

# Continuous Deployment with Spinnaker

This hands-on lab shows you how to create a continuous delivery pipeline using Google Kubernetes Engine, Google Cloud Source Repositories, Google Cloud Container Builder, and Spinnaker. After you create a sample application, you configure these services to automatically build, test, and deploy it. When you modify the application code, the changes trigger the continuous delivery pipeline to automatically rebuild, retest, and redeploy the new version.

## **Objectives**

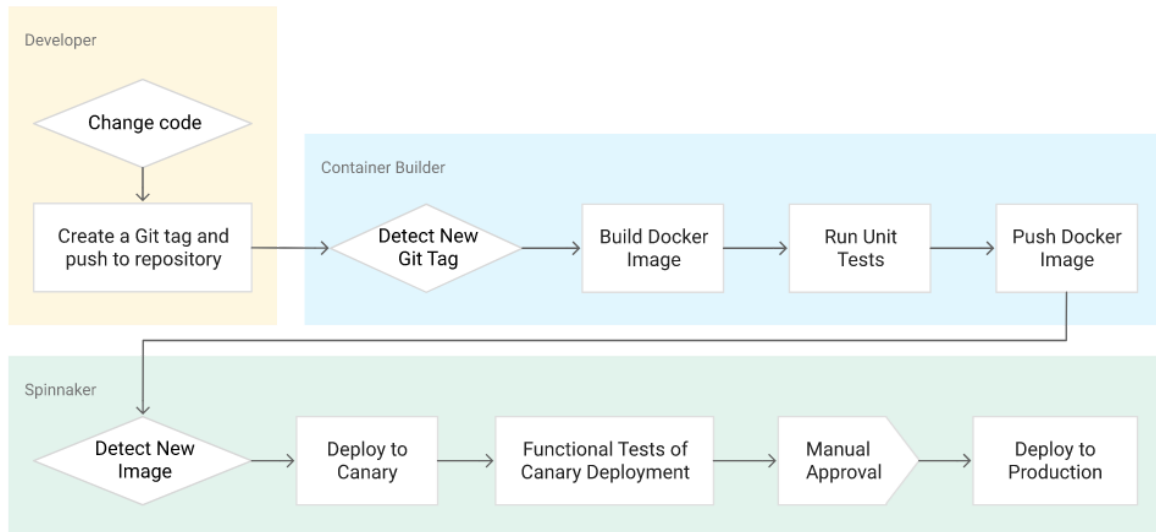
- Set up your environment by launching [Google Cloud Shell](#), creating a Kubernetes Engine cluster, and configuring your identity and user management scheme.
- Download a sample application, create a Git repository then upload it to a Google Cloud Source Repository.
- Deploy Spinnaker to Kubernetes Engine using [Helm](#).
- Build your Docker image.
- Create triggers to create Docker images when your application changes.
- Configure a Spinnaker pipeline to reliably and continuously deploy your application to Kubernetes Engine.
- Deploy a code change, triggering the pipeline, and watch it roll out to production.

## Pipeline architecture

To continuously deliver application updates to your users, you need an automated process that reliably builds, tests, and updates your software. Code changes should automatically flow through a pipeline that includes artifact creation, unit testing, functional testing, and production rollout. In some cases, you want a code update to apply to only a subset of your users, so that it is exercised realistically before you push it to your entire user base. If one of these [canary](#) releases proves unsatisfactory, your automated procedure must be able to quickly roll back the software changes.

## Application delivery pipeline

In this lab, you will build the continuous delivery pipeline shown in the following diagram.



## Setup and Requirements

### Qwiklabs setup

#### What you'll need

To complete this lab, you'll need:

- Access to a standard internet browser (Chrome browser recommended).

- Time. Note the lab's **Completion** time in Qwiklabs. This is an estimate of the time it should take to complete all steps. Plan your schedule so you have time to complete the lab. Once you start the lab, you will not be able to pause and return later (you begin at step 1 every time you start a lab).
- The lab's **Access** time is how long your lab resources will be available. If you finish your lab with access time still available, you will be able to explore the Google Cloud Platform or work on any section of the lab that was marked "if you have time". Once the Access time runs out, your lab will end and all resources will terminate.
- You **DO NOT** need a Google Cloud Platform account or project. An account, project and associated resources are provided to you as part of this lab.
- If you already have your own GCP account, make sure you do not use it for this lab.
- If your lab prompts you to log into the console, **use only the student account provided to you by the lab**. This prevents you from incurring charges for lab activities in your personal GCP account.

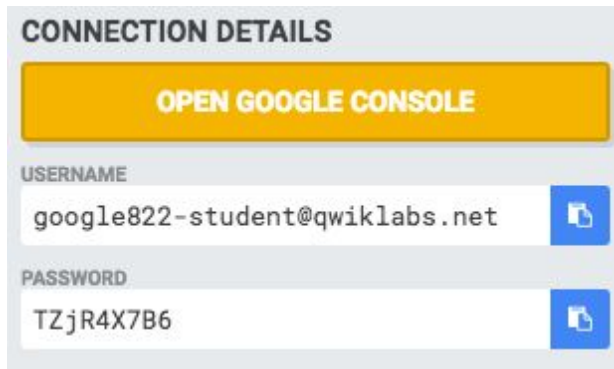
### Start your lab

When you are ready, click **Start Lab**. You can track your lab's progress with the status bar at the top of your screen.

