

App Dev - Deploying the Application into Kubernetes Engine - Java

GSP171



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 that 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:

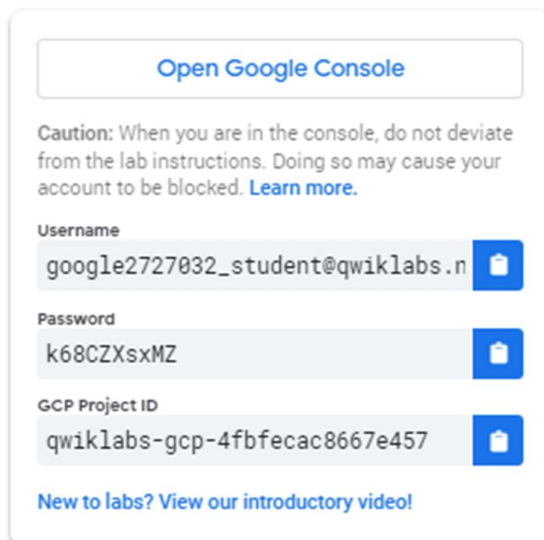
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

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.

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

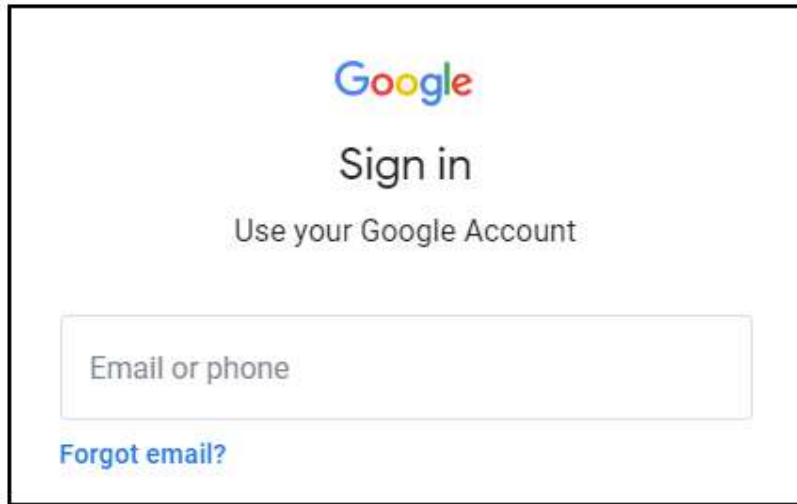
1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



The screenshot shows a panel with the following content:

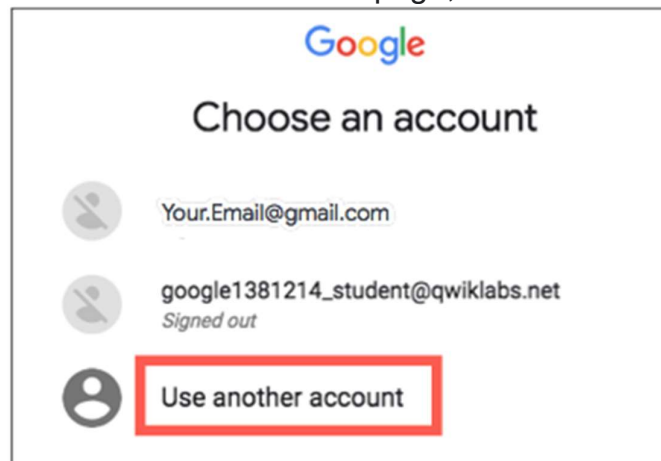
- A button at the top labeled "Open Google Console".
- A caution message: "Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)"
- Three credential fields, each with a copy icon to its right:
 - Username: google2727032_student@qwiklabs.n
 - Password: k68CZXsxMZ
 - GCP Project ID: qwiklabs-gcp-4fbfecac8667e457
- A link at the bottom: "New to labs? View our introductory video!"

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.

