Spinnaker Pipeline

[Continuous Delivery Pipelines with Spinnaker and Google Kubernetes Engine 2](#_Toc4405523)

[Pipeline architecture 2](#_Toc4405524)

[App delivery pipeline 3](#_Toc4405525)

[Objectives 3](#_Toc4405526)

[Costs 3](#_Toc4405527)

[Before you begin 4](#_Toc4405528)

[Set up your environment 4](#_Toc4405529)

[Start a Cloud Shell instance and create a GKE cluster 4](#_Toc4405530)

[Configure identity and access management 4](#_Toc4405531)

[Set up Cloud Pub/Sub to trigger Spinnaker pipelines 5](#_Toc4405532)

[Deploying Spinnaker using Helm 6](#_Toc4405533)

[Install Helm 6](#_Toc4405534)

[Configure Spinnaker 7](#_Toc4405535)

[Deploy the Spinnaker chart 8](#_Toc4405536)

[Building the Docker image 10](#_Toc4405537)

[Create your source code repository 10](#_Toc4405538)

[Configure your build triggers 11](#_Toc4405539)

[Prepare your Kubernetes Manifests for use in Spinnaker 13](#_Toc4405540)

[Build your image 14](#_Toc4405541)

[Configuring your deployment pipelines 15](#_Toc4405542)

[Install the spin CLI for managing Spinnaker 15](#_Toc4405543)

[Create the deployment pipeline 15](#_Toc4405544)

[View your pipeline execution 16](#_Toc4405545)

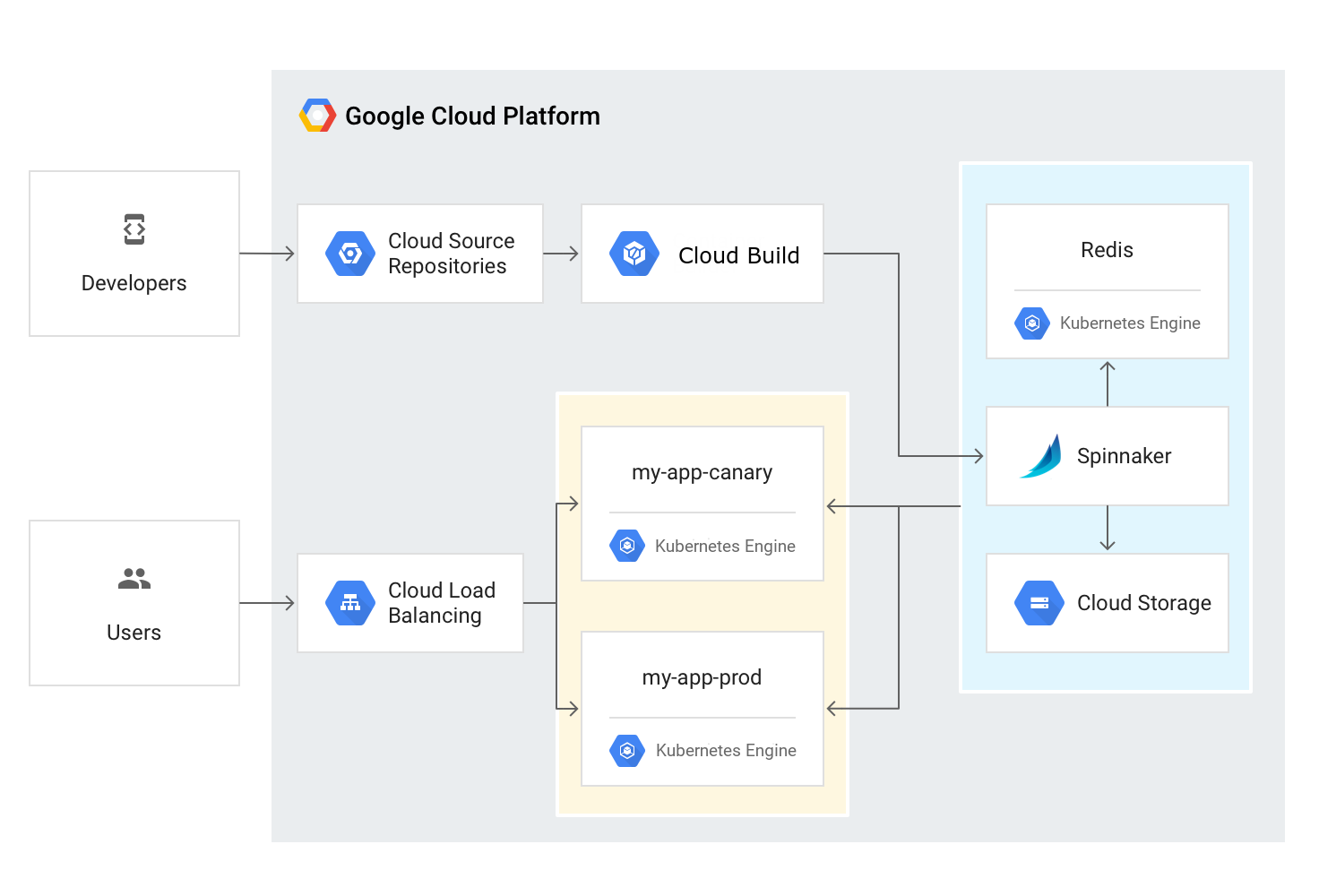
[Triggering your pipeline from code changes 18](#_Toc4405546)

