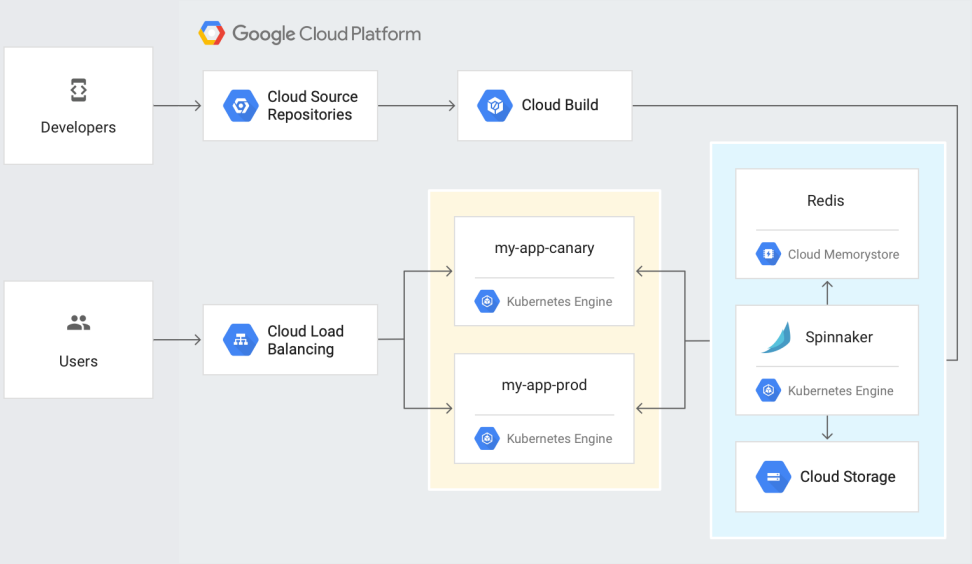
This tutorial shows you how to create a continuous delivery pipeline using Google Kubernetes Engine (GKE), Cloud Source Repositories, Cloud Build, and Spinnaker for Google Cloud. After you create a sample app, you configure these services to automatically build, test, and deploy it. When you modify the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

The following diagram illustrates the architecture of the continuous delivery pipeline.

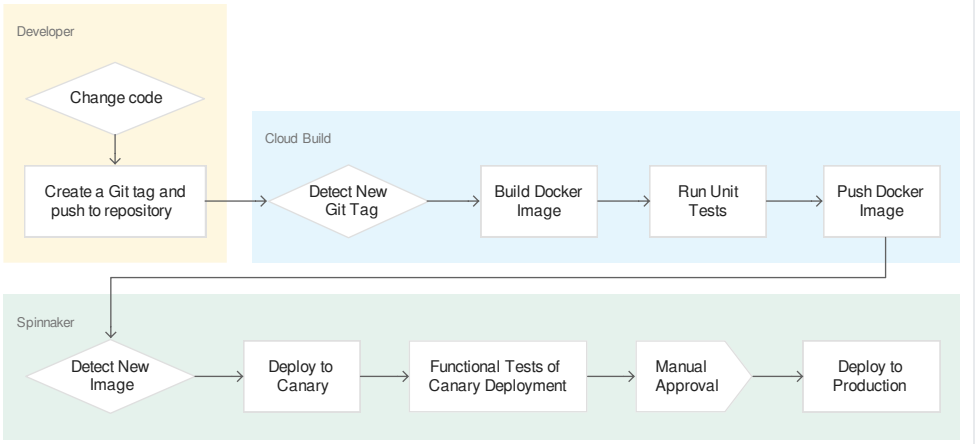


To continuously deliver app updates to your users, you need an automated process that reliably builds, tests, and updates your software. Code changes should automatically flow through a pipeline that includes artifact creation, unit testing, functional testing, and production rollout. In some cases, 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.

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



The high-level steps of this pipeline are as follows:

1. A developer changes code and pushes it to a repository.
2. Cloud Build detects the changes, builds the Docker image, tests the image, and pushes the image to Spinnaker.
3. Spinnaker detects the image, deploys image to Canary, and tests the Canary deployment. After a manual approval, Spinnaker deploys the image to production.

- Set up your environment by launching Cloud Shell (/shell/) and deploying Spinnaker for Google Cloud (https://cloud.google.com/docs/ci-cd/spinnaker/spinnaker-for-gcp).
- Create a GKE cluster to deploy the sample application to.
- Download a sample app, create a Git repository, and upload it to a Cloud Source Repositories.
- 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.

