Deploying the Application into Kubernetes Engine - Python

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.

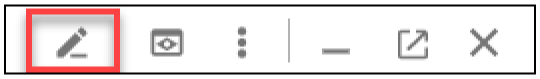
In this lab, you deploy the Quiz application into Kubernetes Engine, leveraging Google Cloud resources, including Container Builder and Container Registry, and Kubernetes resources, such as Deployments, Pods, and Services.

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.

Launch the Cloud Shell code editor

From Cloud Shell, click **Launch the code editor** icon (looks like a pencil) to launch the code editor.



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

1- 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**

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.

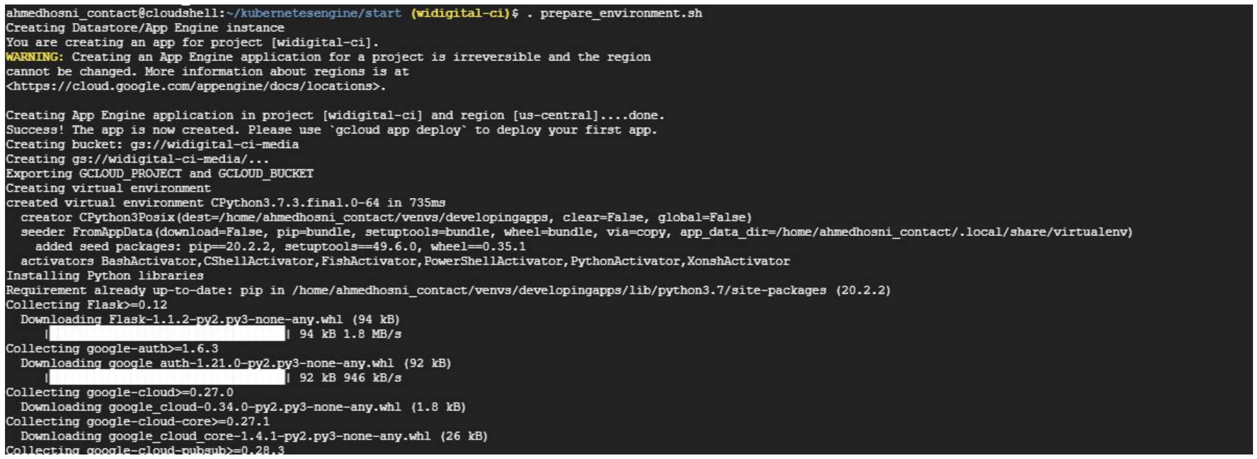
```
ln -s ~/training-data-analyst/courses/developingapps/v1.2/python/kubernetesengine
~/kubernetesengine
```

- **Configure the Quiz application**

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

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

Configure the Quiz application.



