App Dev: Deploying the Application into Kubernetes Engine - Python

GSP188

Google Cloud Self-Paced Labs

Overview

Google Kubernetes Engine provides a managed environment for deploying, managing, and scaling your containerized applications using Google infrastructure. The environment Kubernetes Engine provides consists of multiple machines (specifically, Compute Engine instances) grouped together to form a cluster.

Kubernetes provides the mechanisms through which you interact with your cluster. You use Kubernetes commands and resources to deploy and manage your applications, perform administration tasks and set policies, and monitor the health of your deployed workloads.

In this lab, you deploy the Quiz application into Kubernetes Engine, leveraging Google Cloud resources, including Container Builder and Container Registry, and Kubernetes resources, such as Deployments, Pods, and Services.

Objectives

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

- Create Dockerfiles to package up the Quiz application frontend and backend code for deployment.
- Harness Container Builder to produce Docker images.
- Provision a Kubernetes Engine cluster to host the Quiz application.
- Employ Kubernetes deployments to provision replicated Pods into Kubernetes Engine.
- Leverage a Kubernetes service to provision a load balancer for the quiz frontend.

Setup and requirements

Qwiklabs setup

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

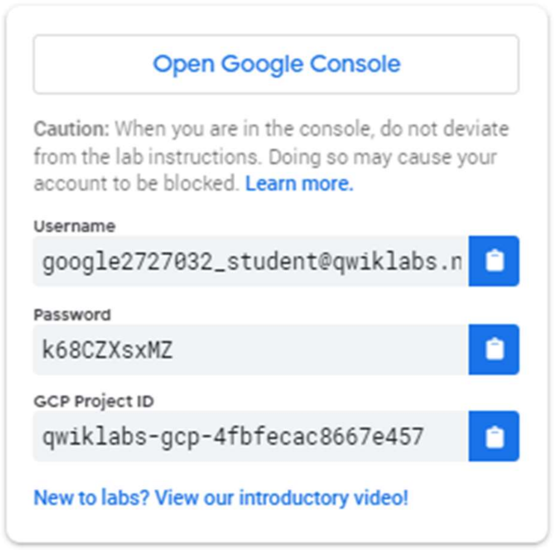
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.
  Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.

Cloud Console

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



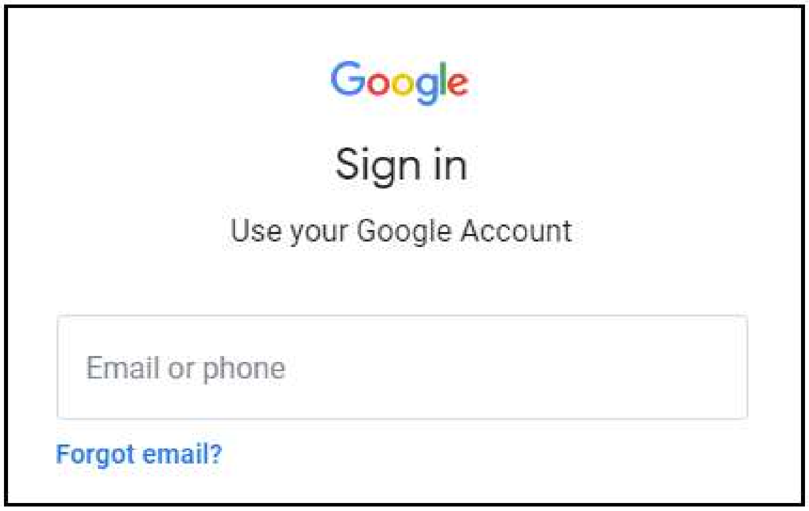
Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username
google2727032_student@qwiklabs.n

Password
k68CZXsxMZ

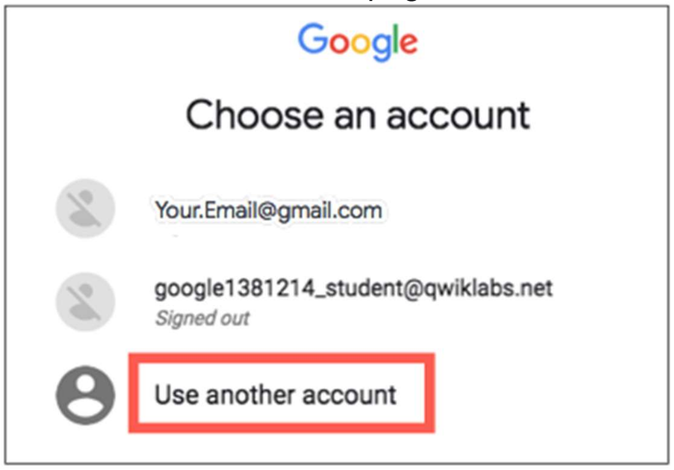
GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another Account**.



