Deploying an application into Kubernetes Engine

# References and support

This is a readapted tutorial from Qwiklabs, if you want to see the original one, click on this [link](https://www.qwiklabs.com/focuses/1061?catalog_rank=%7B%22rank%22%3A1%2C%22num_filters%22%3A0%2C%22has_search%22%3Atrue%7D&parent=catalog&search_id=14671174).

Working with the cloud platforms is difficult because they change frequently, so this tutorial couldn’t be updated. For any problem, please contact us at this email: tcc@uniupo.it.

# Overview

This tutorial has the purpose of Deploying a Java application on Google Kubernetes Engine.

Kubernetes is an environment that allows the deployment and the management of the applications using Google infrastructure to make it scalable and containerized.

To do this, Kubernetes provides multiple machines grouped to form a cluster, also it provides a mechanism for interaction with it.

# What we need for this project

* First of all, we need an application to deploy. It’s better this application is a versioned project. If you haven’t got a project, you can use this one:
* A Linux shell or the Google cloud shell, we use the last one for this tutorial because it has an embedded editor and.

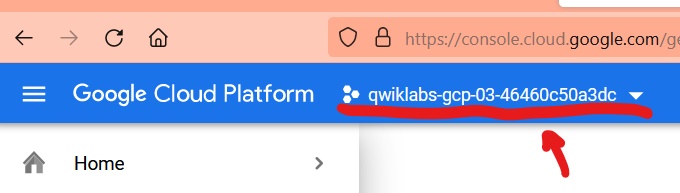
# Tutorial

## 1 - Create a Google Cloud Project

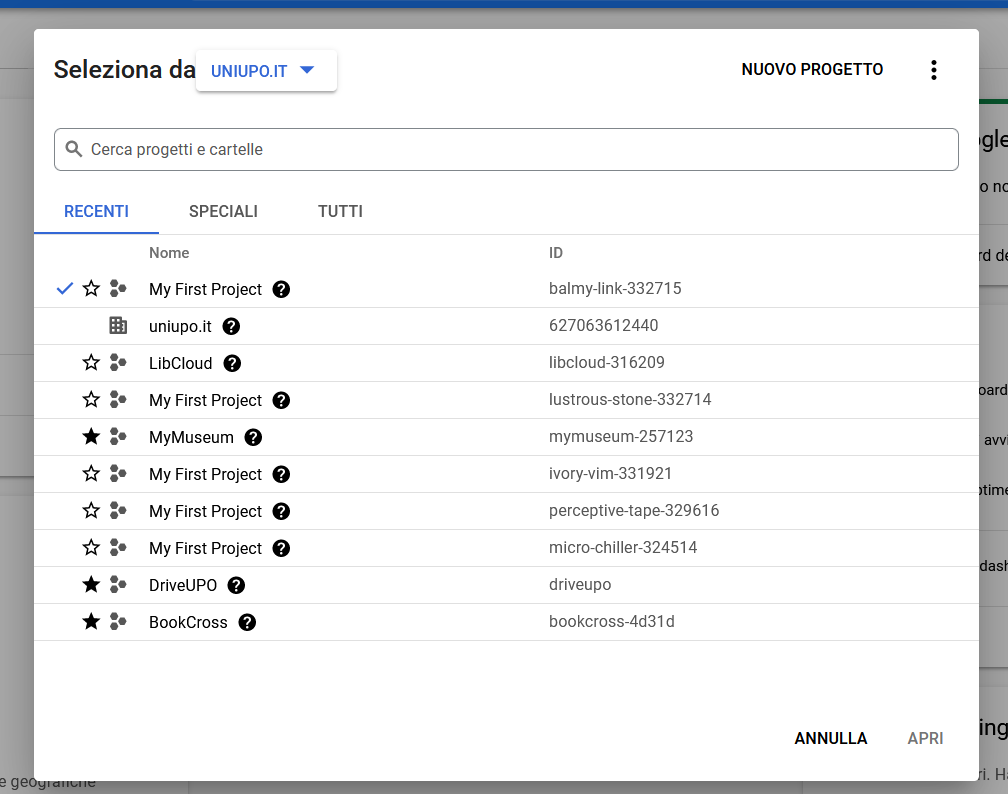
This step is optional for users who already have a project, so you can skip it if you have one.

