Continuous Deployment with Spinnaker v1.6

3 hours9 Credits

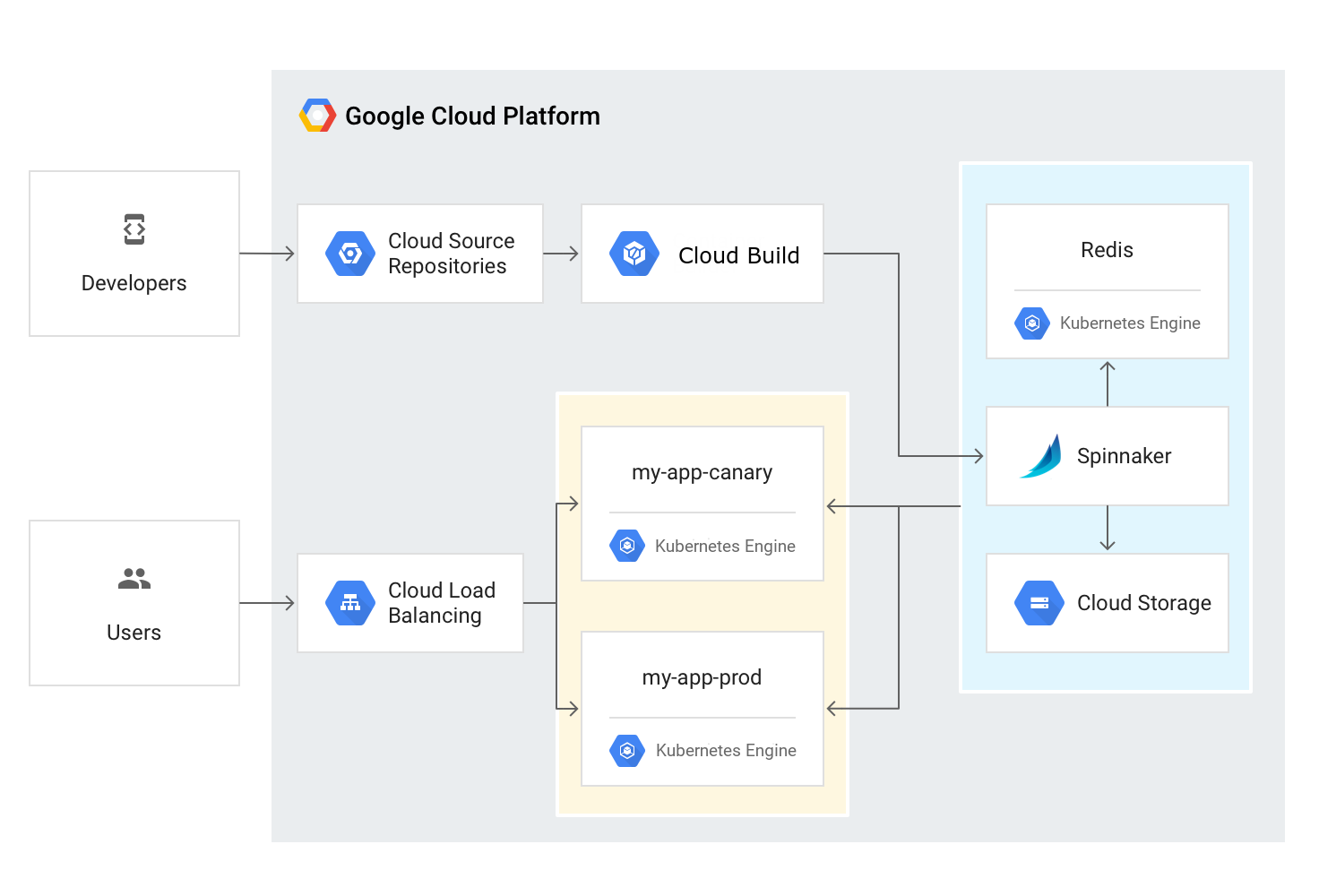
Rate Lab

This lab 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.

**Objectives**

* Set up your environment by launching [Cloud Shell](https://cloud.google.com/shell) and deploying [Spinnaker for Google Cloud](https://cloud.google.com/docs/ci-cd/spinnaker/spinnaker-for-gcp).
* Create a second GKE cluster to deploy the sample application to.
* Configure Spinnaker pipelines to reliably and continuously deploy your app to GKE environments.
* Copy a sample application, create a Git repository, and upload it to a Cloud Source Repositories.
* Create a Cloud Build trigger that builds Docker images when your application changes.
* Deploy a code change, watch your trigger build a Docker image, then observe your Spinnaker pipleline roll it out to staging and production.
* Roll back your change.