3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

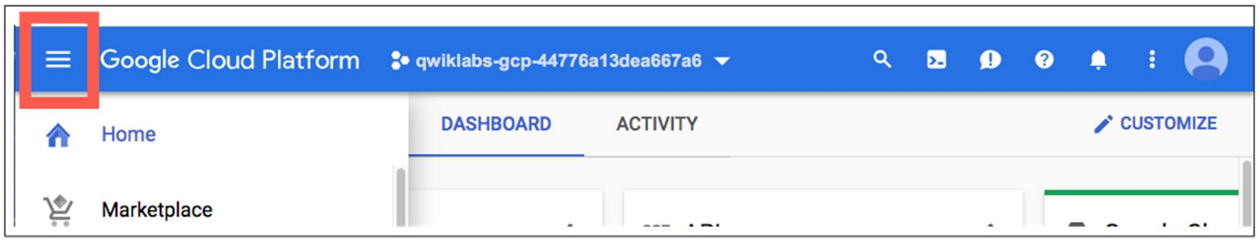
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

   - Accept the terms and conditions.
   - Do not add recovery options or two-factor authentication (because this is a temporary account).
   - Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

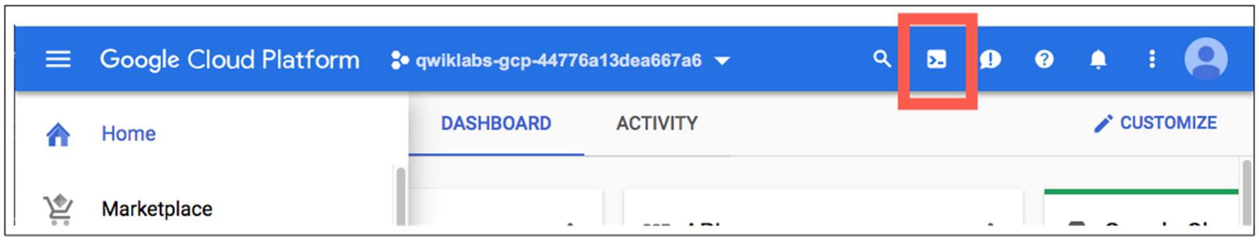
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-
left.



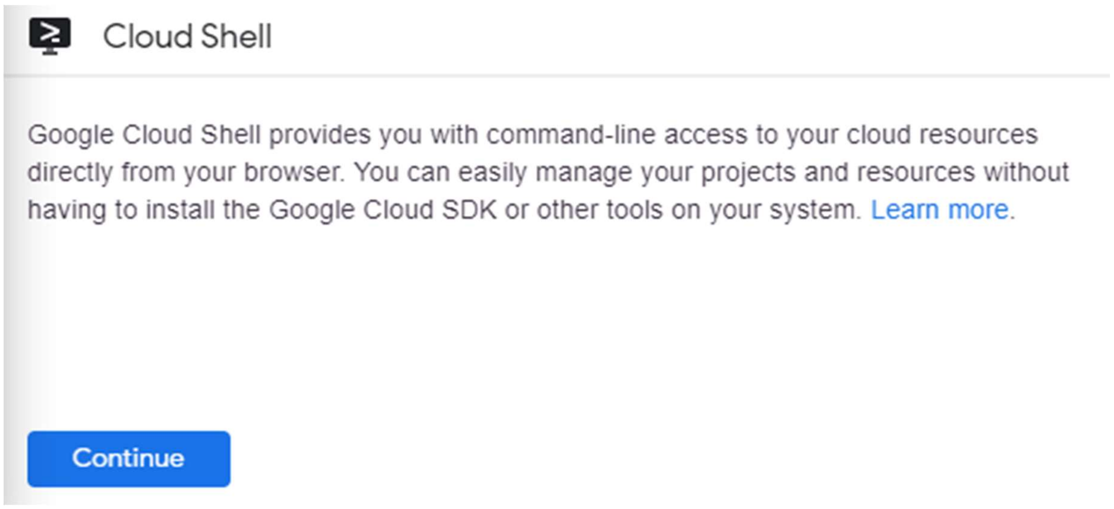
Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.