For creating a project:

* Go to the **Cloud console** page;
* Click the dropdown menu (it is on the left of “Google Cloud Platform”);



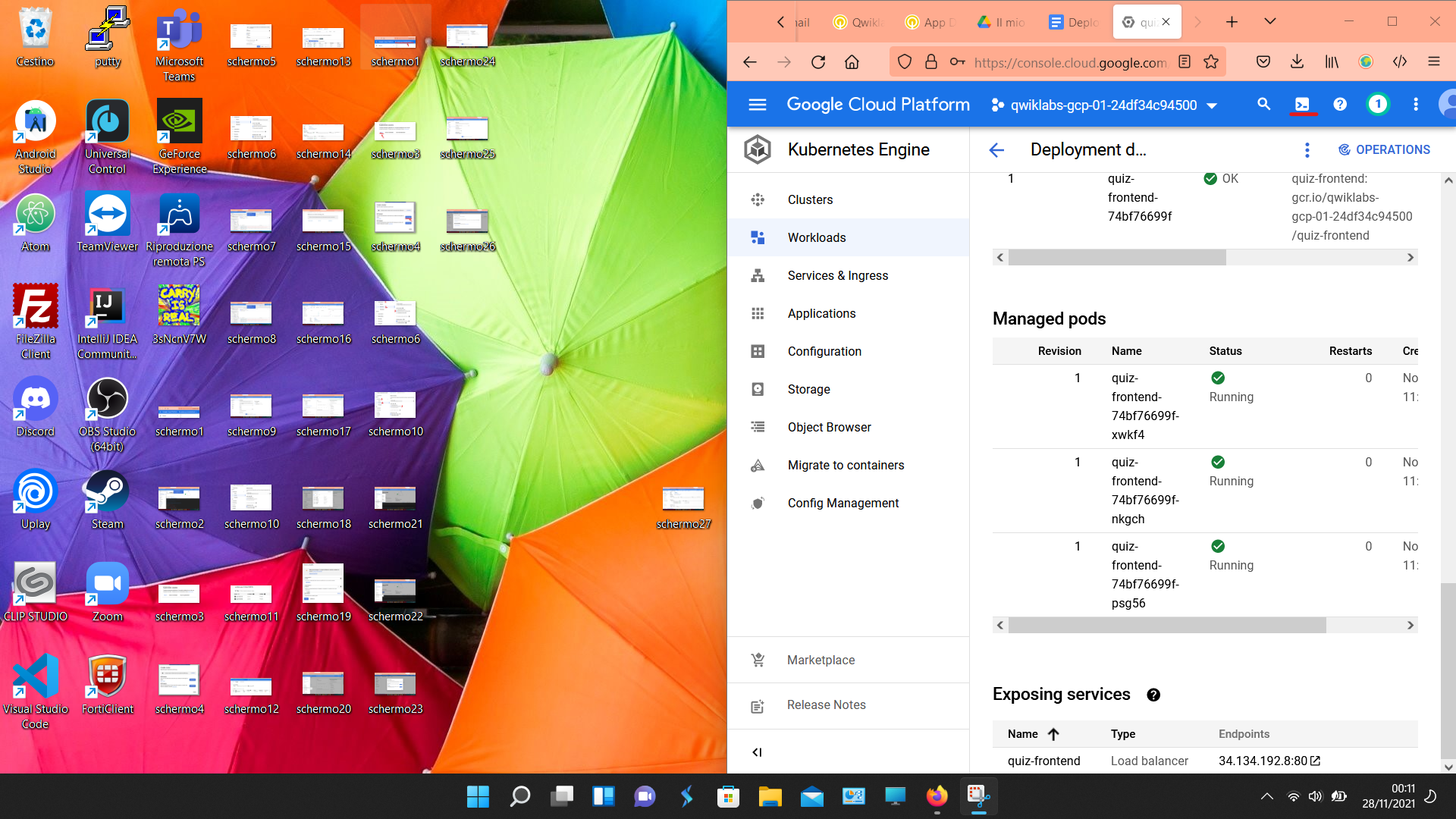
* Click on **New Project**;
* fill all the fields and click **Create**;
* Click the dropdown menu again;
* Select the project you created.



