**Deploying Apps to Google Cloud**

Toggle Lab Panel



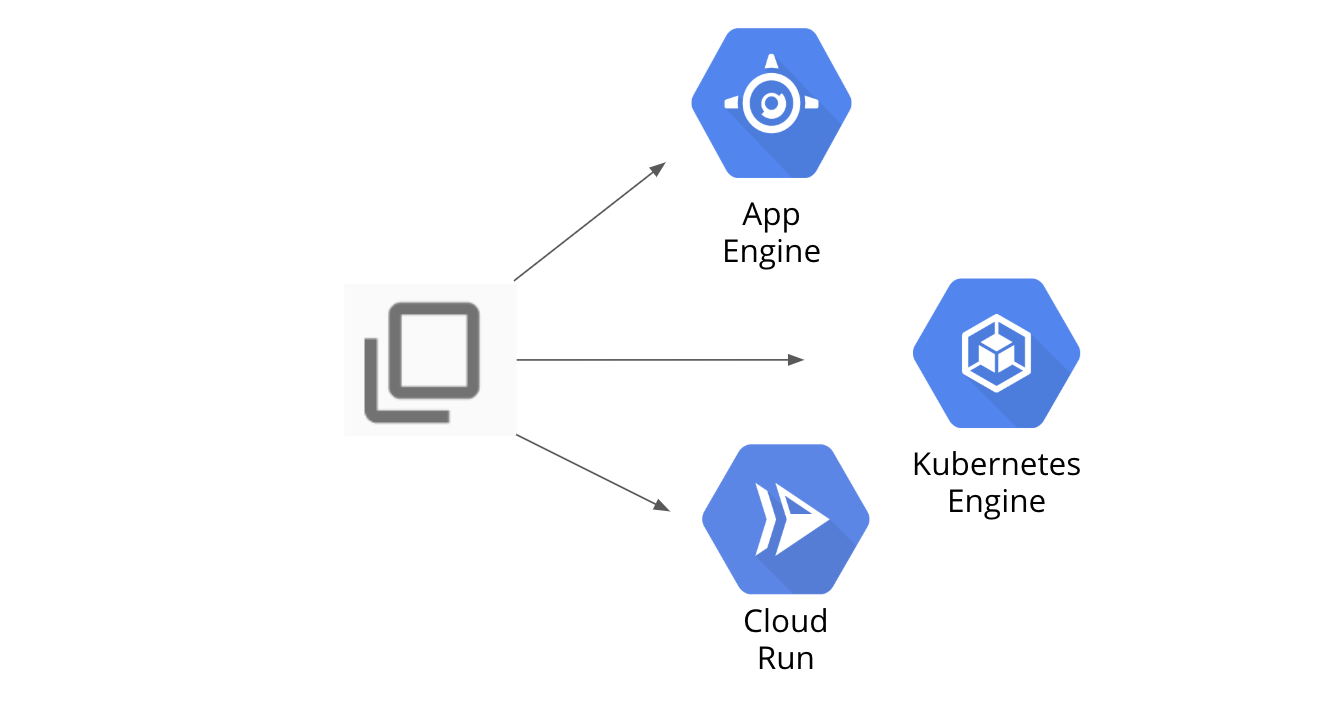
**Deploying Apps to Google Cloud**

1 hour 30 minutes Free

Rate Lab

**Overview**

In this lab, you will deploy applications to the Google Cloud services App Engine, Kubernetes Engine, and Cloud Run.



**Objectives**

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

* Download a sample app from GitHub
* Deploy to App Engine
* Deploy to Kubernetes Engine
* Deploy to Cloud Run

**Set up your lab environment**

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

1. Make sure you signed into Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, img/time.pngand make sure you can finish in that time block.

There is no pause feature. You can restart if needed, but you have to start at the beginning.

1. When ready, click img/start_lab.png.
2. Note your lab credentials. You will use them to sign in to Cloud Platform Console. 
3. Click **Open Google Console**.
4. Click **Use another account** and copy/paste credentials for **this** lab into the prompts.

If you use other credentials, you'll get errors or **incur charges**.

1. Accept the terms and skip the recovery resource page.

Do not click **End Lab** unless you are finished with the lab or want to restart it. This clears your work and removes the project.

**Task 1: Download a sample app from GitHub**

Download a sample application from GitHub and preview it in Cloud Shell.

1. In the Cloud Console, click **Activate Cloud Shell** (Cloud Shell).
2. If prompted, click **Continue**.
3. To create a new folder, run the following command:

mkdir gcp-course

1. Change to the folder you just created:

cd gcp-course

1. Clone a simple Python Flask app from GitHub:

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

1. Change to the deploying-apps-to-gcp folder:

cd training-data-analyst/courses/design-process/deploying-apps-to-gcp

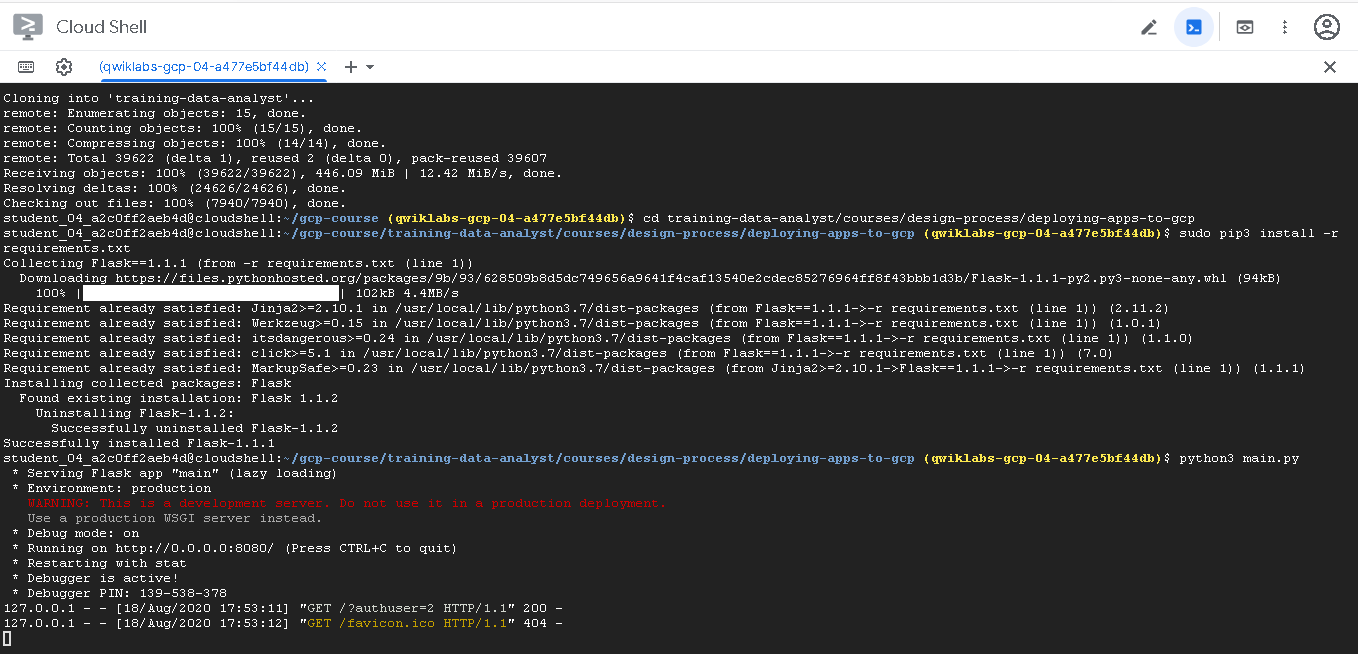
1. To test the program, enter the following command to install requirements and then start the program:

sudo pip3 install -r requirements.txt

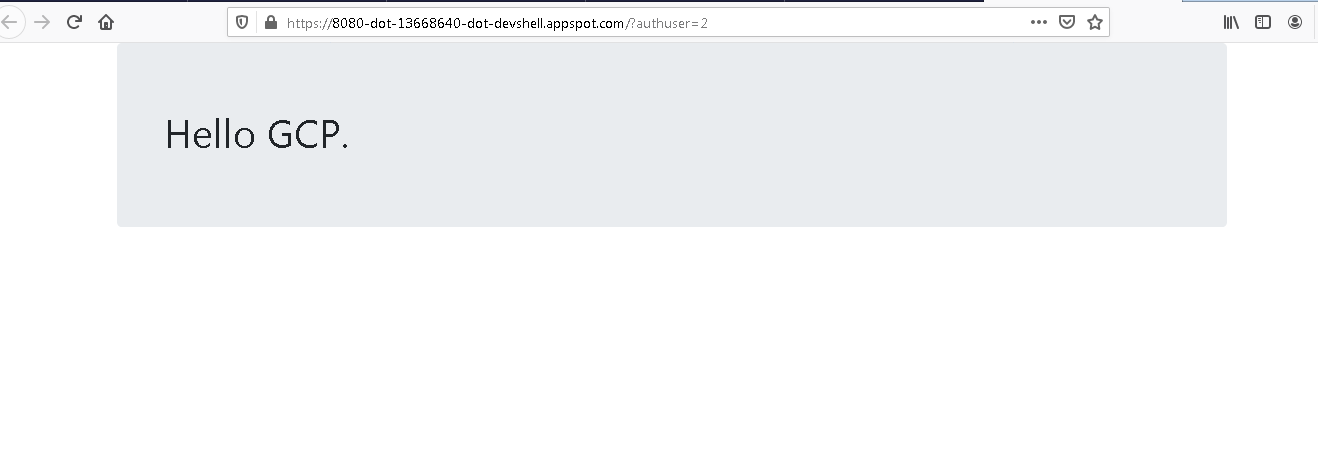
python3 main.py

1. To see the program running, click **Web Preview** (Web Preview) in the toolbar of Google Cloud Shell. Then, select **Preview on port 8080**.