**Important** What is happening during this time? Your lab is spinning up GCP resources for you behind the scenes, including an account, a project, resources within the project, and permission for you to control the resources needed to run the lab. This means that instead of spending time manually setting up a project and building resources from scratch as part of your lab, you can begin learning more quickly.

### Find Your Lab's GCP Username and Password

To access the resources and console for this lab, locate the Connection Details panel in Qwiklabs. Here you will find the account ID and password for the account you will use to log in to the Google Cloud Platform:



**CONNECTION DETAILS**

**OPEN GOOGLE CONSOLE**

USERNAME  
google822-student@qwiklabs.net

PASSWORD  
TZjR4X7B6

If your lab provides other resource identifiers or connection-related information, it will appear on this panel as well.

### **Log in to Google Cloud Console**

Using the Qwiklabs browser tab/window or the separate browser you are using for the Qwiklabs session, copy the Username from the Connection Details panel and click the **Open Google Console** button.

You'll be asked to Choose an account. Click **Use another account**.



## Choose an account



gcpstaging10382\_student@qwiklabs.net  
*Signed out*



gcpstaging10408\_student@qwiklabs.net  
*Signed out*



Use another account

Paste in the Username, and then the Password as prompted:



## Sign in

to continue to Google Cloud Platform

Enter your email

gcpstaging277-student@qwiklabs.net

[More options](#)

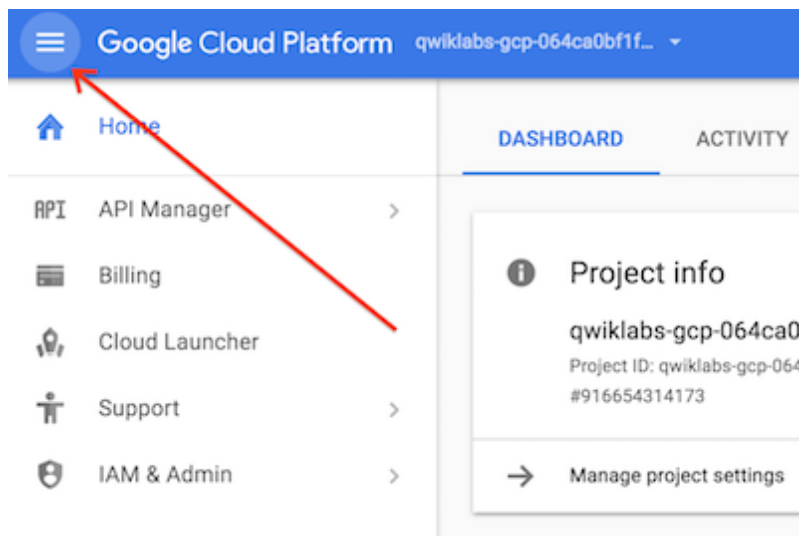
NEXT

Accept the terms and conditions.

Since this is a temporary account, which you will only have to access for this one lab:

- Do not add recovery options
- Do not sign up for free trials

**Note:** You can view the list of services by clicking the GCP Navigation menu button at the top-left next to "Google Cloud Platform".



## The Google Cloud Shell

### Activate Google Cloud Shell

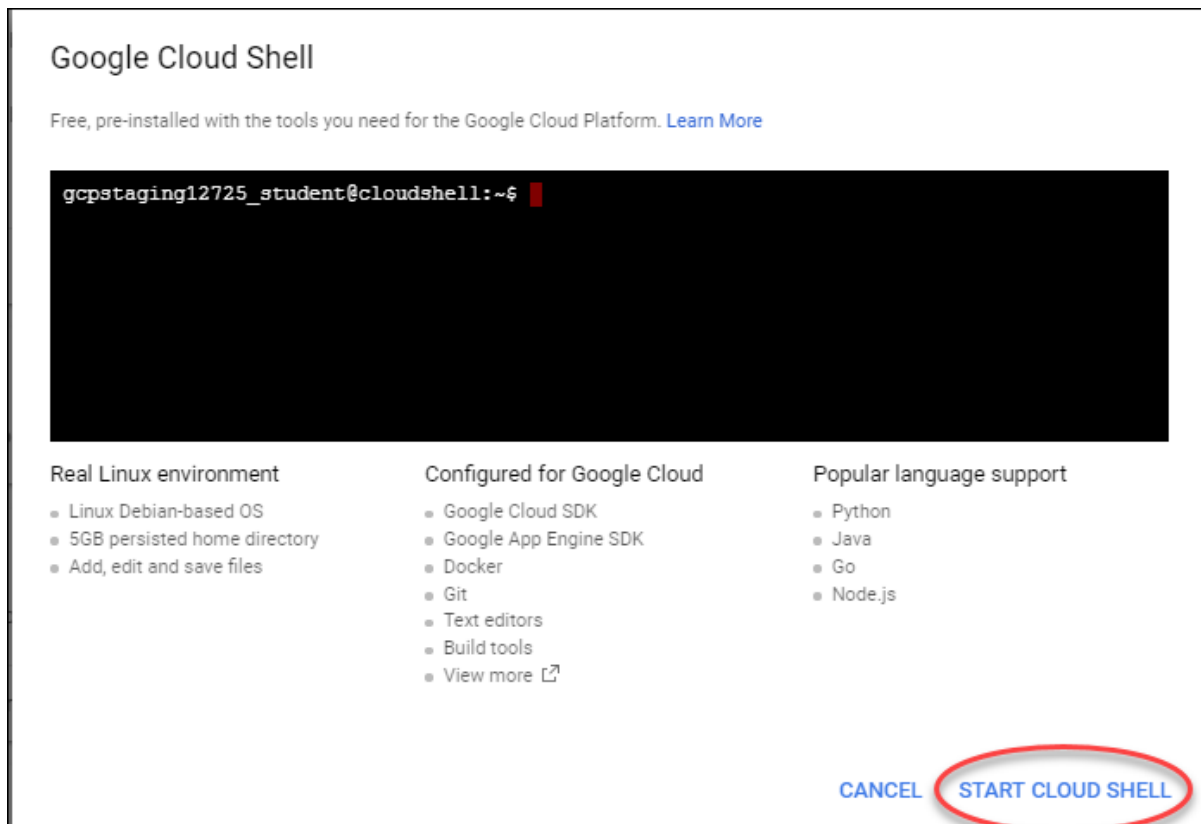
Google Cloud Shell provides command-line access to your GCP resources.