## 2 - Open Cloud Shell and Editor

As I said before, in this tutorial we’ll use the Google Cloud Shell, an embedded command-line interface embedded in the Cloud Console. The use of this shell is only a convenience, the tutorial will work with a classic shell too with the help of Cloud SDK.

To open the Cloud Shell, you have to click on **Activate Cloud Shell** in the navbar.



After the opening of the shell, click on **Open editor** to open the Cloud Editor. The cloud editor is a useful tool that allows the navigation of the file system using the browser.

To come back to the shell, click on **Open terminal**.

## 3 - Download the project

If your application is versioned with Github or similar, you can download in your machine using this command:

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

## 4 - Preparation of the environment

In the project we use for this tutorial, the setup of the environment is done with a bash script, so we have to find it navigating the file system and then execute it:

cd ~/training-data-analyst/courses/developingapps/java/kubernetesengine/start

./prepare\_environment.sh

If you haven’t this bash file or it doesn’t work, you have to do the preparation manually:

1. Create the **App Engine** app:

gcloud app create --region "us-central"

1. Make the bucket:

gsutil mb gs://$DEVSHELL\_PROJECT\_ID-media

1. Export two environment variables, they are project id and the bucket id:

export GCLOUD\_BUCKET=$DEVSHELL\_PROJECT\_ID-media

export GCLOUD\_PROJECT=$DEVSHELL\_PROJECT\_ID

1. Install the dependencies:

mvn clean install

1. Create datastore entities:

mvn exec:java@create-entities

1. Create a pub/sub topic:

gcloud beta pubsub topics create feedback

1. Create a **Cloud Spanner** instance with a database and a table:

gcloud spanner instances create quiz-instance --config=regional-us-central1 --description="Quiz app" --nodes=1

gcloud spanner databases create quiz-database --instance quiz-instance --ddl "CREATE TABLE Feedback ( feedbackId STRING(100) NOT NULL, email STRING(100), quiz STRING(20), feedback STRING(MAX), rating INT64, score FLOAT64, timestamp INT64 ) PRIMARY KEY (feedbackId);"

After the preparation of the environment, check if GCLOUD\_PROJECT and GCLOUD\_BUCKET are setted.

echo $GCLOUD\_PROJECT

echo $GCLOUD\_BUCKET

If these two environment variables are not set, export them manually using the third step above.

## 5 - Review the code

To review the code, we’ll use the cloud editor, but if you want, you can use shell editors like vim.

Navigate to training-data-analyst/courses/developingapps/java/kubernetesengine/start , then:

* copy the file quiz-frontend-0.0.1.jar (it is in the folder target) into the folder frontend;

cp ./target/quiz-frontend-0.0.1.jar ./frontend/

* Configure the backend application;

mvn package -f pom-backend.xml

* copy the output jar file (it is in the target folder too) to the backend folder.

cp ./target/quiz-backend-0.0.1.jar ./backend/

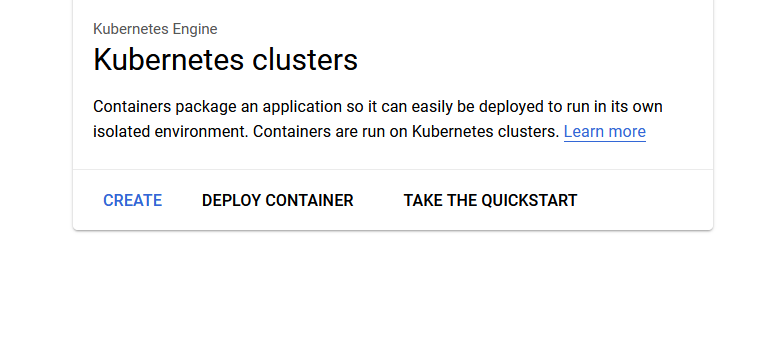
## 6 - Create the Kubernetes Engine Cluster