The program should be displayed in a new browser tab.



1. In Cloud Shell, type **Ctrl+C** to stop the program.



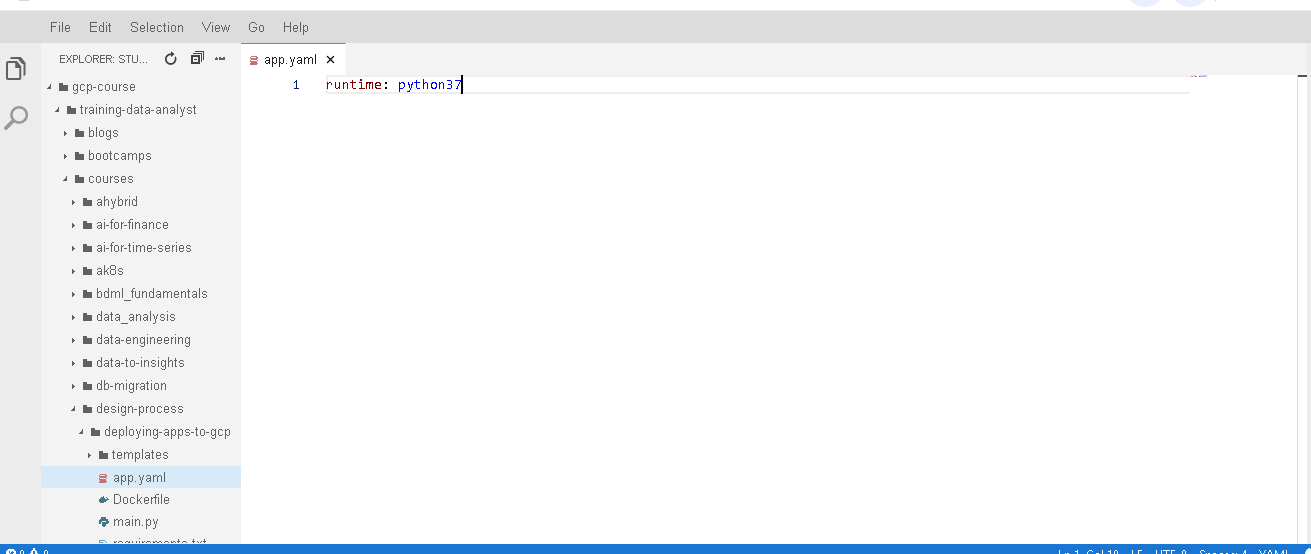
**Task 2: Deploy to App Engine**

App Engine is a completely automated deployment platform. It supports many languages, including Python, Java, JavaScript, and Go. To use it, you create a configuration file and deploy your applications with a couple of simple commands. In this task, you create a file named *app.yaml* and deploy it to App Engine.

1. In Cloud Shell, click **Open Editor** (Cloud Shell Editor).
2. Select the **training-data-analyst/courses/design-process/deploying-apps-to-gcp** folder in the explorer tree on the left.
3. From the **File** menu, select **New File**, and name the file **app.yaml**.
4. Paste the following into the file you just created:

runtime: python37

1. Save your changes.



**Note:**

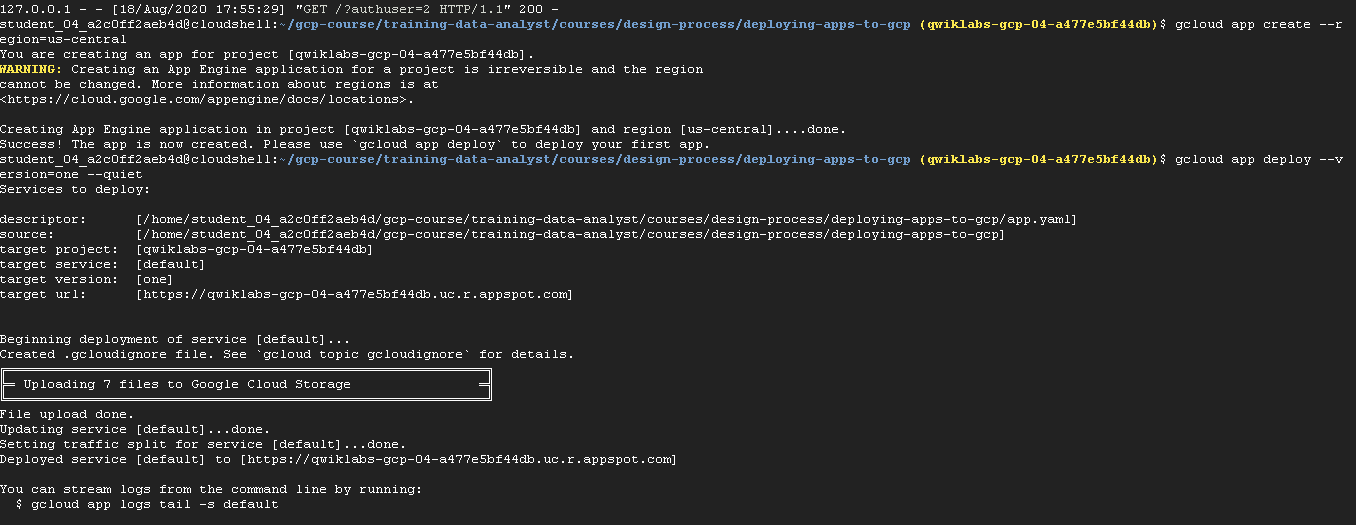
There are other settings you can add to the app.yaml file, but in this case only the language runtime is required.

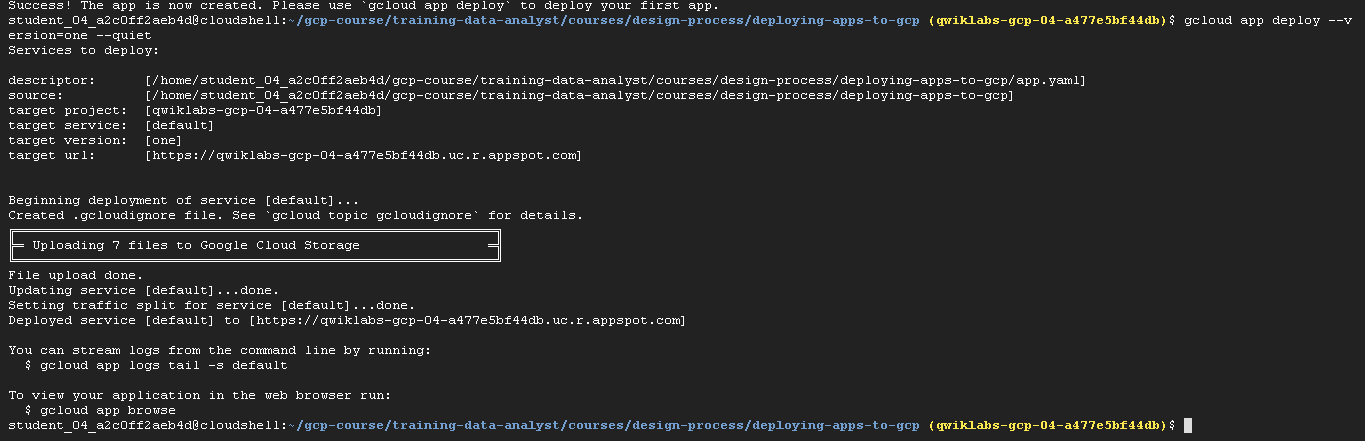
1. In a project, an App Engine application has to be created. This is done just once using the gcloud app create command and specifying the region where you want the app to be created. Enter the following command:

gcloud app create --region=us-central

1. Now deploy your app with the following command:

gcloud app deploy --version=one –quiet

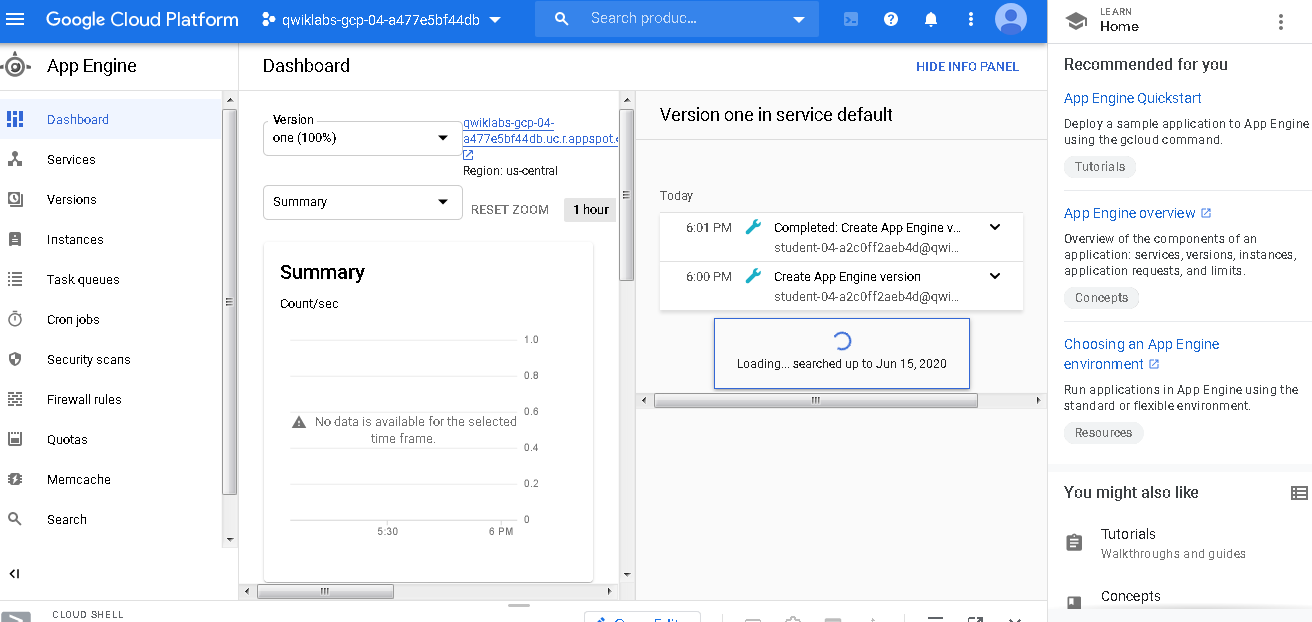


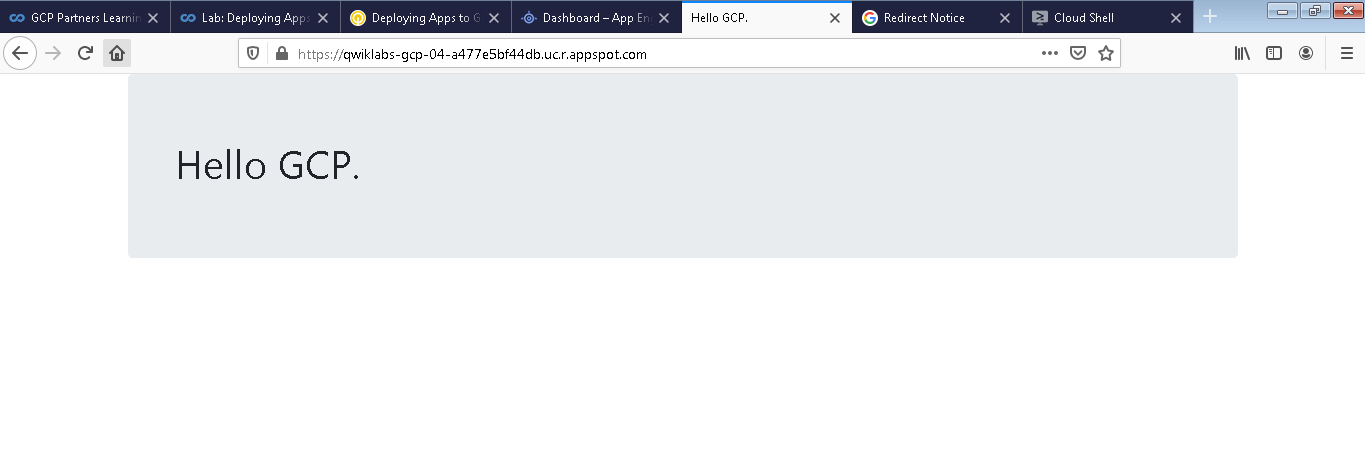


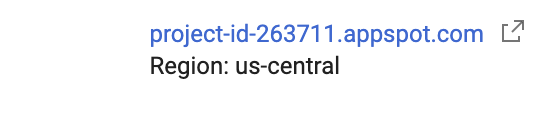
**Note:**

This command will take a couple of minutes to complete.

1. On the **Navigation menu** (Navigation menu), click **App Engine > Dashboard**. In the upper-right corner of the dashboard is a link to your application, similar to this:



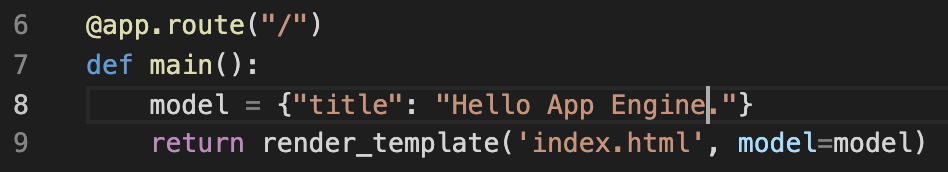




**Note:**

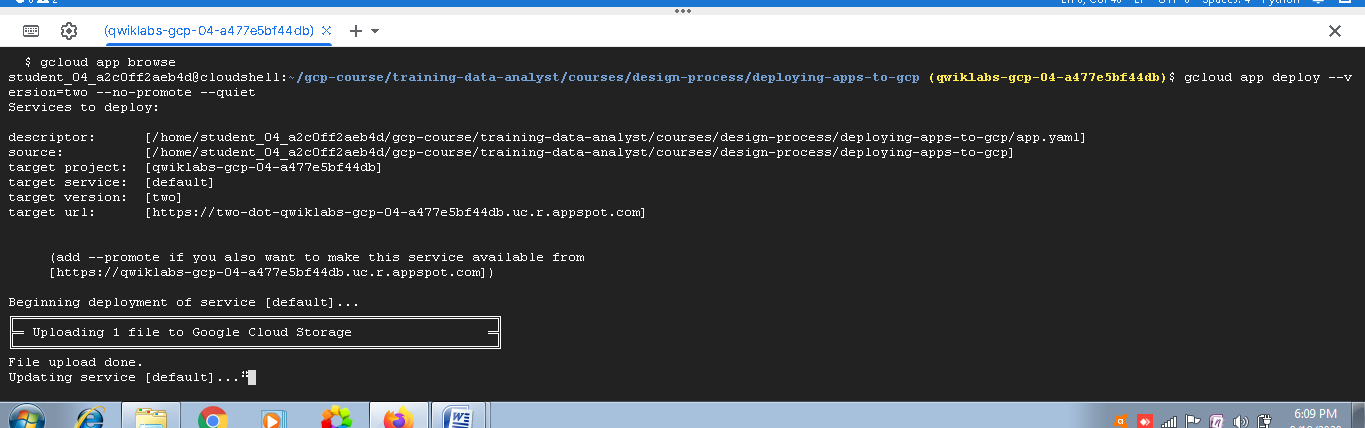
By default, the URL to an App Engine application is in the form of https://project-id.appspot.com.

1. Click on the link to test your program.
2. Make a change to the program to see how easy the App Engine makes managing versions.
3. In the code editor, expand the **training-data-analyst/courses/design-process/deploying-apps-to-gcp** folder in the navigation pane on the left. Then, click **main.py** to open it.
4. In the **main()** function, change the title to Hello App Engine as shown below:



1. Click **File > Save** in the code editor toolbar to save your change.
2. Now, deploy version two with the following command:

gcloud app deploy --version=two --no-promote –quiet



**Note:**

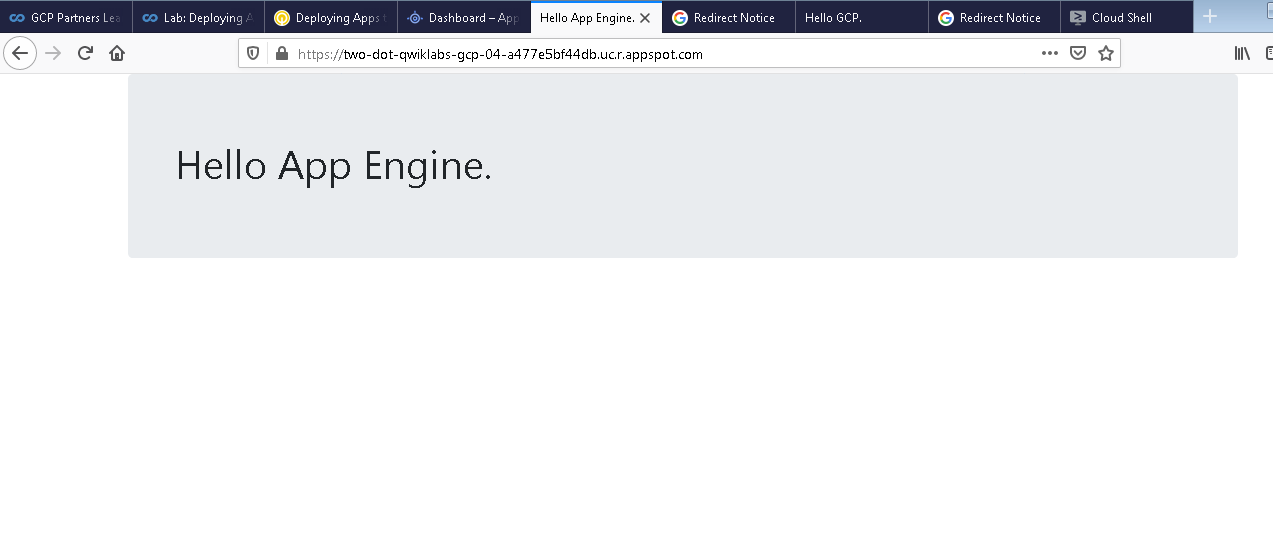
The --no-promote parameter tells App Engine to continue serving requests with the old version. This allows you to test the new version before putting it into production.

