GSP114

Google Cloud Self-Paced Labs

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.

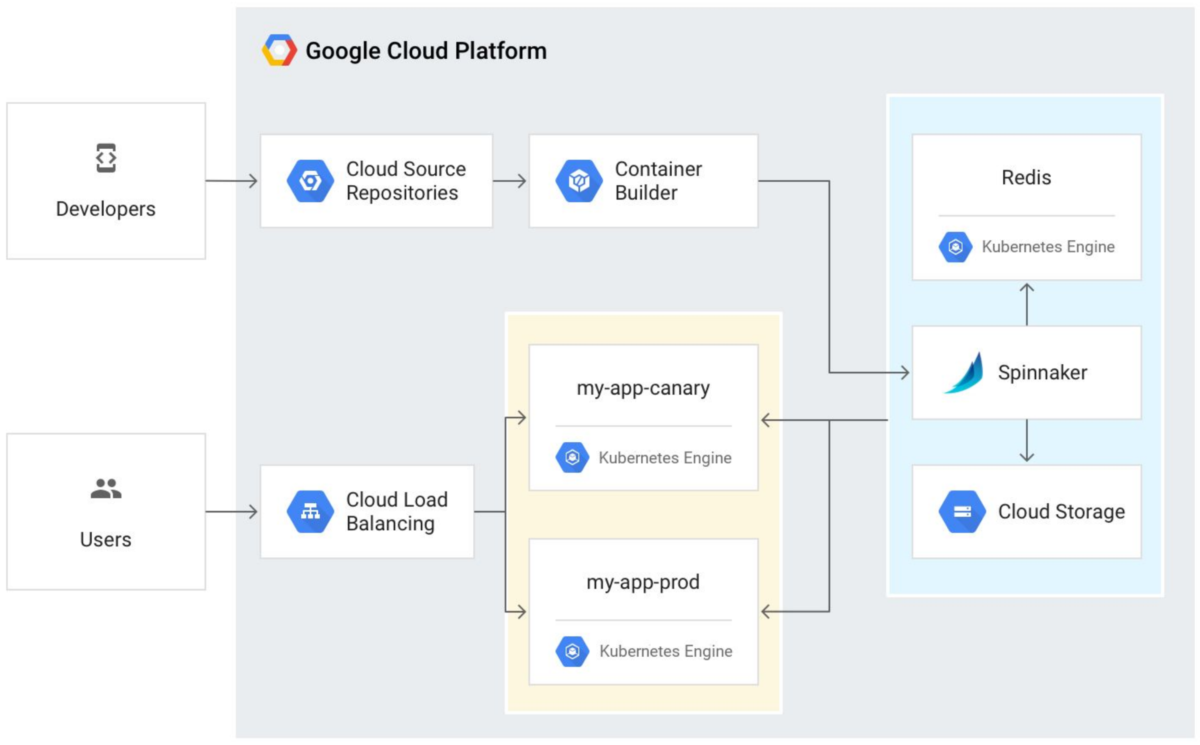
Objectives

- 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.

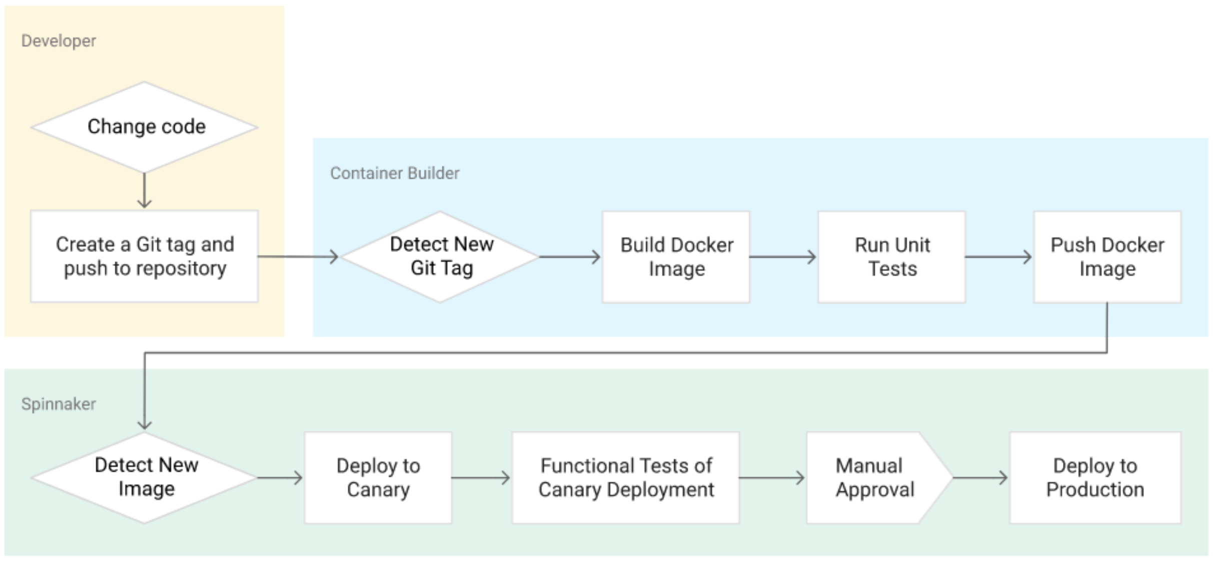


With Kubernetes Engine and Spinnaker you can create a robust continuous delivery flow that helps to ensure your software is shipped as quickly as it is developed and validated. Although rapid iteration is your end goal, you must first ensure that each application revision passes through a gamut of automated validations before becoming a candidate for production rollout. When a given change has been vetted through automation, you can also validate the application manually and conduct further pre-release testing.

After your team decides the application is ready for production, one of your team members can approve it for production deployment.

Application delivery pipeline

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



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.

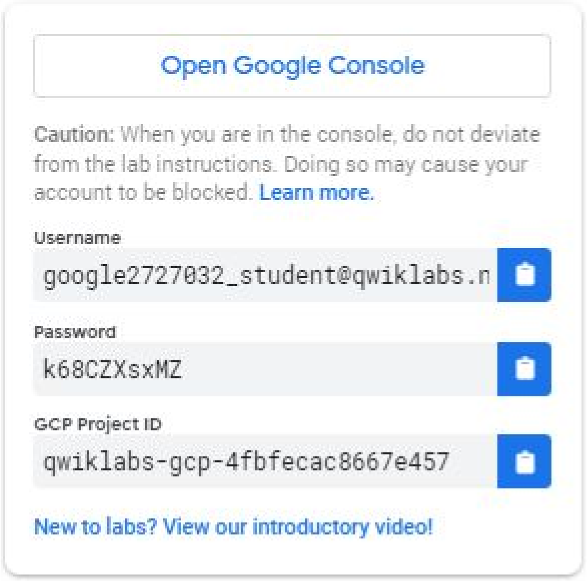
Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

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.


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

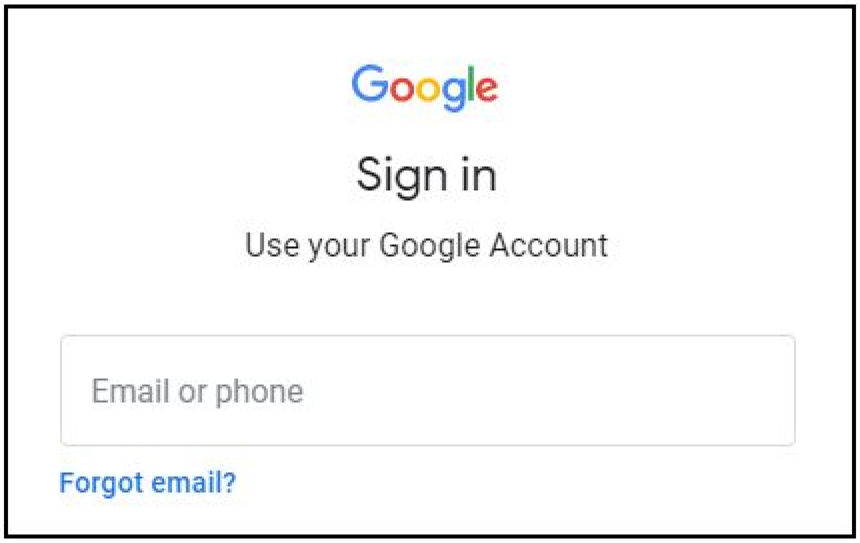
k68CZXsxMZ

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.


Google

Sign in

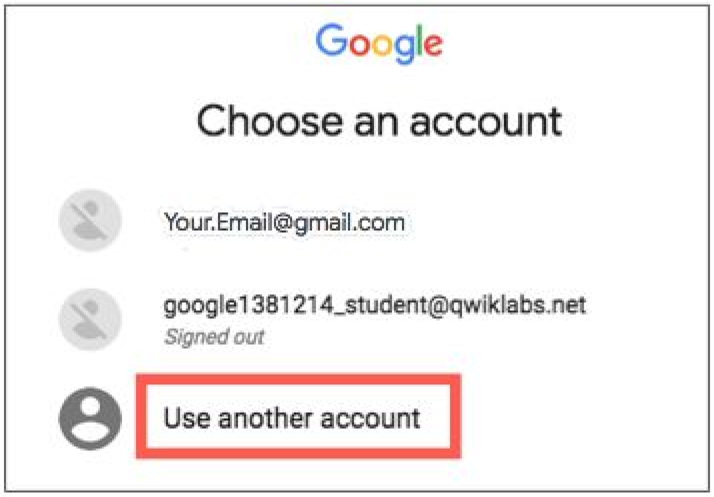
Use your Google Account

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**.