This tutorial uses billable components of Google Cloud, including:

- GKE
- Cloud Load Balancing
- Cloud Build
- Cloud Source Repositories
- Container Registry

Use the Pricing Calculator (/products/calculator) to generate a cost estimate based on your projected usage.

New Google Cloud users might be eligible for a free trial (/free-trial).

1. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a Cloud project.

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

Go to the project selector page (https://console.cloud.google.com/projectselector2/home/dashboard)

3. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (/billing/docs/how-to/modify-project).

4. 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)

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

Run all the terminal commands in this tutorial from Cloud Shell.

With Spinnaker for Google Cloud (https://cloud.google.com/docs/ci-cd/spinnaker/spinnaker-for-gcp), you can set up and manage Spinnaker in a production-ready configuration, optimized for Google Cloud. Spinnaker for Google Cloud sets up resources (GKE, Memorystore, Cloud Storage buckets and service accounts), integrates Spinnaker with related services such as Cloud Build, and provides a Cloud Shell-based management environment for your Spinnaker installations, with helpers and common tools such as `spin` and `hal`.

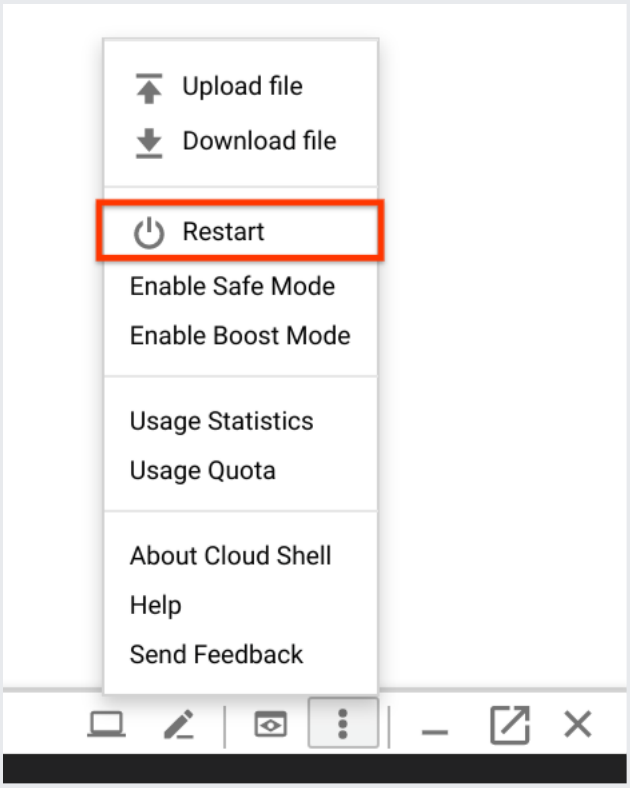
1. Open Spinnaker for Google Cloud in Cloud Shell. This clones the Spinnaker for Google Cloud repository (https://github.com/GoogleCloudPlatform/spinnaker-for-gcp.git) into your Cloud Shell environment and launches the detailed installation instructions.

GO TO CLOUD SHELL (https://console.cloud.google.com/cloudshell/editor?cloudshell_git_repo=https://github.com/GoogleCloudPlatform/spinnaker-for-gc

2. Install Spinnaker for Google Cloud.

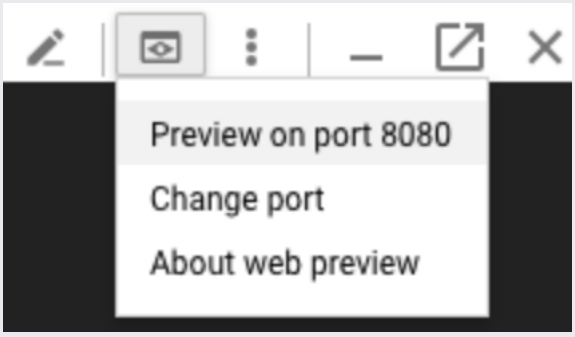
Note: This installation command will take several minutes to complete and is a basic setup for the purposes of this tutorial. For a production setup, follow the detailed instructions included with Spinnaker for Google Cloud.