[Cleaning up 20](#_Toc4405547)

Continuous Delivery Pipelines with Spinnaker and Google Kubernetes Engine

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

Pipeline architecture



To continuously deliver app 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](https://martinfowler.com/bliki/CanaryRelease.html) proves unsatisfactory, your automated procedure must be able to quickly roll back the software changes.

With GKE 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 app revision passes through a series of automated validations before becoming a candidate for production rollout. When a given change has been vetted through automation, you can also validate the app manually and conduct further prerelease testing.

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

App delivery pipeline

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

Objectives

* Set up your environment by launching [Cloud Shell](https://cloud.google.com/shell/), creating a GKE cluster, and configuring your identity and user management scheme.
* Download a sample app, create a Git repository, and upload it to a Cloud Source Repository.
* Deploy Spinnaker to GKE using [Helm](https://github.com/kubernetes/helm).
* Build your Docker image.
* Create triggers to create Docker images when your app changes.
* Configure a Spinnaker pipeline to reliably and continuously deploy your app to GKE.
* Deploy a code change, triggering the pipeline, and watch it roll out to production.

Costs

This tutorial uses billable components of Google Cloud Platform (GCP), including:

* GKE
* Cloud Load Balancing
* Cloud Build

Use the [Pricing Calculator](https://cloud.google.com/products/calculator) to generate a cost estimate based on your projected usage.

Before you begin

1. Select or create a Google Cloud Platform project.

**Note**: If you don't plan to keep the resources you create in this tutorial, create a new project instead of selecting an existing project. After you finish, you can delete the project, removing all resources associated with the project and tutorial.

[GO TO THE MANAGE RESOURCES PAGE](https://console.cloud.google.com/cloud-resource-manager)

1. Make sure that billing is enabled for your Google Cloud Platform project.

[LEARN HOW TO ENABLE BILLING](https://cloud.google.com/billing/docs/how-to/modify-project)

1. Enable the GKE, Cloud Build, and Cloud Source Repositories APIs.

[ENABLE THE APIS](https://console.cloud.google.com/flows/enableapi?apiid=container,cloudbuild.googleapis.com,sourcerepo.googleapis.com)

Set up your environment

In this section, you configure the infrastructure and identities required to complete the tutorial.

Start a Cloud Shell instance and create a GKE cluster

You run all the terminal commands in this tutorial from Cloud Shell.

1. Open Cloud Shell:

[OPEN CLOUD SHELL](https://console.cloud.google.com/?cloudshell=true)

1. Create a GKE cluster to deploy Spinnaker and the sample app with the following commands:
2. gcloud config set compute/zone us-central1-f
3. gcloud container clusters create spinnaker-tutorial \
4. --machine-type=n1-standard-2

Configure identity and access management

You create a Cloud Identity and Access Management (Cloud IAM) [service account](https://cloud.google.com/iam/docs/service-accounts) 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.