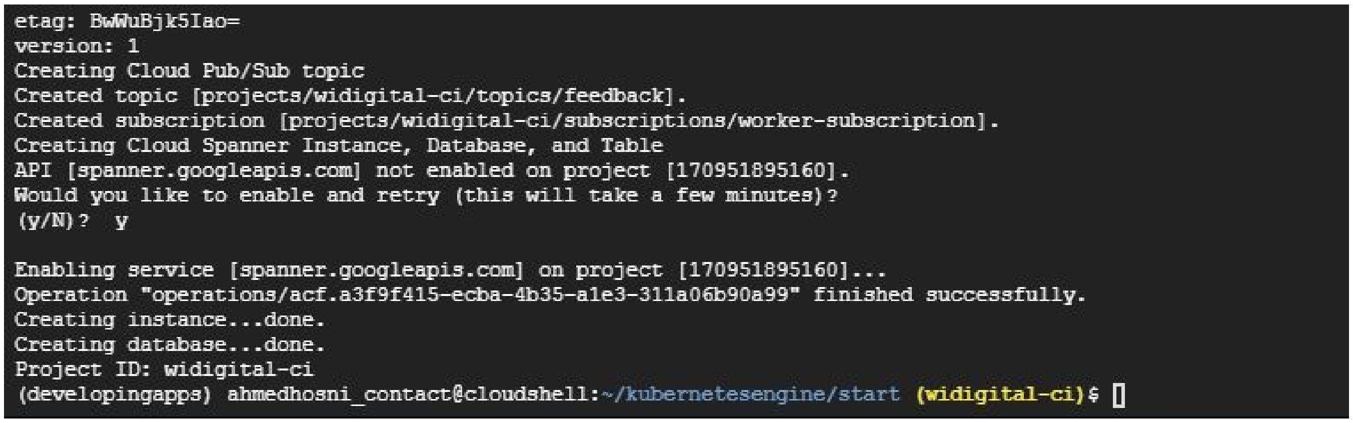
```
ahmedhosni_contact@cloudshell:~/kubernetesengine/start (widigital-ci)$ . prepare_environment.sh
Creating Datastore/App Engine instance
You are creating an app for project [widigital-ci].
WARNING: Creating an App Engine application for a project is irreversible and the region
cannot be changed. More information about regions is at
<https://cloud.google.com/appengine/docs/locations>.

Creating App Engine application in project [widigital-ci] and region [us-central]....done.
Success! The app is now created. Please use `gcloud app deploy` to deploy your first app.
Creating bucket: gs://widigital-ci-media
Creating gs://widigital-ci-media/...
Exporting GCLOUD_PROJECT and GCLOUD_BUCKET
Creating virtual environment
created virtual environment CPython3.7.3.final.0-64 in 735ms
  creator CPython3Posix(dest=/home/ahmedhosni_contact/venvs/developingapps, clear=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=/home/ahmedhosni_contact/.local/share/virtualenv)
    added seed packages: pip==20.2.2, setuptools==49.6.0, wheel==0.35.1
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator
Installing Python libraries
Requirement already up-to-date: pip in /home/ahmedhosni_contact/venvs/developingapps/lib/python3.7/site-packages (20.2.2)
Collecting Flask>=0.12
  Downloading Flask-1.1.2-py2.py3-none-any.whl (94 kB)
     |████████████████████████████████| 94 kB 1.8 MB/s
Collecting google-auth>=1.6.3
  Downloading google_auth-1.21.0-py2.py3-none-any.whl (92 kB)
     |████████████████████████████████| 92 kB 946 kB/s
Collecting google-cloud>=0.27.0
  Downloading google_cloud-0.34.0-py2.py3-none-any.whl (1.8 kB)
Collecting google-cloud-core>=0.27.1
  Downloading google_cloud_core-1.4.1-py2.py3-none-any.whl (26 kB)
Collecting google-cloud-pubsub>=0.28.3
```

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:



```
etag: BwWuBjk5Iao=
version: 1
Creating Cloud Pub/Sub topic
Created topic [projects/widigital-ci/topics/feedback].
Created subscription [projects/widigital-ci/subscriptions/worker-subscription].
Creating Cloud Spanner Instance, Database, and Table
API [spanner.googleapis.com] not enabled on project [170951895160].
Would you like to enable and retry (this will take a few minutes)?
(y/N)?  y

Enabling service [spanner.googleapis.com] on project [170951895160]...
Operation "operations/acf.a3f9f415-ecba-4b35-a1e3-311a06b90a99" finished successfully.
Creating instance...done.
Creating database...done.
Project ID: widigital-ci
(developingapps) ahmedhosni_contact@cloudshell:~/kubernetesengine/start (widigital-ci)$ 
```

2- 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.

