

Deploying to Kubernetes v1.6

1 hour 30 minutes1 Credit

Rate Lab

Overview

The new goal of this lab is to get you ready for scaling and managing containers in production.

And that's where [deployments](#) come in. Deployments are a declarative way to ensure that the number of pods running is equal to the desired number of pods specified by the user.

Introduction to Deployments

Deployments abstract away the low level details of managing pods. They provide a single stable name that you can use to update an application. Behind the scenes, deployments rely on [ReplicaSets](#) to manage starting, stopping, scaling, and restarting the pods if they happen to go down for some reason. If pods need to be updated or scaled, the deployment will handle all of the details for you.

Deployments (and ReplicaSets) are powered by control loops. Control loops are a design pattern for distributed software that allows you to declaratively define your desired state and have the software implement the desired state for you based on the current state. You'll see more about how that works below.

Setup

Step 1

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 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 the Google Cloud Platform 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 GCP account or project, do not use it for this lab.


How to start your lab and sign in to the 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.

[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 

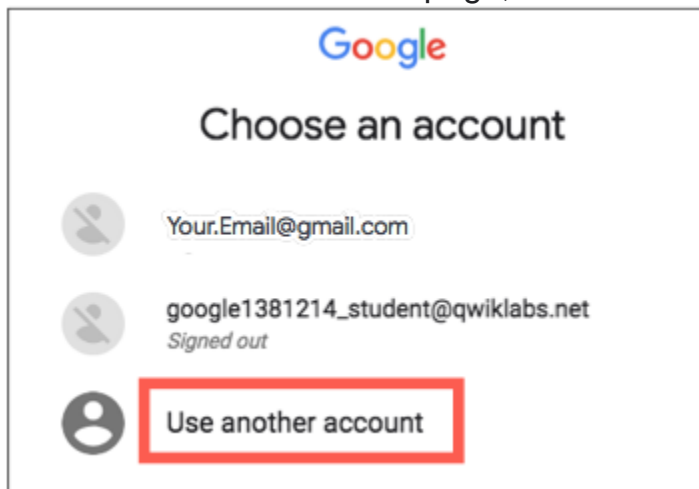
GCP Project ID
qwiklabs-gcp-4fbfecac8667e457 

[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 **Choose an account** page.

Tip: Open the tabs in separate windows, side-by-side.

3. On the Choose an account page, click **Use Another Account**.



4. The Sign in page opens. Paste the username that you copied from the Connection Details panel. Then copy and paste the password.

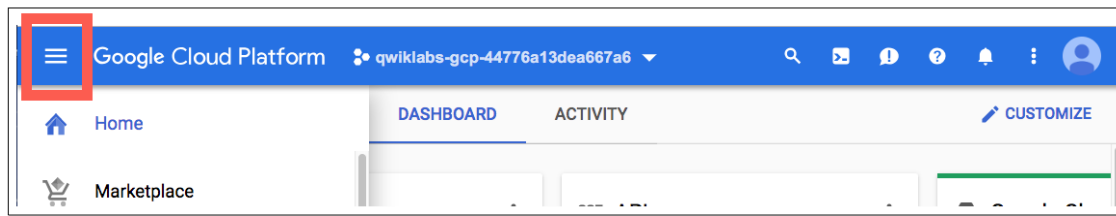
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own GCP account, do not use it for this lab (avoids incurring charges).

5. 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 GCP console opens in this tab.

Note: You can view the menu with a list of GCP Products and Services by clicking the **Navigation menu** at the top-left, next to “Google Cloud Platform”.



Step 2

Make sure the following APIs are enabled in Cloud Platform Console:

- Kubernetes Engine API
- Container Registry API

On the **Navigation menu** (≡), click **APIs & services**.
Scroll down and confirm that your APIs are enabled.

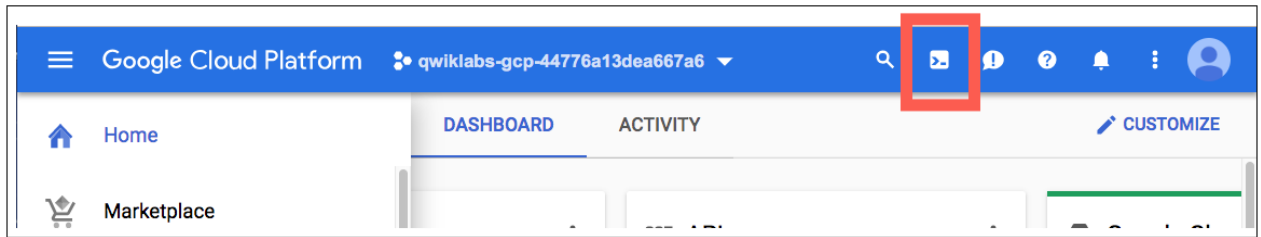
If an API is missing, click **ENABLE APIS AND SERVICES** at the top, search for the API by name, and enable it for your project.

