

# Deploy to Kubernetes in Google Cloud: Challenge Lab

**GSP318**



# Overview

In a challenge lab you're given a scenario and a set of tasks. Instead of following step-by-step instructions, you will use the skills learned from the labs in the quest to figure out how to complete the tasks on your own! An automated scoring system (shown on this page) will provide feedback on whether you have completed your tasks correctly.

When you take a challenge lab, you will not be taught new Google Cloud concepts. You are expected to extend your learned skills, like changing default values and reading and researching error messages to fix your own mistakes.

To score 100% you must successfully complete all tasks within the time period!

This lab is recommended for students who have enrolled in the [Deploy to Kubernetes in Google Cloud](#) quest. Are you ready for the challenge?

Topics tested:

- Creating Docker images on a host.
- Running Docker containers on a host.
- Storing Docker images in the Google Container Repository (GCR).
- Deploying GCR images on Kubernetes.
- Pushing updates onto Kubernetes.
- Automating deployments to Kubernetes using Jenkins.

## 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:

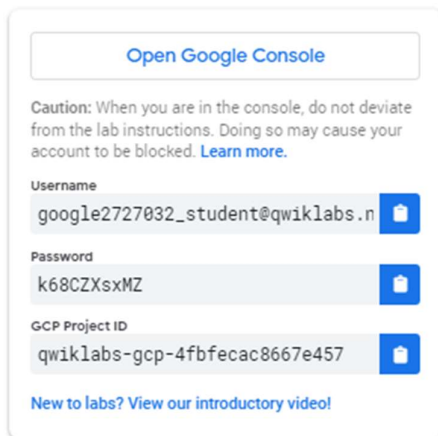
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

**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.



A screenshot of a web panel titled "Open Google Console". It contains a caution message, a username field with "google2727032\_student@qwiklabs.n", a password field with "k68CZXsxMZ", and a GCP Project ID field with "qwiklabs-gcp-4fbfecac8667e457". Each field has a copy icon to its right. At the bottom, there is a link "New to labs? View our introductory video!"

Open Google Console

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.](#)

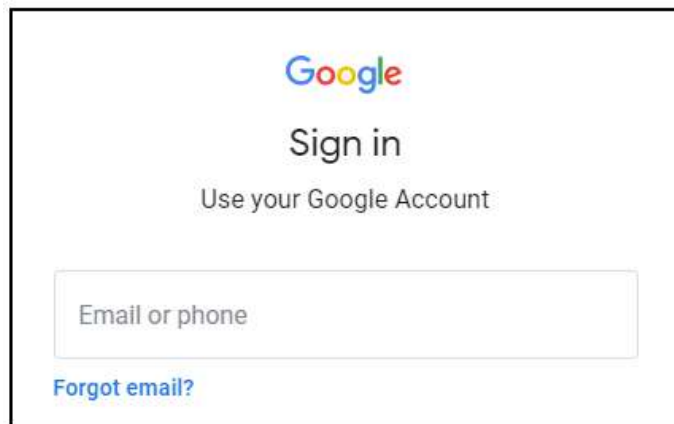
Username  
google2727032\_student@qwiklabs.n

Password  
k68CZXsxMZ

GCP Project ID  
qwiklabs-gcp-4fbfecac8667e457

[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.



A screenshot of the Google Sign in page. It features the Google logo at the top, followed by "Sign in" and "Use your Google Account". Below this is a text input field labeled "Email or phone" and a link "Forgot email?" at the bottom left.