1. Create the service account:
2. gcloud iam service-accounts create spinnaker-account \
3. --display-name spinnaker-account
4. Store the service account email address and your current project ID in environment variables for use in later commands:
5. export SA\_EMAIL=$(gcloud iam service-accounts list \
6. --filter="displayName:spinnaker-account" \
7. --format='value(email)')
8. export PROJECT=$(gcloud info --format='value(config.project)')
9. Bind the storage.admin role to your service account:
10. gcloud projects add-iam-policy-binding \
11. $PROJECT --role roles/storage.admin --member serviceAccount:$SA\_EMAIL
12. Download the service account key. You need this key later when you install Spinnaker and upload the key to GKE.
13. gcloud iam service-accounts keys create spinnaker-sa.json --iam-account $SA\_EMAIL

Set up Cloud Pub/Sub to trigger Spinnaker pipelines

1. Create the Cloud Pub/Sub topic for notifications from Container Registry. This command may fail with the error "Resource already exists in the project", which means that the topic has already been created for you.
2. gcloud beta pubsub topics create projects/$PROJECT/topics/gcr
3. Create a subscription that Spinnaker can read from to receive notifications of images being pushed.
4. gcloud beta pubsub subscriptions create gcr-triggers \
5. --topic projects/${PROJECT}/topics/gcr
6. Give Spinnaker's service account permissions to read from the gcr-triggers subscription.
7. export SA\_EMAIL=$(gcloud iam service-accounts list \
8. --filter="displayName:spinnaker-account" \
9. --format='value(email)')
10. gcloud beta pubsub subscriptions add-iam-policy-binding gcr-triggers \
11. --role roles/pubsub.subscriber --member serviceAccount:$SA\_EMAIL

Deploying Spinnaker using Helm

In this section, you use [Helm](https://github.com/kubernetes/helm) to deploy Spinnaker from the [Charts](https://github.com/kubernetes/charts) repository. Helm is a package manager you can use to configure and deploy [Kubernetes apps](https://hub.kubeapps.com/).

Install Helm

1. Download and install the helm binary:
2. wget https://storage.googleapis.com/kubernetes-helm/helm-v2.10.0-linux-amd64.tar.gz
3. Unzip the file to your local system:
4. tar zxfv helm-v2.10.0-linux-amd64.tar.gz
5. cp linux-amd64/helm .
6. Grant Tiller, the server side of Helm, the cluster-admin role in your cluster:
7. kubectl create clusterrolebinding user-admin-binding --clusterrole=cluster-admin --user=$(gcloud config get-value account)
8. kubectl create serviceaccount tiller --namespace kube-system
9. kubectl create clusterrolebinding tiller-admin-binding --clusterrole=cluster-admin --serviceaccount=kube-system:tiller
10. Grant Spinnaker the cluster-admin role so it can deploy resources across all namespaces:
11. kubectl create clusterrolebinding --clusterrole=cluster-admin --serviceaccount=default:default spinnaker-admin
12. Initialize Helm to install Tiller in your cluster:

./helm init --service-account=tiller

./helm update

1. Ensure that Helm is properly installed by running the following command. If Helm is correctly installed, v2.10.0 appears for both client and server.

./helm version

Client: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b115f93090", GitTreeState:"clean"}

Server: &version.Version{SemVer:"v2.10.0", GitCommit:"9ad53aac42165a5fadc6c87be0dea6b115f93090", GitTreeState:"clean"}

Configure Spinnaker

1. Create a bucket for Spinnaker to store its pipeline configuration:
2. export PROJECT=$(gcloud info \
3. --format='value(config.project)')
4. export BUCKET=$PROJECT-spinnaker-config

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

1. Create the file (spinnaker-config.yaml) describing the configuration for how Spinnaker should be installed:

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:  
  spinnakerVersion: 1.10.2  
  image:  
    tag: 1.12.0  
  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

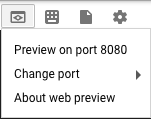
1. Use the Helm command-line interface to deploy the chart with your configuration set. This command typically takes five to ten minutes to complete.
2. ./helm install -n cd stable/spinnaker -f spinnaker-config.yaml --timeout 600 \