Step 3

Activate Google Cloud Shell

Google 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. Google Cloud Shell provides command-line access to your GCP resources.

1. In GCP console, on the top right toolbar, click the Open Cloud Shell button.



2. In the dialog box that opens, click **START CLOUD SHELL**:

Google Cloud Shell

Free, pre-installed with the tools you need for the Google Cloud Platform. [Learn More](#)

```
example-vm-2    europe-west1-b f1-micro           10.240.119.112  104.155.36.122  RUNN
example-vm-3    us-central1-f  f1-micro           10.240.57.1     104.154.76.241  RUNN
google1530209_student@cloudshell:~$
google1530209_student@cloudshell:~$ git clone https://github.com/GoogleCloud/appengine-exa
Cloning into 'appengine-example'...
remote: Counting objects: 476, done.
remote: Total 476 (delta 0), reused 0 (delta 0), pack-reused 476
Receiving objects: 100% (476/476), 432.65 KiB | 0 bytes/s, done.
Checking connectivity... done.
google1530209_student@cloudshell:~$ cd appengine-example
google1530209_student@cloudshell:~/appengine-example$ appcfg.py -
```


Real Linux environment	Configured for Google Cloud	Popular language support
<ul style="list-style-type: none">• Linux Debian-based OS• 5GB persisted home directory• Add, edit and save files	<ul style="list-style-type: none">• Google Cloud SDK• Google App Engine SDK• Docker• Git• Text editors• Build tools• View more	<ul style="list-style-type: none">• Python• Java• Go• Node.js

[CANCEL](#) [START CLOUD SHELL](#)

You can click "START CLOUD SHELL" immediately when the dialog box opens.

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 Platform. 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
```

Full documentation of **gcloud** is available on [Google Cloud gcloud Overview](#).

Step 4

Define your zone as a project default zone. This way you do not need to specify `-zone` parameter in **gcloud** commands.

```
gcloud config set compute/zone us-central1-a
```

Get the sample code for creating and running containers and deployments:

```
git clone https://github.com/googlecodelabs/orchestrate-with-kubernetes.git
```

Step 5

Start your Kubernetes cluster with 5 nodes.

```
cd orchestrate-with-kubernetes/kubernetes
```

In Cloud Shell, run the following command to start a Kubernetes cluster called bootcamp that runs 5 nodes.

```
gcloud container clusters create bootcamp --num-nodes 5 --scopes "https://www.googleapis.com/auth/projecthosting,storage-rw"
```

The scopes argument provides access to project hosting and Google Cloud Storage APIs that you'll use later.

It takes several minutes to create a cluster as Kubernetes Engine provisions virtual machines for you. It spins up one or more master nodes and multiple configured worker nodes. This is one of the advantages of a managed service.

Learn About Deployment Objects

Step 1

Run the `explain` command in `kubectl` to tell you about the deployment object.

```
kubectl explain deployment
```

Step 2

Run the command with the `--recursive` option to see all of the fields.

```
kubectl explain deployment --recursive
```

Step 3

Use the `explain` command as you go through the lab to help you understand the structure of a deployment object and understand what the individual fields do.

```
kubectl explain deployment.metadata.name
```

Create a Deployment

Create a simple deployment.

Step 1

Examine the deployment configuration file.

```
cat deployments/auth.yaml
```

`kubectl create` will create the `auth` deployment with one replica, using version 1.0.0 of the `auth` container. To scale the number of pods, you simply change the `replicas` field.

Step 2

Create the deployment object using `kubectl create`.

```
kubectl create -f deployments/auth.yaml
```

Step 3

Verify that it was created.

```
kubectl get deployments
```

Step 4

Kubernetes creates a `ReplicaSet` for the deployment.

Run the following command to verify it. You should see a `ReplicaSet` with a name like `auth-xxxxxxx`.

```
kubectl get replicaset
```

Step 5

Run the following command to view the pods created for your deployment. A single pod was created when the `ReplicaSet` was created.

```
kubectl get pods
```

Step 6

With your pod running, it's time to put it behind a service. Use the `kubectl create` command to create the `auth` service.

```
kubectl create -f services/auth.yaml
```

Step 7

Do the same to create and expose the `hello` and `frontend` deployments.