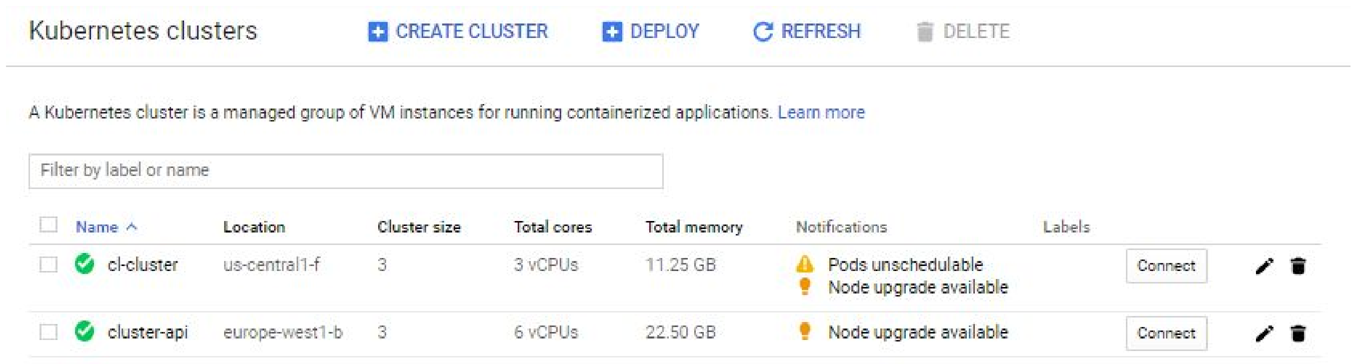
Examine the code

- Navigate to `training-data-analyst/courses/developingapps/v1.2/python/kubernetes engine/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.

3- Connect to the cluster

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

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



A Kubernetes cluster is a managed group of VM instances for running containerized applications. Learn more

Name	Location	Cluster size	Total cores	Total memory	Notifications	Labels
✓ cl-cluster	us-central1-f	3	3 vCPUs	11.25 GB	Pods unschedulable Node upgrade available	
✓ cluster-api	europe-west1-b	3	6 vCPUs	22.50 GB	Node upgrade available	

2. 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.