1. Restart Cloud Shell to load new environment settings.



2. Connect to Spinnaker.

3. In Cloud Shell, click the **Web Preview** icon and select **Preview on port 8080**.



★ **Note:** For now, this Spinnaker instance isn't publicly accessible. Only you have access to it, with no authentication. A production Spinnaker instance is a critical component of your infrastructure, so you must properly secure it. Several options are available to you for security and authentication:

- Spinnaker for Google Cloud provides tools (https://github.com/GoogleCloudPlatform/spinnaker-for-gcp/blob/master/scripts/install/provision-spinnaker.md#expose-spinnaker-publicly) to help secure your deployment using IAP with a SSL Certificate.
- Take a look at the security documentation (https://www.spinnaker.io/setup/security/) of Spinnaker.
- Use G Suite as an identity provider (https://www.spinnaker.io/setup/security/authentication/oauth/google/) for Spinnaker authentication.
- Use Google Groups (https://www.spinnaker.io/setup/security/authorization/google-groups/) for Spinnaker authorization.
- Configure a Identity-Aware Proxy (/iap/docs/enabling-kubernetes-howto) in front of Spinnaker to further control who has access to it.

A common pattern is to have a GKE cluster used for builds, deployments, and so on, and then other GKE clusters for running applications. In this section you create another GKE cluster, `app-cluster`, to deploy the sample application to.

1. In Cloud Shell, create a new GKE cluster:

2. Add the new GKE cluster to Spinnaker. The default values should be correct.

   Example output and values:

3. Change the kubernetes context back to your Spinnaker cluster:

4. Push and apply the configuration changes to Spinnaker:

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.

1. In Cloud Shell, download the sample source code:

2. Unpack the source code:

3. Change directories to the source code:

4. 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.

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

6. Create a repository to host your code:

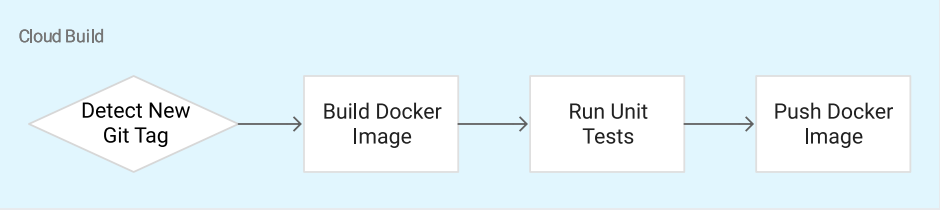
7. Add your newly created repository as remote:

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

9. 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)

The following diagram illustrates the trigger that you build in this section.