From the GCP Console click the **Cloud Shell** icon on the top right toolbar:



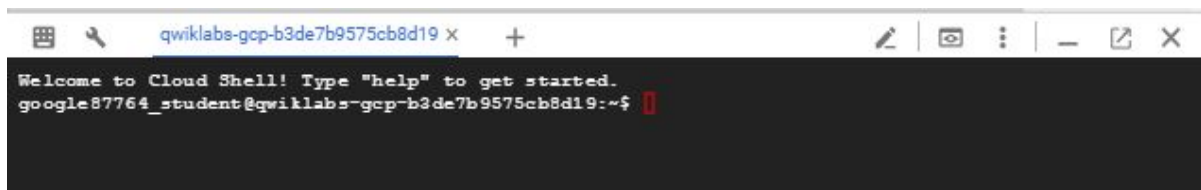


Then click **START CLOUD SHELL**:



You can click **START CLOUD SHELL** immediately when the dialog comes up instead of waiting in the dialog until the Cloud Shell provisions.

It takes a few moments to provision and connects to the environment:



The Cloud Shell is a virtual machine loaded with all the development tools you'll need. It offers a persistent 5GB home directory, and runs on the Google Cloud, greatly enhancing network performance and authentication.

Once connected to the cloud shell, you'll see that you are already authenticated and the project is set to your *PROJECT\_ID*:

```
gcloud auth list
```

Output:

```
Credentialed accounts:
```

```
- <myaccount>@<mydomain>.com (active)
```

**Note:** gcloud is the powerful and unified command-line tool for Google Cloud Platform. Full documentation is available on [Google Cloud gcloud Overview](#). It comes pre-installed on Cloud Shell and supports tab-completion.

```
gcloud config list project
```

Output:

```
[core]
```

```
project = <PROJECT_ID>
```

## Set up your environment

Configure the infrastructure and identities required for this lab. First, you'll create a Kubernetes Engine cluster to deploy Spinnaker and the sample application.

Set the default compute zone:

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

Create a Kubernetes Engine using the Spinnaker tutorial sample application.

```
gcloud container clusters create spinnaker-tutorial  
--machine-type=n1-standard-2 --enable-legacy-authorization
```

This deployment will take between **five to ten minutes** to complete. You may see a warning about default scopes that you can safely ignore as it has no impact on this lab.

Once the command completes you'll have a report detailing the name, location, version, ip-address, machine-type, node version, number of nodes and status of the cluster indicating that the cluster is running.

## Configure identity and access management

Create a Cloud Identity Access Management (Cloud IAM) [service account](#) to delegate permissions to Spinnaker, allowing it to store data in Cloud Storage. Spinnaker stores its pipeline data in Cloud Storage to ensure reliability and resiliency. If your Spinnaker deployment unexpectedly fails, you can create an

identical deployment in minutes with access to the same pipeline data as the original.

Upload your startup script to a Cloud Storage bucket by following these steps:

Create the service account:

```
gcloud iam service-accounts create spinnaker-storage-account \
  --display-name spinnaker-storage-account
```

Store the service account email address and your current project ID in environment variables for use in later commands:

```
export SA_EMAIL=$(gcloud iam service-accounts list \
  --filter="displayName:spinnaker-storage-account" \
  --format='value(email)')
```

```
export PROJECT=$(gcloud info --format='value(config.project)')
```

Bind the `storage.admin` role to your service account:

```
gcloud projects add-iam-policy-binding \
  $PROJECT --role roles/storage.admin --member serviceAccount:$SA_EMAIL
```

Download the service account key. In a later step, you will install Spinnaker and upload this key to Kubernetes Engine.

```
gcloud iam service-accounts keys create spinnaker-sa.json \
  --iam-account $SA_EMAIL
```

Output (do not copy)

```
created key [f36140d95582818bcefc6fe7c3a270494a00323] of type [json] as  
[spinnaker-sa.json] for  
[spinnaker-storage-account@qwiklabs-gcp-60ca368a34737ebf.iam.gserviceaccount.com]
```

## Deploying Spinnaker using Helm

In this section, you'll use [Helm](#) to deploy Spinnaker from the [Charts](#) repository. Helm is a package manager you can use to configure and deploy [Kubernetes applications](#).