Google

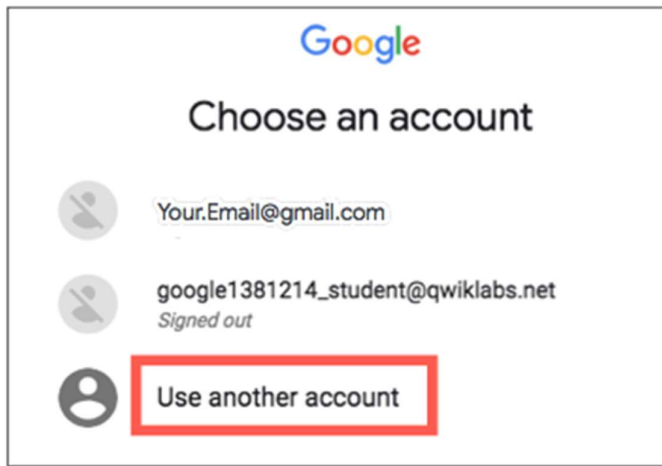
Sign in

Use your Google Account

Email or phone

[Forgot email?](#)

**Tip:** Open the tabs in separate windows, side-by-side.  
If you see the **Choose an account** page, click **Use Another Account**.



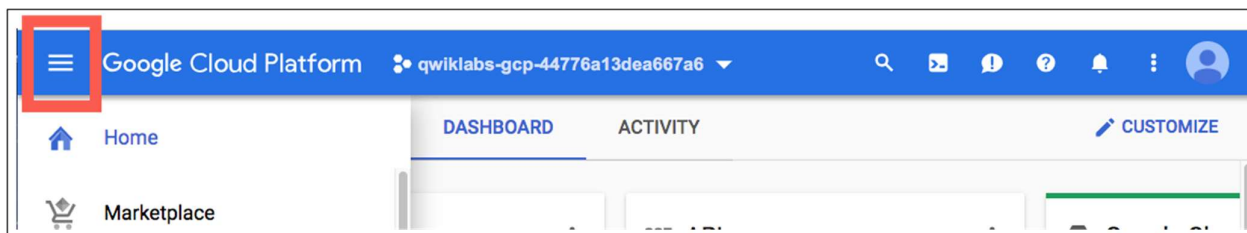
3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

**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.



# Challenge scenario

You have just completed training on containers and their creation and management and now you need to demonstrate to the Jooli Inc. development team your new skills. You have to help with some of their initial work on a new project around an application environment utilizing Kubernetes. Some of the work was already done for you, but other parts require your expert skills.

You are expected to create container images, store the images in a repository, and configure a Jenkins CI/CD pipeline to automate the build for the product. Your know that Kurt, your supervisor, will ask you to complete these tasks:

- Create a Docker image and store the Dockerfile.
- Test the created Docker image.
- Push the Docker image into the Container Repository.
- Use the image to create and expose a deployment in Kubernetes
- Update the image and push a change to the deployment.
- Create a pipeline in Jenkins to deploy a new version of your image when the source code changes.

Some Jooli Inc. standards you should follow:

- Create all resources in the `us-east1` region and `us-east1-b` zone, unless otherwise directed.
- Use the project VPCs.
- Naming is normally *team-resource*, e.g. an instance could be named **kraken-webserver1**.
- Allocate cost effective resource sizes. Projects are monitored and excessive resource use will result in the containing project's termination (and possibly yours), so beware. This is the guidance the monitoring team is willing to share: unless directed, use `n1-standard-1`.

## Your challenge

As soon as you sit down at your desk and open your new laptop you receive the following request to complete these tasks. Good luck!

Do not wait for the lab to provision! You can work through tasks 1-3 before you need the provisioning to be finished. Just ensure the workstation exists before starting Task 1.

# Task 1: Create a Docker image and store the Dockerfile

Open Cloud Shell and run `source <(gsutil cat gs://cloud-training/gsp318/marketing/setup_marking.sh)`. This command will install marking scripts you can use to help check your progress.

Use Cloud Shell to clone the `valkyrie-app` source code repository (it is in your project).

The app source code is in `valkyrie-app/source`. Create `valkyrie-app/Dockerfile` and add the configuration below.

```
FROM golang:1.10
WORKDIR /go/src/app
COPY source .
RUN go install -v
ENTRYPOINT ["app", "-single=true", "-port=8080"]
content copy
```

Use `valkyrie-app/Dockerfile` to create a Docker image called **valkyrie-app** with the tag **v0.0.1**

Once you have created the Docker image, and before clicking **Check my progress**, run `step1.sh` to perform the local check of your work. After you get a successful response from the local marking you can check your progress.