```
kubectl create -f deployments/hello.yaml
kubectl create -f services/hello.yaml
kubectl create configmap nginx-frontend-conf --from-file=nginx/frontend.conf

kubectl create secret generic tls-certs --from-file tls/

kubectl create -f deployments/frontend.yaml
kubectl create -f services/frontend.yaml
```

You created a ConfigMap and secret for the frontend.

Click *Check my progress* to verify the objective.

Create cluster and deployments (Auth, Hello, and Frontend)

Check my progress

Step 8

Interact with the frontend.

Get its external IP.

```
kubectl get services frontend
```

You may need to re-run this command every few seconds until the External IP is populated.

And `curl` the service.

```
curl -ks https://<EXTERNAL-IP>
```

You get the "hello" response. Use the output templating feature of `kubectl` to run `curl` as a one-line command.

```
curl -ks https://`kubectl get svc frontend -o=jsonpath='{.status.loadBalancer.ingress[0].ip}'`
```

Scale a Deployment

Update the `spec.replicas` field to scale the deployment.

Step 1

Run the `kubectl explain` command to see an explanation of the field.

```
kubectl explain deployment.spec.replicas
```

Step 2

You can update the `replicas` field most easily using the `kubectl scale` command.

```
kubectl scale deployment hello --replicas=5
```

It may take a minute or so for all the new pods to start up.

Step 3

Kubernetes updates the ReplicaSet and starts new pods to equal 5.

Verify there are 5 pods running.

```
kubectl get pods | grep hello- | wc -l
```

Step 4

Scale back the application.

```
kubectl scale deployment hello --replicas=3
```

Step 5

Verify the correct number of pods.

```
kubectl get pods | grep hello- | wc -l
```

Congratulations!

You learned about Kubernetes deployments and how to manage and scale a group of pods.

Rolling Updates

Deployments update images to new versions through rolling updates. When a deployment is updated with a new version, it creates a new ReplicaSet and slowly increases the number of replicas in the new ReplicaSet as it decreases the replicas in the old ReplicaSet.

Trigger a Rolling Update

Step 1

Run the following command to update your deployment.

```
kubectl edit deployment hello
```

Step 2

Change the `image` in `containers` section to the following, then save and exit.

```
containers:
- name: hello
  image: kelseyhightower/hello:2.0.0
```

The editor uses **vi** commands:

1. Use arrow keys to hover over version number **1**
2. Type **r** to replace it, and enter **2**
3. Type **:wq!** and hit **Enter** to write and quit the file.

If you have difficulty and are in a class, ask your instructor for help.

The updated deployment is saved to your cluster and Kubernetes begins a rolling update.

Step 3

You can see the new ReplicaSet that Kubernetes creates.

```
kubectl get replicaset
```

If you fail to see a new ReplicaSet, make sure you changed the `image` in `containers`, and not one of the other references in `labels`.

Step 4

View the new entry in the rollout history.

```
kubectl rollout history deployment/hello
```

Pause a Rolling Update

If you detect problems with a running rollout, pause it to stop the update.

Step 1

Pause the update.

```
kubectl rollout pause deployment/hello
```

Step 2

Verify the current state of the rollout.

```
kubectl rollout status deployment/hello
```

Step 3

Verify this with the pods.

```
kubectl get pods -o jsonpath --template='{range .items[*]}{.metadata.name}{"\t"}{"\t"}{.spec.containers[0].image}{"\n"}{end}'
```

Resume a Rolling Update

The rollout is paused which means that some pods are at the new version and some pods are at the older version.

Step 1

Use the `resume` command to continue the rollout.

```
kubectl rollout resume deployment/hello
```

Step 2

Run the status command to verify the rollout is complete.

```
kubectl rollout status deployment/hello
```

You'll get the following:

```
deployment "hello" successfully rolled out
```

Rollback an Update

If a bug occurs in your new version, users connected to new pods will experience the issue.

Step 1

Use the `rollout undo` command to roll back to the previous version, then fix any bugs.

```
kubectl rollout undo deployment/hello
```

Step 2

Verify the rollback in the deployment's history.

```
kubectl rollout history deployment/hello
```

Step 3

Verify all pods have rolled back to the previous version.

```
kubectl get pods -o jsonpath --template='{range .items[*]}{.metadata.name}{ "\t"}{ "\t"}{.spec.containers[0].image}{ "\n"}{end} '
```

Congratulations!

You learned how to roll out application updates without downtime.

Canary Deployments

Run a canary deployment to test a new deployment in production with a subset of users. This mitigates risk with new releases.

Create a Canary Deployment

A canary deployment consists of a separate deployment from your stable deployment and a service that targets them both at the same time.

Step 1

Examine the file that creates a canary deployment for your new version.

```
cat deployments/hello-canary.yaml
```

It includes the following:

- the deployment `hello-canary`
- 1 pod (replica)
- selectors `app: hello` and `track: canary`
- an image with version 2.0.0.

Step 2

Create the canary deployment.