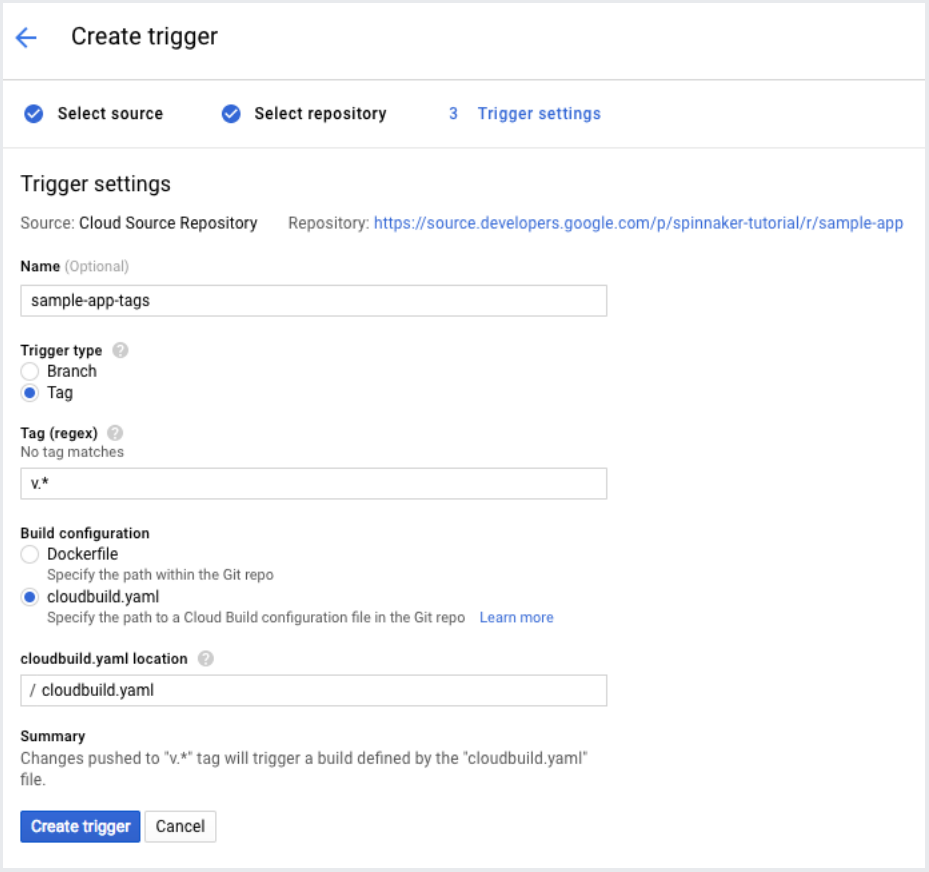


You configure Cloud Build to build and push your Docker images every time you push Git tags (https://git-scm.com/book/en/v2/Git-Basics-Tagging) to your source repository. Cloud Build automatically checks out your source code, builds the Docker image from the Dockerfile in your repository, and pushes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

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

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

2. Select **Cloud Source Repository** and click **Continue**.
3. Select your newly created `sample-app` repository from the list, and click **Continue**.
4. Set the following trigger settings:
    - **Name**: `sample-app-tags`
    - **Trigger type**: Tag
    - **Tag (regex)**: `v.*`
    - **Build configuration**: `cloudbuild.yaml`
    - **cloudbuild.yaml location**: `cloudbuild.yaml`
5. 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.

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.

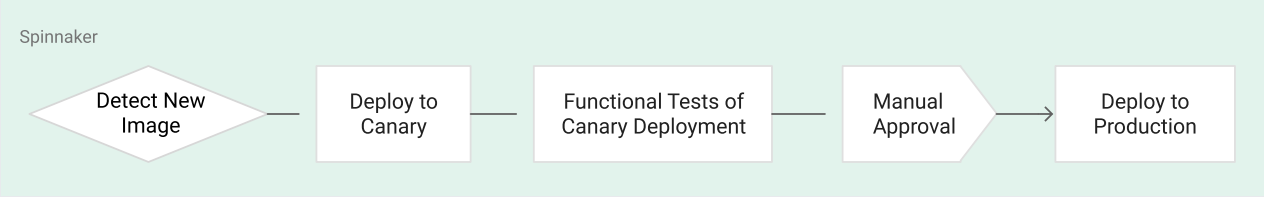
2. Enable versioning on the bucket so that you have a history of your manifests.

3. Set the correct Google Cloud project ID in your Kubernetes deployment manifests:

4. Commit the changes to the repository:

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

The following diagram illustrates the deployment pipeline steps.



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.

1. Use `spin` to create an app in Spinnaker.

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:

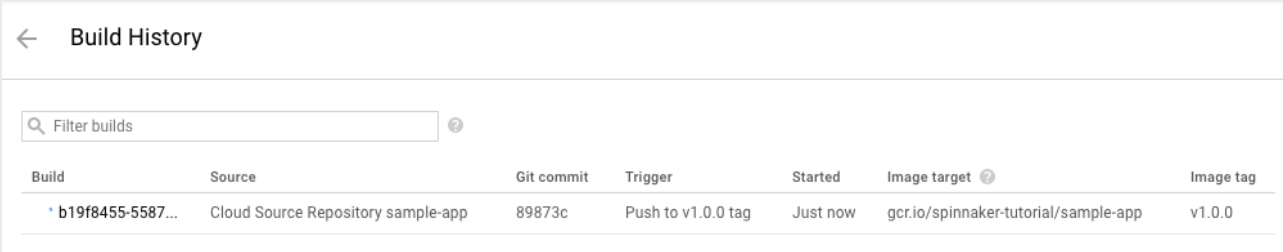
Push your first image using the following steps:

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

3. Push the tag:

4. 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)

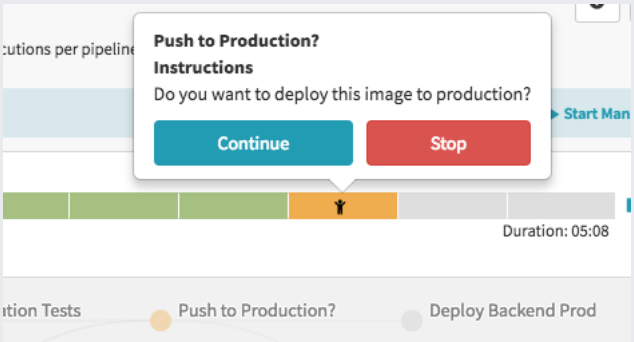


The configuration you just created uses notifications of newly tagged images being pushed to trigger a Spinnaker pipeline. In a previous step, you pushed a tag to the Cloud Source Repositories which triggered Cloud Build to build and push your image to Container Registry. You can now check on the pipeline 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.