### Install Helm

Download and install the Helm binary:

```
wget  
https://storage.googleapis.com/kubernetes-helm/helm-v2.5.0-linux-amd64.tar.  
gz
```

Unzip the file to your local system:

```
tar xzfv helm-v2.5.0-linux-amd64.tar.gz  
  
cp linux-amd64/helm .
```

Initialize Helm to install Tiller, the server side of Helm, in your cluster:

```
./helm init
```

```
./helm repo update
```

Ensure that Helm is properly installed by running the following command. If Helm is correctly installed, v2.5.0 appears for both client and server.

```
./helm version
```

Output (do not copy)

```
Client: &version.Version{SemVer:"v2.5.0",  
GitCommit:"012cb0ac1a1b2f888144ef5a67b8dab6c2d45be6",  
GitTreeState:"clean"}Server: &version.Version{SemVer:"v2.5.0",  
GitCommit:"012cb0ac1a1b2f888144ef5a67b8dab6c2d45be6",  
GitTreeState:"clean"}
```

When installing and configuring Helm you may see errors related to the configuration of listener services or port binding. These relate to issues this version of Helm has with the ipv6 configuration of Cloud Shell. The errors can be ignored as they do not affect this lab.

## Configure Spinnaker

In Cloud Shell, create a bucket for Spinnaker to store its pipeline configuration:

```
export PROJECT=$(gcloud info \\\n--format='value(config.project)')
```

```
export BUCKET=$PROJECT-spinnaker-config
```

```
gsutil mb -c regional -l us-central1 gs://$BUCKET
```

Create your configuration file by executing the following:

```
export SA_JSON=$(cat spinnaker-sa.json)
```

```
export PROJECT=$(gcloud info --format='value(config.project)')
```

```
export BUCKET=$PROJECT-spinnaker-config
```

```
cat > spinnaker-config.yaml <<EOF
```

```
storageBucket: $BUCKET
```

```
gcs:
```

```
  enabled: true
```

```
  project: $PROJECT
```

```
  jsonKey: '$SA_JSON'
```

```
# Disable minio the default
```

```
minio:
```

```
  enabled: false
```

```
# Configure your Docker registries here
```

```
accounts:
```

```
- name: gcr
```

```
  address: https://gcr.io
```

```
  username: _json_key
```

```
  password: '$SA_JSON'
```

```
  email: 1234@5678.com
```

```
EOF
```

## Deploy the Spinnaker chart

Use the Helm command-line interface to deploy the chart with your configuration set. This command typically takes **five to ten minutes** to complete.

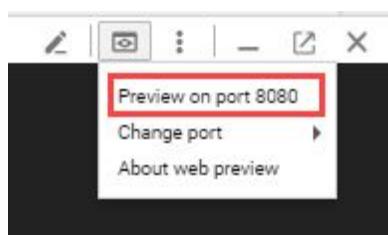
```
./helm install -n cd stable/spinnaker -f spinnaker-config.yaml --timeout  
600 \  
--version 0.3.1
```

You will see a warning here about a listener not getting created - you can ignore it. The installation will proceed after a few minutes.

After the command completes, run the following command to set up port forwarding to Spinnaker from Cloud Shell:

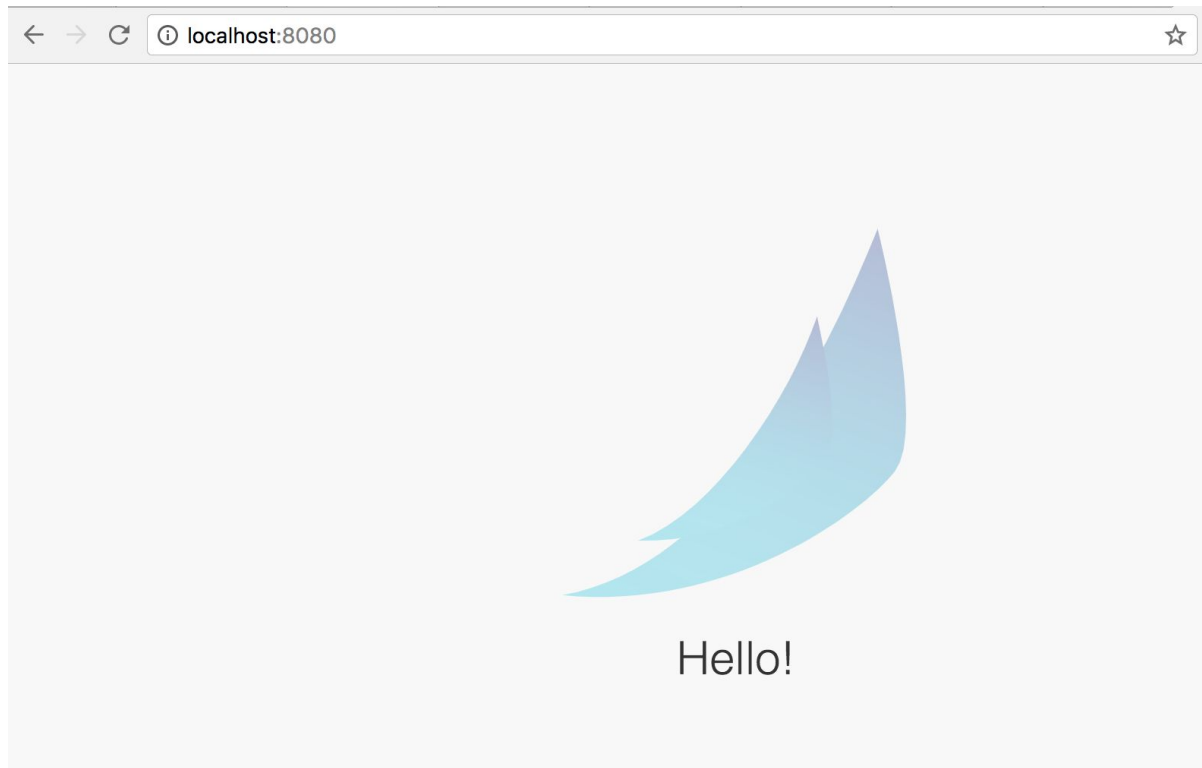
```
export DECK_POD=$(kubectl get pods --namespace default -l "component=deck"  
\  
-o jsonpath="{.items[0].metadata.name}")  
kubectl port-forward --namespace default $DECK_POD 8080:9000 >> /dev/null &
```

To open the Spinnaker user interface, click the **Web Preview** icon at the top of the Cloud Shell window and select **Preview on port 8080**.