--version 1.1.6 --wait

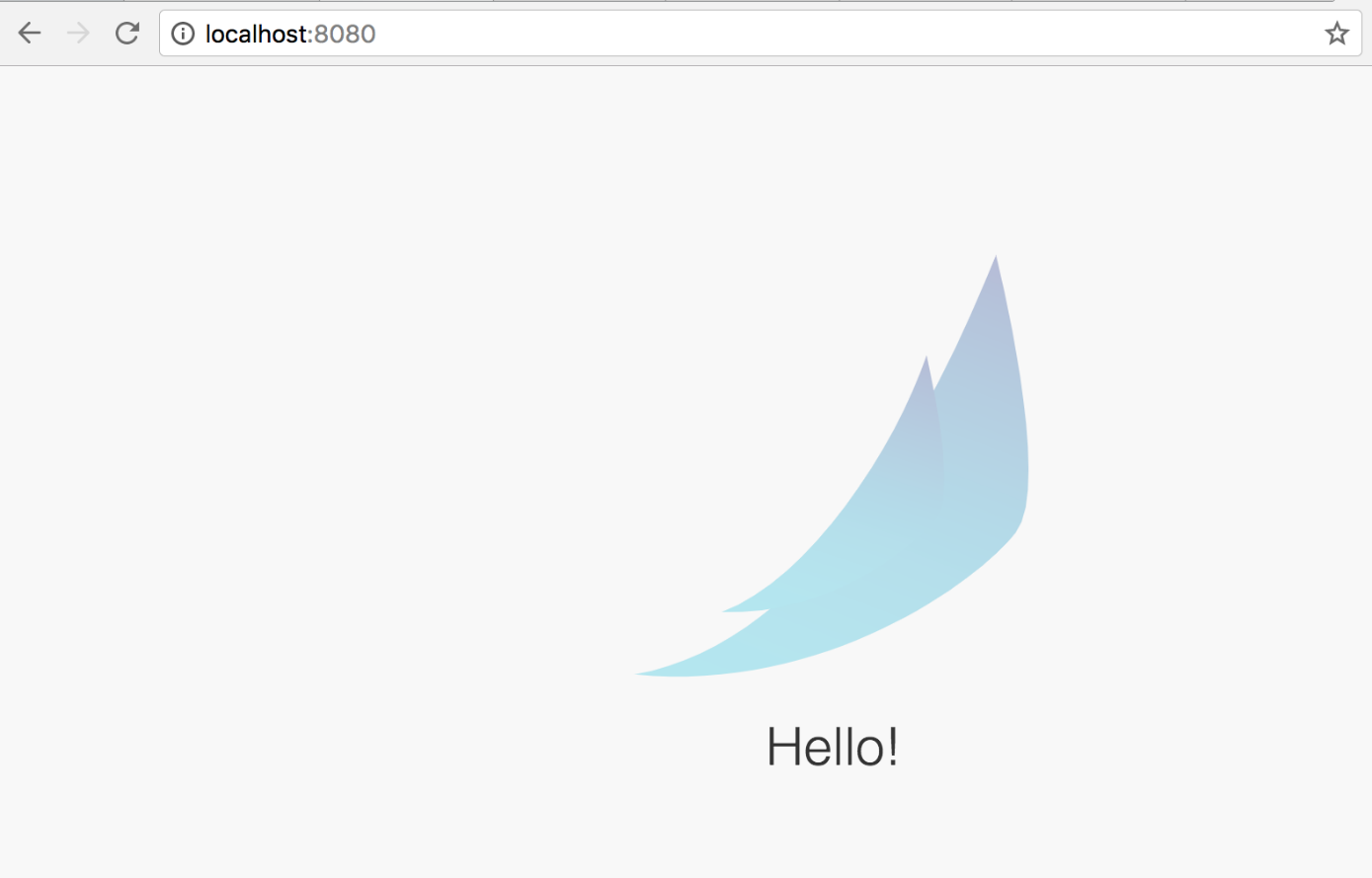
1. After the command completes, run the following command to set up port forwarding to the Spinnaker UI from Cloud Shell:
2. export DECK\_POD=$(kubectl get pods --namespace default -l "cluster=spin-deck" \
3. -o jsonpath="{.items[0].metadata.name}")
4. kubectl port-forward --namespace default $DECK\_POD 8080:9000 >> /dev/null &