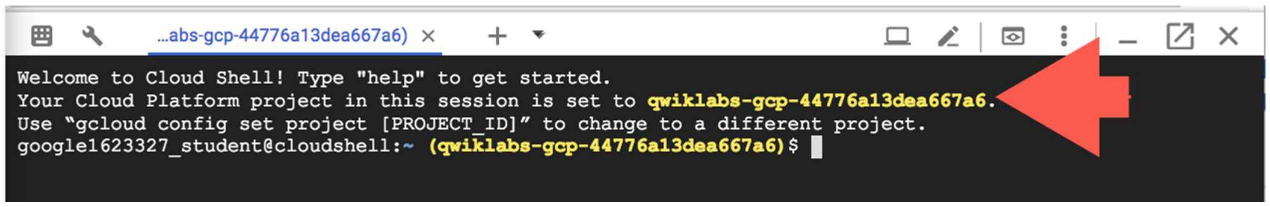


Click **Continue**.


Cloud Shell

Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6)$
```

`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:
 - <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:
 - google1623327_student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]
project = <project_ID>
```

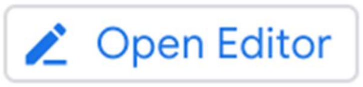
(Example output)

```
[core]
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the gcloud command-line tool overview.

Launch the Cloud Shell code editor

From Cloud Shell, click **Open Editor** icon to launch the code editor.



The code editor launches in a separate tab of your browser, along with Cloud Shell.

Prepare the Quiz Application

In this section, you access Cloud Shell, clone the git repository containing the Quiz application, configure environment variables, and run the application.

Clone source code in Cloud Shell

Click **Open Terminal** and clone the repository for the lab.

```
git clone https://github.com/GoogleCloudPlatform/training-data-analyst
```

Create a soft link as a shortcut to the working directory.

```
ln -s ~/training-data-analyst/courses/developingapps/v1.2/python/kubernetesengine
~/kubernetesengine
```

Configure the Quiz application

Change the directory that contains the sample files for this lab.

```
cd ~/kubernetesengine/start
```

Configure the Quiz application.

```
. prepare_environment.sh
```

This script file:

- Creates a Google App Engine application.
- Exports environment variables `GCLOUD_PROJECT` and `GCLOUD_BUCKET`.
- Updates pip then runs `pip install -r requirements.txt`.
- Creates entities in Google Cloud Datastore.
- Creates a Google Cloud Pub/Sub topic.
- Creates a Cloud Spanner Instance, Database, and Table.
- Prints out the Project ID.

The Quiz application is configured when you see the following message:

Example output message