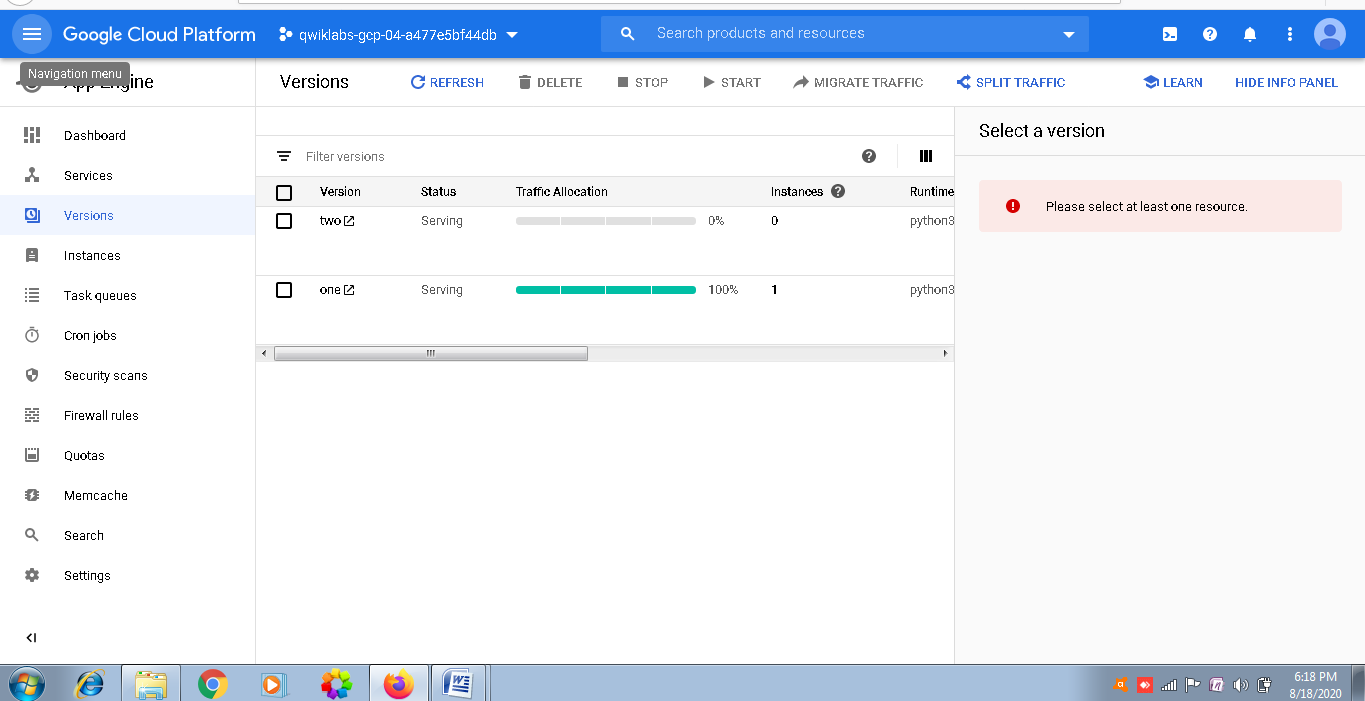
1. When the command completes, return to the App Engine dashboard. Click the link again, and version one will still be returned. It should return Hello GCP. This is because of the --no-promote parameter in the previous command.
2. On the left, click the **Versions** link. Notice that two versions are listed.

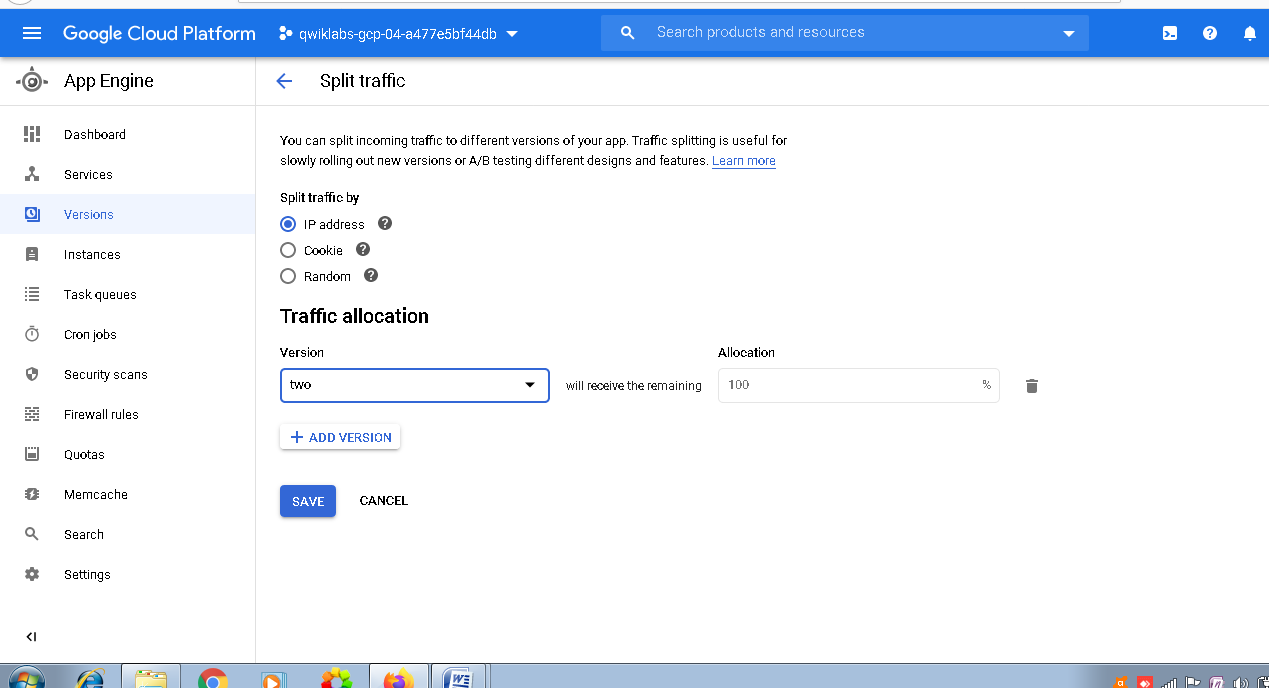
**Note:**

You might have to click **Refresh** to see version **two**.

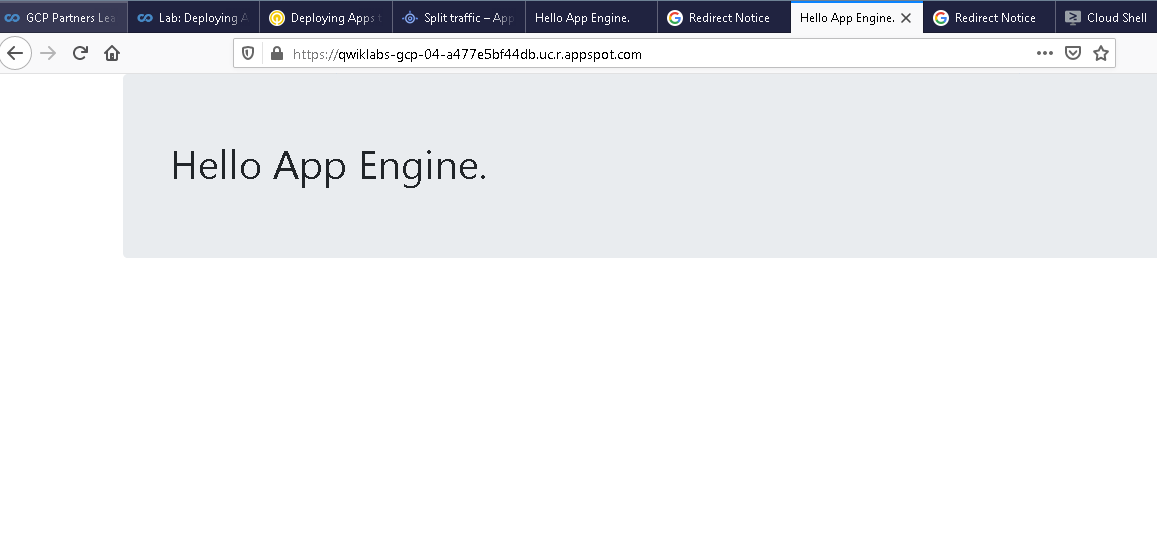
1. Click on the version two link to test it. It should return Hello App Engine.