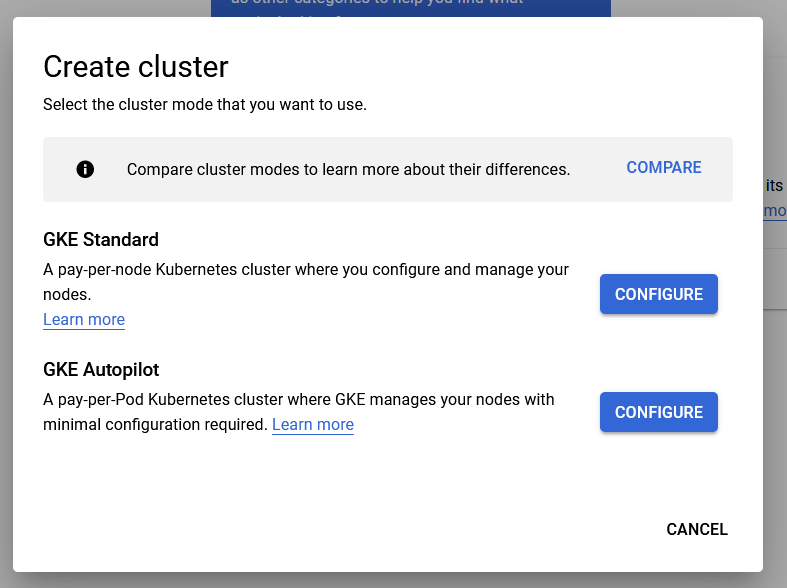
Now we will create a cluster using Kubernetes! From the console:

1. Click the **Navigation menu**;
2. Search and click on **Kubernetes Engine**;
3. Click on **Clusters**.

## 

After these operations, it will be shown a web page like this:

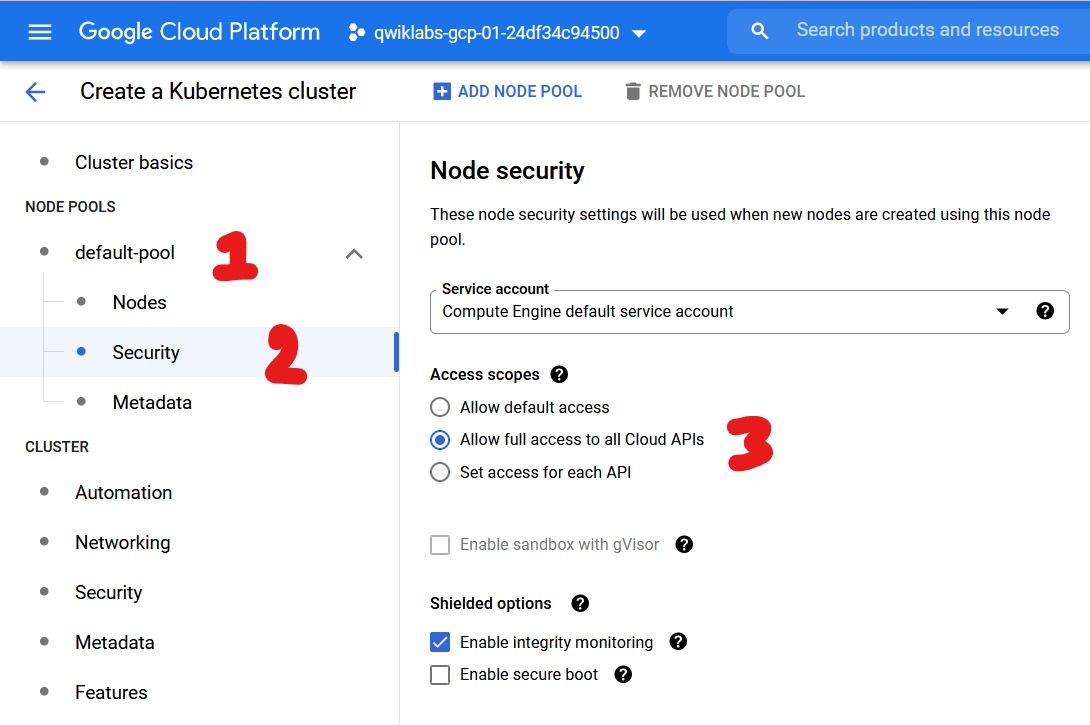




So, Click on **Create**, then click on **Configure** in **GKE Standard** section and fill the fields:

* in the name field, write **quiz-cluster**;
* in the zone field, search and click on **us-central1-b**.