Click *Check my progress* to verify the objective.

If you don't get a green check mark, click on the **Score** fly-out on the top right and click **Run Step** on the relevant step. A hint pop up opens to give you advice.

## Task 2: Test the created Docker image

Launch a container using the image **valkyrie-app:v0.0.1**. You need to map the host's port 8080 to port 8080 on the container. Add `&` to the end of the command to cause the container to run in the background.

When your container is running you will see the page by **Web Preview**.

Once you have your container running, and before clicking **Check my progress**, run `step2.sh` to perform the local check of your work. After you get a successful response from the local marking you can check your progress.

Click *Check my progress* to verify the objective.

If you don't get a green check mark, click on the **Score** fly-out on the top right and click **Run Step** on the relevant step. A hint pop up opens to give you advice.

## Task 3: Push the Docker image in the Container Repository

Push the Docker image **valkyrie-app:v0.0.1** into the Container Registry.

Make sure you re-tag the container to **gcr.io/YOUR\_PROJECT/valkyrie-app:v0.0.1**.

Click *Check my progress* to verify the objective.

If you don't get a green check mark, click on the **Score** fly-out on the top right and click **Run Step** on the relevant step. A hint pop up opens to give you advice.

## Task 4: Create and expose a deployment in Kubernetes

Kurt created the `deployment.yaml` and `service.yaml` to deploy your new container image to a Kubernetes cluster (called `valkyrie-dev`). The two files are in `valkyrie-app/k8s`.

Remember you need to get the Kubernetes credentials before you deploy the image onto the Kubernetes cluster.

Before you create the deployments make sure you check the `deployment.yaml` and `service.yaml` files. Kurt thinks they need some values set (he thinks he left some placeholder values).

You can check the load balancer once it's available.

Click *Check my progress* to verify the objective.

If you don't get a green check mark, click on the **Score** fly-out on the top right and click **Run Step** on the relevant step. A hint pop up opens to give you advice.



## Task 5: Update the deployment with a new version of valkyrie-app

Before deploying the new code, increase the replicas from 1 to 3 to ensure you don't cause an outage.

Click *Check my progress* to verify the objective.

If you don't get a green check mark, click on the **Score** fly-out on the top right and click **Run Step** on the relevant step. A hint pop up opens to give you advice.

Kurt made changes to the source code (he put the changes in a branch called **kurt-dev**). You need to merge **kurt-dev** into **master** (you should use `git merge origin/kurt-dev`).

Build the new code as version v0.0.2 of valkyrie-app, push the updated image to the Container Repository, and then redeploy to the valkyrie-dev cluster. You will know you have the new v0.0.2 version because the titles for the cards will be green.

Click *Check my progress* to verify the objective.

If you don't get a green check mark, click on the **Score** fly-out on the top right and click **Run Step** on the relevant step. A hint pop up opens to give you advice.

# Task 6: Create a pipeline in Jenkins to deploy your app

This process of building the container and pushing to the container repository can be automated using Jenkins. There is a Jenkins deployment in your `valkyrie-dev` cluster - connect to Jenkins and configure a job to build when you push a change to the source code.

Remember with Jenkins:

- Get the password with `printf $(kubectl get secret cd-jenkins -o jsonpath="{.data.jenkins-admin-password}" | base64 --decode);echo.`
- Connect to the Jenkins console using the commands below (but make sure you don't have a running container `docker ps`; if you do, kill it):

```
export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/component=jenkins-master" -l "app.kubernetes.io/instance=cd" -o
jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:8080 >> /dev/null &
content copy
```

- Setup your credentials to use **Google Service Account from metadata**.
- Create a pipeline job that points to your `*/master` branch on your source code. Make two changes to your files before you commit and build:
- Edit `valkyrie-app/Jenkinsfile` and change `YOUR_PROJECT` to your actual project id.
- Edit `valkyrie-app/source/html.go` and change the two occurrences of green to orange. Use `git` to:
- Add all the changes then commit those changes to the master branch.
- Push the changes back to the repository.