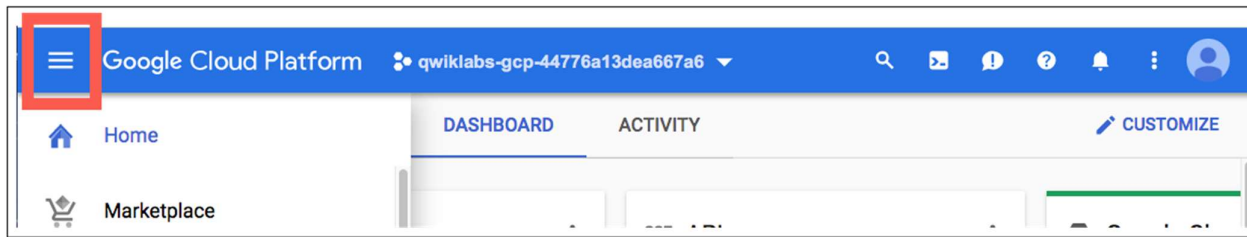
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

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.

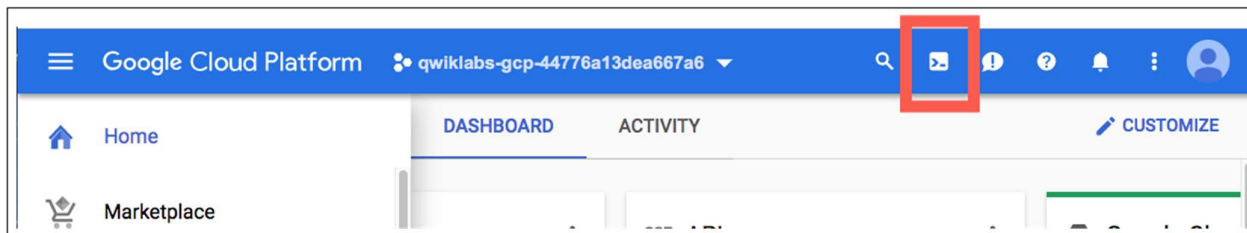
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



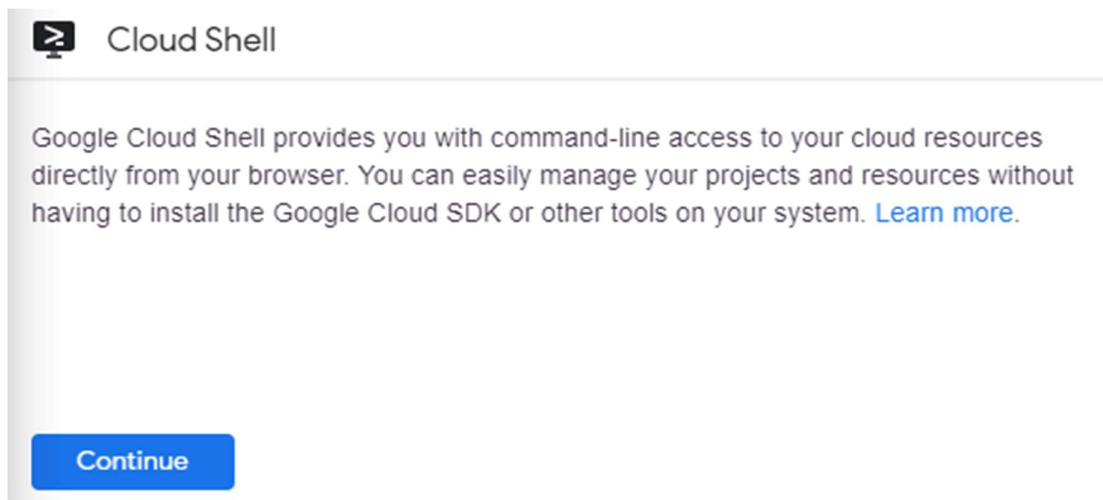
Activate Cloud Shell

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.

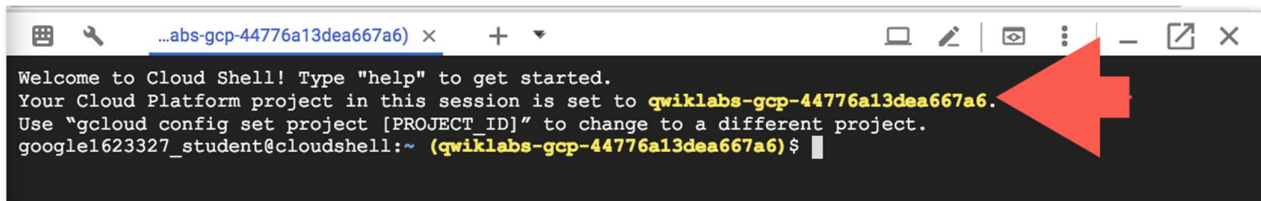
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