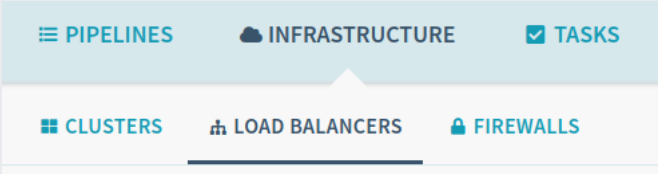
   After 3 to 5 minutes the integration test phase completes and the pipeline requires manual approval to continue the deployment.

3. Hold the pointer over **Push to production**, and then click **Continue**.

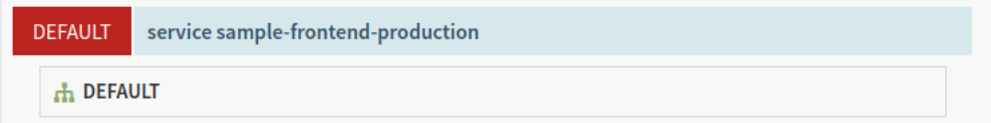


   Your rollout continues to the production frontend and backend deployments. It completes after a few minutes.

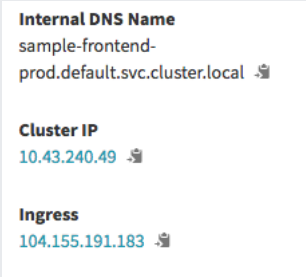
4. To view the app, select **Infrastructure** > **Load Balancers** in the top of the Spinnaker UI.



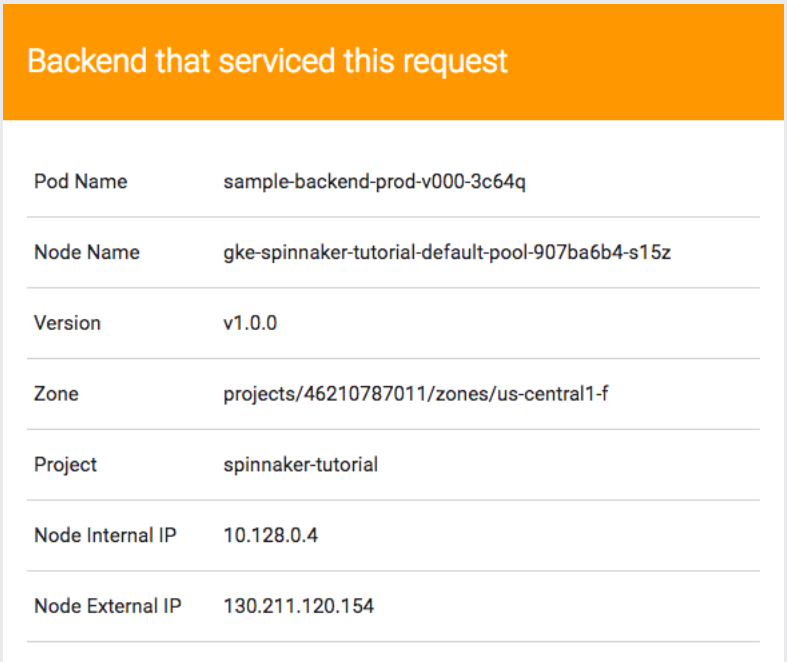
5. Scroll down the list of load balancers and click **Default**, under **sample-frontend-production**.



6. 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.



7. 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.