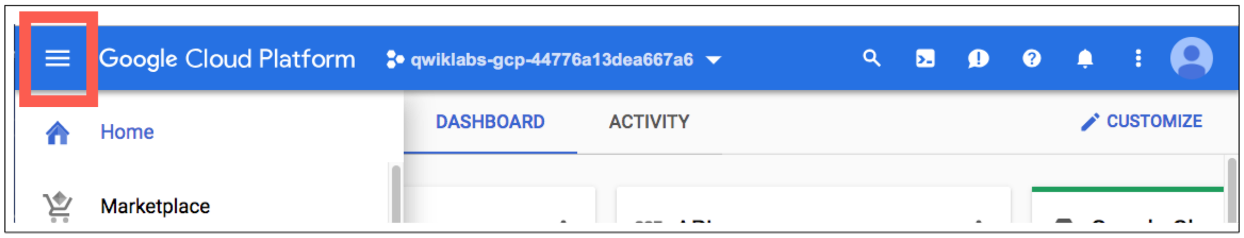


3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

   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.

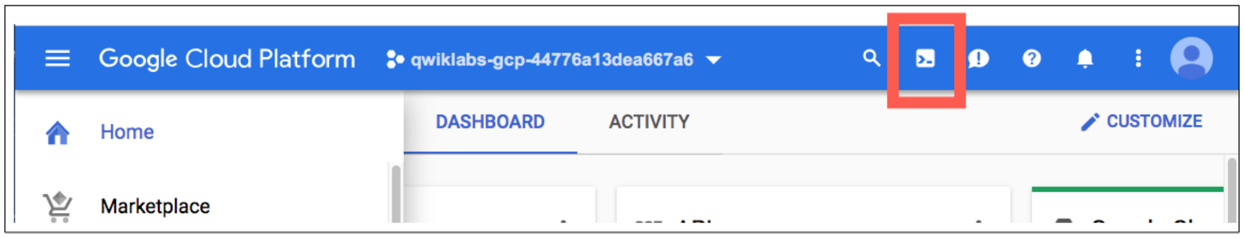
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



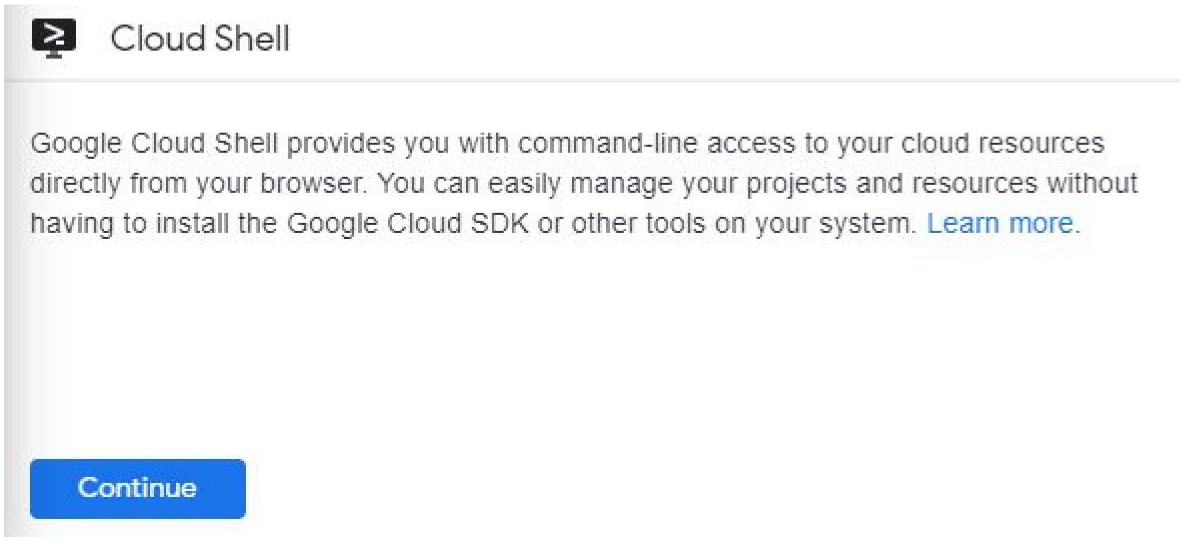
Activate Cloud Shell

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.



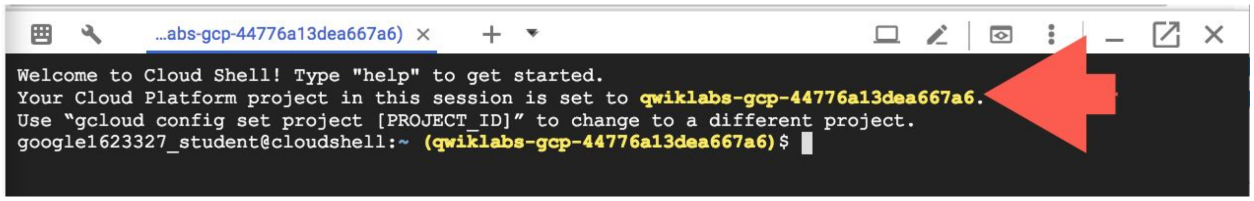
Click **Continue**.



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.

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.

1. Set the default compute zone:

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

2. Create a Kubernetes Engine using the Spinnaker tutorial sample application:

```
gcloud container clusters create spinnaker-tutorial \
    --machine-type=n1-standard-2
```

This deployment takes between **five and 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. Wait for the deployment to complete before proceeding.

When completed you see a report detailing the name, location, version, ip-address, machine-type, node version, number of nodes and status of the cluster that indicates 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:

3. Create the service account:

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

4. 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-account" \
    --format='value(email)')

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

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

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

6. 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)

```
created key [12f224e036437704b91a571792462ca6fc4cd438] of type [json] as
[spinnaker-sa.json] for
[spinnaker-account@qwiklabs-gcp-gcpd-f5e16da10e5d.iam.gserviceaccount.com]
```

Set up Cloud Pub/Sub to trigger Spinnaker pipelines

1. Create the Cloud Pub/Sub topic for notifications from Container Registry.

```
gcloud pubsub topics create projects/$PROJECT/topics/gcr
```

2. Create a subscription that Spinnaker can read from to receive notifications of images being pushed.

```
gcloud pubsub subscriptions create gcr-triggers \
    --topic projects/${PROJECT}/topics/gcr
```

3. Give Spinnaker's service account permissions to read from the gcr-triggers subscription.

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

gcloud beta pubsub subscriptions add-iam-policy-binding gcr-triggers \
    --role roles/pubsub.subscriber --member serviceAccount:$SA_EMAIL
```

Test Completed Task

Click **Check my progress** to verify your performed task. If you have successfully setup the environment, you will see an assessment score.

Set up your environment

Check my progress

Deploying Spinnaker using Helm

In this section you 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

1. Download and install the `helm` binary:

```
wget https://get.helm.sh/helm-v3.1.0-linux-amd64.tar.gz
```

2. Unzip the file to your local system:

```
tar zxfv helm-v3.1.0-linux-amd64.tar.gz
```

```
cp linux-amd64/helm .
```

3. Grant Helm the cluster-admin role in your cluster:

```
kubectl create clusterrolebinding user-admin-binding \
    --clusterrole=cluster-admin --user=$(gcloud config get-value account)
```

4. Grant Spinnaker the `cluster-admin` role so it can deploy resources across all namespaces:

```
kubectl create clusterrolebinding --clusterrole=cluster-admin \
    --serviceaccount=default:default spinnaker-admin
```

5. Add the stable charts deployments to Helm's usable repositories (includes Spinnaker):

```
./helm repo add stable https://kubernetes-charts.storage.googleapis.com
./helm repo update
```

Configure Spinnaker

6. Still in Cloud Shell, create a bucket for Spinnaker to store its pipeline configuration:

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

export BUCKET=$PROJECT-spinnaker-config

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

7. Run the following command to create a `spinnaker-config.yaml` file, which describes how Helm should install Spinnaker:

```
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
gcs:
  enabled: true
  bucket: $BUCKET
  project: $PROJECT
  jsonKey: '$SA_JSON'

dockerRegistries:
- name: gcr
  address: https://gcr.io
  username: _json_key
  password: '$SA_JSON'
  email: 1234@5678.com

# Disable minio as the default storage backend
minio:
  enabled: false

# Configure Spinnaker to enable GCP services
halyard:
  additionalScripts:
    create: true
    data:
      enable_gcs_artifacts.sh: |-
        \$HAL_COMMAND config artifact gcs account add gcs-$PROJECT --json-path
/opt/gcs/key.json
        \$HAL_COMMAND config artifact gcs enable
      enable_pubsub_triggers.sh: |-
        \$HAL_COMMAND config pubsub google enable
        \$HAL_COMMAND config pubsub google subscription add gcr-triggers \
          --subscription-name gcr-triggers \
          --json-path /opt/gcs/key.json \
          --project $PROJECT \
          --message-format GCR
EOF
```

Deploy the Spinnaker chart

8. Use the Helm command-line interface to deploy the chart with your configuration set:

```
./helm install -n default cd stable/spinnaker -f spinnaker-config.yaml \
           --version 2.0.0-rc9 --timeout 10m0s --wait
```

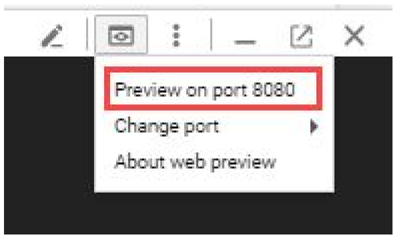
The installation typically takes **5-8 minutes** to complete.

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

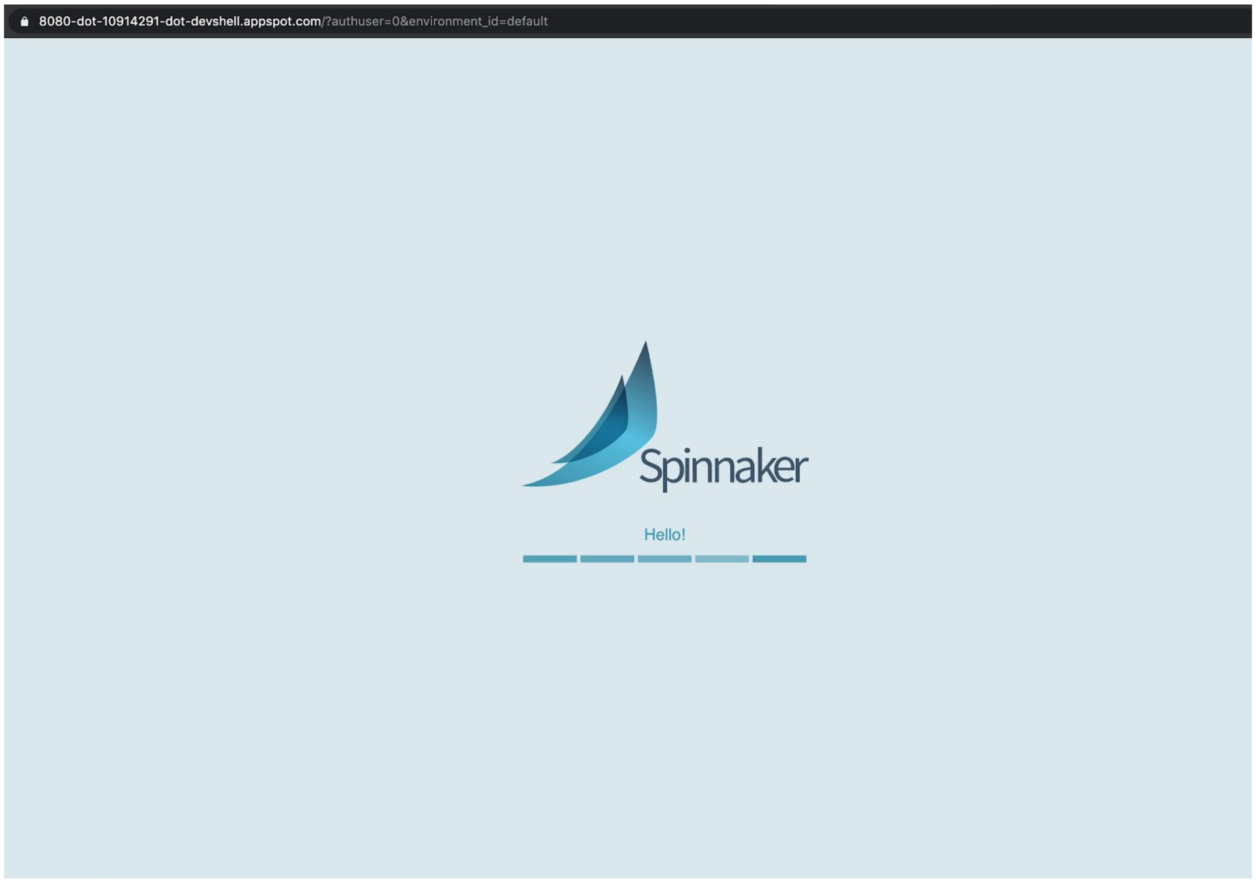
```
export DECK_POD=$(kubectl get pods --namespace default -l "cluster=spin-deck"
\
    -o jsonpath="{.items[0].metadata.name}")

kubectl port-forward --namespace default $DECK_POD 8080:9000 >> /dev/null &
```

10. 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**.



The welcome screen opens, followed by the Spinnaker user interface:


8080-dot-10914291-dot-devshell.appspot.com/?authuser=0&environment_id=default
Spinnaker
Hello!

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

Test Completed Task

Click **Check my progress** to verify your performed task. If you have successfully deployed the Spinnaker chart using Kubernetes Helm, you will see an assessment score.

Deploy the Spinnaker chart using Kubernetes Helm

Check my progress

Building the Docker image

In this section, you configure Cloud Build to detect changes to your app source code, build a Docker image, and then push it to Container Registry.

Create your source code repository

1. In Cloud Shell tab and download the sample application source code:

```
gsutil -m cp -r gs://spls/gsp114/sample-app.tar .
```

2. Unpack the source code:

```
mkdir sample-app
tar xvf sample-app.tar -C ./sample-app
```

3. Change directories to the source code:

```
cd sample-app
```

4. Set the username and email address for your Git commits in this repository. Replace `[USERNAME]` with a username you create:

```
git config --global user.email "$(gcloud config get-value core/account)"
```

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

5. Make the initial commit to your source code repository:

```
git init
git add .
```

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

6. Create a repository to host your code:

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

Disregard the "you may be billed for this repository" message.

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

7. Add your newly created repository as remote:

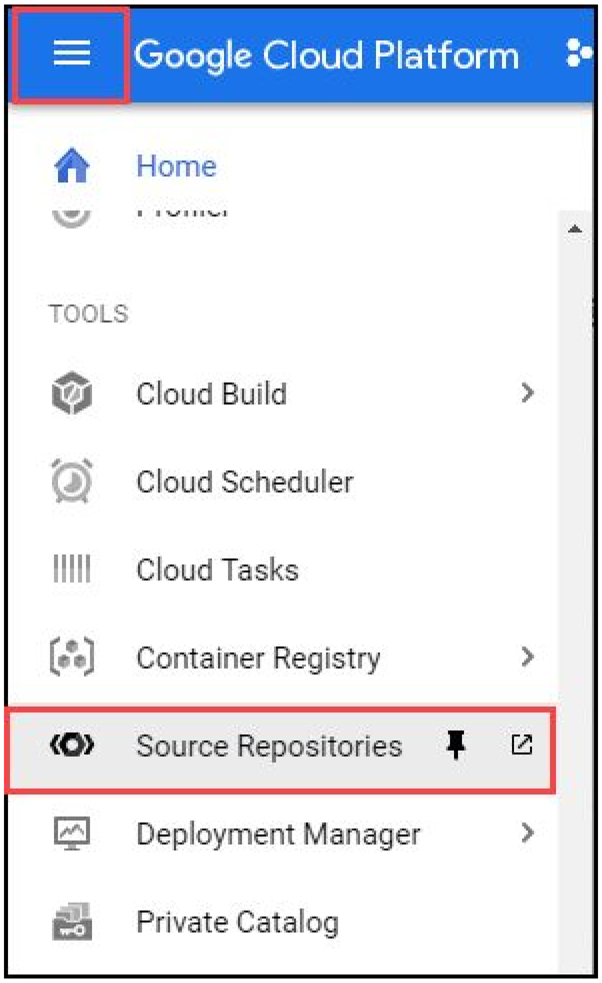
```
export PROJECT=$(gcloud info --format='value(config.project)')
```

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

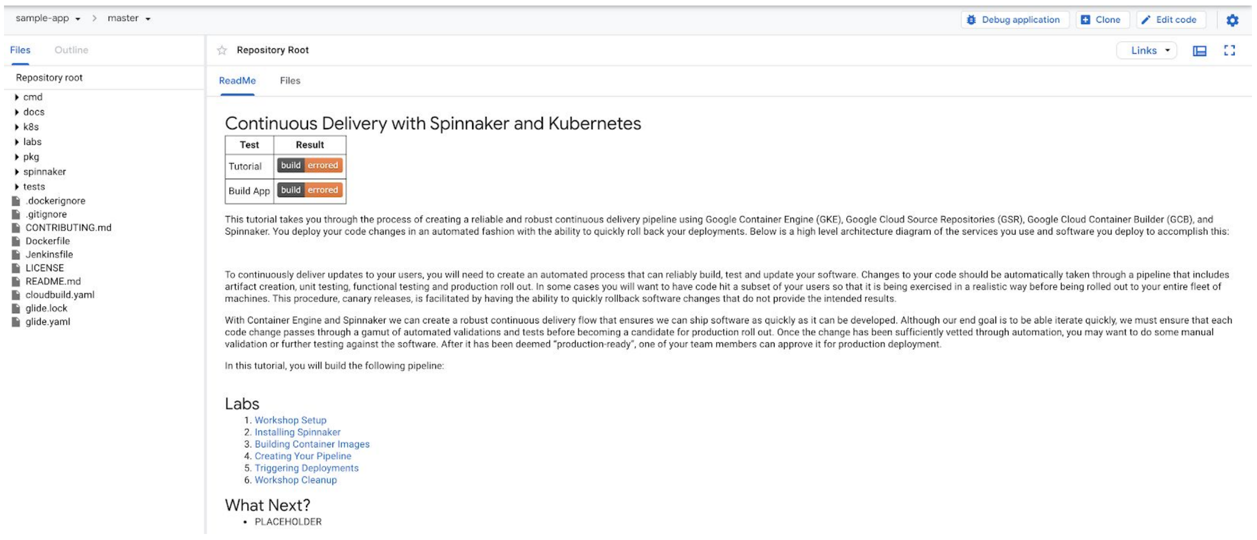
8. Push your code to the new repository's master branch:

```
git push origin master
```

9. Check that you can see your source code in the Console by clicking **Navigation Menu** > **Source Repositories**.



10. Click **View All Repositories** and select **sample-app**.



cmd
docs
k8s
labs
pkg
spinnaker
tests
.dockerignore
.gitignore
CONTRIBUTING.md
Dockerfile
Jenkinsfile
LICENSE
README.md
cloudbuild.yaml
glide.lock
glide.yaml

Continuous Delivery with Spinnaker and Kubernetes

Test	Result
Tutorial	build errored
Build App	build errored

This tutorial takes you through the process of creating a reliable and robust continuous delivery pipeline using Google Container Engine (GKE), Google Cloud Source Repositories (GSR), Google Cloud Container Builder (GCB), and Spinnaker. You deploy your code changes in an automated fashion with the ability to quickly roll back your deployments. Below is a high level architecture diagram of the services you use and software you deploy to accomplish this:

To continuously deliver updates to your users, you will need to create an automated process that can reliably build, test and update your software. Changes to your code should be automatically taken through a pipeline that includes artifact creation, unit testing, functional testing and production roll out. In some cases you will want to have code hit a subset of your users so that it is being exercised in a realistic way before being rolled out to your entire fleet of machines. This procedure, canary releases, is facilitated by having the ability to quickly rollback software changes that do not provide the intended results.

With Container Engine and Spinnaker we can create a robust continuous delivery flow that ensures we can ship software as quickly as it can be developed. Although our end goal is to be able iterate quickly, we must ensure that each code change passes through a gamut of automated validations and tests before becoming a candidate for production roll out. Once the change has been sufficiently vetted through automation, you may want to do some manual validation or further testing against the software. After it has been deemed "production-ready", one of your team members can approve it for production deployment.

In this tutorial, you will build the following pipeline:

Labs

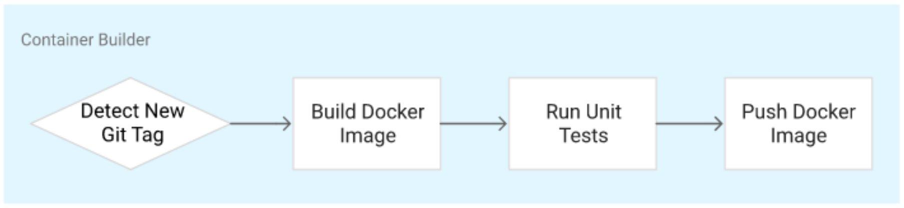
1. Workshop Setup
2. Installing Spinnaker
3. Building Container Images
4. Creating Your Pipeline
5. Triggering Deployments
6. Workshop Cleanup

What Next?

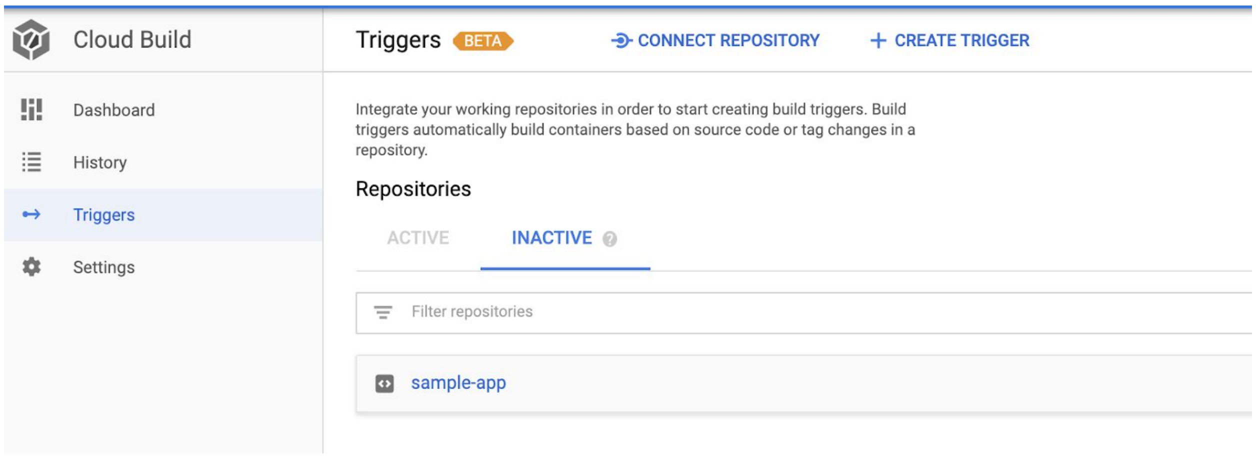
- PLACEHOLDER

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.



11. In the Cloud Platform Console, click **Navigation menu** > **Cloud Build** > **Triggers**.
12. Click **Create trigger**.



Integrate your working repositories in order to start creating build triggers. Build triggers automatically build containers based on source code or tag changes in a repository.

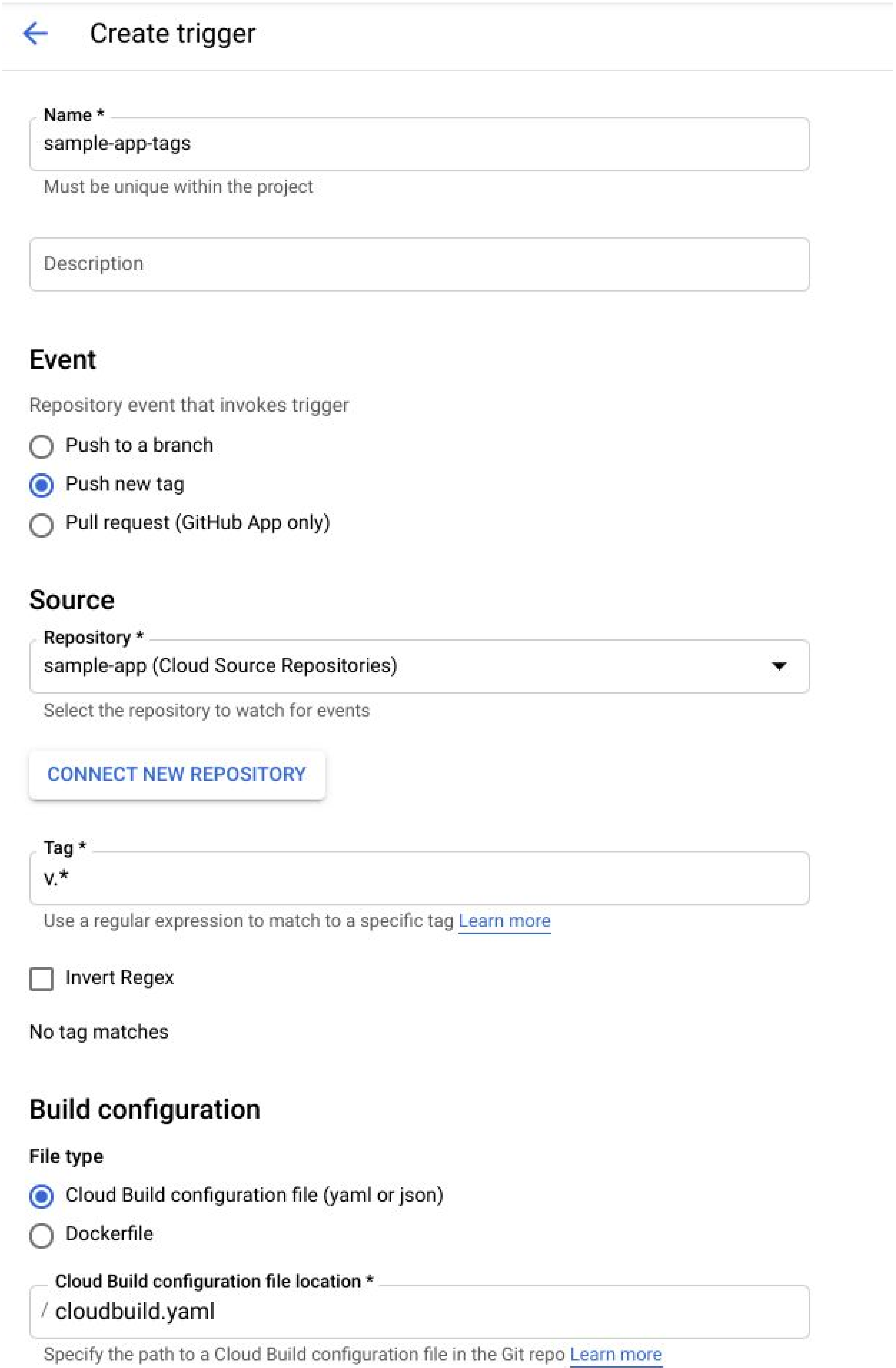
Repositories

13. Set the following trigger settings:

- **Name**:`sample-app-tags`
- **Event**: Push new tag

- Select your newly created `sample-app` repository.
- **Tag**: `v.*`
- **Build configuration**: `Cloud Build configuration file (yaml or json)`
- **Cloud Build configuration file location**: `/cloudbuild.yaml`

14. Click **CREATE**.



Create trigger

Name *
sample-app-tags
Must be unique within the project

Description

Event

Repository event that invokes trigger

- Push to a branch
- [✓ Push new tag]
- Pull request (GitHub App only)

Source

Repository *
sample-app (Cloud Source Repositories)
Select the repository to watch for events

CONNECT NEW REPOSITORY

Tag *
v.*
Use a regular expression to match to a specific tag Learn more

☐ Invert Regex

No tag matches

Build configuration

File type

- [✓ Cloud Build configuration file (yaml or json)]
- Dockerfile

Cloud Build configuration file location *
/ cloudbuild.yaml
Specify the path to a Cloud Build configuration file in the Git repo Learn more

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.

Prepare your Kubernetes Manifests for use in Spinnaker

Spinnaker needs access to your Kubernetes manifests in order to deploy them to your clusters. This section creates a Cloud Storage bucket that will be populated with your manifests during the CI process in Cloud Build. After your manifests are in Cloud Storage, Spinnaker can download and apply them during your pipeline's execution.

15. Create the bucket:

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

```
gsutil mb -l us-central1 gs://$PROJECT-kubernetes-manifests
```

16. Enable versioning on the bucket so that you have a history of your manifests:

```
gsutil versioning set on gs://$PROJECT-kubernetes-manifests
```

17. Set the correct project ID in your kubernetes deployment manifests:

```
sed -i s/PROJECT/$PROJECT/g k8s/deployments/*
```

18. Commit the changes to the repository:

```
git commit -a -m "Set project ID"
```

Build your image

Push your first image using the following steps:

19. In Cloud Shell, still in the `sample-app` directory, create a Git tag:

```
git tag v1.0.0
```

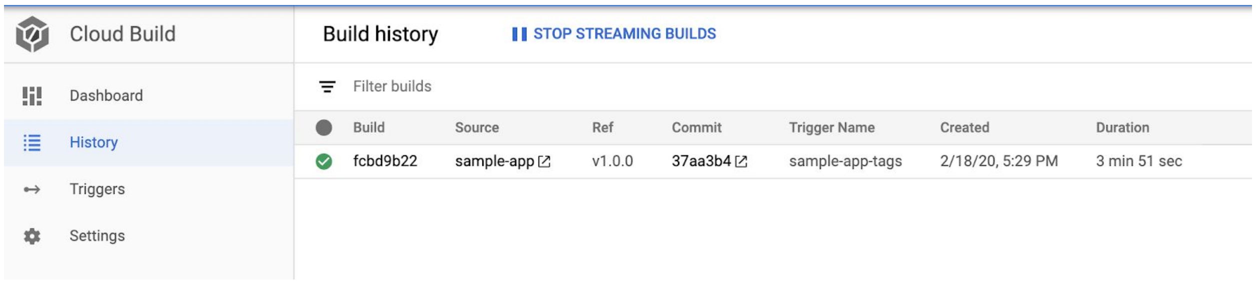
20. Push the tag:

```
git push --tags
```

(output)

```
To
https://source.developers.google.com/p/qwiklabs-gcp-ddf2925f84de0b16/r/sample-
app
* [new tag]      v1.0.0 -> v1.0.0
```

21. Go to the Cloud Console. Still in Cloud Build, click **History** in the left pane to check that the build has been triggered. If not, verify that the trigger was configured properly in the previous section.



	Build	Source	Ref	Commit	Trigger Name	Created	Duration
✓	fcbd9b22	sample-app	v1.0.0	37aa3b4	sample-app-tags	2/18/20, 5:29 PM	3 min 51 sec

Stay on this page and wait for the build to complete before going on to the next section.

Test Completed Task

Click **Check my progress** to verify your performed task. If you have successfully build the Docker image, you will see an assessment score.

Build the Docker image

Check my progress

Configuring your deployment pipelines

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

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

Install the spin CLI for managing Spinnaker

spin is a command-line utility for managing Spinnaker's applications and pipelines.

1. Download the 1.14.0 version of `spin`:

```
curl -LO
https://storage.googleapis.com/spinnaker-artifacts/spin/1.14.0/linux/amd64/spi
n
```

2. Make `spin` executable:

```
chmod +x spin
```

Create the deployment pipeline

3. Use spin to create an app called sample in Spinnaker. Set the owner email address for the app in Spinnaker:

```
./spin application save --application-name sample \
                        --owner-email "$(gcloud config get-value
core/account)" \
                        --cloud-providers kubernetes \
                        --gate-endpoint http://localhost:8080/gate
```

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

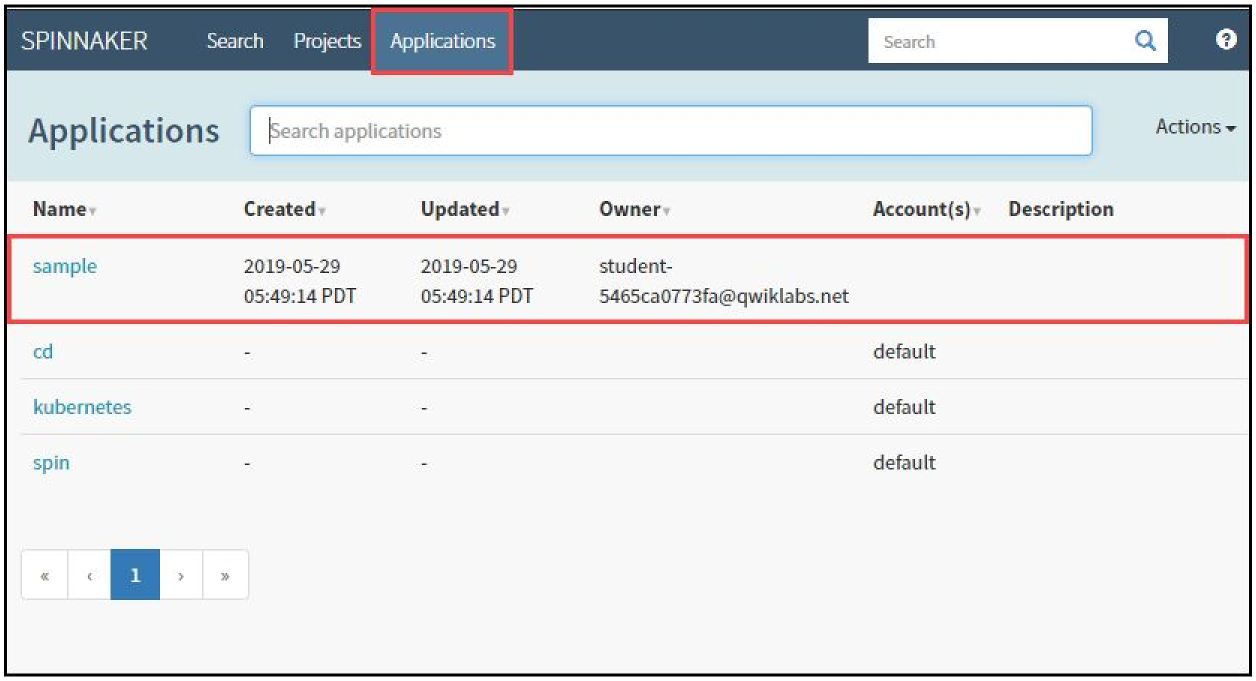
4. From your `sample-app` source code directory, run the following command to upload an example pipeline to your Spinnaker instance:

```
export PROJECT=$(gcloud info --format='value(config.project)')
sed s/PROJECT/$PROJECT/g spinnaker/pipeline-deploy.json > pipeline.json
./spin pipeline save --gate-endpoint http://localhost:8080/gate -f
pipeline.json
```

Manually Trigger and View your pipeline execution

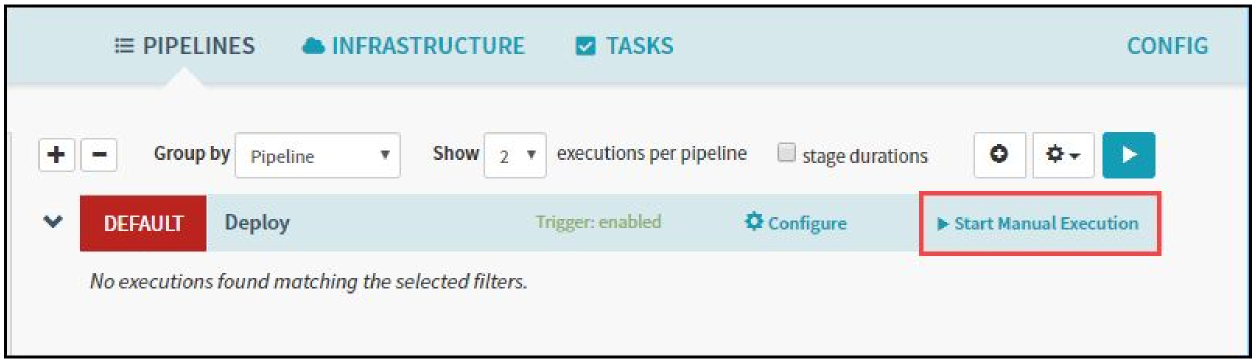
The configuration you just created uses notifications of newly tagged images being pushed to trigger a Spinnaker pipeline. In a previous step, you pushed a tag to the Cloud Source Repositories which triggered Cloud Build to build and push your image to Container Registry. To verify the pipeline, `manually trigger` it.

5. In the Spinnaker UI and click **Applications** at the top of the screen to see your list of managed applications. **sample** is your application. If you don't see **sample**, try refreshing the Spinnaker Applications tab.



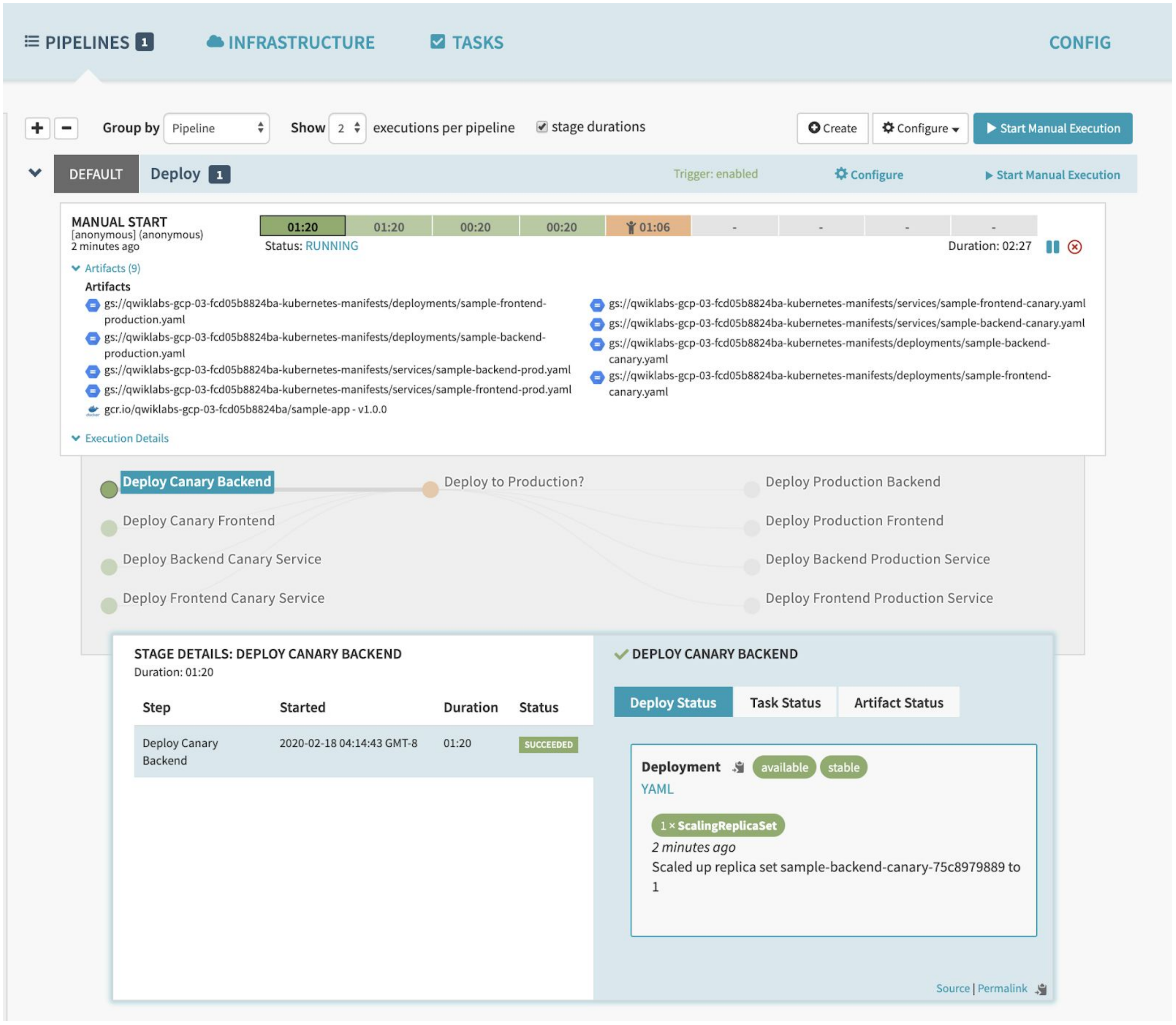
Name	Created	Updated	Owner	Account(s)	Description
sample	2019-05-29 05:49:14 PDT	2019-05-29 05:49:14 PDT	student-5465ca0773fa@qwiklabs.net		
cd	-	-		default	
kubernetes	-	-		default	
spin	-	-		default	

6. Click **sample** to view your application deployment.
7. Click **Pipelines** at the top to view your applications pipeline status.
8. Click **Start Manual Execution** to trigger the pipeline this first time.



No executions found matching the selected filters.

9. Click **Run**.
10. Click **Execution Details** to see more information about the pipeline's progress.



MANUAL START
[anonymous] (anonymous)
2 minutes ago

01:20	01:20	00:20	00:20	01:06	-	-	-	-

Status: RUNNING
Duration: 02:27

Artifacts (9)

Artifacts

- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-frontend-production.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-backend-production.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-backend-prod.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-frontend-prod.yaml
- gcr.io/qwiklabs-gcp-03-fcd05b8824ba/sample-app - v1.0.0
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-frontend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-backend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-backend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-frontend-canary.yaml

Execution Details

Deploy Canary Backend
Deploy Canary Frontend
Deploy Backend Canary Service
Deploy Frontend Canary Service
Deploy to Production?
Deploy Production Backend
Deploy Production Frontend
Deploy Backend Production Service
Deploy Frontend Production Service

STAGE DETAILS: DEPLOY CANARY BACKEND
Duration: 01:20

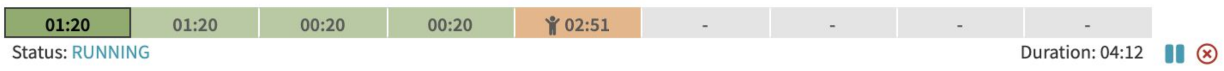
Step	Started	Duration	Status
Deploy Canary Backend	2020-02-18 04:14:43 GMT-8	01:20	SUCCEEDED

DEPLOY CANARY BACKEND

Deployment available stable
YAML

1 × ScalingReplicaSet
2 minutes ago
Scaled up replica set sample-backend-canary-75c8979889 to 1

The progress bar shows the status of the deployment pipeline and its steps.



01:20	01:20	00:20	00:20	02:51	-	-	-	-

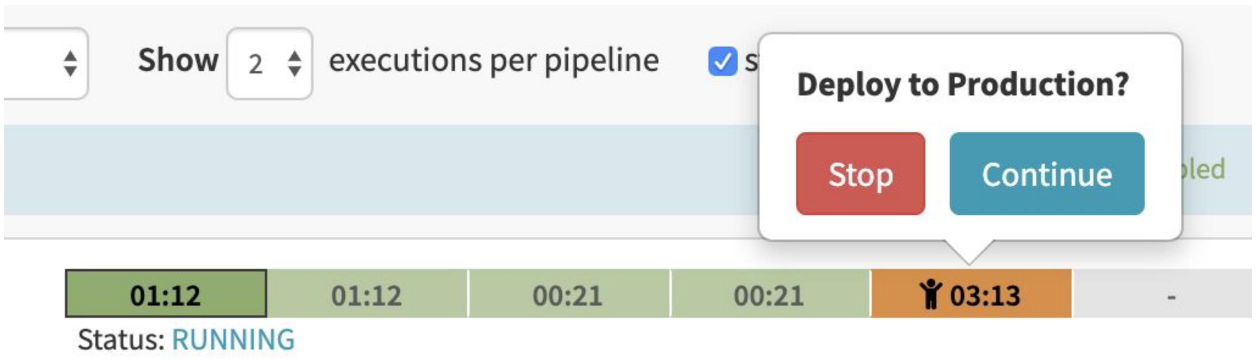
Status: RUNNING
Duration: 04:12

Steps in blue are currently running, green ones have completed successfully, and red ones have failed.

11. Click a stage to see details about it.

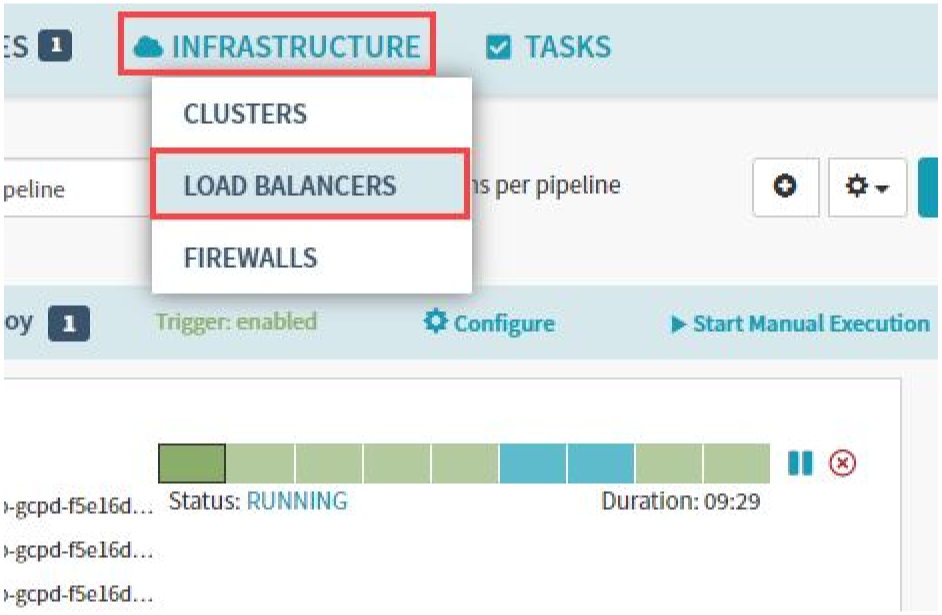
After **3 to 5 minutes** the integration test phase completes and the pipeline requires manual approval to continue the deployment.

12. Hover over the yellow "person" icon and click **Continue**.

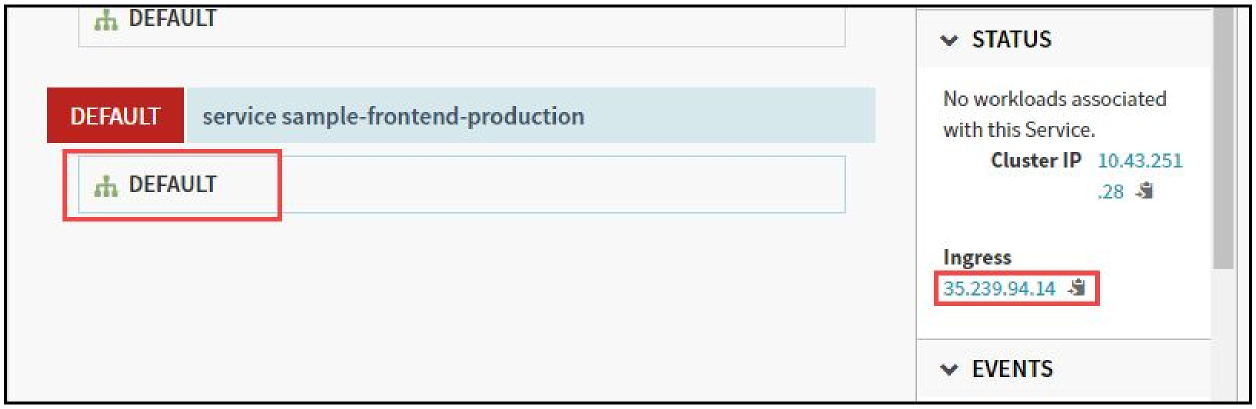


Your rollout continues to the production frontend and backend deployments. It completes after a few minutes.

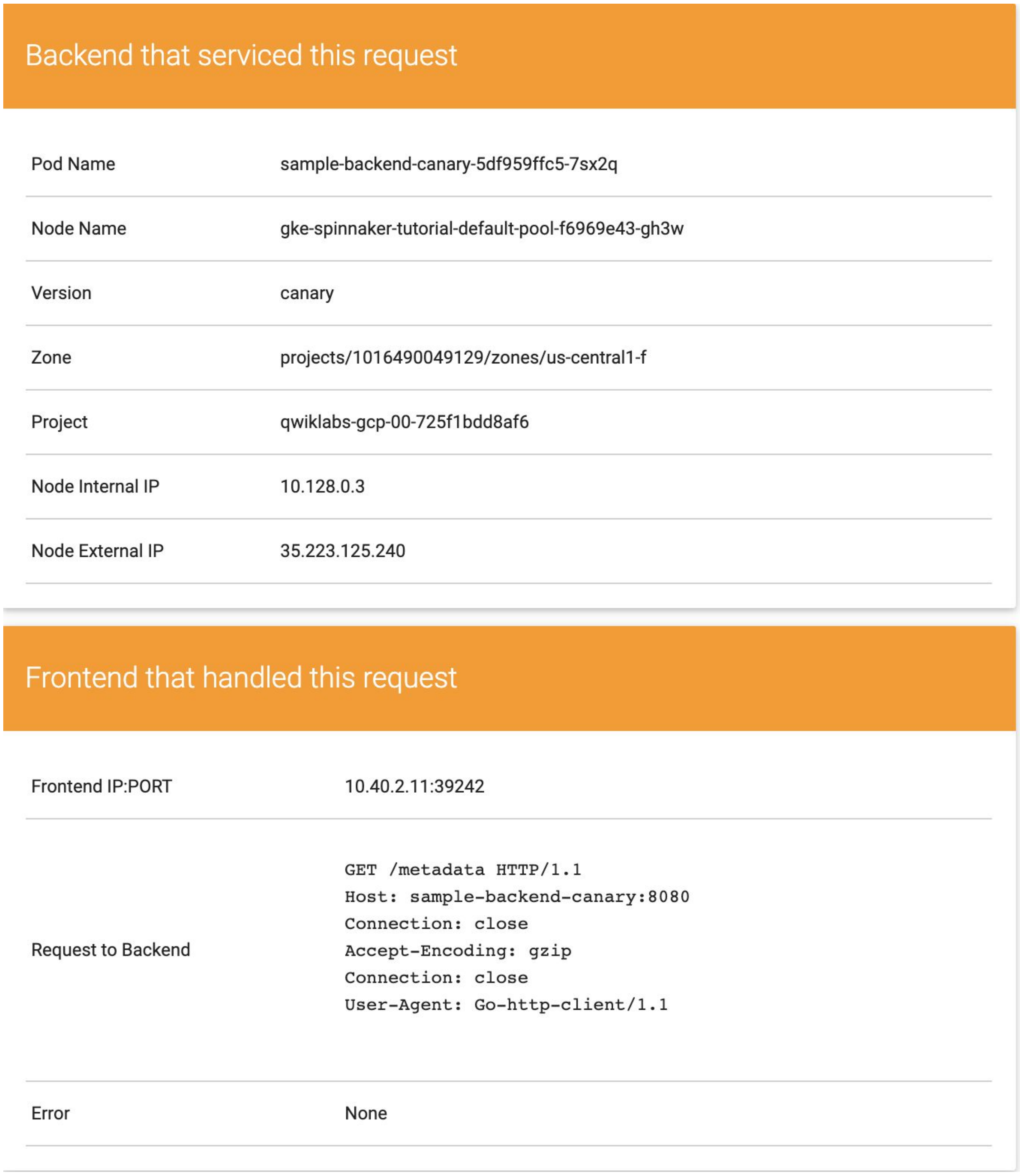
13. To view the app, select **Infrastructure** > **Load Balancers** in the top of the Spinnaker UI.



14. Scroll down the list of load balancers and click Default, under `service sample-frontend-production`.
15. Scroll down the details pane on the right and copy your app's IP address by clicking the clipboard button on the **Ingress** IP. The ingress IP link from the Spinnaker UI may use HTTPS by default, while the application is configured to use HTTP.


DEFAULT
DEFAULT
service sample-frontend-production
DEFAULT
STATUS
No workloads associated with this Service.
Cluster IP 10.43.251.28
Ingress
35.239.94.14
EVENTS

16. Paste the address into a new browser tab to view the application. You might see the canary version displayed, but if you refresh you will also see the production version.



Backend that serviced this request

Pod Name	sample-backend-canary-5df959ffc5-7sx2q
Node Name	gke-spinnaker-tutorial-default-pool-f6969e43-gh3w
Version	canary
Zone	projects/1016490049129/zones/us-central1-f
Project	qwiklabs-gcp-00-725f1bdd8af6
Node Internal IP	10.128.0.3
Node External IP	35.223.125.240

Frontend that handled this request

Frontend IP:PORT 10.40.2.11:39242

Request to Backend