4- 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

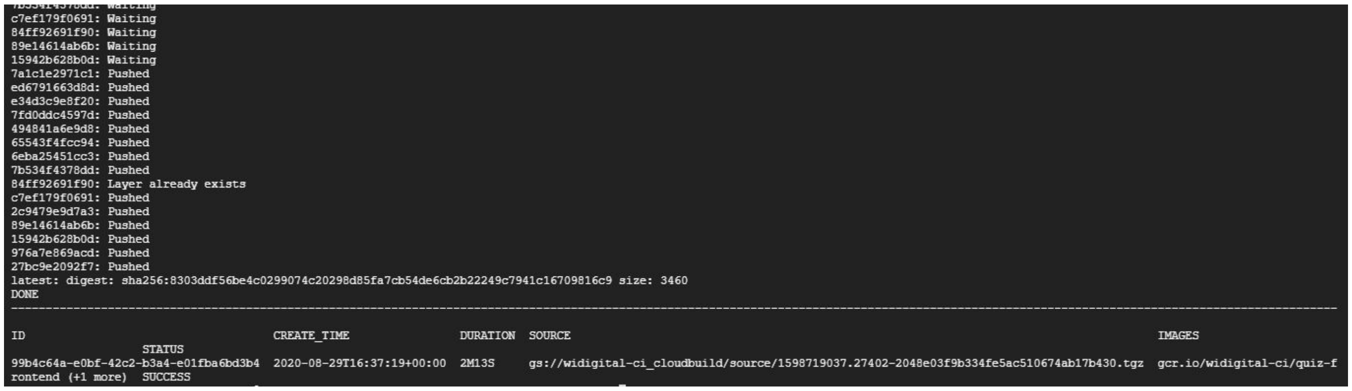
1- In Cloud Shell, make sure you are in the `start` folder:

```
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.



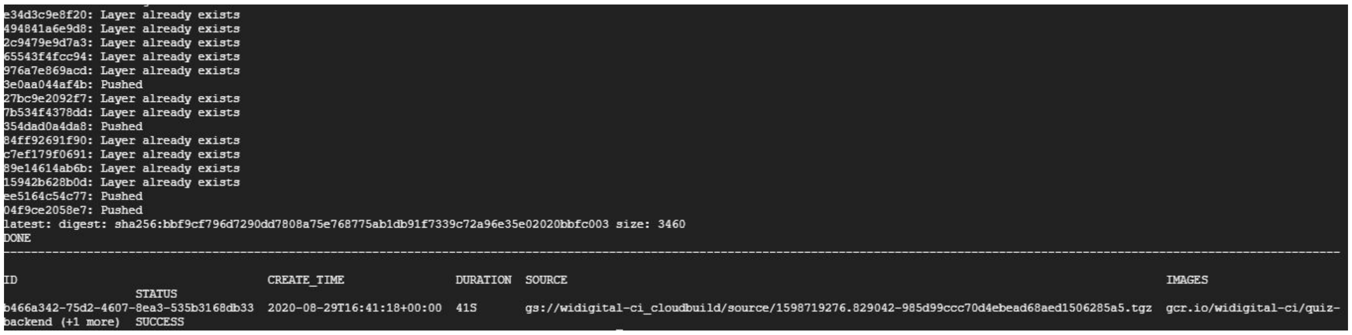
```
c7ef179f0691: Waiting
84ff92691f90: Waiting
89e14614ab6b: Waiting
15942b628b0d: Waiting
7a1c1e2971c1: Pushed
ed6791663d8d: Pushed
e34d3c9e8f20: Pushed
7fd0ddc4597d: Pushed
494841a6e9d8: Pushed
65543f4fcc94: Pushed
6eba25451cc3: Pushed
7b534f4378dd: Pushed
84ff92691f90: Layer already exists
c7ef179f0691: Pushed
2c9479e9d7a3: Pushed
89e14614ab6b: Pushed
15942b628b0d: Pushed
976a7e869acd: Pushed
27bc9e2092f7: Pushed
latest: digest: sha256:8303ddf56be4c0299074c20298d85fa7cb54de6cb2b22249c7941c16709816c9 size: 3460
DONE
------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ID                                    CREATE_TIME                DURATION  SOURCE                                                                                          IMAGES
                   STATUS
99b4c64a-e0bf-42c2-b3a4-e01fba6bd3b4  2020-08-29T16:37:19+00:00  2M13S     gs://widigital-ci_cloudbuild/source/1598719037.27402-2048e03f9b334fe5ac510674ab17b430.tgz  gcr.io/widigital-ci/quiz-f
rontend (+1 more)  SUCCESS
```

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

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

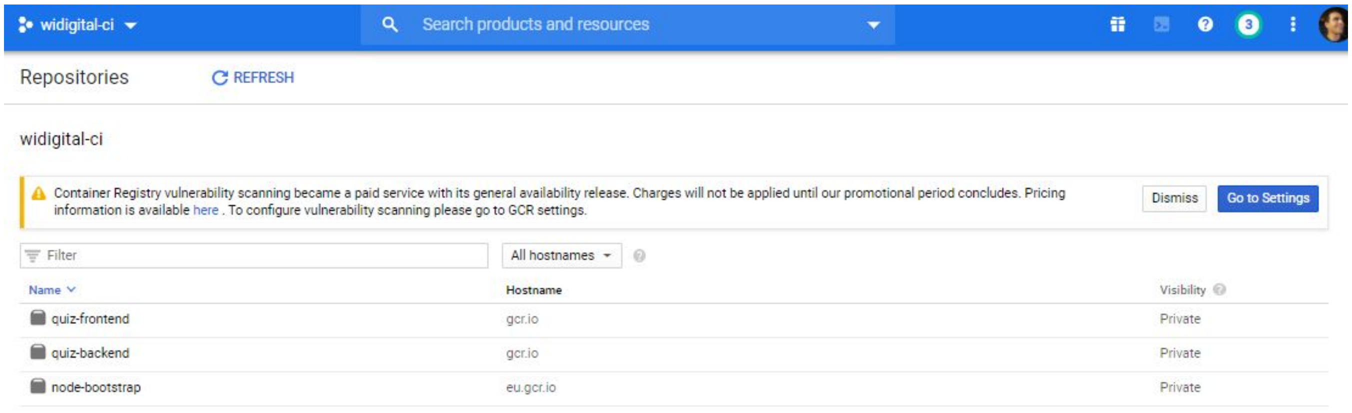
When the backend Docker image is ready you see these last messages:



```
e34d3c9e8f20: Layer already exists
494841a6e9d8: Layer already exists
2c9479e9d7a3: Layer already exists
65543f4fcc94: Layer already exists
976a7e869acd: Layer already exists
3e0aa044af4b: Pushed
27bc9e2092f7: Layer already exists
7b534f4378dd: Layer already exists
354dad0a4da8: Pushed
84ff92691f90: Layer already exists
c7ef179f0691: Layer already exists
89e14614ab6b: Layer already exists
15942b628b0d: Layer already exists
ee5164c54c77: Pushed
04f9ce2058e7: Pushed
latest: digest: sha256:bbf9cf796d7290dd7808a75e768775ab1db91f7339c72a96e35e02020bbfc003 size: 3460
DONE
------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ID                                    CREATE_TIME                DURATION  SOURCE                                                                                           IMAGES
                   STATUS
b466a342-75d2-4607-8ea3-535b3168db33  2020-08-29T16:41:18+00:00  41S       gs://widigital-ci_cloudbuild/source/1598719276.829042-985d99ccc70d4ebead68aed1506285a5.tgz  gcr.io/widigital-ci/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`.



widigital-ci

Container Registry vulnerability scanning became a paid service with its general availability release. Charges will not be applied until our promotional period concludes. Pricing information is available here . To configure vulnerability scanning please go to GCR settings.

Name	Hostname	Visibility
quiz-frontend	gcr.io	Private
quiz-backend	gcr.io	Private
node-bootstrap	eu.gcr.io	Private

5- Click **quiz-frontend**.

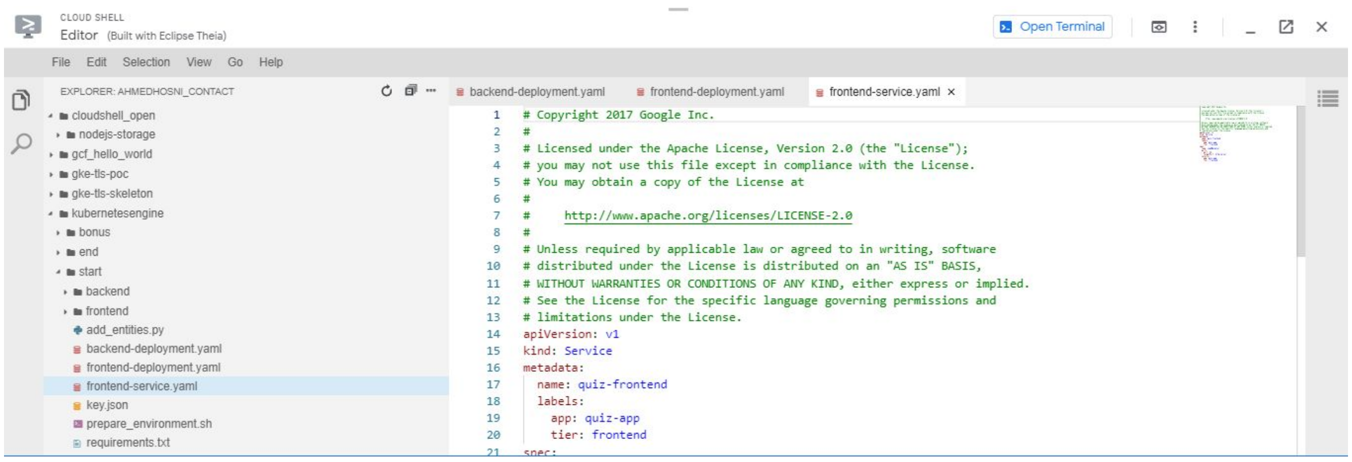
5- Create Kubernetes Deployment and Service Resources