```
Creating Cloud Pub/Sub topic
Created topic [projects/qwiklabs-gcp-92b7e5716e0cbf7e/topics/feedback].
Created subscription [projects/qwiklabs-gcp-92b7e5716e0cbf7e/subscriptions/worker-
subscription].
Creating Cloud Spanner Instance, Database, and Table
Creating instance...done.
Creating database...done.
Project ID: qwiklabs-gcp-92b7e5716e0cbf7e
```

Click *Check my progress* to verify the objective.

Review the code

In this section you examine the application files.

To view and edit files, you can use the shell editors that are installed in Cloud Shell, such as `nano` or `vim` or the Cloud Shell code editor. This lab uses the Cloud Shell code editor.

Examine the code

Navigate to `training-data-analyst/courses/developingapps/v1.2/python/kubernetesengine/start`.

The folder structure for the Quiz application reflects how it will be deployed in Kubernetes Engine.

The web application is in a folder called `frontend`.

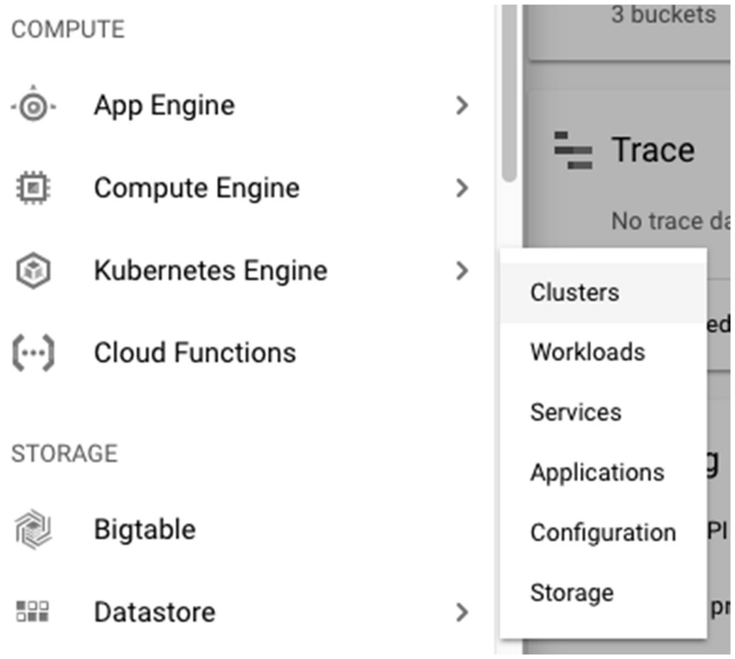
The worker application code that subscribes to Cloud Pub/Sub and processes messages is in a folder called `backend`.

There are configuration files for Docker (a `Dockerfile` in the `frontend` and `backend` folder) and `backend-deployment` and `frontend-deployment` Kubernetes Engine `.yaml` files.

Create and connect to a Kubernetes Engine Cluster

Create a Kubernetes Engine Cluster

1. In the Cloud Platform Console, click **Navigation menu** > **Kubernetes Engine** > **Clusters**.



2. Click **Create cluster**.

3. Configure the cluster. Set the following fields to the provided values, leave all others at the default value:

Property	**Value**
Name	`quiz-cluster`
Zone	`us-central1-b`
default Pool > Security > Access scopes	Select **Allow full access to all Cloud APIs**, and then click **Save**

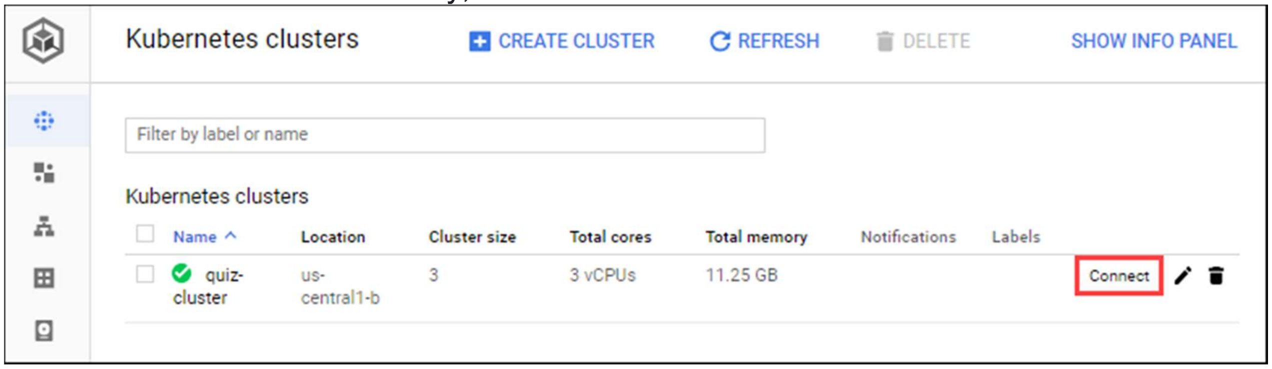
4. Click **Create**. The cluster takes a few minutes to provision.

   Click *Check my progress* to verify the objective.

Connect to the cluster

In this section you connect the Quiz application to the kubernetes cluster.

1. When the cluster is ready, click **Connect**.



Kubernetes clusters

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels	
✓ quiz-cluster	us-central1-b	3	3 vCPUs	11.25 GB			Connect

2. In **Connect to the cluster**, click **Run in Cloud Shell** to populated Cloud shell with the command that resembles `gcloud container clusters get-credentials quiz-cluster --zone us-central1-b --project [Project-ID]`. Press **Enter** to run the commnand in Cloud Shell.

3. Run the following command to list the pods in the cluster:

```
kubectl get pods
```

The response should be `No resources found` because there are no pods in the cluster. It confirms that you have configured security to allow the `kubectl` command-line tool to perform operations against the cluster.

Build Docker Images using Container Builder

In this section, you create a Dockerfile for the application frontend and backend, and then employ Container Builder to build images and store them in the Container Registry.

Create the Dockerfile for the frontend and backend

In the Cloud Shell code editor, open `frontend/Dockerfile`. You will now add a block of code that does the following:

- Enters the Dockerfile command to initialize the creation of a custom Docker image using Google's Python App Engine image as the starting point.
- Writes the Dockerfile commands to activate a virtual environment.
- Writes the Dockerfile command to execute `pip install` as part of the build process.
- Writes the Dockerfile command to add the contents of the current folder to the `/app` path in the container.
- Completes the `Dockerfile` by entering the statement, `gunicorn ...`, that executes when the container runs. Gunicorn (Green Unicorn) is an HTTP server that supports the Python Web Server Gateway Interface (WSGI) specification.
  Copy and paste the following to `Dockerfile`:

```
FROM gcr.io/google_appengine/python

RUN virtualenv -p python3.7 /env

ENV VIRTUAL_ENV /env
ENV PATH /env/bin:$PATH

ADD requirements.txt /app/requirements.txt
RUN pip install -r /app/requirements.txt

ADD . /app

CMD gunicorn -b 0.0.0.0:$PORT quiz:app
```

Open the `backend/Dockerfile` file and copy and paste the following code:

```
FROM gcr.io/google_appengine/python

RUN virtualenv -p python3.7 /env

ENV VIRTUAL_ENV /env
ENV PATH /env/bin:$PATH

ADD requirements.txt /app/requirements.txt
RUN pip install -r /app/requirements.txt

ADD . /app

CMD python -m quiz.console.worker
```

Build Docker images with Container Builder

1. In Cloud Shell, make sure you are in the `start` folder:

```
cd ~/kubernetesengine/start
```

2. Run the following command to build the frontend Docker image:

```
gcloud builds submit -t gcr.io/$DEVSHELL_PROJECT_ID/quiz-frontend ./frontend/
```

The files are staged into Cloud Storage, and a Docker image is built and stored in the Container Registry. It takes a few minutes.

Ignore any incompatibility messages you see in the output messages.

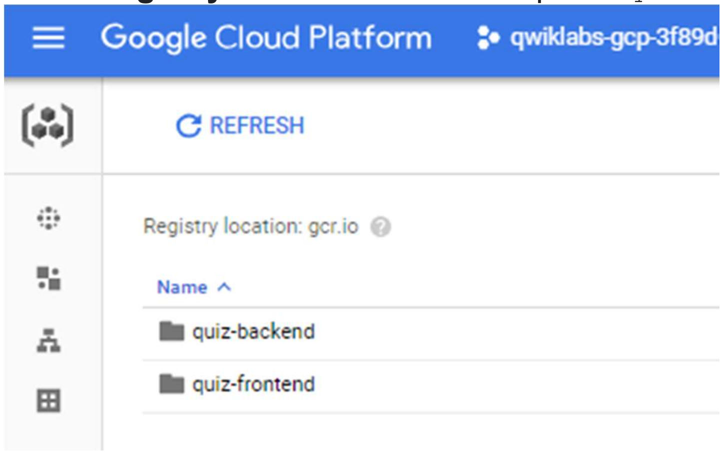
3. Now run the following command to build the backend Docker image:

```
gcloud builds submit -t gcr.io/$DEVSHELL_PROJECT_ID/quiz-backend ./backend/
```

When the backend Docker image is ready you see these last messages:

```
DONE
---------------------------------------------------------------------------------------
-------------------------------
ID                                    CREATE_TIME                DURATION  SOURCE
                                                             IMAGES
   STATUS
be0326f4-3f6f-42d6-850f-547e260dd4d7  2018-06-13T22:20:16+00:00  50S
gs://qwiklabs-gcp-3f89d0745056ee31_cloudbuil
d/source/1528928414.79-4914d2a972f74e188f40ced135662b7d.tgz  gcr.io/qwiklabs-gcp-
3f89d0745056ee31/quiz-backend (+1 more
)   SUCCESS
```

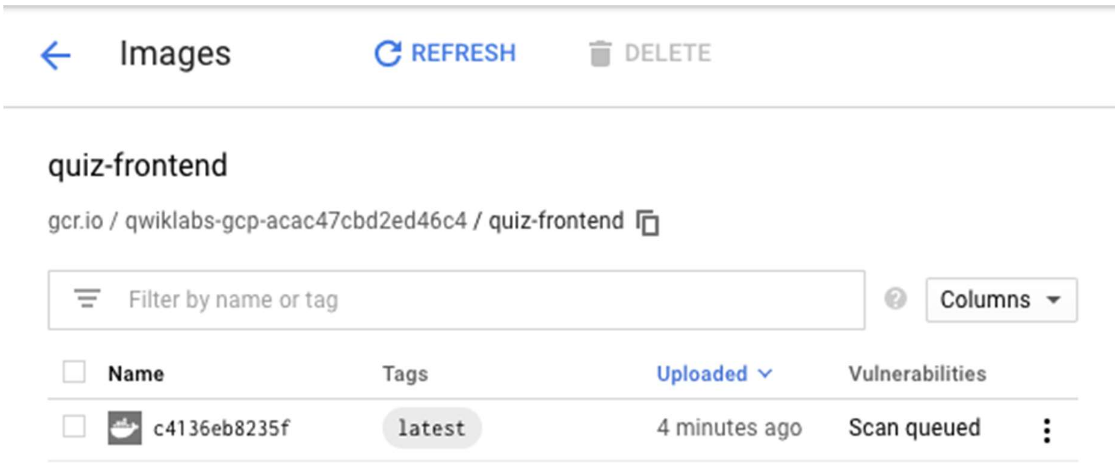
4. In the **Cloud Platform Console**, from the **Navigation menu** menu, click **Container Registry**. You should see two pods: `quiz-frontend` and `quiz-backend`.



quiz-backend
quiz-frontend

5. Click **quiz-frontend**.

You should see the image name and tags (latest).


Images
REFRESH
DELETE

quiz-frontend

gcr.io / qwiklabs-gcp-acac47cbd2ed46c4 / quiz-frontend

Filter by name or tag
Columns

Name	Tags	Uploaded	Vulnerabilities
c4136eb8235f	latest	4 minutes ago	Scan queued

Click *Check my progress* to verify the objective.

Create Kubernetes Deployment and Service Resources

In this section, you will modify the template `yaml` files that contain the specification for Kubernetes Deployment and Service resources, and then create the resources in the Kubernetes Engine cluster.

Create a Kubernetes Deployment file

1. In the **Cloud Shell** code editor, open the `frontend-deployment.yaml` file.

The file skeleton has been created for you. Your job is to replace placeholders with values specific to your project.

2. Replace the placeholders in the `frontend-deployment.yaml` file using the following values:

Placeholder Name	Value
`[GCLOUD_PROJECT]`	Project ID(Display the Project ID by entering`echo $GCLOUD_PROJECT` in **Cloud Shell**)
`[GCLOUD_BUCKET]`	Cloud Storage bucket name for the media bucket in your project(Display the bucket name by entering`echo $GCLOUD_BUCKET` in **Cloud Shell**)
`[FRONTEND_IMAGE_IDENTIFIER]`	The frontend image identified in the form `gcr.io/[Project_ID]/quiz-frontend`

The quiz-frontend deployment provisions three replicas of the frontend Docker image in Kubernetes pods, distributed across the three nodes of the Kubernetes Engine cluster.

3. Save the file.
4. Replace the placeholders in the `backend-deployment.yaml` file using the following values:

Placeholder Name	Value
`[GCLOUD_PROJECT]`	Project ID
`[GCLOUD_BUCKET]`	Cloud Storage bucket ID for the media bucket in your project
`[BACKEND_IMAGE_IDENTIFIER]`	The backend image identified in the form `gcr.io/[Project_ID]/quiz-backend`

The quiz-backend deployment provisions two replicas of the backend Docker image in Kubernetes pods, distributed across two of the three nodes of the Kubernetes Engine cluster.

5. Save the file.
6. Review the contents of the `frontend-service.yaml` file.

The service exposes the frontend deployment using a load balancer. The load balancer sends requests from clients to all three replicas of the frontend pod.

Execute the Deployment and Service Files

1. In Cloud Shell, provision the quiz frontend Deployment.

```
kubectl create -f ./frontend-deployment.yaml
```

2. Provision the quiz backend Deployment.

```
kubectl create -f ./backend-deployment.yaml
```

3. Provision the quiz frontend Service.

