Setting up Jenkins on Kubernetes Engine

GSP117

Google Cloud Self-Paced Labs

In this hands-on lab, you'll learn how to set up Jenkins on Google Kubernetes Engine to help orchestrate your software delivery pipeline.

Objectives

- Creating a Kubernetes cluster with Kubernetes Engine.
- Creating a Jenkins deployment and services.
- Connecting to Jenkins.

Setup and Requirements

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

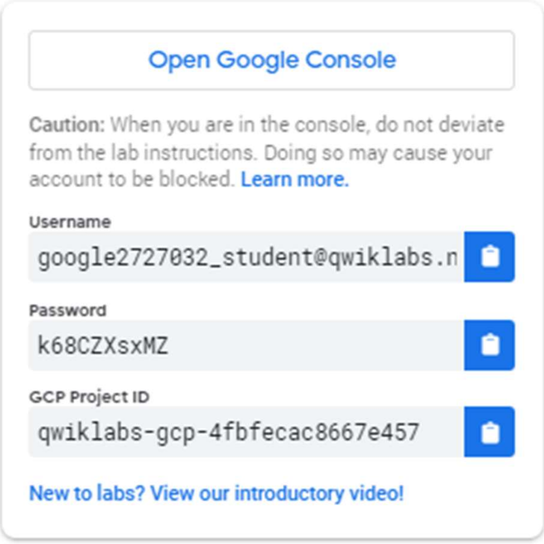
To complete this lab, you need:

- 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.



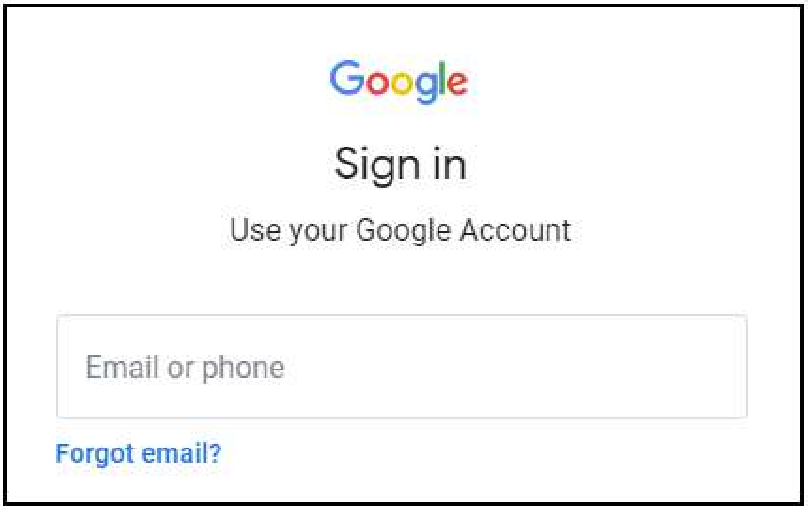
Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. Learn more.

Username
google2727032_student@qwiklabs.n

Password
k68CZXsxMZ

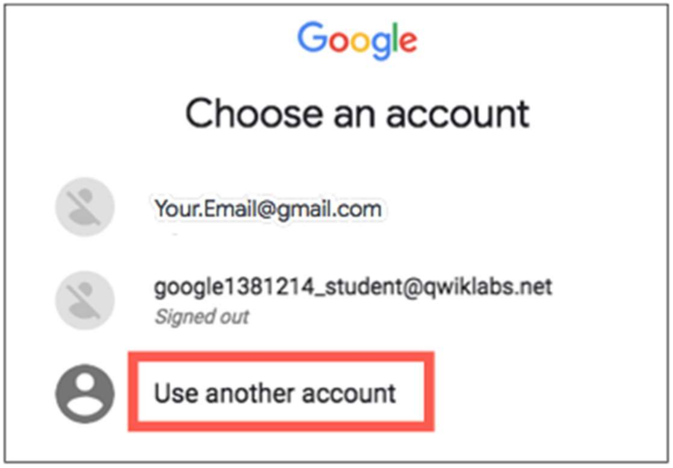
GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

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.

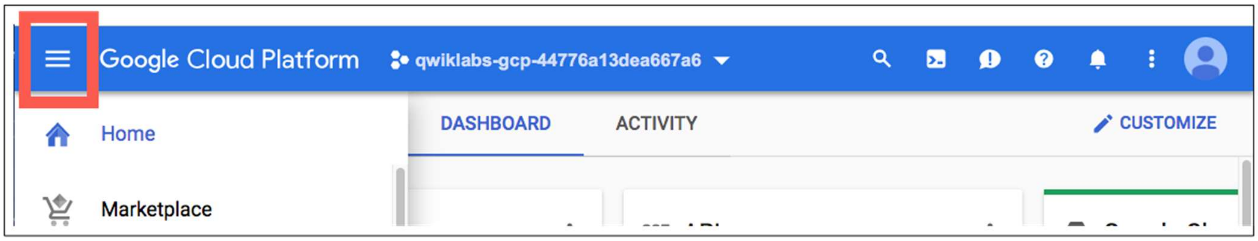
   Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

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.