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:



```
...abs-gcp-44776a13dea667a6) x + ▾
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](#).

Prepare the Quiz Application

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

Clone source code in Cloud Shell

Enter the following command in Cloud Shell to clone the repository for this lab.

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

Configure the Quiz application

Change the working directory:

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

Configure the Quiz frontend application:

```
. prepare_environment.sh
```

This script file:

- Creates a Google App Engine application.
- Exports environment variables `GCP_PROJECT` and `GCP_BUCKET`.
- Runs `mvn clean install`.
- 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.

Click *Check my progress* to verify the objective.

Review the code

Next review and update the Quiz application code in a code editor. You can use the Cloud Shell code editor or the shell editors that are installed on Cloud Shell, such as `nano` or `vim`. This lab uses the Cloud Shell code editor.

Launch the Cloud Shell text editor

Use the Google Cloud Shell Code Editor to easily create and edit directories and files in the Cloud Shell instance.

Once you activate the Google Cloud Shell, click the **Open editor > Open In New Window** to open the Cloud Shell Code Editor.



The code editor launches in a separate tab of your browser.

Examine the code and folder structure

In Cloud Shell code editor, navigate to `training-data-analyst/courses/developingapps/java/kubernetesengine/start`.

In the `kubernetesengine` folder, notice the `end` folder. The `end` folder contains the same files as the `start` folder, but each file in the `end` folder contains the complete code required to perform this lab. The folder structure for the Quiz application now reflects how it's deployed in Kubernetes Engine.

- There is a new folder called `frontend`. This contains the packaged output for the web application.
 - There is a folder called `backend`. This contains the packaged output for the console application.
 - There are configuration files for Docker (a `Dockerfile` in the `frontend` and `backend` folder) and Kubernetes Engine (`*.yaml`).
1. Go back to cloud shell UI and click on **Open Terminal**.
 2. In Cloud Shell, use the following command to copy the output `jar` for the frontend application to the `frontend` folder:

```
3. cp ./target/quiz-frontend-0.0.1.jar ./frontend/
```
 4. Configure the Quiz backend application:

```
5. mvn package -f pom-backend.xml
```
 6. Copy the output `jar` for the backend application to the `backend` folder:

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


Creating a Kubernetes Engine Cluster

In this section you create a Google Kubernetes Engine cluster to host the Quiz application.

1. In the Console, click **Navigation menu > Kubernetes Engine > Clusters**, and then click **CREATE**. Then select **CONFIGURE** for the Standard cluster mode.
2. To configure the cluster, use the specified values for the properties listed in the following table; leave the properties not listed at the default values:

Property	Value
Name	quiz-cluster
Zone	us-central1-b
In the left hand side expand default-pool option under NODE POOLS and select Security	For Access scopes , Select Allow full access to all Cloud APIs

3. Click **CREATE**. The cluster will take a couple of minutes to provision.

Click *Check my progress* to verify the objective.

Connect to the cluster

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

Filter

Enter property name or value

<input type="checkbox"/>	<input checked="" type="radio"/>	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input type="checkbox"/>	<input checked="" type="radio"/>	quiz-cluster	us-central1-b	3	6	12 GB		—

Edit

Connect

Delete

2. In **Connect to the cluster**, copy the first command to the clipboard, and then click **OK** to close the window:

Connect to the cluster

You can connect to your cluster via command-line or using a dashboard.

Command-line access

Configure [kubectl](#) command line access by running the following command:

```
$ gcloud container clusters get-credentials quiz-cluster --zone us-central1-b --project qwiklabs-gcp-03-11b20a4049f0
```

[RUN IN CLOUD SHELL](#)

Cloud Console dashboard

You can view the workloads running in your cluster in the Cloud Console [Workloads dashboard](#).

[OPEN WORKLOADS DASHBOARD](#)

OK

The command is in the form: `gcloud container clusters get-credentials quiz-cluster --zone us-central1-b --project <Project-ID>`

3. Paste the command into Cloud Shell and press **Enter**.
4. List the pods in the cluster.

```
5. kubectl get pods
```

The response should indicate that there are no pods in the cluster.

This 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 employ Container Builder to build images and store them in the Container Registry.

Create the Dockerfile for the frontend

In Cloud Shell code editor, navigate to `training-data-analyst/courses/developingapps/java/kubernetesengine/start` and open `frontend/Dockerfile`.

Copy and paste the following content into `frontend/Dockerfile` then save the file.