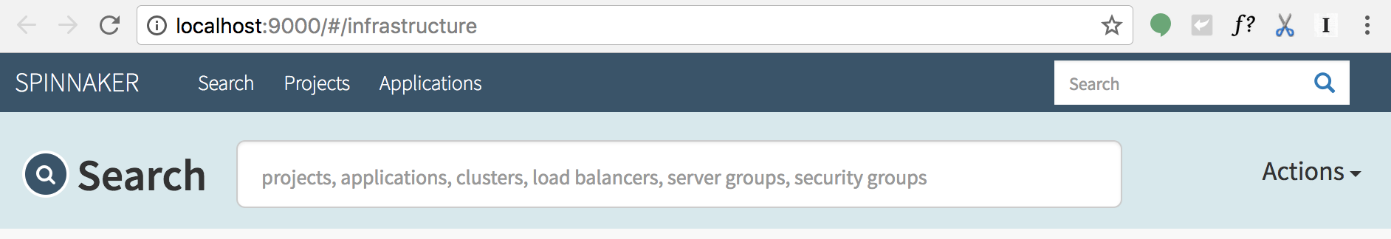
**Note:** This command can take several minutes to complete. Be sure to wait until you see that it has succeeded before proceeding.

1. To open the Spinnaker user interface, click **Web Preview** in Cloud Shell and click **Preview on port 8080**.



1. You should see the welcome screen, followed by the Spinnaker UI:





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, download the sample source code:
2. wget https://gke-spinnaker.storage.googleapis.com/sample-app-v2.tgz
3. Unpack the source code:
4. tar xzfv sample-app-v2.tgz
5. Change directories to the source code:
6. cd sample-app
7. Set the username and email address for your Git commits in this repository. Replace [EMAIL\_ADDRESS] with your Git email address, and replace [USERNAME] with your Git username.
8. git config --global user.email "[EMAIL\_ADDRESS]"
9. git config --global user.name "[USERNAME]"
10. Make the initial commit to your source code repository:
11. git init
12. git add .
13. git commit -m "Initial commit"
14. Create a repository to host your code:
15. gcloud source repos create sample-app
16. git config credential.helper gcloud.sh
17. Add your newly created repository as remote:
18. export PROJECT=$(gcloud info --format='value(config.project)')
19. git remote add origin https://source.developers.google.com/p/$PROJECT/r/sample-app
20. Push your code to the new repository's master branch:

git push origin master

1. Check that you can see your source code in the console:

[GO TO THE SOURCE CODE PAGE](https://console.cloud.google.com/code/develop/browse/sample-app/master)

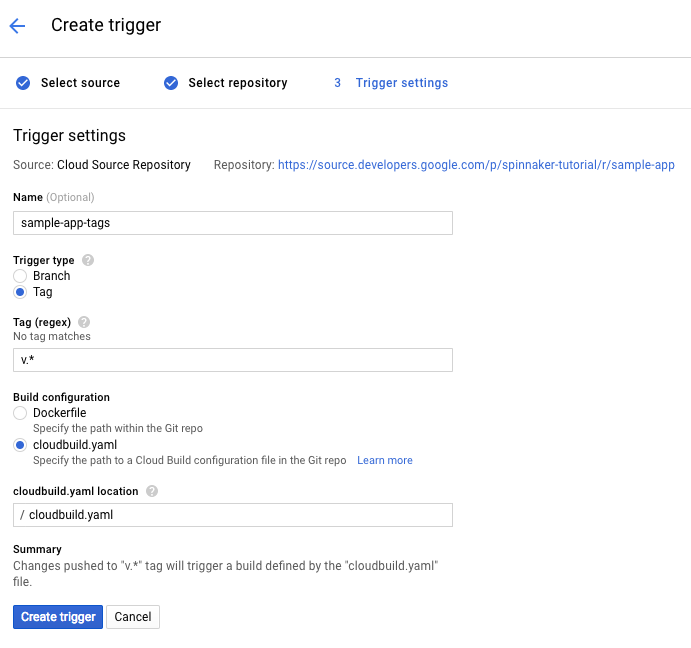
Configure your build triggers

In this section, you configure Cloud Build to build and push your Docker images every time you push[Git tags](https://git-scm.com/book/en/v2/Git-Basics-Tagging) to your source repository. Cloud Build automatically checks out your source code, builds the Docker image from the Dockerfile in your repository, and pushes that image to Container Registry.