You should see the welcome screen, followed by the Spinnaker user interface:





Leave this tab open, this is where you'll access the Spinnaker user interface.

## Building the Docker image

Next you will configure Container Builder to detect changes to your application source code, build a Docker image, and then push it to Container Registry.

### Create your source code repository

Switch back to the Cloud Shell tab and download the sample application source code:

```
wget https://gke-spinnaker.storage.googleapis.com/sample-app.tgz
```

Unpack the source code:

```
tar xzfv sample-app.tgz
```

Change to the sample application source directory:

```
cd sample-app
```

Set the username and email address for your Git commits in this repository.

Replace [EMAIL\_ADDRESS] with the lab username email address, and replace

[USERNAME] with a username you create.

```
git config --global user.email "[EMAIL_ADDRESS]"
```

```
git config --global user.name "[USERNAME]"
```

Make the initial commit to your source code repository:

```
git init
```

```
git add .
```

```
git commit -m "Initial commit"
```

Create a repository to host your code:

```
gcloud source repos create sample-app
```

You can disregard the "you may be billed for this repository" message.

```
git config credential.helper gcloud.sh
```

Add your newly created repository as remote:

```
export PROJECT=$(gcloud info --format='value(config.project)')
```

```
git remote add origin
```

```
https://source.developers.google.com/p/$PROJECT/r/sample-app
```

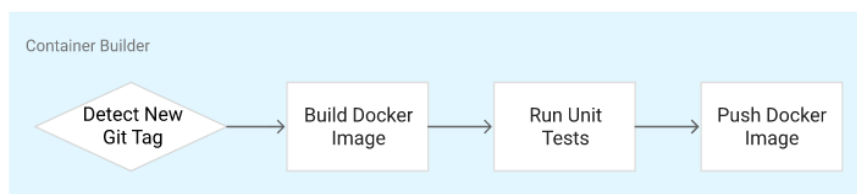
Push your code to the new repository's master branch:

```
git push origin master
```

Check that you can see your source code in the Console by clicking **Navigation menu**, then in the **Tools** section, click **Source Repository > Repositories**.

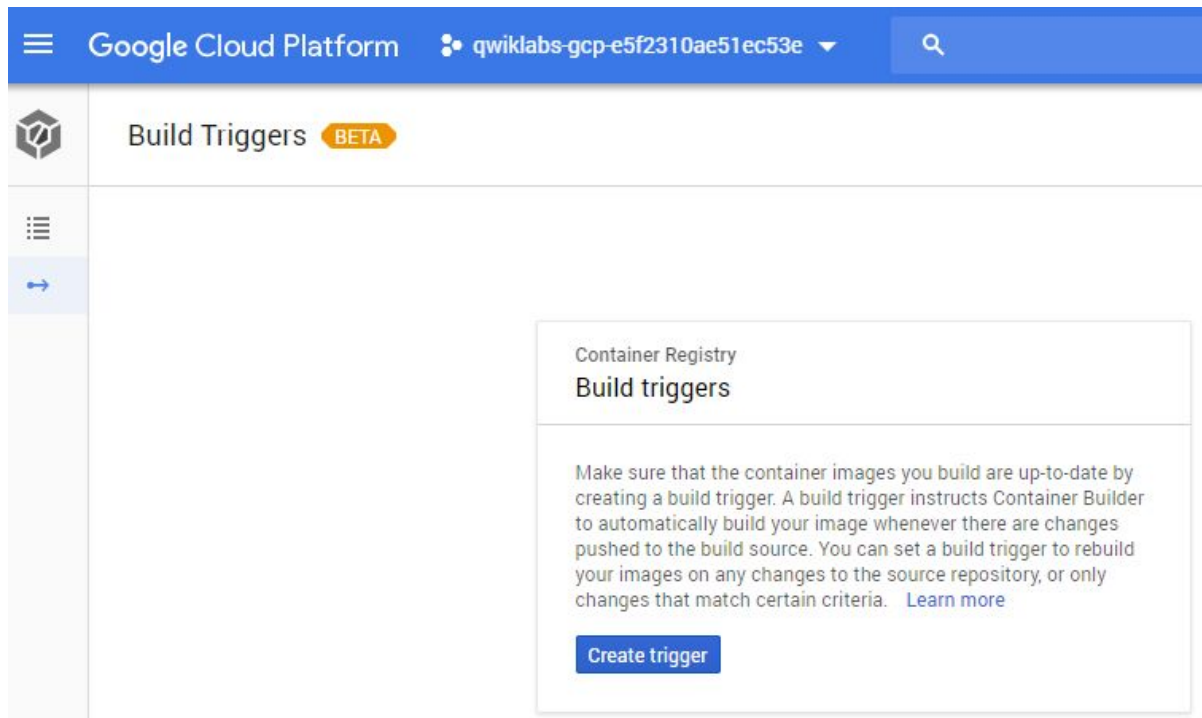
## Configure your build triggers

Configure Container Builder to build and push your Docker images every time you push [Git tags](#) to your source repository. Container Builder automatically checks out your source code, builds the Docker image from the Dockerfile in your repository, and pushes that image to Google Cloud Container Registry.