```
FROM gcr.io/google_appengine/jetty9
VOLUME /tmp
ADD ./quiz-frontend-0.0.1.jar /app.jar
CMD java -jar /app.jar
```

What this script does:

This script is a series of Dockerfile commands.

The first command, `FROM gcr.io/google_appengine/jetty9`, initializes the creation of a custom Docker image using the Google App Engine Jetty 9 image, `gcr.io/google_appengine/jetty9` as the starting point.

This second command, `VOLUME /tmp`, creates a volume in the container's file system with the path of `/tmp`.

The third command, `ADD ./quiz-frontend-0.0.1.jar /app.jar`, adds the Jar file, `quiz-frontend-0.0.1.jar` for the frontend generated by the Maven packaging process as part of the build process.

This fourth and last command, `CMD java -jar /app.jar`, executes when the container runs.

Create the Dockerfile for the backend

You create the dockerfile for the backend the same way you created the frontend, except the `jar` file is added to the backend.

In Cloud Shell code editor, navigate to `training-data-analyst/courses/developingapps/java/kubernetesengine/start` and open `backend/Dockerfile`.

Copy and paste the following content into `backend/Dockerfile` then save the file.

```
FROM gcr.io/google_appengine/jetty9
VOLUME /tmp
ADD ./quiz-backend-0.0.1.jar /app.jar
CMD java -jar /app.jar
```

Build Docker images with Container Builder

1. In Cloud Shell, enter the following command to build the frontend Docker image:











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

The files are staged into Cloud Storage, and a Docker image will be built and stored in the Container Registry. It will take a few seconds.




3. Build the backend Docker image:

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

4. In the console, Click **Navigation menu** > **Container Registry**. You should see two folders: **quiz-frontend** and **quiz-backend**.

 Container Registry	Repositories REFRESH						
 Images	qwiklabs-gcp-03-11b20a4049f0						
 Settings							
	<input type="text" value="Filter"/> All hostnames 						
	<table><thead><tr><th>Name ^</th><th>Hostname</th></tr></thead><tbody><tr><td> quiz-backend</td><td>gcr.io</td></tr><tr><td> quiz-frontend</td><td>gcr.io</td></tr></tbody></table>	Name ^	Hostname	 quiz-backend	gcr.io	 quiz-frontend	gcr.io
Name ^	Hostname						
 quiz-backend	gcr.io						
 quiz-frontend	gcr.io						

5. Click **quiz-frontend**. You should see the image name (a hash), tags (latest), size (about 200 MB), and other details.

<input type="checkbox"/> Name	Tags	Virtual size	Uploaded v	Build	Vulnerabilities	Type
<input type="checkbox"/>  710a4d53ec07	latest 	215.9 MB	6 minutes ago	888f54da-6290...	 Not scanned	Docker Manifest, Schema 2

Click *Check my progress* to verify the objective.

Create a Kubernetes deployment and service resources

In this section you will modify 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

In Cloud Shell code editor, navigate to `training-data-analyst/courses/developingapps/java/kubernetesengine/start`.

1. In the 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
[GCPLOUD_PROJECT]	Project ID (Display the Project ID by entering <code>echo \$GCPLOUD_PROJECT</code> in Cloud Shell)
[GCPLOUD_BUCKET]	Cloud Storage bucket ID for the media bucket in your project. (Display the bucket name by entering <code>echo \$GCPLOUD_BUCKET</code> in Cloud Shell)
[FRONTEND_IMAGE_IDENTIFIER]	The frontend image identified in the form <code>gcr.io/[Google Cloud_Project_ID]/quiz-frontend</code>

3. 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.