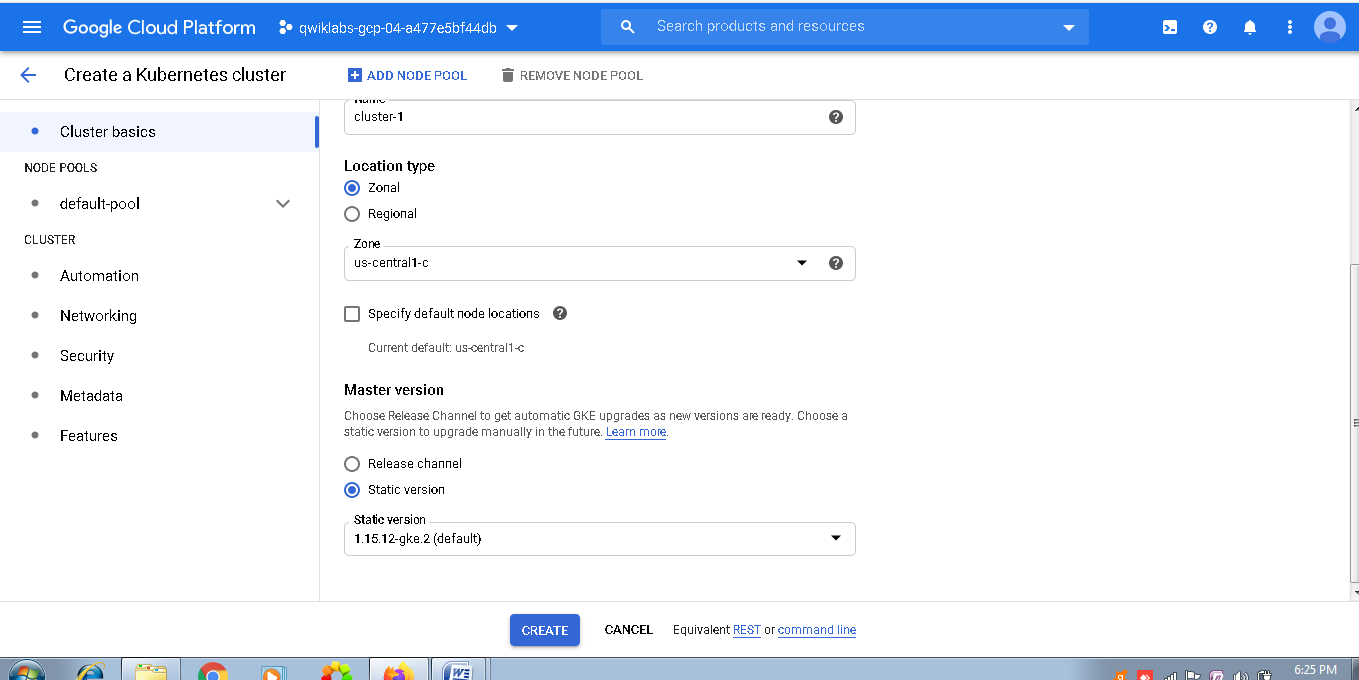


1. To migrate production traffic to version two, click **Split Traffic** at the top. Change the version to two, and click **Save**.
2. Give it a minute to complete. Refresh the browser tab that earlier returned Hello GCP. It should now return the new version.



Click *Check my progress* to verify the objective. Deploy to App Engine

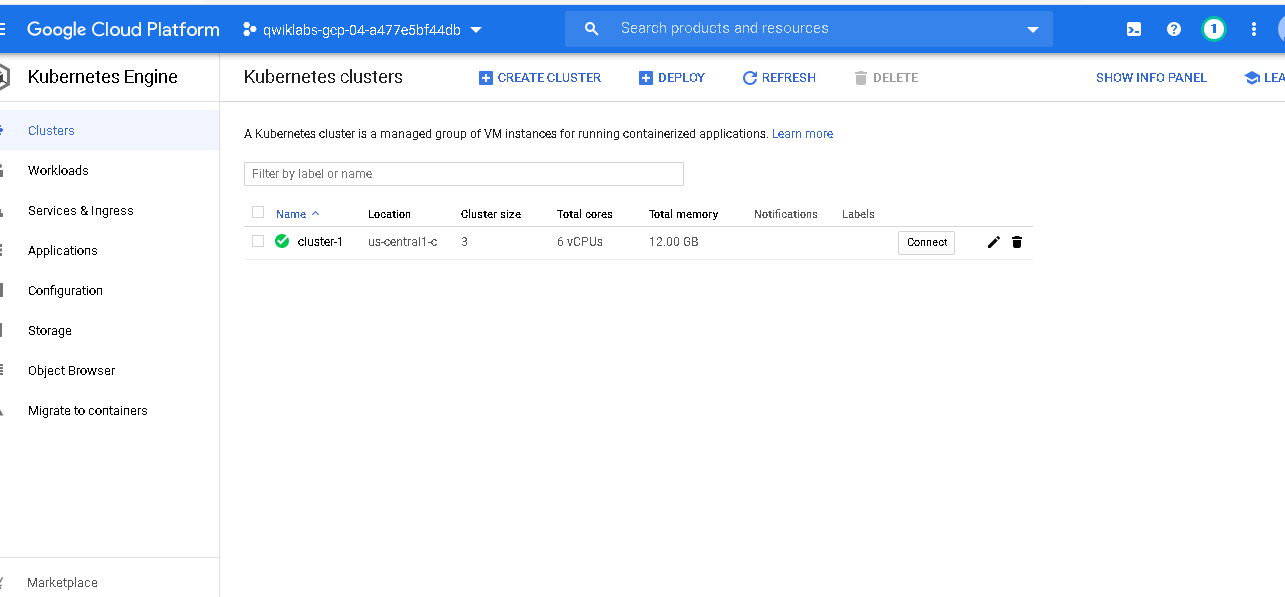
**Task 3: Deploy to Kubernetes Engine**

****

Kubernetes Engine allows you to create a cluster of machines and deploy any number of applications to it. Kubernetes abstracts the details of managing machines and allows you to automate the deployment of your applications with simple CLI commands.

To deploy an application to Kubernetes, you first need to create the cluster. Then you need to add a configuration file for each application you will deploy to the cluster.

1. On the **Navigation menu** (Navigation menu), click **Kubernetes Engine**. If a message appears saying the Kubernetes API is being initialized, wait for it to complete.
2. Click **Create cluster**.
3. Accept all the defaults, and click **Create**. It will take a couple of minutes for the Kubernetes Engine cluster to be created.

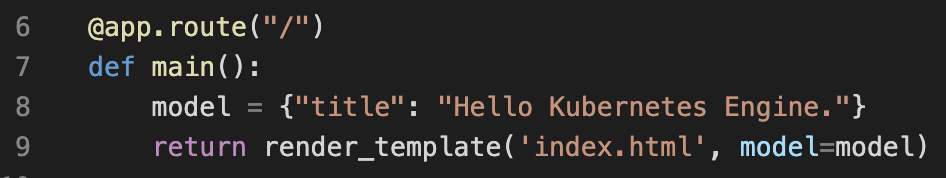


1. When the cluster is ready, a green check appears, click **Connect**.
2. In the **Connect to the cluster** screen, click **Run in Cloud Shell**. This opens Cloud Shell with the connect command entered automatically.
3. Press **Enter** to connect to the cluster.
4. To test your connection, enter the following command:

kubectl get nodes

This command simply shows the machines in your cluster. If it works, you're connected.

1. In Cloud Shell, click **Open Editor** (Cloud Shell Editor).
2. Expand the **training-data-analyst/courses/design-process/deploying-apps-to-gcp** folder in the navigation pane on the left. Then, click **main.py** to open it.
3. In the **main()** function, change the title to Hello Kubernetes Engine as shown below:



1. Save your change.
2. Add a file named kubernetes-config.yaml to the **training-data-analyst/courses/design-process/deploying-apps-to-gcp** folder.
3. Paste the following code in that file to configure the application:

---

apiVersion: apps/v1beta1

kind: Deployment

metadata:

name: devops-deployment

labels:

app: devops

tier: frontend

spec:

replicas: 3

selector:

matchLabels:

app: devops

tier: frontend

template:

metadata:

labels:

app: devops

tier: frontend

spec:

containers:

- name: devops-demo

image: <YOUR IMAGE PATH HERE>

ports:

- containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

name: devops-deployment-lb

labels:

app: devops

tier: frontend-lb

spec:

type: LoadBalancer

ports:

- port: 80

targetPort: 80

selector:

app: devops

tier: frontend

**Note:**

In the first section of the YAML file above, you are configuring a deployment. In this case, you are deploying 3 instances of your Python web app. Notice the image attribute. You will update this value with your image in a minute after you build it.In the second section, you are configuring a service of the type "load balancer." The load balancer will have a public IP address. Users will access your application through the load balancer.

For more information on Kubernetes deployments and services, see the links below:

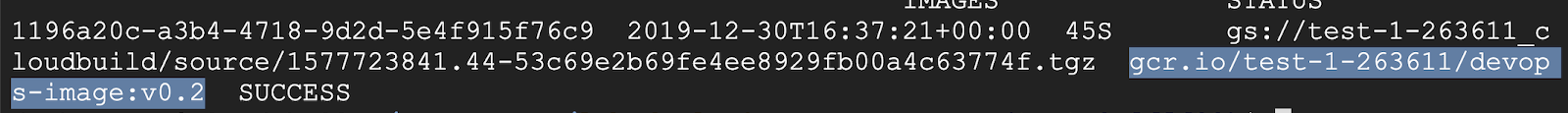
* <https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>
* <https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/>

1. To use Kubernetes Engine, you need to build a Docker image. Enter the following commands to use Cloud Build to create the image and store it in Container Registry:

cd ~/gcp-course/training-data-analyst/courses/design-process/deploying-apps-to-gcp

gcloud builds submit --tag gcr.io/$DEVSHELL\_PROJECT\_ID/devops-image:v0.2 .