In the Cloud Platform Console, click **Navigation menu > Cloud Build > Build triggers**.

Click **Create trigger**.



Select **Cloud Source Repository** and click **Continue**.


Click the radio button for your newly created sample-app-repository then click **Continue**.

Set the following trigger settings:


- **Name:** sample-app-tags
- **Trigger type:** Tag
- **Tag (regex):** v.\*
- **Build configuration:** cloudbuild.yaml


- **cloudbuild.yaml location:** /cloudbuild.yaml

Click **Create trigger**.

 **Create trigger**

---

 **Select source**

 **Select repository**


**3** [Trigger settings](#)

---

### Trigger settings


Source: Cloud Source Repository    Repository: <https://source.developers.google.com/p/spinnaker-tutorial/r/sample-app>

**Name** (Optional)

**Trigger type** 

☐ Branch

☒ Tag


**Tag (regex)** 

No tag matches

**Build configuration**

☐ Dockerfile  
Specify the path within the Git repo

☒ **cloudbuild.yaml**  
Specify the path to a Cloud Build configuration file in the Git repo [Learn more](#)

**cloudbuild.yaml location** 

**Summary**

Changes pushed to "v.\*" tag will trigger a build defined by the "cloudbuild.yaml" file.

Create trigger

Cancel

From now on, whenever you push a Git tag prefixed with the letter "v" to your source code repository, Container Builder automatically builds and pushes your application as a Docker image to Container Registry.

# Build your image

Push your first image using the following steps:

Return to Cloud Shell and create a Git tag:

```
git tag v1.0.0
```

Make sure you are still in the `sample-app` directory and push the tag:

```
git push --tags
```

output (do not copy)

To

```
https://source.developers.google.com/p/qwiklabs-gcp-ddf2925f84de0b16/r/sample-app
```

```
* [new tag]          v1.0.0 -> v1.0.0
```

Switch back to the Console. In **Cloud Build**, click **Build history** to check that the build has been triggered. If not, verify that the trigger was configured properly in the previous section.

Google Cloud Platform					
qwiklabs-gcp-e5f2310ae51ec53e					
Build History					
Filter builds					
Build	Source	Git commit	Trigger	Started	Artifacts
99aae14b-7b84...	Cloud Source Repository sample-app	03b065e	Push to v1.0.0 tag	4 minutes ago	gcr.io/qwiklabs-gcp-e5f2310ae51ec53e/sample-app:v1.0.0

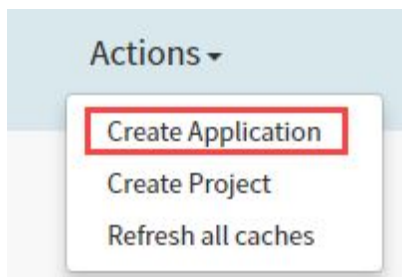
# Configuring your deployment pipelines

Now that your images are building automatically, you need to deploy them to the Kubernetes cluster.

You'll deploy to a scaled-down environment for testing. After the tests pass, you must manually approve the changes to deploy the code to production services.

## Create the application

In the Spinnaker tab click **Actions**, then select **Create Application**.



In the **New Application** dialog, enter the following fields:

- **Name:** sample

- **Owner Email:** [your lab username/email address]

Click **Create**.

## New Application ✕

**Name \***

sample

**Owner Email \***

user@example.org

**Repo Type**

Select Repo Type

**Description**

Enter a description

**Instance Health**

☐ Consider only cloud provider health when executing tasks ?

☐ Show health override option for each operation ?

**Instance Port ?**

**Pipeline Behavior**

☐ Enable restarting running pipelines ?

*\* Required*

Cancel

Create

## Create service load balancers

To avoid having to enter the information manually, use the Kubernetes command-line interface to create load balancers for your services. Alternatively, you can perform this operation on the Spinnaker page.



In Cloud Shell, run the following command from the `sample-app` root directory:

```
kubectl apply -f k8s/services
```

## Create the deployment pipeline

Next, create the continuous delivery pipeline. For this lab, the pipeline is configured to detect when a Docker image with a tag prefixed with "v" has arrived in your Container Registry.

Run the following commands to upload an example pipeline to your Spinnaker instance:

```
export PROJECT=$(gcloud info --format='value(config.project)')
sed s/PROJECT/$PROJECT/g spinnaker/pipeline-deploy.json | curl -d@- -X \
  POST --header "Content-Type: application/json" --header \
  "Accept: /" http://localhost:8080/gate/pipelines
```

In the Spinnaker tab, click **Pipelines** on the top navigation bar. You can see there is 1 pipeline ready for you called Deploy.

The screenshot shows the 'Pipelines' section of the Spinnaker web interface. At the top, there are controls for adding (+) or removing (-) pipelines, a 'Group by' dropdown set to 'Pipeline', a 'Show' dropdown set to '2' with the text 'executions per pipeline', and a checkbox for 'stage durations'. Below these controls is a table of pipelines. The first pipeline is 'LOCAL Deploy', with a status of 'Trigger: enabled' and a 'Configure' link. Below the table, a message states: 'No executions found matching the selected filters.'