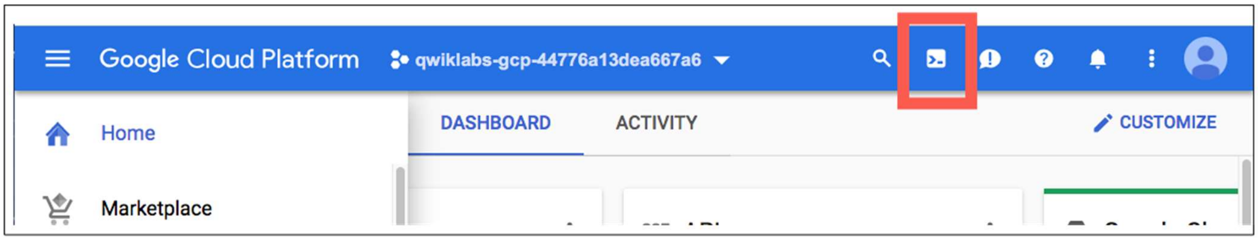
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



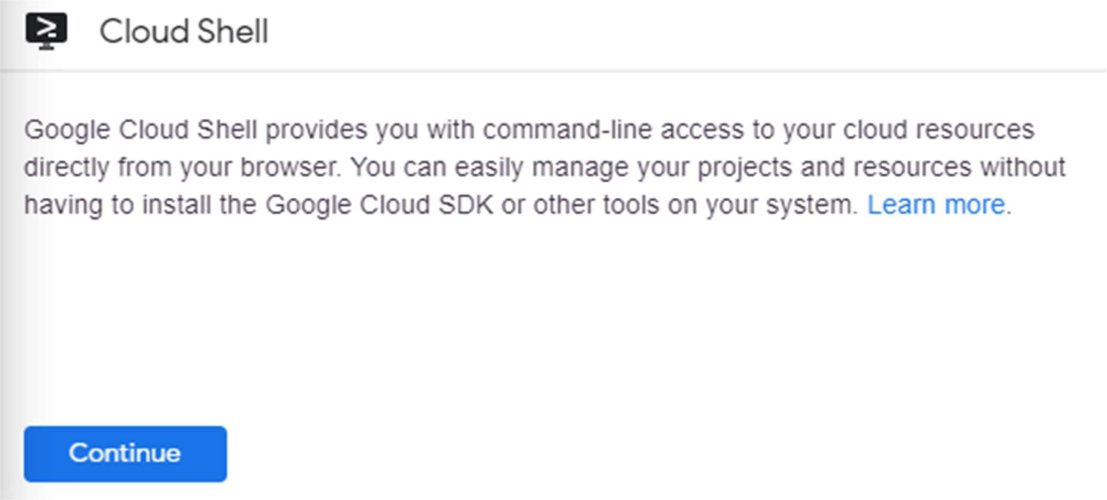
Activate Cloud Shell

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.



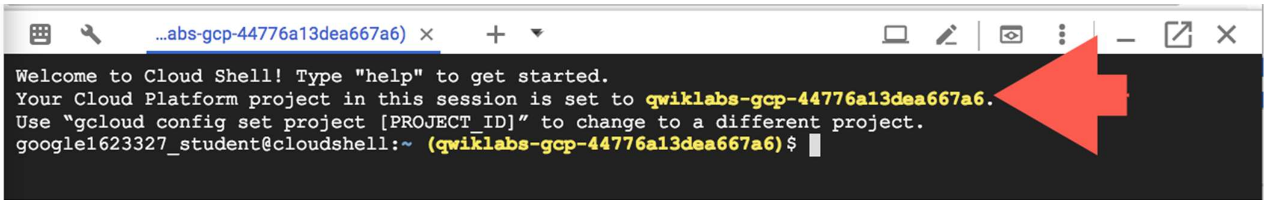
Click **Continue**.



Cloud Shell

Google Cloud Shell provides you with command-line access to your cloud resources directly from your browser. You can easily manage your projects and resources without having to install the Google Cloud SDK or other tools on your system. Learn more.

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:



```
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 Environment

First, you'll prepare your deployment environment and download a sample application.

Set the default Compute Engine zone to `us-east1-d`.

```
gcloud config set compute/zone us-east1-d
```

Clone the sample code.

```
git clone https://github.com/GoogleCloudPlatform/continuous-deployment-on-
kubernetes.git
```

Navigate to the sample code directory.

```
cd continuous-deployment-on-kubernetes
```

Creating a Kubernetes cluster

Now you'll use the Kubernetes Engine to create and manage your Kubernetes cluster.

Next provision a Kubernetes cluster using Kubernetes Engine. This step can take several minutes to complete.

```
gcloud container clusters create jenkins-cd \
--num-nodes 2 \
--machine-type n1-standard-2 \
--scopes "https://www.googleapis.com/auth/projecthosting,cloud-platform"
```

The extra scopes enable Jenkins to access Cloud Source Repositories and Google Container Registry.

Test Completed Task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

Confirm that your cluster is running.

```
gcloud container clusters list
```

Example Output:

Look for `RUNNING` in the `STATUS` column.