```
kubectl create -f ./frontend-service.yaml
```

Each command provisions resources in Kubernetes Engine. It takes a few minutes to complete the process.

Click *Check my progress* to verify the objective.

Test the Quiz Application

In this section you review the deployed Pods and Service and navigate to the Quiz application.

Review the deployed resources

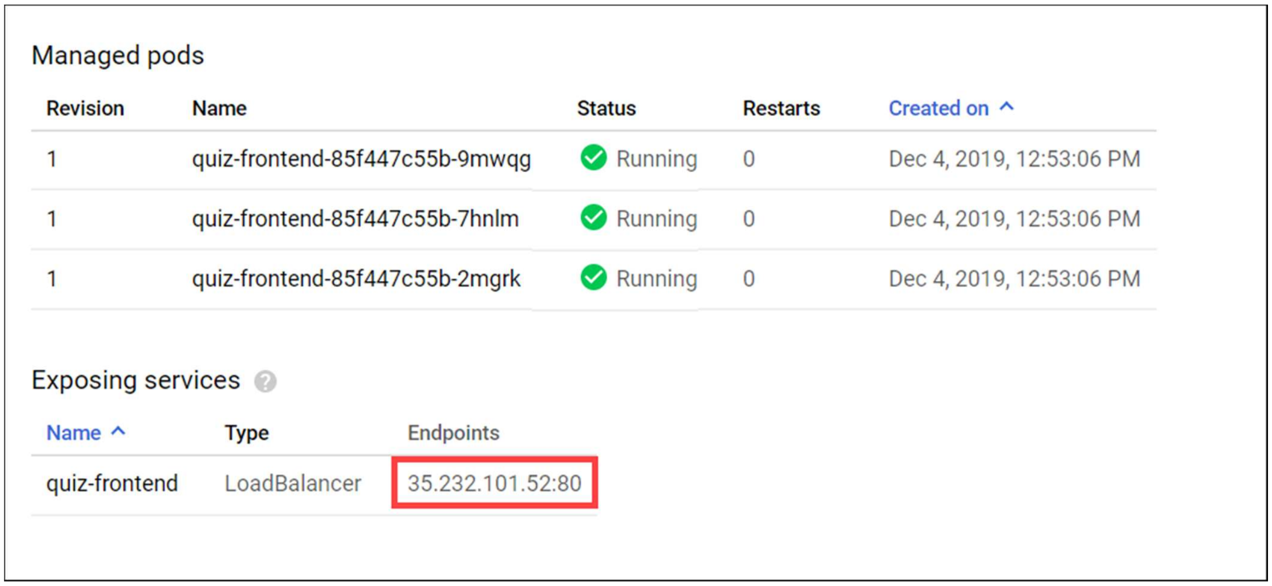
1. In the **Cloud Console**, select **Navigation menu** > **Kubernetes Engine**.
2. Click **Workloads** in the left menu.

You should see two containers: quiz-frontend and quiz-backend. You may see that the status is OK or in the process of being created.

If the status of one or both containers is **Does not have minimum availability**, refresh the window.

3. Click **quiz-frontend**. In the **Managed pods** section that there are three quiz-frontend pods.

4. In the **Services** section near the bottom, find the **Endpoints** section and copy the the IP address and paste it into the URL field of a new browser tab or window:



Managed pods

Revision	Name	Status	Restarts	Created on
1	quiz-frontend-85f447c55b-9mwqg	✓ Running	0	Dec 4, 2019, 12:53:06 PM
1	quiz-frontend-85f447c55b-7hnlm	✓ Running	0	Dec 4, 2019, 12:53:06 PM
1	quiz-frontend-85f447c55b-2mgrk	✓ Running	0	Dec 4, 2019, 12:53:06 PM

Exposing services

Name	Type	Endpoints
quiz-frontend	LoadBalancer	35.232.101.52:80

5. This opens the Quiz application, which means you successfully deployed the application! You can end your lab here or use the remainder of the time to build some quizzes.

Congratulations!

This concludes the self-paced lab, App Dev: Deploying the Application into Kubernetes Engine - Python. You leveraged Google Cloud resources and Kubenetes resources to deploy a Quiz application.

Finish your Quest

This self-paced lab is part of the Application Development - Python and Cloud Development Quests. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in a Quest and get immediate completion credit if you've taken this lab. See other available Qwiklabs Quests.

Next steps / learn more

Learn more about Kubernetes Engine.
Manual last updated October 16, 2020
Lab last tested October 16, 2020