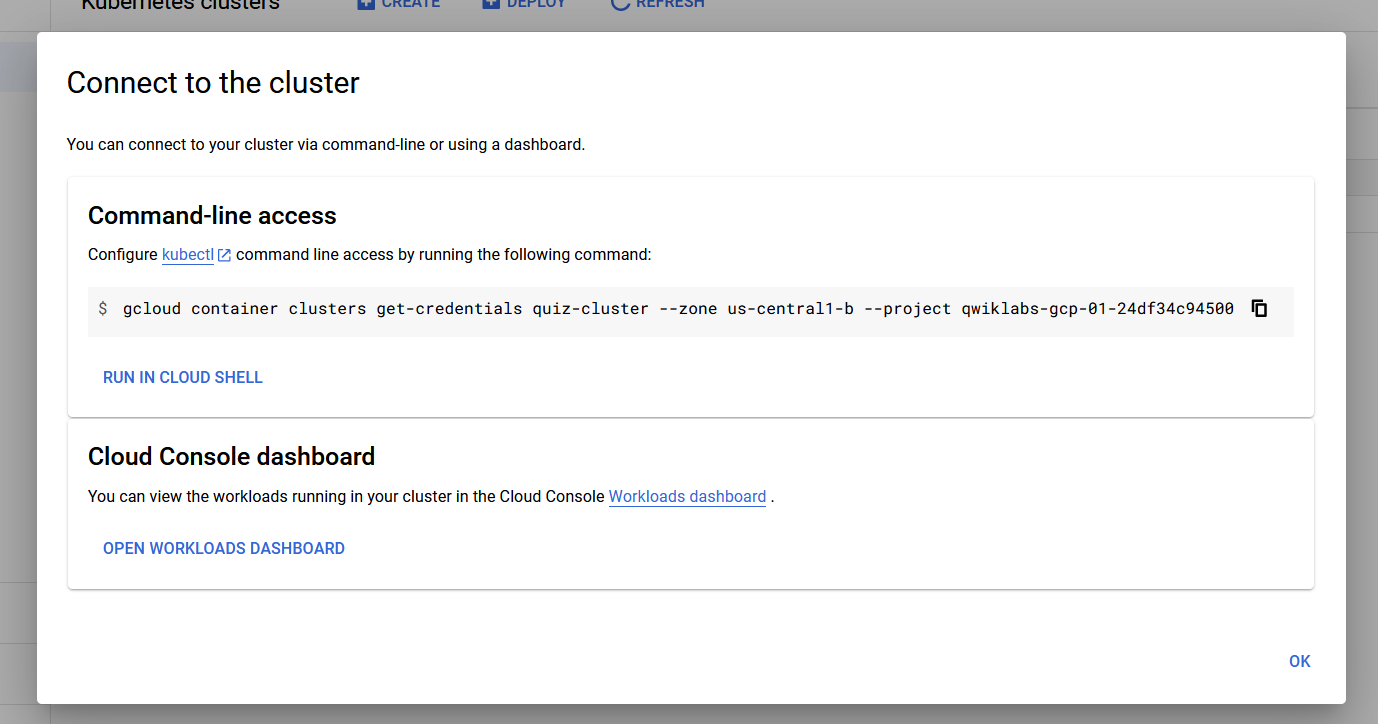
Now, click on **default-pool**, select **Security** and check **Allow full access to all Cloud APIs** in the **Access scopes** section.



Last but not least, click on **Create** to create the cluster, it will become available after a little amount of time.

## 7 - Connect to the cluster

After the cluster is ready, click on the **three dots** and select **Connect**.



In the **Connect to the cluster** window, copy the first command and execute it in the Cloud shell.

Now use this command to list the pods in the cluster:

kubectl get pods

The response you have to receive is that there aren’t pods in the cluster. This is a confirmation that the security is configured well.

## 8 - Build Docker images

Now we build **Docker** images! To do this, we have to do these steps:

* From the Cloud Editor, navigate to

training-data-analyst/courses/developingapps/java/kubernetesengine/start

* Click on frontend and then on Dockerfile;
* Copy and paste the following text:

FROM gcr.io/google\_appengine/jetty9

VOLUME /tmp

ADD ./quiz-frontend-0.0.1.jar /app.jar

CMD java -jar /app.jar

* Click on backend and then on Dockerfile;
* Copy and paste the following text:

FROM gcr.io/google\_appengine/jetty9

VOLUME /tmp

ADD ./quiz-backend-0.0.1.jar /app.jar

CMD java -jar /app.jar

* Save the files.