4. **Save** the file.
5. Open the `backend-deployment.yaml` file and replace the placeholders in the `backend-deployment.yaml` file using the following values:

Placeholder Name	Value
[GCPLOUD_PROJECT]	Project ID

[GCPLOUD_BUCKET]	Cloud Storage bucket ID for the media bucket in your project. This will be the same bucket used in <code>frontend-deployment.yaml</code> . (Display the bucket name by entering <code>echo \$GCPLOUD_BUCKET</code> in Cloud Shell)
[BACKEND_IMAGE_IDENTIFIER]	The backend image identified in the form <code>gcr.io/[Google Cloud_Project_ID]/quiz-backend</code>

- The quiz-backend deployment provisions one replica of the backend Docker image in Kubernetes pods, placed on one of the three nodes of the Kubernetes Engine cluster.

- Save** the file.

- Review the contents of the `frontend-service.yaml` file.

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

Execute the Deployment and Service Files

- In Cloud Shell, provision the quiz frontend deployment.

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

- Provision the quiz backend deployment.

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

- Provision the quiz frontend Service.

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

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

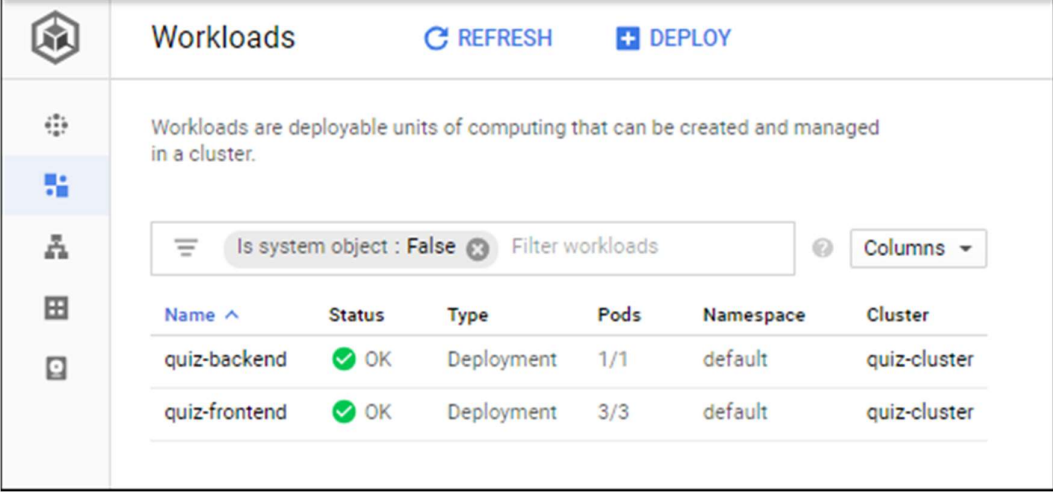
Click *Check my progress* to verify the objective.

Test the Quiz Application

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

Review the deployed resources

1. In the console, click **Navigation menu > Kubernetes Engine > Workloads**. You should see two items: `quiz-frontend` and `quiz-backend`.



Name ^	Status	Type	Pods	Namespace	Cluster
quiz-backend	✓ OK	Deployment	1/1	default	quiz-cluster
quiz-frontend	✓ OK	Deployment	3/3	default	quiz-cluster

You may see that the pod status is OK or in the process of being created.

2. Click **quiz-frontend**.
3. Scroll down to **Managed pods** to see an overview of `quiz-frontend`, including the three managed pods and services.

Managed pods

Revision	Name	Status	Restarts	Created on ^
1	quiz-frontend-6f5997697b-flkt5	✓ Running	0	Mar 22, 2019, 2:58:26 PM
1	quiz-frontend-6f5997697b-kjdx	✓ Running	0	Mar 22, 2019, 2:58:27 PM
1	quiz-frontend-6f5997697b-4c9qh	✓ Running	0	Mar 22, 2019, 2:58:27 PM

Exposing services ?

Name ^	Type	Endpoints
quiz-frontend	LoadBalancer	35.188.35.201:80

You may see that the `quiz-frontend` load balancer is being created or is OK. Wait until the Service is OK before continuing. You should see an IP address endpoint when the service is ready.

4. In **Exposing services**, under **Endpoints**, select the IP address and open in the new browser tab.
5. Take a test to verify the application works as expected.

Congratulations!

You learned how to package frontend and backend application code for deployment, use Container Builder to produce Docker images, and provision a Kubernetes Engine cluster to host the application.



Finish your Quest

This lab is part of the [Application Development - Java](#) 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 this Quest and get immediate completion credit if you've taken this lab. See other available [Qwiklabs Quests](#).

Next steps / learn more

- Learn more about [Java on the Google Cloud](#).
- Get up to speed with Firebase, take [Firebase SDK for Cloud Functions](#), or check out [Google Cloud Datastore Documentation](#).

Manual last updated: April 12, 2021

Lab last tested: April 12, 2021

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.