1. In the GCP Console, in the **Cloud Build** section, click **Build Triggers**.

[GO TO THE BUILD TRIGGERS PAGE](https://console.cloud.google.com/gcr/triggers/add)

1. Select **Cloud Source Repository** and click **Continue**.
2. Select your newly created sample-app repository from the list, and click **Continue**.
3. Set the following trigger settings:
   * **Name**: sample-app-tags
   * **Trigger type**: Tag
   * **Tag (regex)**: v.\*
   * **Build configuration**: cloudbuild.yaml
   * **cloudbuild.yaml location**: cloudbuild.yaml
4. Click **Create trigger**.



From now on, whenever you push a Git tag prefixed with the letter "v" to your source code repository, Cloud Build automatically builds and pushes your app 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.

1. Create the bucket.
2. export PROJECT=$(gcloud info --format='value(config.project)')
3. gsutil mb -l us-central1 gs://$PROJECT-kubernetes-manifests
4. Enable versioning on the bucket so that you have a history of your manifests.
5. gsutil versioning set on gs://$PROJECT-kubernetes-manifests
6. Set the correct project ID in your kubernetes deployment manifests:
7. sed -i s/PROJECT/$PROJECT/g k8s/deployments/\*
8. Commit the changes to the repository:
9. git commit -a -m "Set project ID"

Build your image

Push your first image using the following steps:

1. Go to your source code folder in Cloud Shell.
2. Create a Git tag:

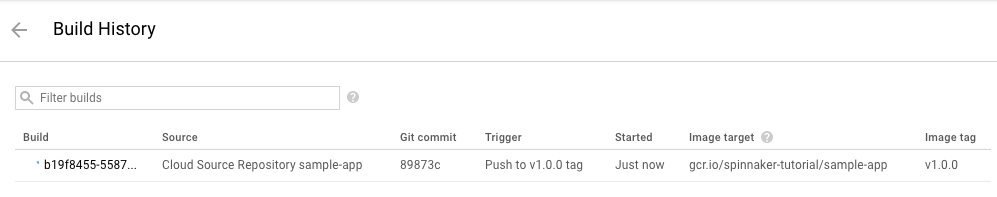
git tag v1.0.0

1. Push the tag:

git push --tags

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

[GO TO BUILD HISTORY](https://console.cloud.google.com/cloud-build/builds)



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](https://www.spinnaker.io/guides/spin/cli/) is a command-line utility for managing Spinnaker's applications and pipelines.

1. Download the latest version of spin.
2. curl -LO https://storage.googleapis.com/spinnaker-artifacts/spin/$(curl -s https://storage.googleapis.com/spinnaker-artifacts/spin/latest)/linux/amd64/spin
3. Make spin executable.
4. chmod +x spin

Create the deployment pipeline

1. Use spin to create an app in Spinnaker.
2. ./spin application save --application-name sample \
3. --owner-email example@example.com \
4. --cloud-providers kubernetes \
5. --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.

1. In a new tab of Cloud Shell, run the following command in the source code directory to upload an example pipeline to your Spinnaker instance:
2. export PROJECT=$(gcloud info --format='value(config.project)')
3. sed s/PROJECT/$PROJECT/g spinnaker/pipeline-deploy.json > pipeline.json
4. ./spin pipeline save --gate-endpoint http://localhost:8080/gate -f pipeline.json

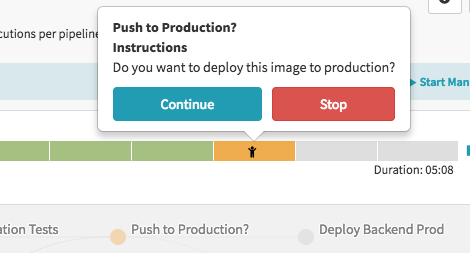
View your pipeline execution

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. You can now check on the pipeline that was triggered.

1. Return to the Pipelines page by clicking **Pipelines**.
2. Click **Details** to see more information about the pipeline's progress. This section shows the status of the deployment pipeline and its steps. Steps in blue are currently running, green ones have completed successfully, and red ones have failed. Click a stage to see details about it.

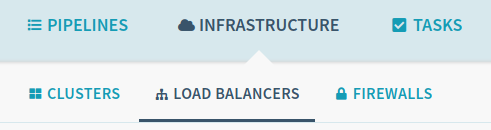
After 3 to 5 minutes the integration test phase completes and the pipeline requires manual approval to continue the deployment.