```
NAME        LOCATION    MASTER_VERSION  MASTER_IP      MACHINE_TYPE   NODE_VERSION
NUM_NODES  STATUS
jenkins-cd  us-east1-d  1.9.7-gke.3     35.237.126.84  n1-standard-2  1.9.7-gke.3   2
RUNNING
```

Get the credentials for your cluster. Kubernetes Engine uses these credentials to access your newly provisioned cluster.

```
gcloud container clusters get-credentials jenkins-cd
```

Confirm that you can connect to your cluster.

```
kubectl cluster-info
```

Example Output: If the cluster is running, the URLs of where your Kubernetes components are accessible display.

```
Kubernetes master is running at https://130.211.178.38
GLBCDefaultBackend is running at https://130.211.178.38/api/v1/proxy/namespaces/kube-
system/services/default-http-backend
Heapster is running at https://130.211.178.38/api/v1/proxy/namespaces/kube-
system/services/heapster
KubeDNS is running at https://130.211.178.38/api/v1/proxy/namespaces/kube-
system/services/kube-dns
kubernetes-dashboard is running at https://130.211.178.38/api/v1/proxy/namespaces/kube-
system/services/kubernetes-dashboard
```

Configure Helm

In this lab, you will use Helm to install Jenkins from the Charts repository. Helm is a package manager that makes it easy to configure and deploy Kubernetes applications. Your Cloud Shell will already have a recent, stable version of Helm pre-installed.

If curious, you can run `helm version` in Cloud Shell to check which version you are using and also ensure that Helm is installed.

1. Add Helm's stable chart repository:

```
helm repo add stable https://charts.helm.sh/stable
```

2. Update the repo to ensure you get the latest list of charts:

```
helm repo update
```

Configure and Install Jenkins

You will use a custom values file to add the Google Cloud specific plugin necessary to use service account credentials to reach your Cloud Source Repository.

1. Use the Helm CLI to deploy the chart with your configuration set.

```
helm install cd stable/jenkins -f jenkins/values.yaml --version 1.2.2 --wait
```

Test Completed Task

Click **Check my progress** to verify your performed task. If you have completed the task successfully you will granted with an assessment score.

2. Once that command completes ensure the Jenkins pod goes to the Running state and the container is in the READY state:

```
kubectl get pods
```

Example Output:

```
NAME                          READY     STATUS    RESTARTS   AGE
cd-jenkins-7c786475dd-vbhg4   1/1       Running   0          1m
```

3. Run the following command to setup port forwarding to the Jenkins UI from the Cloud Shell

```
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 &
```

4. Now, check that the Jenkins Service was created properly:

```
kubectl get svc
```

Example Output:

```
NAME               CLUSTER-IP     EXTERNAL-IP   PORT(S)     AGE
cd-jenkins         10.35.249.67   <none>        8080/TCP    3h
cd-jenkins-agent   10.35.248.1    <none>        50000/TCP   3h
kubernetes         10.35.240.1    <none>        443/TCP     9h
```

We are using the Kubernetes Plugin so that our builder nodes will be automatically launched as necessary when the Jenkins master requests them. Upon completion of their work, they will automatically be turned down and their resources added back to the clusters resource pool.

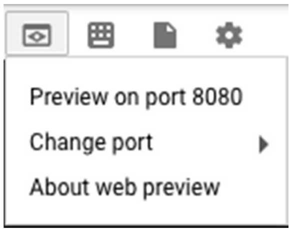
Notice that this service exposes ports `8080` and `50000` for any pods that match the `selector`. This will expose the Jenkins web UI and builder/agent registration ports within the Kubernetes cluster. Additionally, the `jenkins-ui` service is exposed using a ClusterIP so that it is not accessible from outside the cluster.

Connect to Jenkins

1. The Jenkins chart will automatically create an admin password for you. To retrieve it, run:

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

2. To get to the Jenkins user interface, click on the Web Preview button in cloud shell, then click **Preview on port 8080**:



3. You should now be able to log in with username admin and your auto-generated password.

You now have Jenkins set up in your Kubernetes cluster!

Test your Understanding

Below are multiple-choice questions to reinforce your understanding of this lab's concepts. Answer them to the best of your abilities.

The additional scope enables access to Kubernetes cluster for interacting with other Google Cloud services.
True

The role-based access control (RBAC) is an approach to restricting system access to authorized users.
True

Congratulations!

Finish Your Quest

This self-paced lab is part of the Qwiklabs Quest, Google Cloud Solutions I: Scaling Your Infrastructure. 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.

Take Your Next Lab

Continue your Quest with Continuous Delivery Pipelines with Spinnaker and Kubernetes Engine, or check out these suggestions:

- Autoscaling an Instance Group with Stackdriver Custom Metrics
- Deploying Memcached on Kubernetes Engine

Next Steps / Learn More

Here are some follow-up steps :

- Learn more about Jenkins on Kubernetes Engine best practices.
- Learn about how to configure Jenkins for Kubernetes Engine.
- Learn about how to set up continuous deployment to Kubernetes Engine using Jenkins.

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 December 21, 2020

Lab Last Tested October 19, 2020