When you are ready, manually trigger a build (the initial build will take some time, so just monitor the process). The build will replace the running containers with containers with different tags; you will see orange colored headings.

Click *Check my progress* to verify the objective.

If you don't get a green check mark, click on the **Score** fly-out on the top right and click **Run Step** on the relevant step. A hint pop up opens to give you advice.

# Congratulations!



## Earn Your Next Skill Badge

This self-paced lab is part of the [Deploy to Kubernetes in Google Cloud](#) Quest. Completing this skill badge quest earns you the badge above, to recognize your achievement. Share your badge on your resume and social platforms, and announce your accomplishment using #GoogleCloudBadge.

This skill badge quest is part of Google Cloud's [Hybrid and Multi-Cloud Cloud Architect](#) learning path. Continue your learning journey by enrolling in the [Secure Workloads in Google Kubernetes Engine](#) quest. [See other available Qwiklabs Quests](#) available in the catalog.

## Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated April 09, 2021

Lab Last Tested April 09, 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.

**SOLUTION:** [youtube link: <https://www.youtube.com/watch?v=AJna2uyI9hw>]

Task 1:

```
gsutil cat gs://cloud-training/gsp318/marketing/setup_marking.sh | bash
gcloud source repos clone valkyrie-app
cd valkyrie-app
cat > Dockerfile <<EOF
FROM golang:1.10
WORKDIR /go/src/app
COPY source .
RUN go install -v
ENTRYPOINT ["app","-single=true","-port=8080"]
EOF
docker build -t valkyrie-app:v0.0.1 .
cd ..
cd marketing
./step1.sh
```

Task 2:

```
cd ..
cd valkyrie-app
docker run -p 8080:8080 valkyrie-app:v0.0.1 &
cd ..
cd marketing
./step2.sh
```

Task 3:

```
cd ..
cd valkyrie-app
docker tag valkyrie-app:v0.0.1 gcr.io/$GOOGLE_CLOUD_PROJECT/valkyrie-app:v0.0.1
docker push gcr.io/$GOOGLE_CLOUD_PROJECT/valkyrie-app:v0.0.1

sed -i s#IMAGE_HERE#gcr.io/$GOOGLE_CLOUD_PROJECT/valkyrie-app:v0.0.1#g
k8s/deployment.yaml
```

Task 4:

```
sed -i s#IMAGE_HERE#gcr.io/$GOOGLE_CLOUD_PROJECT/valkyrie-
app:v0.0.1#g k8s/deployment.yaml

gcloud container clusters get-credentials valkyrie-dev --zone us-east1-d
kubectl create -f k8s/deployment.yaml
kubectl create -f k8s/service.yaml
```

#### Task 5:

```
git merge origin/kurt-dev
kubectl edit deployment valkyrie-dev

# replace replicas from 1 to 3 at 2 places

docker build -t gcr.io/$GOOGLE_CLOUD_PROJECT/valkyrie-app:v0.0.2 .
docker push gcr.io/$GOOGLE_CLOUD_PROJECT/valkyrie-app:v0.0.2
kubectl edit deployment valkyrie-dev

# replace 0.0.1 to 0.0.2 at spec > containers > image

docker ps
```

#### Task 6:

```
docker kill container_id

export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/component=jenkins-master" -l "app.kubernetes.io/instance=cd" -
o jsonpath="{.items[0].metadata.name}")

kubectl port-forward $POD_NAME 8080:8080 >> /dev/null &

printf $(kubectl get secret cd-jenkins -o jsonpath="{.data.jenkins-admin-password}" |
base64 --decode);echo

gcloud source repos list

sed -i "s/green/orange/g" source/html.go

sed -i "s/YOUR_PROJECT/$GOOGLE_CLOUD_PROJECT/g" Jenkinsfile
git config --global user.email "you@example.com" <----- put from first consol
git config --global user.name "student" <----- from login status
git add .
git commit -m "built pipeline init"
git push
```