP project and set of resources for a fixed time at no cost.

1. Open your terminal and make sure you signed in:

```
gcloud auth login
```

2. List available projects and take a note of your project ID:

```
gcloud project list
```

3. Set the project ID:

```
gcloud config set project <PROJECT_ID>
```

## Task 2: Confirm that needed APIs are enabled

1. Make a note of the name of your GCP project:

```
gcloud project list
```

2. To see a list of available services for a project, run:

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

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

- Kubernetes Engine API
- Container Registry API

If either API is missing, use the  command below to enable the services

```
gcloud services enable Kubernetes Engine API Container
```

```
Registry API
```

## Task 3: Start a Kubernetes Engine cluster

1. Launch the cloud shell:

```
gcloud alpha cloud-shell
```

2. 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
```

3. 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.

4. 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.

5. View your running nodes:

```
gcloud compute instances list
```

## Task 4: Run and deploy a container

1. 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
   ```

   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.

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!

In this lab, you configured a Kubernetes cluster in Kubernetes Engine. You populated the cluster with several pods containing an application, exposed the application, and scaled the application.

## End your lab

When you have completed your lab, click **End Lab**. Qwiklabs removes the resources you've used and cleans the account for you.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.