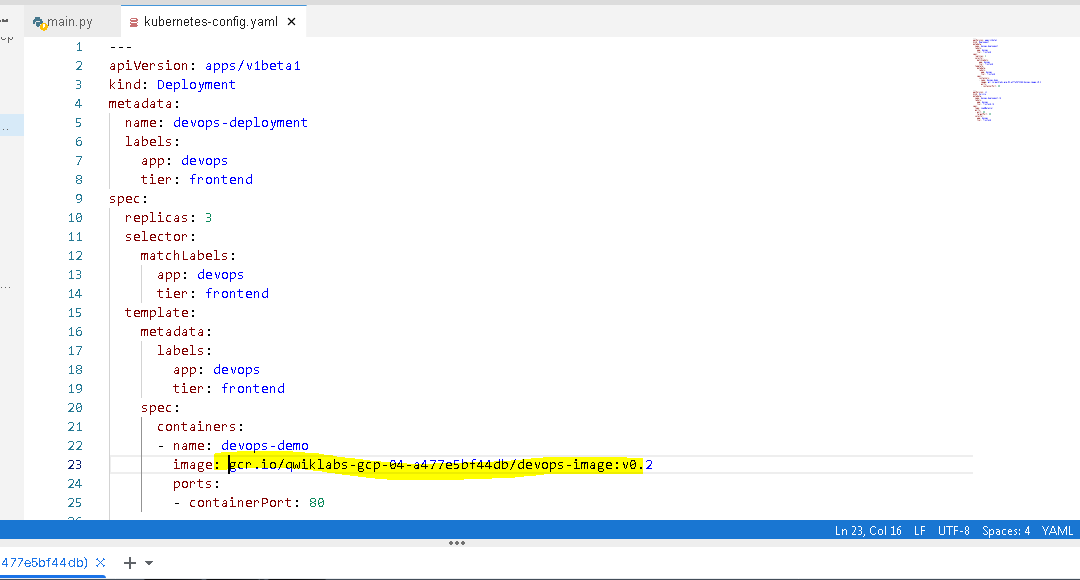
1. When the previous command completes, the image named will be listed in the output. The image name is in the form gcr.io/project-id/devops-image:v0.2. It should appear as shown below:



1. Highlight your image name and copy it to the clipboard. Paste that value in the kubernetes-config.yaml file, overwriting the string <YOUR IMAGE PATH HERE>.

You should see something similar to below:





1. Enter the following Kubernetes command to deploy your application:

kubectl apply -f kubernetes-config.yaml



1. In the configuration file, three replicas of the application were specified. Type the following command to see whether three instances have been created:

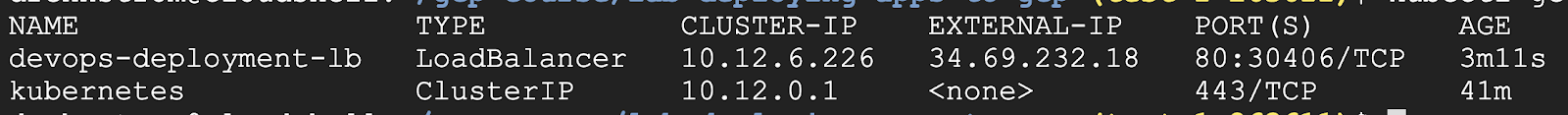
kubectl get pods

Make sure all the pods are ready. If they aren't, wait a few seconds and try again.

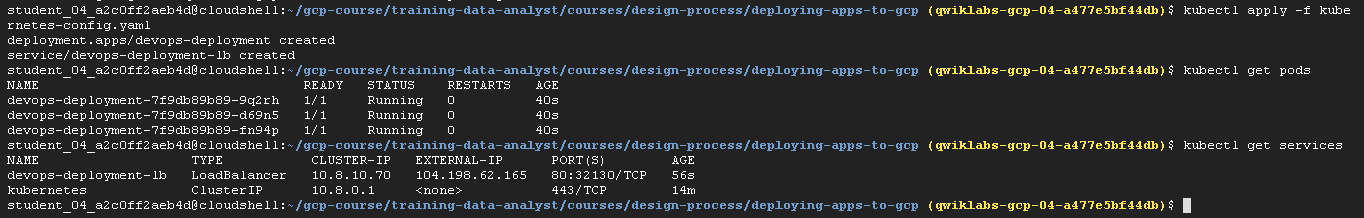
1. A load balancer was also added in the configuration file. Type the following command to see whether it was created:

kubectl get services

You should see something similar to below:

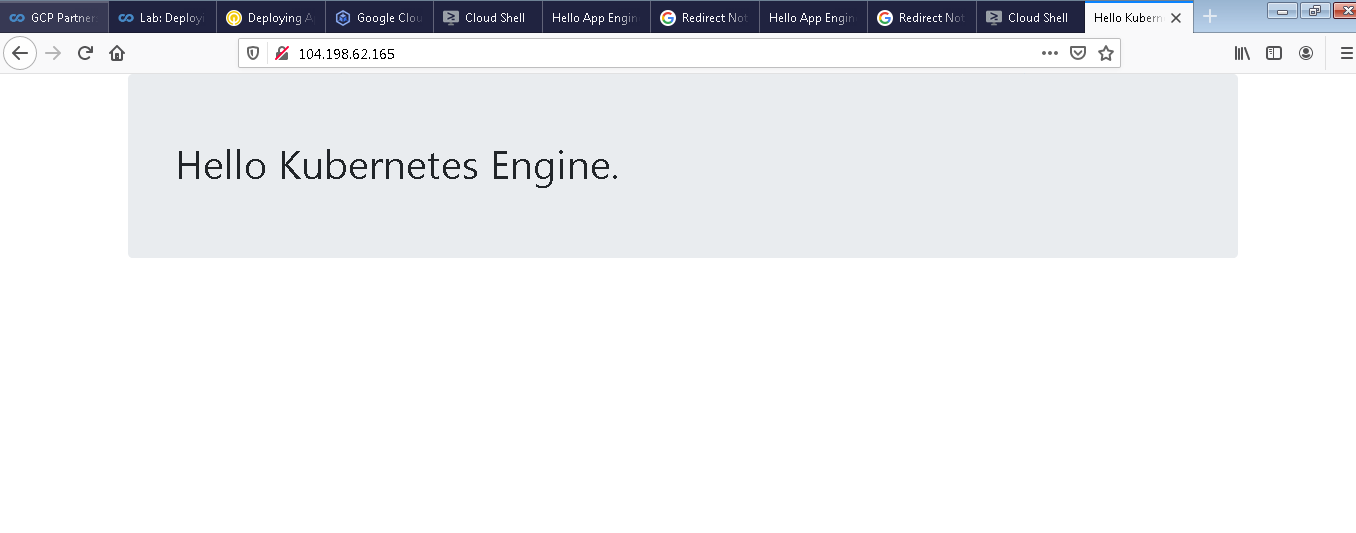


If the load balancer's external IP address says "pending," wait a few seconds and try again.



1. When you have an external IP, open a browser tab and make a request to it. It should return Hello Kubernetes Engine. It might take a few seconds to be ready.

Click *Check my progress* to verify the objective. Deploy to Kubernetes Engine



**Task 4: Deploy to Cloud Run**

Cloud Run simplifies and automates deployments to Kubernetes. When you use Cloud Run, you don't need a configuration file. You simply choose a cluster for your application. With Cloud Run, you can use a cluster managed by Google, or you can use your own Kubernetes cluster.

To use Cloud Run, your application needs to be deployed using a Docker image and it must be stateless.

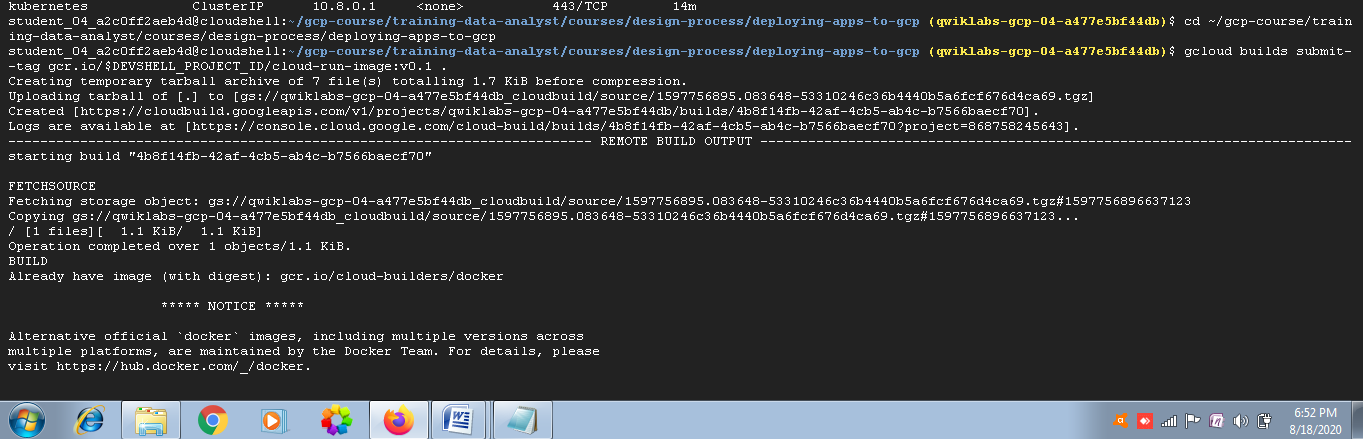
1. Open the Cloud Shell code editor and expand the **training-data-analyst/courses/design-process/deploying-apps-to-gcp** folder in the navigation pane on the left. Then, click **main.py** to open it.
2. In the **main()** function, change the title to Hello Cloud Run as shown below:



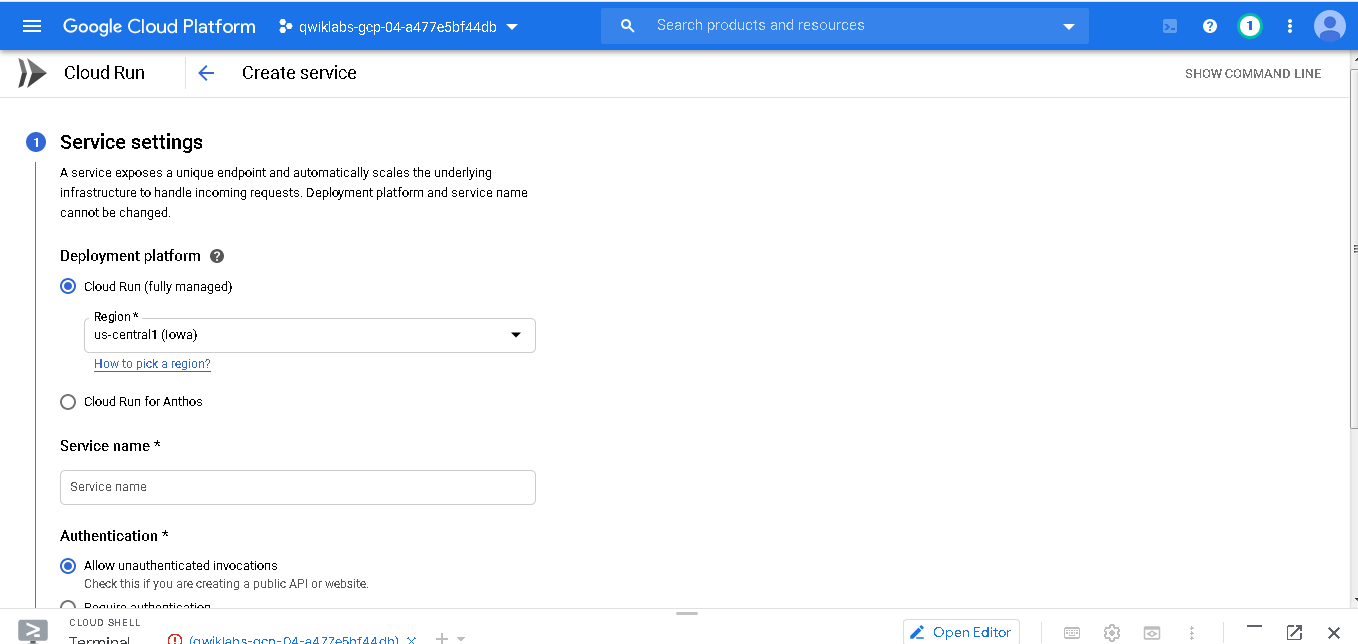
1. Save your change.
2. To use Cloud Run, you need to build a Docker image. In Cloud Shell, enter the following commands to use Cloud Build to create the image and store it in Container Registry:

cd ~/gcp-course/training-data-analyst/courses/design-process/deploying-apps-to-gcp

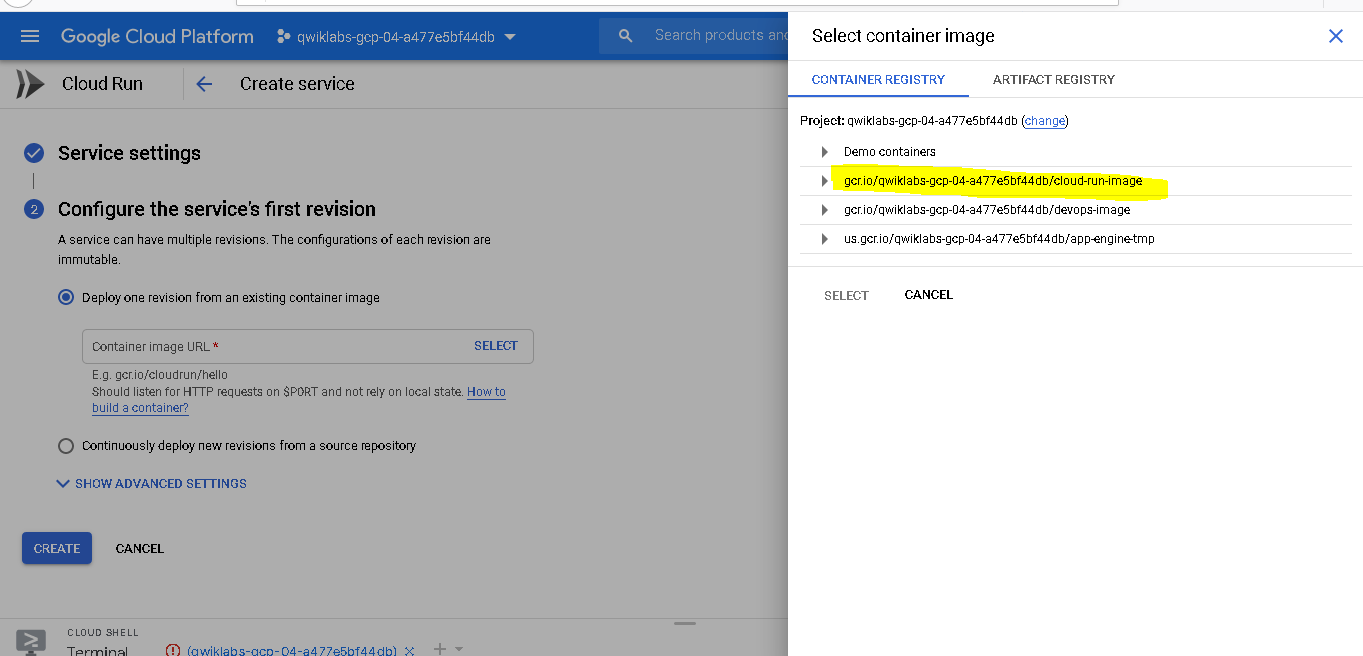
gcloud builds submit --tag gcr.io/$DEVSHELL\_PROJECT\_ID/cloud-run-image:v0.1 .

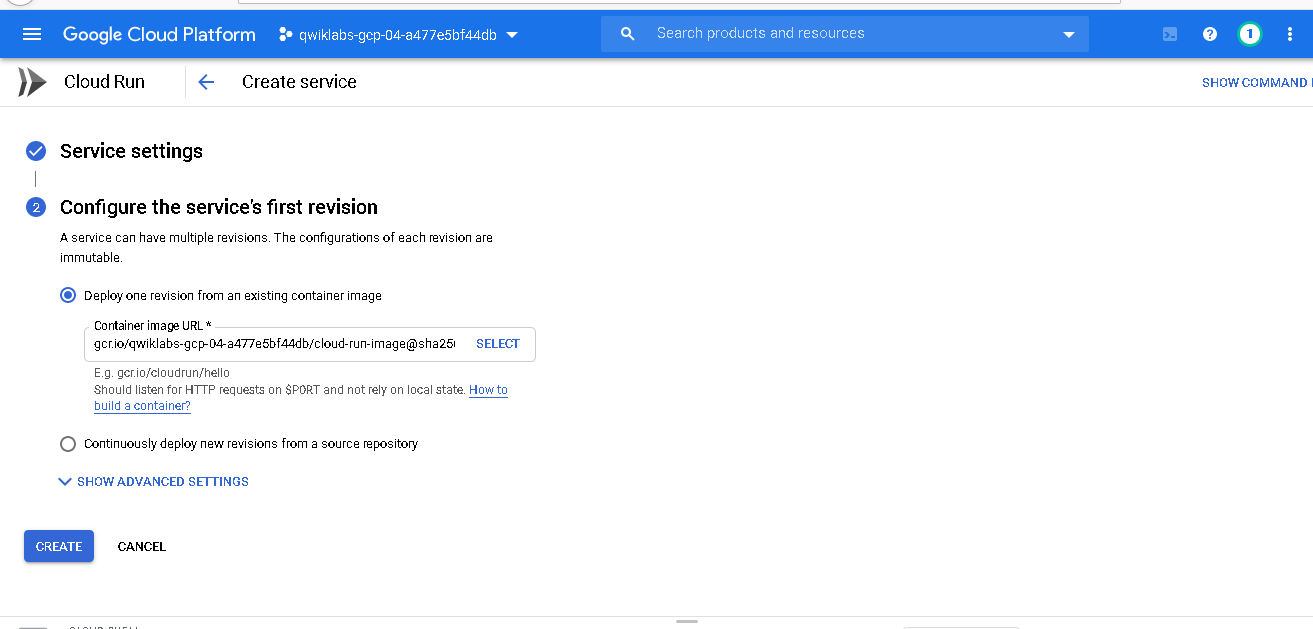


1. When the build completes, in the **Navigation menu** (Navigation menu), click **Cloud Run**.
2. Cloud Run is not enabled by default. Click **Start using Cloud Run** to enable the API.
3. Click **Create service**.
4. Accept the defaults in the **Deployment platform** section.
5. Give any name to the service and select the **Allow unauthenticated invocations** option.