**Pipeline architecture**



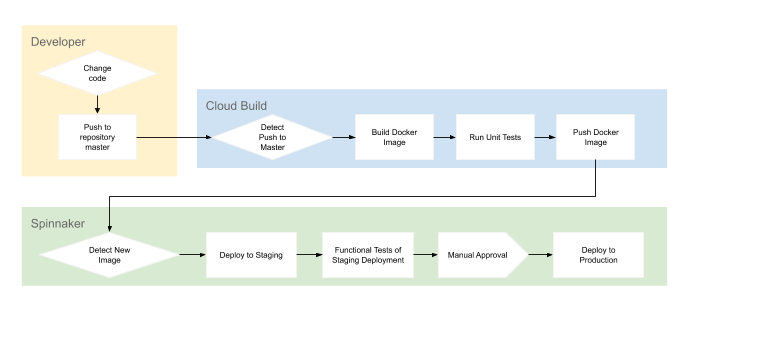
To continuously deliver app updates to your users, you need an automated process that reliably builds, tests, and updates your software. Code changes should automatically flow through a pipeline that includes artifact creation, unit testing, functional testing, and production rollout. In some cases,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

**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 Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access the Google Cloud Platform for the duration of the lab.

**What you need**

To complete this lab, you need:

* Access to a standard internet browser (Chrome browser recommended).
* Time to complete the lab.

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

**How to start your lab and sign in to the Console**

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



1. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Choose an account** page.

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

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



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

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

1. Click through the subsequent pages:
   * Accept the terms and conditions.
   * Do not add recovery options or two-factor authentication (because this is a temporary account).
   * Do not sign up for free trials.

After a few moments, the GCP console opens in this tab.

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

Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Google Cloud Shell provides command-line access to your GCP resources.

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



1. Click **Continue**. 

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:



**gcloud** is the command-line tool for Google Cloud Platform. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

gcloud auth list

Output:

Credentialed accounts:

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

Example output:

Credentialed accounts:

- google1623327\_student@qwiklabs.net

You can list the project ID with this command:

gcloud config list project

Output:

[core]

project = <project\_ID>

Example output:

[core]

project = qwiklabs-gcp-44776a13dea667a6

Full documentation of **gcloud** is available on [Google Cloud gcloud Overview](https://cloud.google.com/sdk/gcloud).

**Creating a cluster for Spinnaker operation**

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

Deploy the Spinnaker for Google Cloud solution using 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. Clone the Spinnaker for Google Cloud [repository](https://github.com/GoogleCloudPlatform/spinnaker-for-gcp.git) into your Cloud Shell environment.
2. mkdir ~/cloudshell\_open && cd ~/cloudshell\_open
3. git clone https://github.com/GoogleCloudPlatform/spinnaker-for-gcp.git
4. cd spinnaker-for-gcp
5. Create a script with your Spinnaker environment properties.
6. ./scripts/install/setup\_properties.sh

This creates the ./scripts/install/properties file.

1. Review your Spinnaker environment properties.
2. cat ./scripts/install/properties
3. Apply the properties to your environment.
4. source ./scripts/install/properties
5. Set up a git user which is used to create a git repo for Spinnaker config.
6. git config --global user.email "$USER@qwiklabs.net"
7. git config --global user.name "$USER"
8. Start the Spinnaker installation script.
9. ./scripts/install/setup.sh

Necessary APIs are enabled, which takes a couple minutes, and then you can follow along with the script output to see the steps:

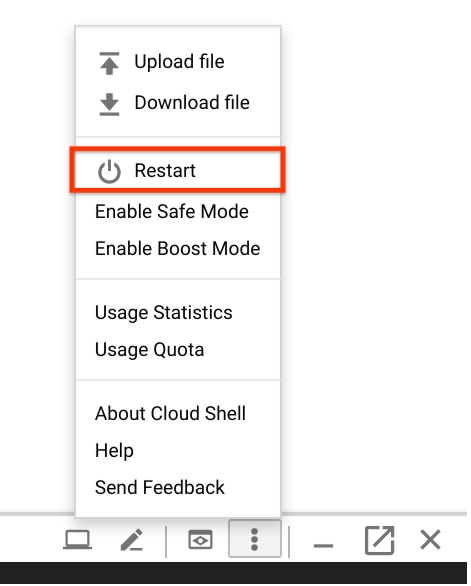
* + A service account is created and IAM roles assigned for Spinnaker tasks
  + A Redis instance is created in Cloud Memorystore
  + A Cloud Storage bucket is created
  + A GKE cluster is created for Spinnaker application pipelines
  + PubSub topics and subscriptions are created
  + Spinnaker resources are provisioned in the GKE cluster
  + A Spinnaker config git repo is configured in Cloud Source Repositories
  + The Halyard management tool for configuring Spinnaker: hal is installed in Cloud Shell
  + The Spinnaker tool for managing applications and pipelines as code: spin is installed in Cloud Shell

Altogether these tasks can take >15min to complete. You may see warnings that you can safely ignore as they should have no impact on this lab.