After these steps, return on the shell and execute this command:

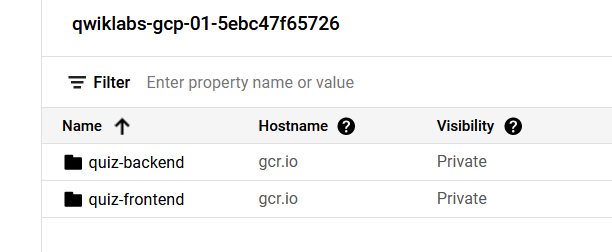
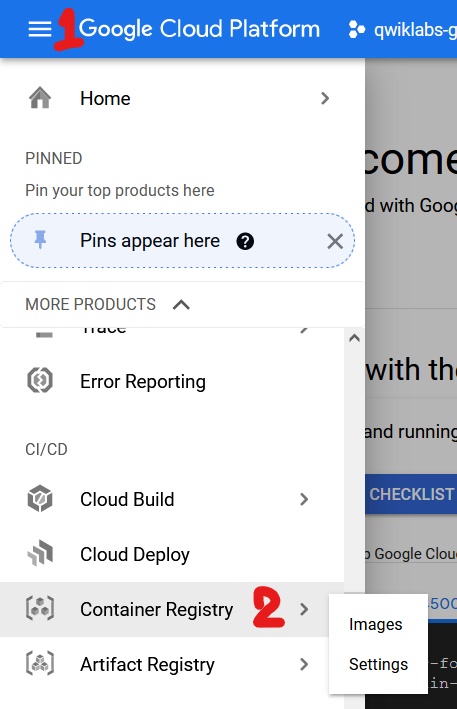
gcloud builds submit -t gcr.io/$DEVSHELL\_PROJECT\_ID/quiz-frontend ./frontend/

gcloud builds submit -t gcr.io/$DEVSHELL\_PROJECT\_ID/quiz-backend ./backend/

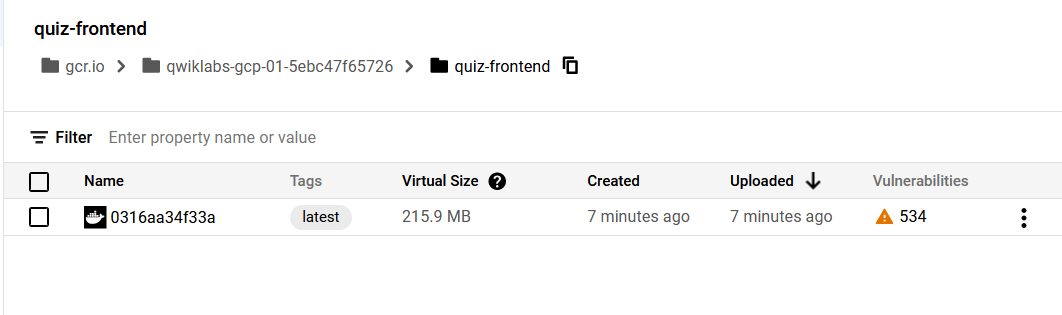
These commands create two files in the Cloud Storage, build and store a Docker image in the Container Registry.

To check if everything is done, click on **Navigation Menu** and then on **Container Registry**, you should see two folders:

* the first one is about the backend;
* the second one is about frontend;



If you click one of these folders, you should find one file, this is the **Docker** image.



## 9 - Create a Kubernetes deployment

In this step we’ll create a Kubernetes deployment, to do this we have to modify some .yaml files typing specific information fields.

First of all, navigate to training-data-analyst/courses/developingapps/java/kubernetesengine/start , then open the frontend-deployment.yaml file, here do the following things:

* replace [GCLOUD\_PROJECT] with your project id, you can find it with the following command:

echo $GCLOUD\_PROJECT

* replace [GCLOUD\_BUCKET] with your bucket id, you can find it with the following command:

echo $GCLOUD\_BUCKET

* replace [FRONTEND\_IMAGE\_IDENTIFIER] with gcr.io/<your-project-id>/quiz-frontend .
* save the file.