Group by	Show	stage durations
Pipeline	2 executions per pipeline	<input type="checkbox"/>

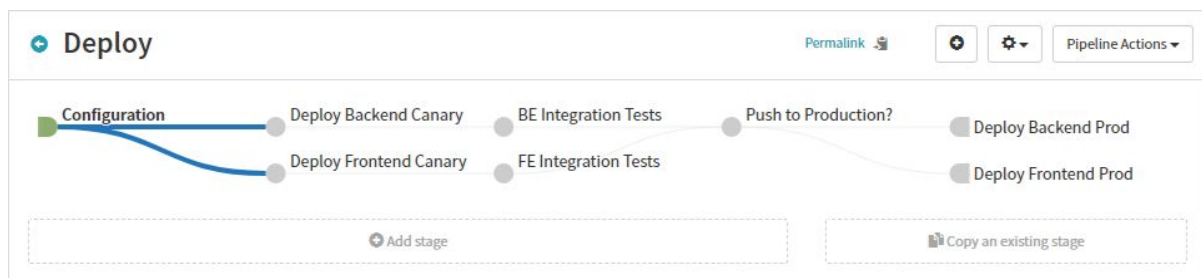
  

LOCAL	Deploy	Trigger: enabled	Configure
No executions found matching the selected filters.			

Click **Configure** and select **Deploy**.



The continuous delivery pipeline configuration appears:



The configuration you just created contains a trigger to start the pipeline when you push a new Git tag containing the prefix "v". You'll next test the pipeline by running it manually, then test it by pushing a Git tag and watching the pipeline run automatically.

## Run your pipeline manually

Return to the Pipelines page by clicking **Pipelines**.

Click **Start Manual Execution** next to Deploy.

The v1.0.0 tag from the Tag drop-down list is selected by default. Click **Run**.

## Select Execution Parameters ✕

This will start a new run of **Deploy**.

**Trigger** (Docker Registry) gcr: qwiklabs-gcp-fbb6dc4c1b962b80/sample-app

**Tag** v1.0.0

**Notifications** ☐ Notify me when the pipeline completes

Cancel ▶ Run

LOCAL

Deploy 1

Trigger: enabled

Configure

▶ Start Manual Execution

MANUAL START

vic-google/sample-app:v1.0.0

[anonymous]

7 hours ago

Details

Status: RUNNING

Duration: 07:00:09

Deploy Backend Canary

BE Integration Tests

Push to Production?

Deploy Backend Prod

Deploy Frontend Canary

FE Integration Tests

Deploy Frontend Prod

STAGE DETAILS: DEPLOY FRONTEND CANARY

Duration: 07:00:08

Step	Started	Duration	Status
Deploy in default	2017-07-24 10:50:46 PDT	07:00:08	RUNNING

You may have to select the Deploy pipeline from the Pipeline drop-down list first if you clicked the global Start Manual Execution button at the top of the page.

After the pipeline starts, expand **Details** to see more information about the build's progress.

**Pipelines**

[Create](#)
[Configure](#)
[Start Manual Execution](#)

+ - Group by Pipeline Show 2 executions per pipeline ☐ stage durations

LOCAL Deploy 1 Trigger: enabled [Configure](#) [Start Manual Execution](#)

MANUAL START

qwiklabs-gcp-e5f2310ae51ec53e/sample-app:v1.0.0

[anonymous] 2 minutes ago

Status: RUNNING

Duration: 03:30

[Details](#)

This section shows the status of the deployment pipeline and its steps:

- blue - currently running
- green - completed successfully
- red - failed

Click on a stage to see details about it.

After about three minutes the test phase completes and the pipeline requires manual approval to continue the deployment.

Click the yellow "person" icon and click **Continue**.

utions per pipeline

**Push to Production?**

**Instructions**

Do you want to deploy this image to production?

[Continue](#) [Stop](#)

Duration: 05:08

ation Tests Push to Production? Deploy Backend Prod

Your rollout continues to the production frontend and backend deployments. It completes after a few minutes.

To view the app, click **Load Balancers** in the top right of the Spinnaker page.



Scroll down the list of load balancers. Under **sample-frontend-prod** click **Default**.



Scroll down the details pane on the right and copy the application's IP address by clicking the clipboard button for **Ingress**.

**Internal DNS Name**  
sample-frontend-  
prod.default.svc.cluster.local 

**Cluster IP**  
10.43.240.49 

**Ingress**  
104.155.191.183 

Paste the address into a new browser tab to view the production version of the application.

## Backend that serviced this request

Pod Name	sample-backend-prod-v000-3c64q
Node Name	gke-spinnaker-tutorial-default-pool-907ba6b4-s15z
Version	v1.0.0
Zone	projects/46210787011/zones/us-central1-f
Project	spinnaker-tutorial
Node Internal IP	10.128.0.4
Node External IP	130.211.120.154

You have now manually triggered the pipeline to build, test, and deploy your application.

## Triggering your pipeline from code changes

Now test the pipeline end to end by making a code change, pushing a Git tag, and watching the pipeline run in response. By pushing a Git tag that starts with "v", you trigger Container Builder to build a new Docker image and push it to Container Registry. Spinnaker detects that the new image tag begins with "v" and triggers a pipeline to deploy the image to canaries, run tests, and roll out the same image to all pods in the deployment.