```
GET /metadata HTTP/1.1
Host: sample-backend-canary:8080
Connection: close
Accept-Encoding: gzip
Connection: close
User-Agent: Go-http-client/1.1
```

Error None

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

Test Completed Task

Click **Check my progress** to verify your performed task. If you have successfully created service load balancers, you will see an assessment score.

Create service load balancers

Check my progress

Test Completed Task

Click **Check my progress** to verify your performed task. If you have successfully deployed an image to production, you will see an assessment score.

Deploy an image to production

Check my progress

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.

1. From your sample-app directory, change the color of the app from orange to blue:

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

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

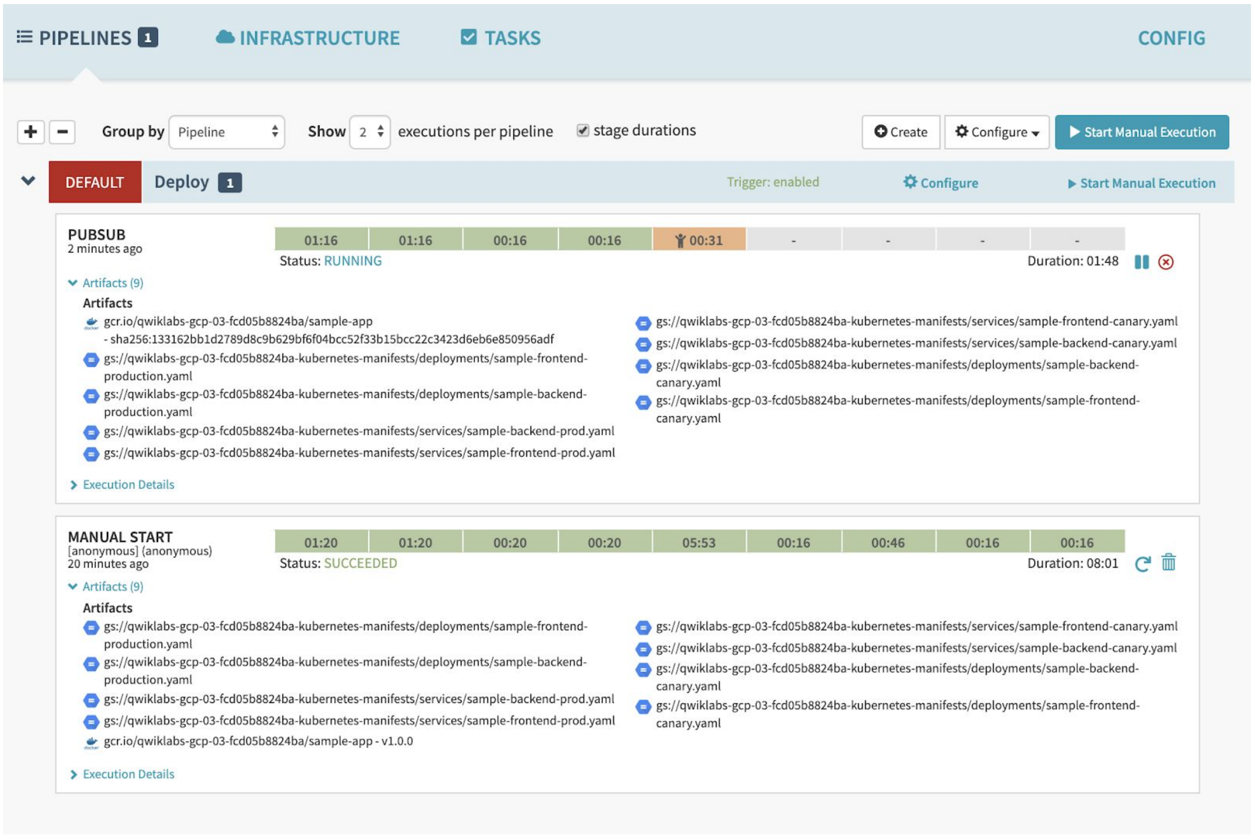
```
git commit -a -m "Change color to blue"