You have to do the same things in the backend-deployment.yaml, the only difference is that you have to digit this text in [BACKEND\_IMAGE\_IDENTIFIER]:

gcr.io/<your-project-id>/quiz-backend

After these operations, we have to execute the deployment using these commands:

kubectl create -f ./frontend-deployment.yaml

kubectl create -f ./backend-deployment.yaml

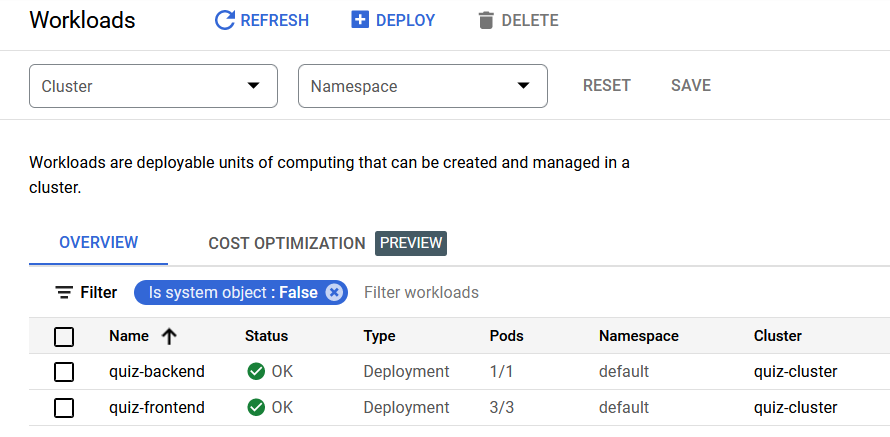
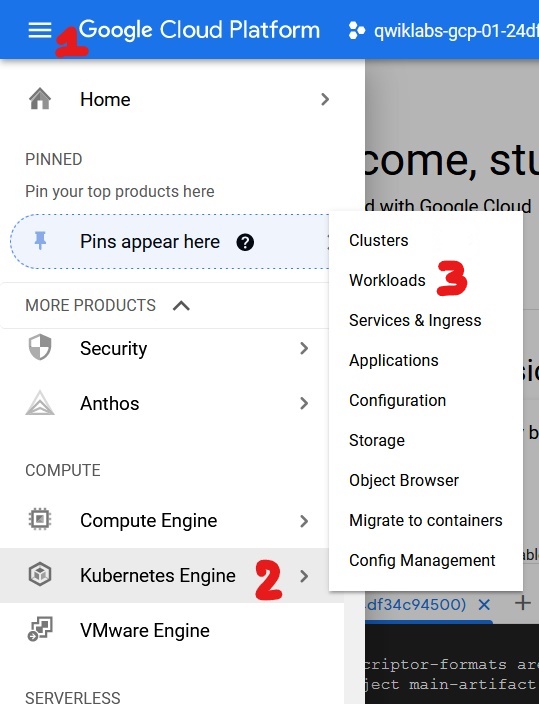
Then we provide the frontend service using this command:

kubectl create -f ./frontend-service.yaml

## 10 - Test your application

Now you can test your application! To do this:

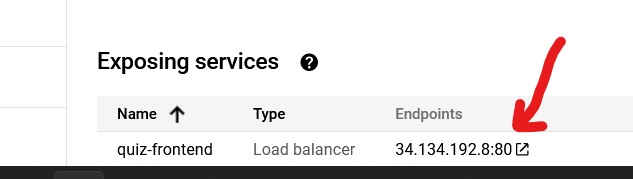
* click on Navigation menu;
* search and click Kubernetes Engine;
* click on Workloads.



Now click on frontend, in the webpage you’ll go you’ll find some information about the situation of the frontend:

* there are some graphics about the amount of traffic of your application;
* in the managed pods section, there are the status of every machine;
* In the exposing services section, a load balancer was created with an IP address nearby. If you don’t see the IP address, wait a little amount of time and/or reload the page.

To test your application, click on the IP address.



Now enjoy your application!