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

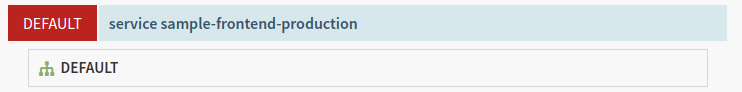


Your rollout continues to the production frontend and backend deployments. It completes after a few minutes.

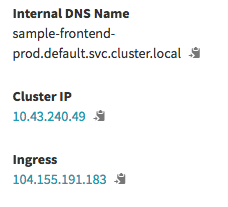
1. To view the app, select **Infrastructure** > **Load Balancers** in the top of the Spinnaker UI.



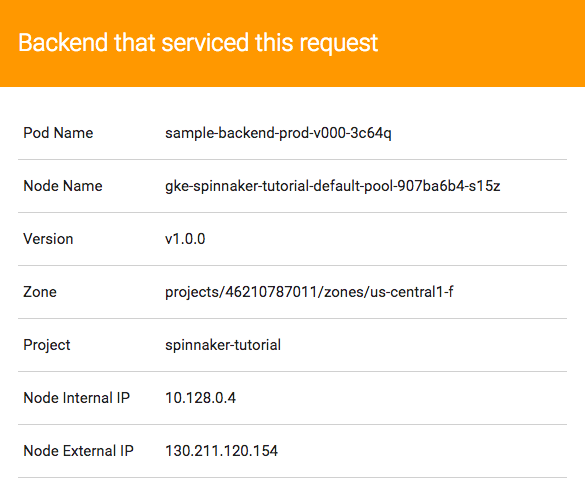
1. Scroll down the list of load balancers and click **Default**, under **sample-frontend-production**.



1. 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 uses HTTPS by default, but the application is configured to use HTTP.