git tag v1.0.1

git push --tags
```

3. In the Console, in **Cloud Build** > **History**, wait a couple of minutes for the new build to appear. You may need to refresh your page. Wait for the new build to complete, before going to the next step.

4. Return to the Spinnaker UI 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.



PUBSUB
2 minutes ago

01:16 | 01:16 | 00:16 | 00:16 | 00:31 | - | - | - | -

Status: RUNNING
Duration: 01:48

Artifacts (9)

Artifacts
- gcr.io/qwiklabs-gcp-03-fcd05b8824ba/sample-app - sha256:133162bb1d2789d8c9b629bf6f04bcc52f33b15bcc22c3423d6eb6e850956adf
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-frontend-production.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-backend-production.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-backend-prod.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-frontend-prod.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-frontend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-backend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-backend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-frontend-canary.yaml

Execution Details

MANUAL START
[anonymous] (anonymous)
20 minutes ago

01:20 | 01:20 | 00:20 | 00:20 | 05:53 | 00:16 | 00:46 | 00:16 | 00:16

Status: SUCCEEDED
Duration: 08:01

Artifacts (9)

Artifacts
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-frontend-production.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-backend-production.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-backend-prod.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-frontend-prod.yaml
- gcr.io/qwiklabs-gcp-03-fcd05b8824ba/sample-app - v1.0.0
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-frontend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/services/sample-backend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-backend-canary.yaml
- gs://qwiklabs-gcp-03-fcd05b8824ba-kubernetes-manifests/deployments/sample-frontend-canary.yaml

Execution Details

Test Completed Task

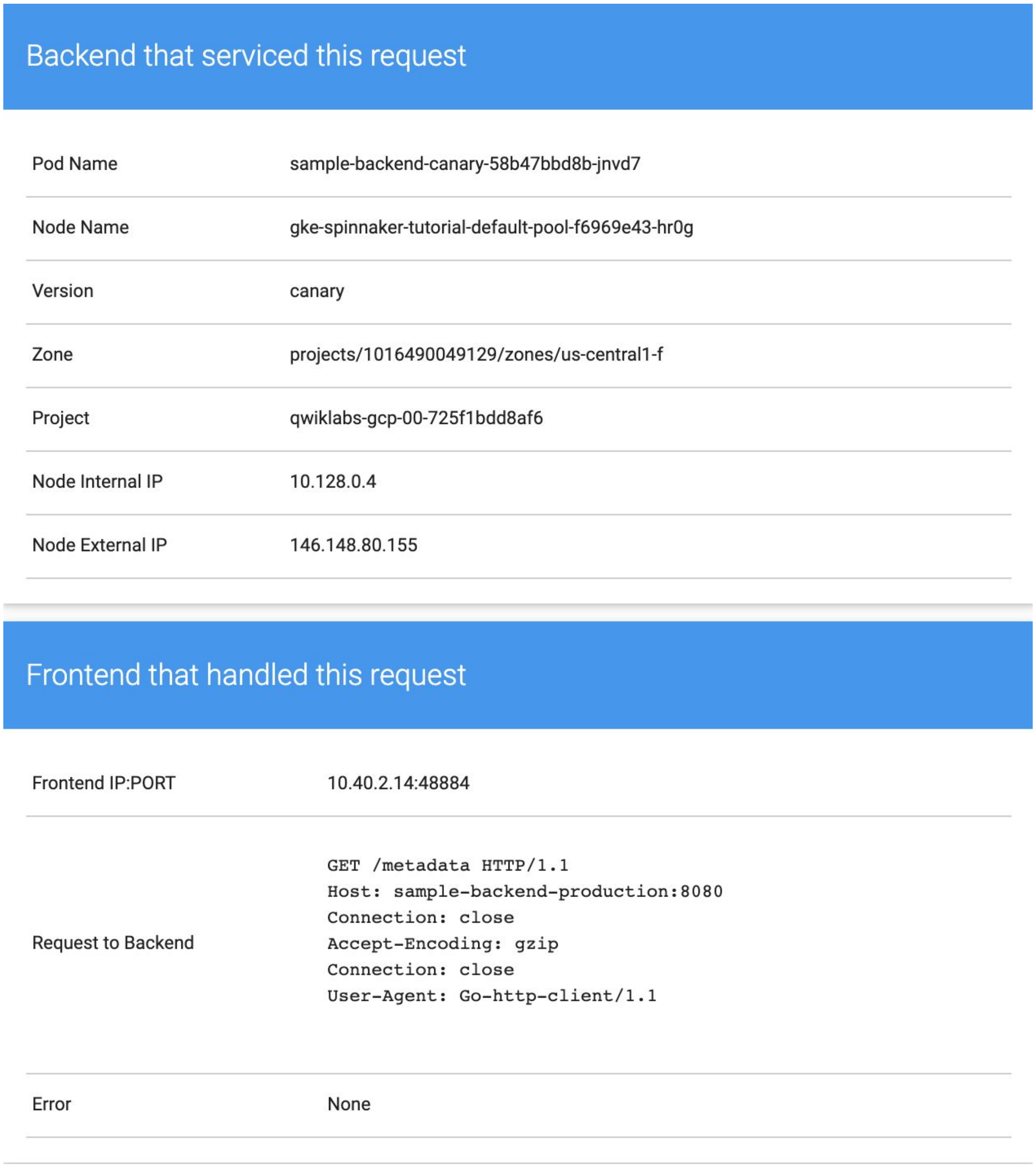
Click **Check my progress** to verify your performed task. If you have successfully triggered pipeline from code changes, you will see an assessment score.

Triggering pipeline from code changes

Check my progress

Observe the canary deployments

1. When the deployment is paused, waiting to roll out to production, return to the web page displaying your running application and start refreshing the tab that contains your app. Four of your backends are running the previous version of your app, while only one backend is running the canary. You should see the new, blue version of your app appear about every fifth time you refresh.
2. When the pipeline completes, your app looks like the following screenshot. Note that the color has changed to blue because of your code change, and that the **Version** field now reads `canary`.


Backend that serviced this request

Pod Name	sample-backend-canary-58b47bbd8b-jnvd7
Node Name	gke-spinnaker-tutorial-default-pool-f6969e43-hr0g
Version	canary
Zone	projects/1016490049129/zones/us-central1-f
Project	qwiklabs-gcp-00-725f1bdd8af6
Node Internal IP	10.128.0.4
Node External IP	146.148.80.155

Frontend that handled this request

Frontend IP:PORT	10.40.2.14:48884
Request to Backend	see below
Error	None

```
GET /metadata HTTP/1.1
Host: sample-backend-production:8080
Connection: close
Accept-Encoding: gzip
Connection: close
User-Agent: Go-http-client/1.1
```

You have now successfully rolled out your app to your entire production environment!

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

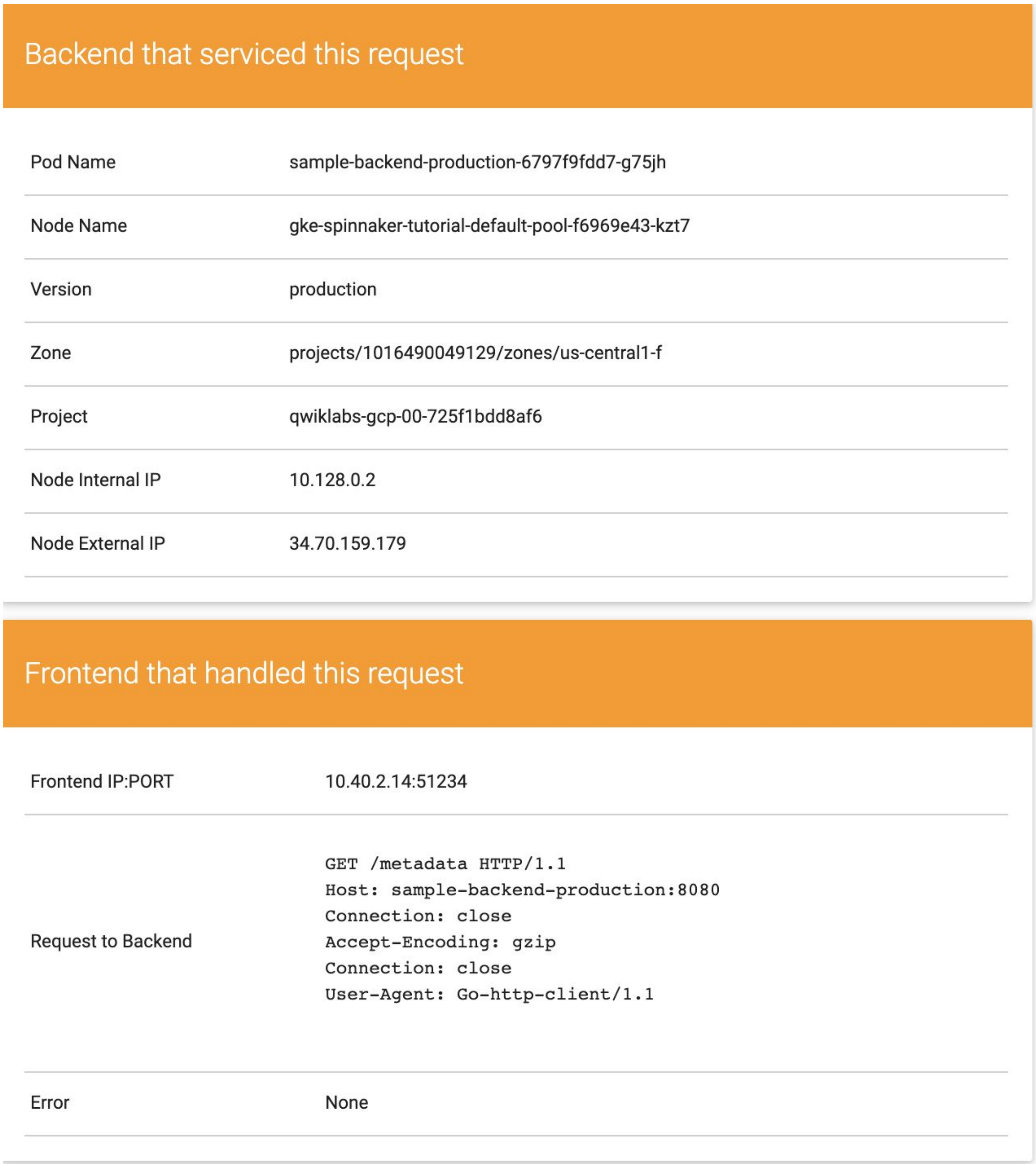
```
git revert v1.0.1
```

```
git tag v1.0.2
```

```
git push --tags
```

4. When the build and then the pipeline completes, verify the roll back by clicking **Infrastructure** > **Load Balancers**, then click the **service sample-frontend-production Default** and copy the Ingress IP address into a new tab.

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



Backend that serviced this request

Pod Name	sample-backend-production-6797f9fdd7-g75jh
Node Name	gke-spinnaker-tutorial-default-pool-f6969e43-kzt7
Version	production
Zone	projects/1016490049129/zones/us-central1-f
Project	qwiklabs-gcp-00-725f1bdd8af6
Node Internal IP	10.128.0.2
Node External IP	34.70.159.179

Frontend that handled this request

Frontend IP:PORT 10.40.2.14:51234

Request to Backend

```
GET /metadata HTTP/1.1
Host: sample-backend-production:8080
Connection: close
Accept-Encoding: gzip
Connection: close
User-Agent: Go-http-client/1.1
```

Error None

Congratulations

You have now successfully completed the Continuous Delivery Pipelines with Spinnaker and Kubernetes Engine lab.

Finish Your Quest

This self-paced lab is part of the Qwiklabs Google Cloud Solutions I: Scaling Your Infrastructure and DevOps Essentials 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 a 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 Running Dedicated Game Servers in Google Kubernetes Engine, or check out these suggestions:

- Deployment Manager - Full Production
- Site Reliability Troubleshooting with Stackdriver APM

Next Steps / Learn More

- Learn about Continuous Deployment with Jenkins
- Learn about Deploying Spinnaker to Compute Engine
- Deploy Jenkins to Kubernetes Engine
- Use Spinnaker with Ansible to Do Continuous Delivery

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: October 7, 2020
Lab Last Tested: October 7, 2020