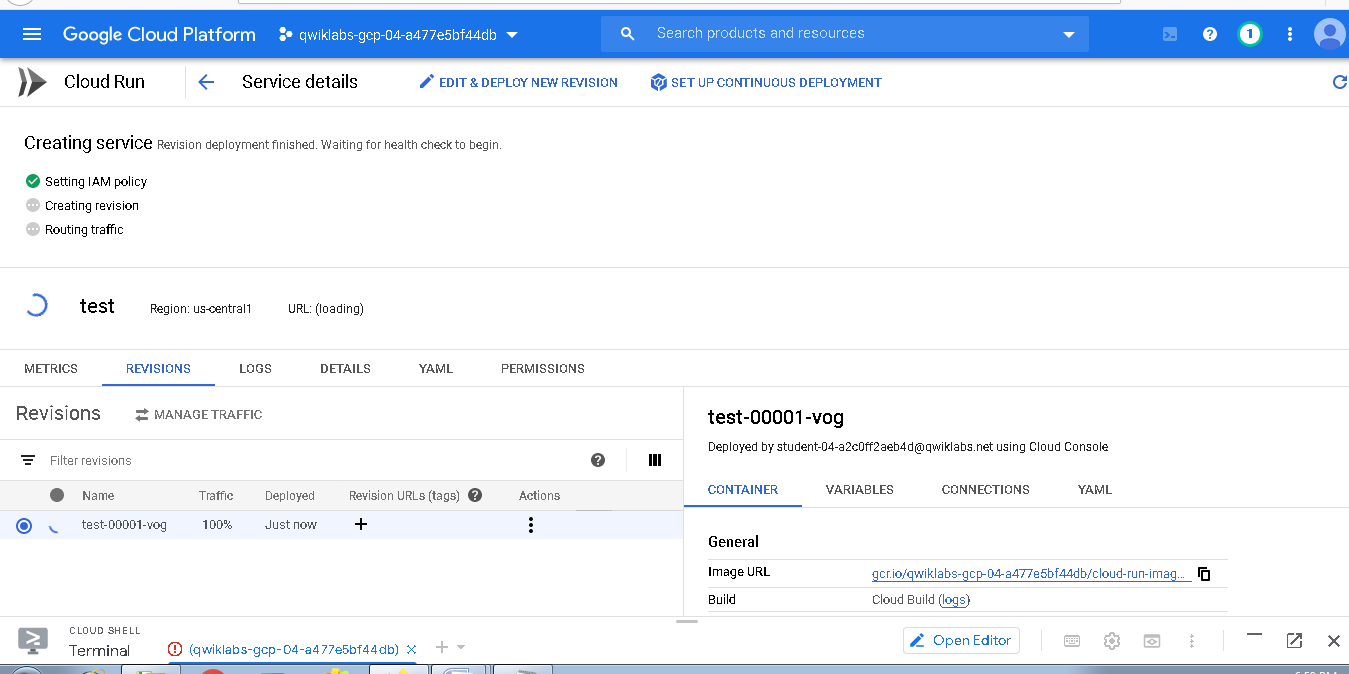


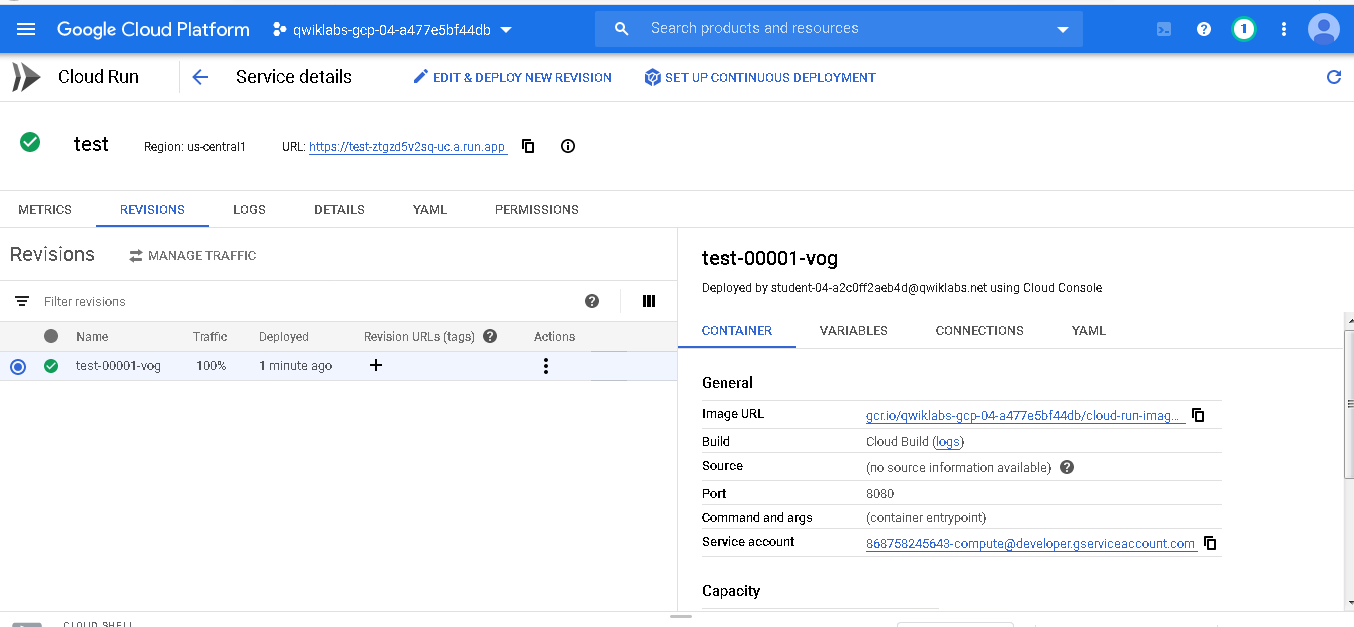
1. Click **Next**.
2. Click the **Select** link in the Container image URL text box. In the resulting dialog, expand cloud-run-image and select the image listed. Then click **Continue**.

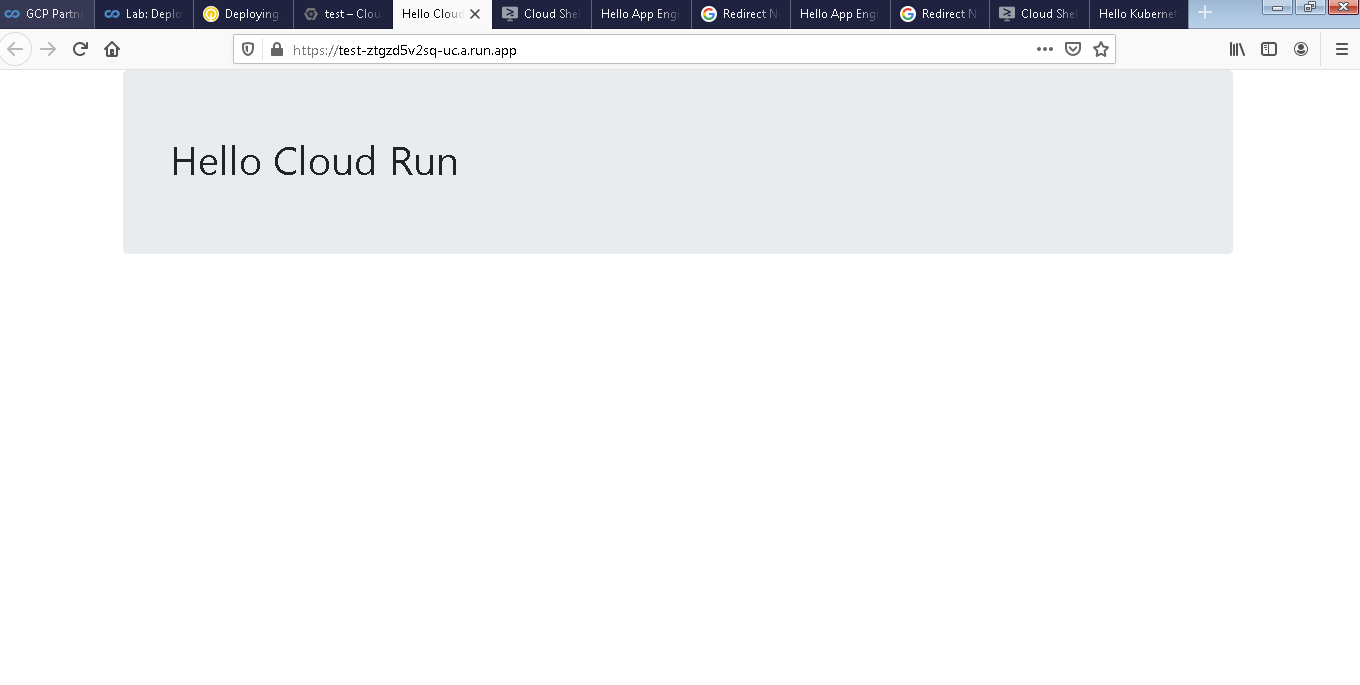




1. Finally, click **Create**.
2. It shouldn't take long for the service to deploy. When a green check appears, click on the URL that is automatically generated for the application. It should return Hello Cloud Run.







Click *Check my progress* to verify the objective. Deploy to Cloud Run

**Congratulations!**

In this lab, you deployed applications to the Google Cloud services App Engine, Kubernetes Engine, and Cloud Run.

**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.