1. Paste the address into your browser to view the production version of the app.

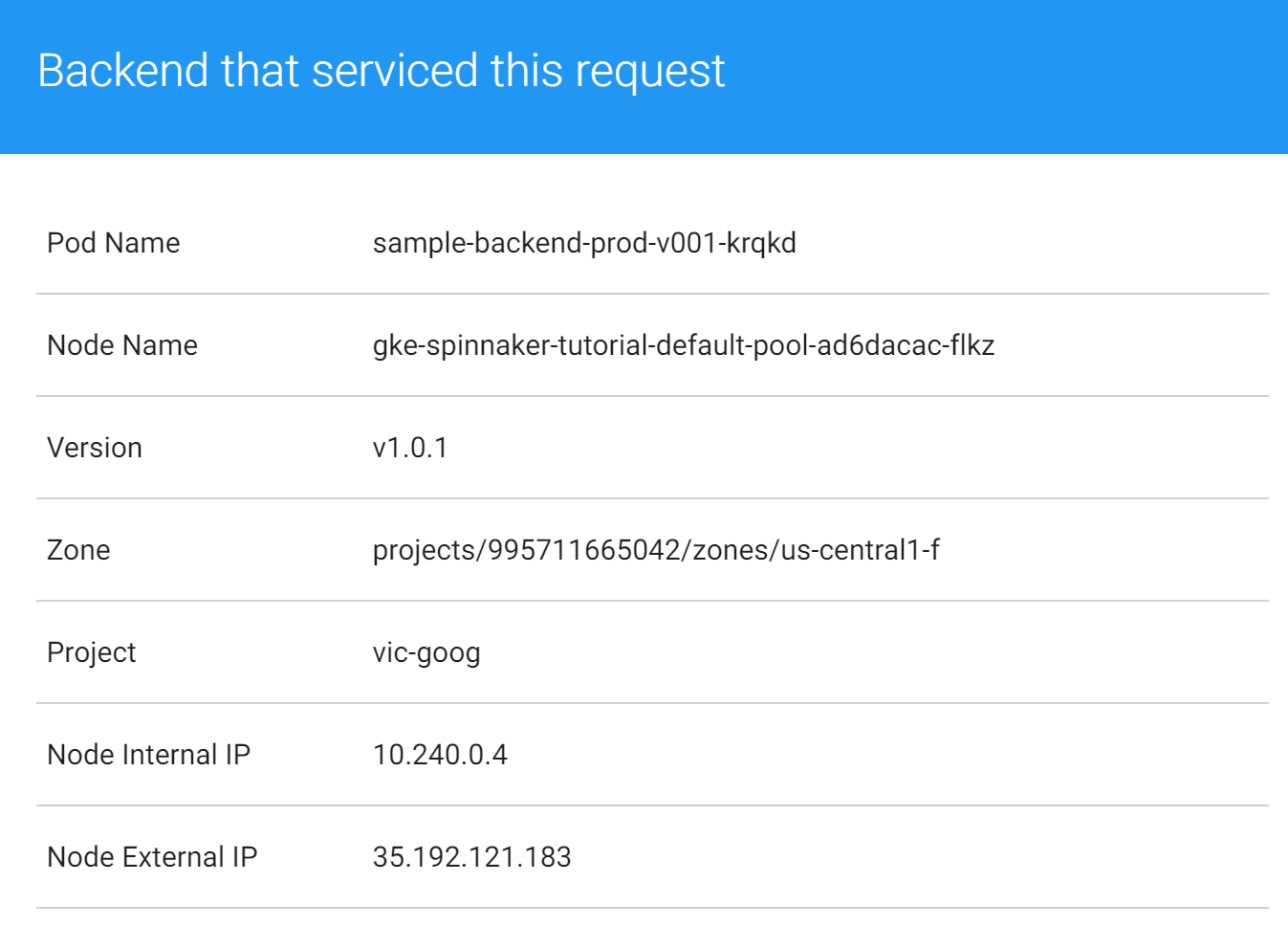


You have now manually triggered the pipeline to build, test, and deploy your app.

Triggering your pipeline from code changes

In this section, you 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 Cloud Build 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. Change the color of the app from orange to blue:
2. sed -i 's/orange/blue/g' cmd/gke-info/common-service.go
3. Tag your change and push it to the source code repository:
4. git commit -a -m "Change color to blue"
5. git tag v1.0.1
6. git push --tags
7. See the new build appear in the [Cloud Build Build History](https://console.cloud.google.com/gcr/builds).
8. Click **Pipelines** to watch the pipeline start to deploy the image.
9. Observe the canary deployments. When the deployment is paused, waiting to roll out to production, 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 tenth time you refresh.
10. After testing completes, return to the **Spinnaker** tab and approve the deployment.
11. 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 v1.0.1.



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

1. 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:
2. git revert v1.0.1
3. git tag v1.0.2

git push --tags

Cleaning up

To avoid incurring charges to your Google Cloud Platform account for the resources used in this tutorial:

1. Delete the Spinnaker installation:
2. ../helm delete --purge cd
3. Delete the sample app services:
4. kubectl delete -f k8s/services
5. Remove the service account IAM bindings:
6. export SA\_EMAIL=$(gcloud iam service-accounts list \
7. --filter="displayName:spinnaker-account" --format='value(email)')
8. export PROJECT=$(gcloud info --format='value(config.project)')
9. gcloud projects remove-iam-policy-binding $PROJECT --role roles/storage.admin --member serviceAccount:$SA\_EMAIL
10. Delete the service account:
11. export SA\_EMAIL=$(gcloud iam service-accounts list \
12. --filter="displayName:spinnaker-account" --format='value(email)')
13. gcloud iam service-accounts delete $SA\_EMAIL
14. Delete the GKE cluster:
15. gcloud container clusters delete spinnaker-tutorial --zone=us-central1-f
16. Delete the repository:
17. gcloud source repos delete sample-app
18. Delete the bucket:
19. export PROJECT=$(gcloud info --format='value(config.project)')
20. export BUCKET=$PROJECT-spinnaker-config
21. gsutil -m rm -r gs://$BUCKET
22. Delete your container images:
23. export PROJECT=$(gcloud info --format='value(config.project)')
24. gcloud container images delete gcr.io/$PROJECT/sample-app:v1.0.0
25. gcloud container images delete gcr.io/$PROJECT/sample-app:v1.0.1
26. If you created v1.0.2 in the optional rollback step above, delete that container image:

gcloud container images delete gcr.io/$PROJECT/sample-app:v1.0.2