Switch back to Cloud Shell and run this code to change the color of the app from orange to blue.

```
sed -i 's/orange/blue/g' cmd/gke-info/common-service.go
```

Tag your change and push it to the source code repository.

```
git commit -a -m "Change color to blue"
```

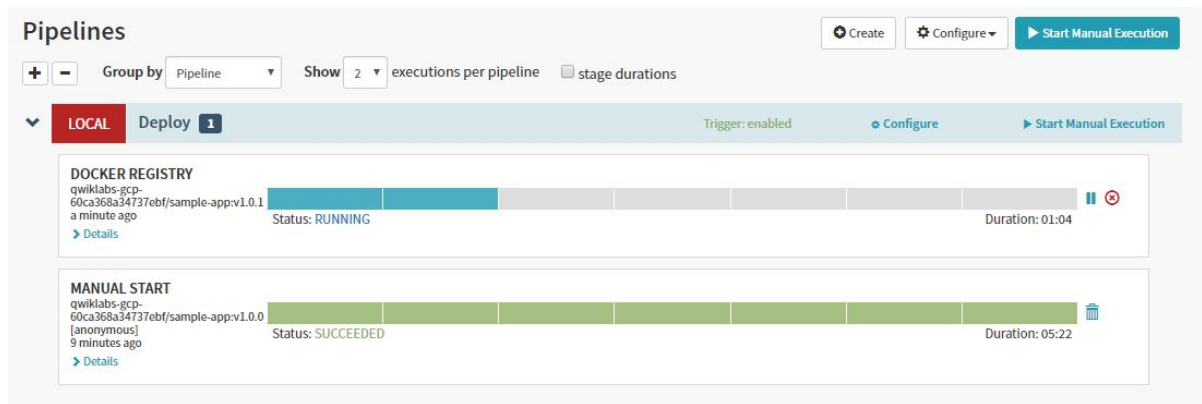
```
git tag v1.0.1
```

```
git push --tags
```

In the Console open **Cloud Build > Build history** and wait a couple of minutes for the new build to appear. You may need to refresh your page.

Switch back to the Spinnaker tab and click **Pipelines** to watch the pipeline start to deploy the image. The automatically triggered pipeline will take a few

minutes to appear. You may need to refresh your page.



The screenshot shows the Spinnaker Pipelines interface. At the top, there are buttons for 'Create', 'Configure', and 'Start Manual Execution'. Below these, there are filters for 'Group by Pipeline' and 'Show 2 executions per pipeline'. The main section displays a pipeline named 'Deploy' with a status of 'Trigger: enabled'. It contains two stages: 'DOCKER REGISTRY' (status: RUNNING, duration: 01:04) and 'MANUAL START' (status: SUCCEEDED, duration: 05:22). Each stage has a 'Details' link.

To see the canary deployments of the app, click **Load Balancers** in the top right of the Spinnaker page.



The screenshot shows a navigation bar with three buttons: 'LOAD BALANCERS', 'SECURITY GROUPS', and 'CONFIG'.

Scroll down the list of load balancers. Under **sample-frontend-canary** click **Default**.



The screenshot shows the Spinnaker Load Balancers page. It displays a load balancer named 'sample-frontend-canary' with a status of 'LOCAL'. Below it, there is a table showing the 'DEFAULT' instance with a status of '1 ▲ : 100%'.

Scroll down the details pane on the right and copy your application's IP address by clicking the clipboard button for **Ingress IP**.



**Internal DNS Name**

sample-frontend-  
canary.default.svc.cluster.local 

**Cluster IP**

10.43.248.62 

**Ingress**

35.202.128.205 

Paste the address into a new tab in your browser. You should see the new, blue version of your application, validating that it has been successfully deployed.

After testing completes, return to the Spinnaker tab and click **Pipelines**.

Click the yellow approval button (with the person icon) and click **Continue** to approve the deployment.

When the pipeline completes, refresh your production application tab. The application now also looks blue and the **Version** field now reads v1.0.1.

## Backend that serviced this request

Pod Name	sample-backend-prod-v001-krqkd
Node Name	gke-spinnaker-tutorial-default-pool-ad6dacac-flkz
Version	v1.0.1
Zone	projects/995711665042/zones/us-central1-f
Project	vic-goog
Node Internal IP	10.240.0.4
Node External IP	35.192.121.183

You have successfully rolled out your application to your entire production environment!

## Roll back a change (optional)

You can roll back the change you just made by reverting your previous commit. Rolling back adds a new tag (v1.0.2), and pushes the tag back through the same pipeline used to deploy v1.0.1.

Want to check it out? Switch back to Cloud Shell and enter the following commands to roll back the change:

```
git revert v1.0.1 --no-edit
```

```
git tag v1.0.2
```

```
git push --tags
```

To verify the roll back, click on **Load Balancer**, to **sample-frontend-canary**, click **Default** and copy the Ingress IP address into a new tab.

Now your app is back to orange and you can see the version number change.

## Congratulations

You have now successfully completed the Continuous Deployment with Spinnaker lab.



## 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.

You will be given an opportunity to rate the lab experience. Select the applicable number of stars, type a comment, and then click **Submit**.

The number of stars indicates the following:

- 1 star = Very dissatisfied
- 2 stars = Dissatisfied
- 3 stars = Neutral
- 4 stars = Satisfied
- 5 stars = Very satisfied

You can close the dialog box if you don't want to provide feedback.

For feedback, suggestions, or corrections, please use the **Support** tab.

Manual Last Updated October 16, 2018

Lab Last Tested October 16, 2018

©2019 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.