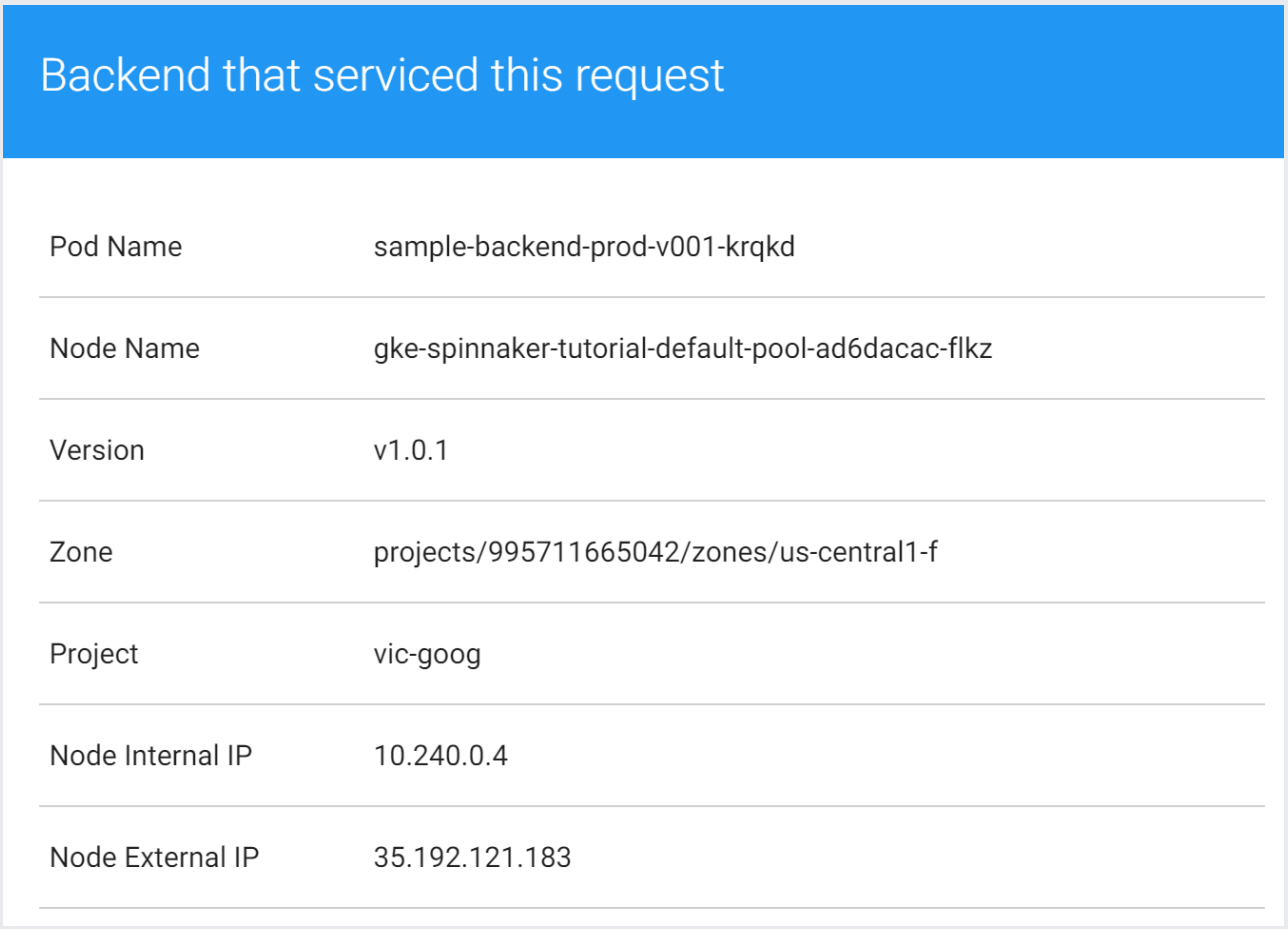
In this section, you test the pipeline end to end by making a code change, pushing a Git tag, and watching the pipeline run in response. By pushing a Git tag that starts with "v", you trigger Cloud Build to build a new Docker image and push it to Container Registry. Spinnaker detects

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

3. See the new build appear in the Cloud Build Build History (https://console.cloud.google.com/gcr/builds).
4. Click **Pipelines** to watch the pipeline start to deploy the image.
5. 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.
6. After testing completes, return to the **Spinnaker** tab and approve the deployment.
7. When the pipeline completes, your app looks like the following screenshot. The **Version** field now reads `v1.0.1`.



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

8. 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`:

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

1. Delete Spinnaker for Google Cloud:

2. Delete the GKE cluster:

3. Delete the repository:

4. Delete the bucket:

5. Delete your container images:

6. If you created `v1.0.2` in the optional rollback step above, delete that container image:

- Register existing Google Kubernetes Engine clusters in Spinnaker (https://www.spinnaker.io/setup/install/providers/kubernetes-v2/).
- Learn about continuous deployment with Jenkins (/solutions/continuous-delivery-jenkins-kubernetes-engine).
- Deploy Jenkins to GKE (/solutions/jenkins-on-kubernetes-engine-tutorial).

- Learn about the Kubernetes Provider in Spinnaker (https://www.spinnaker.io/reference/providers/kubernetes-v2/).
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).