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

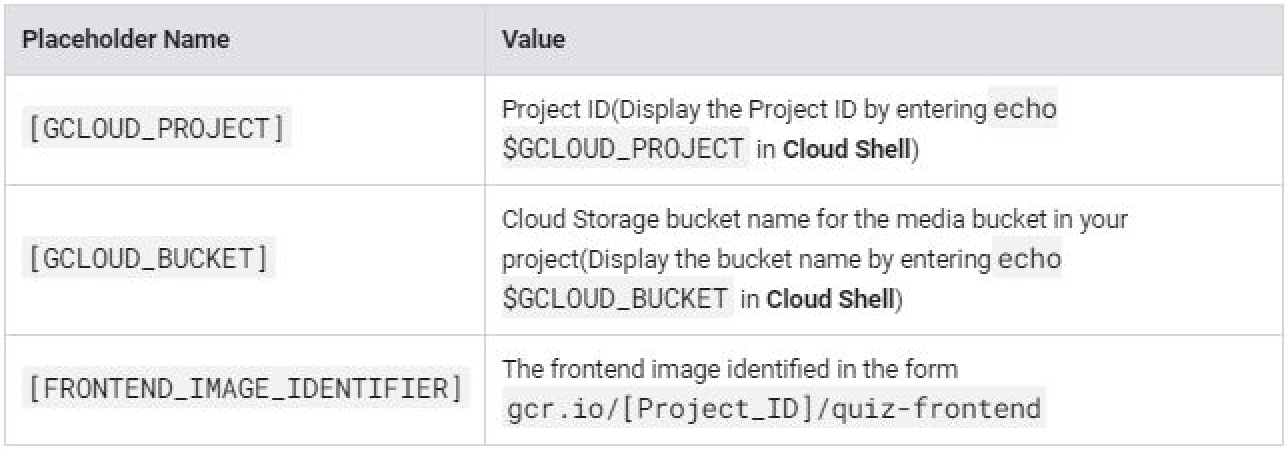
1. In the **Cloud Shell** code editor, open the `frontend-deployment.yaml` file.

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



```
# Copyright 2017 Google Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
apiVersion: v1
kind: Service
metadata:
  name: quiz-frontend
  labels:
    app: quiz-app
    tier: frontend
spec:
```

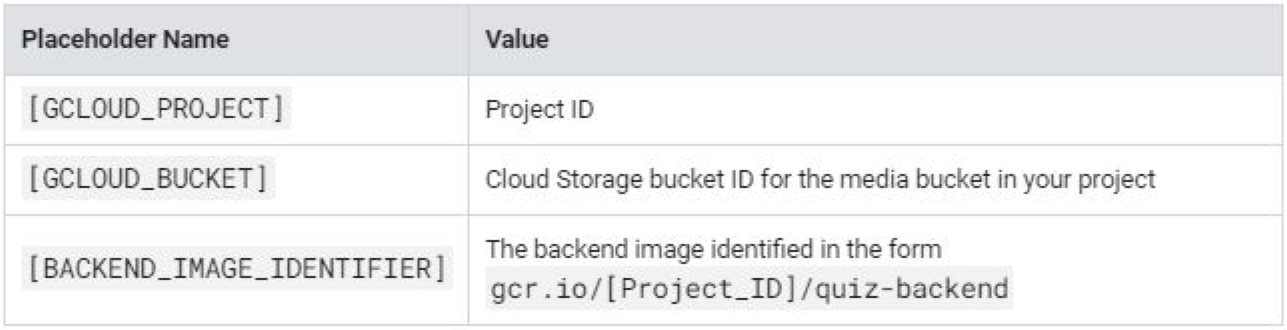
2. Replace the placeholders in the `frontend-deployment.yaml` file using the following values:



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.

6- 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.