```
kubectl create -f deployments/hello-canary.yaml
```

Click *Check my progress* to verify the objective.

Canary Deployments

Check my progress

Step 3

After the canary deployment is created, verify you have two deployments `hello` and `hello-canary`.

```
kubectl get deployments
```

The `hello` service selector uses `app: hello`, which matches pods in both deployments. However, the canary deployment has fewer pods, and is only used by a subset of users.

Verify the Canary Deployment

You can verify both `hello` versions being served by requests.

```
curl -ks https://`kubectl get svc frontend -o=jsonpath='{.status.loadBalancer.ingress[0].ip}'`/version
```

Run the command several times and confirm that `hello 1.0.0` serves about $\frac{3}{4}$ (75%) of requests and `2.0.0` serves about $\frac{1}{4}$ (25%).

By default, every request has a chance to be served by the canary deployment. If you want users to get all their responses from the same version, enable session affinity in the configuration file as follows:

```
spec:  
sessionAffinity: ClientIP
```

Clean Up

You're done using the canary deployment.

Delete it and the service as follows.

```
kubectl delete deployment hello-canary
```

Congratulations!

You learned about canary deployments and how to test new versions of an application in a live environment.

Blue-Green Deployments

You can use blue-green deployments if it's more beneficial to modify load balancers to point to a new, fully-tested deployment all at once.

A downside is you need double the resources to host both versions of your application during the switch.

The Service

You use the existing `hello` deployment for the blue version and a new `hello-green` deployment for the green version.

Deployments have the following label:

Deployment	Label Name	Label Value
hello (blue)	version	1.0.0
hello-green	version	2.0.0

You use two nearly-identical service files (`hello-blue` and `hello-green`) to switch between versions. The only difference between these files is their `version` selector. You could edit the service while it's running and change the `version` selector, but switching files is easier for labs.

First, update the service to use the blue deployment:

```
kubectl apply -f services/hello-blue.yaml
```

Create a Blue-Green Deployment

Step 1

Create the green deployment.

```
kubectl create -f deployments/hello-green.yaml
```

Step 2

Verify the blue deployment (1.0.0) is still being used.

```
curl -ks https://`kubectl get svc frontend -o=jsonpath='{.status.loadBalancer.ingress[0].ip}'`/version
```

Step 3

Run the following command to update the service to use the green deployment.

```
kubectl apply -f services/hello-green.yaml
```

Step 4

Verify the green deployment is being used.

```
curl -ks https://`kubectl get svc frontend -o=jsonpath='{.status.loadBalancer.ingress[0].ip}'`/version
```


Rollback a Blue-Green Deployment

You can roll back to the old version.

Step 1

While the green deployment is still running, simply update the service to the old (blue) deployment.

```
kubectl apply -f services/hello-blue.yaml
```

Step 2

Verify that the blue deployment is being used.

```
curl -ks https://`kubectl get svc frontend -o=jsonpath='{.status.loadBalancer.ingress[0].ip}'` /version
```

Congratulations!

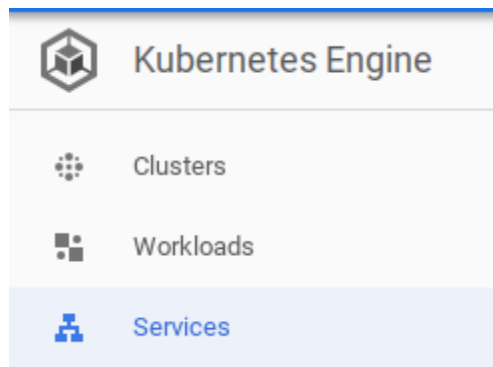
You learned how to use blue-green deployments to switch application versions all at once.

Use the Web UI

You can also use Admin Console to view and interact with your Kubernetes Engine objects in the web interface rather than the command-line.

Step 1

In Admin Console, click **Navigation menu** and select **Kubernetes Engine**. Select a resource in the tab on the left, for example **Services**.



Step 2

Investigate each set of objects by navigating through the following pages:

- **Clusters**
- **Workloads** (Pods)
- **Services**
- **Configuration** (ConfigMaps and Secrets).

On each page, scroll over an item in the **Name** field and click its link to view more information about it. For example, on the **Services** page, click **frontend**. Your front end Services will display as follows with status, types, endpoints, and number of running Pods:

Name ^	Status	Service Type	Endpoints	Pods
auth	✓ Ok	Cluster IP	10.7.251.93	1 / 1
frontend	✓ Ok	Load balancer	35.203.136.90:443 ↗	1 / 1
hello	✓ Ok	Cluster IP	10.7.253.127	3 / 3

End your lab