7- Test the Quiz Application

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

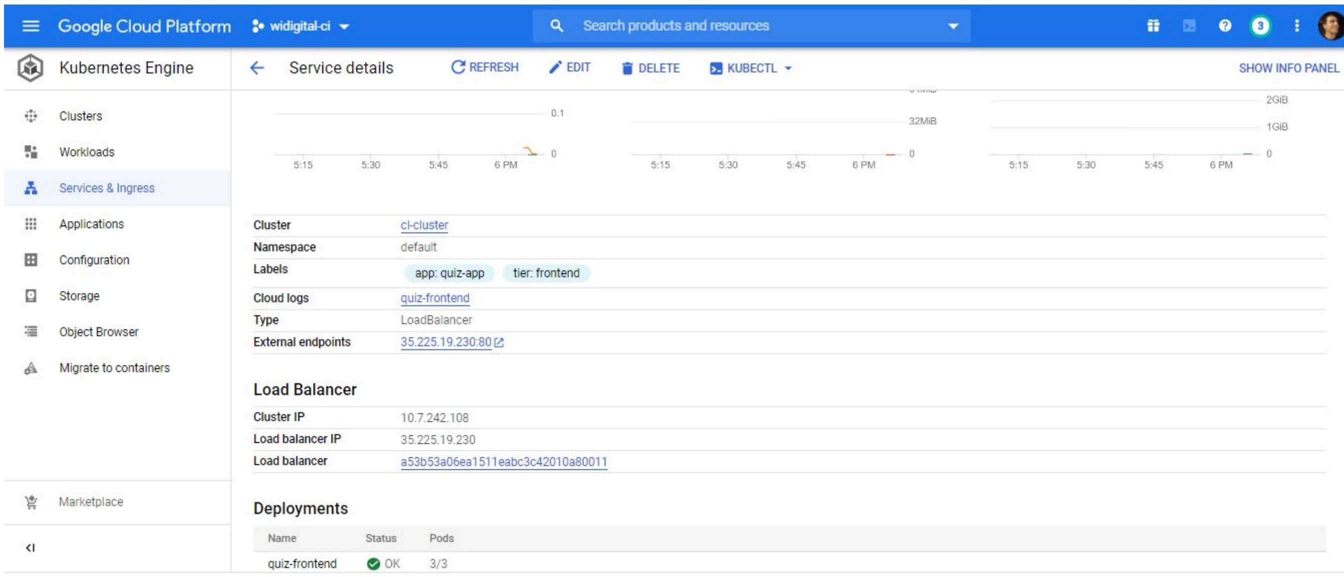
Review the deployed resources

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:



Cluster	cl-cluster
Namespace	default
Labels	app: quiz-app tier: frontend
Cloud logs	quiz-frontend
Type	LoadBalancer
External endpoints	35.225.19.230:80

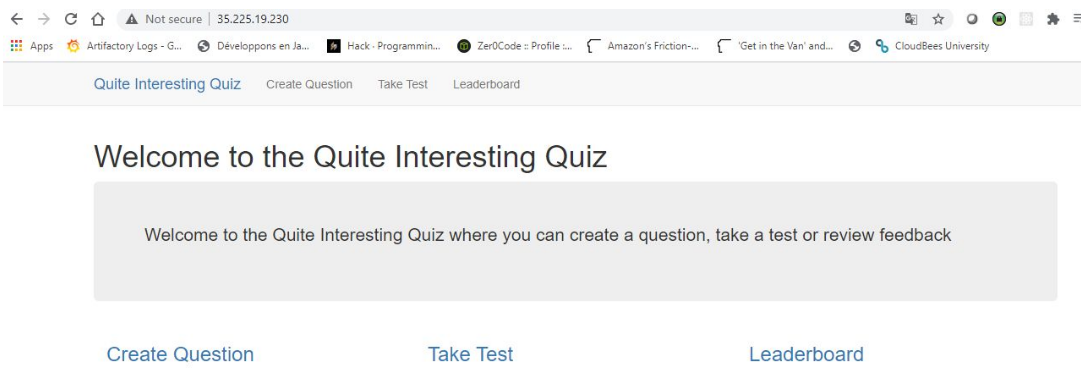
Load Balancer

Cluster IP	10.7.242.108
Load balancer IP	35.225.19.230
Load balancer	a53b53a06ea1511eabc3c42010a80011

Deployments

Name	Status	Pods
quiz-frontend	OK	3/3

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.



Welcome to the Quite Interesting Quiz

Welcome to the Quite Interesting Quiz where you can create a question, take a test or review feedback

Create Question Take Test Leaderboard