**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.

Click **Restart** from the Cloud Shell *more* menu, then **Restart** again. You don't have to select a reason for restarting.



1. Restore your environment and directory.
2. cd ~/cloudshell\_open/spinnaker-for-gcp
3. source ./scripts/install/properties

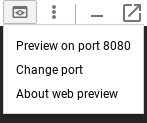
Connect to Spinnaker UI

1. Set up port-forwarding to connect from Cloud Shell.
2. ./scripts/manage/connect\_unsecured.sh

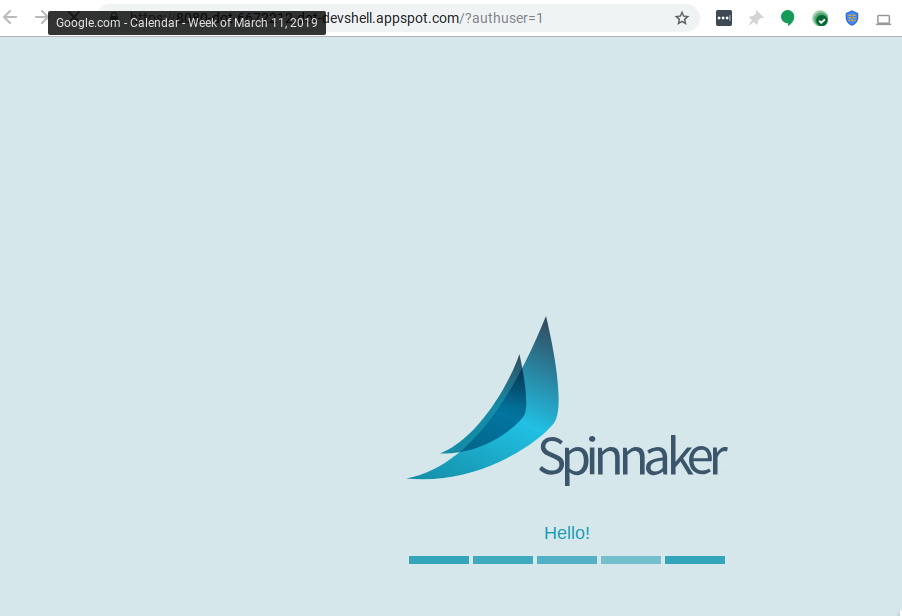
**Note:** If you encounter **error: You must be logged in to the server Unauthorized)()** re-run the connect\_unsecured.sh command.

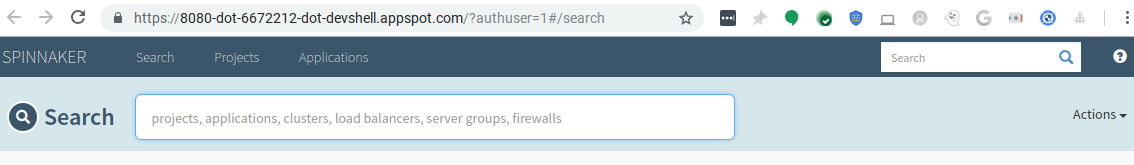
1. Open the Spinnaker web user interface.

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



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





**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 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](https://cloud.google.com/iap/docs/enabling-kubernetes-howto) in front of Spinnaker to further control who has access to it.

Click *Check my progress* to verify the objective.

Set up the environment

Check my progress

**Creating a second GKE cluster for application deployments**

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 for your application.

For this example, use a different *region* than the one used by the Spinnaker deployment.

APP\_REGION=us-west1-b; gcloud config set compute/zone $APP\_REGION

gcloud container clusters create app-cluster --machine-type=n1-standard-2

This can take ~4 min to complete. You may see warnings that you can safely ignore as they should have no impact on this lab.

1. Grant Spinnaker access to the new applications cluster.

Confirm the default script values by pressing **[Enter]** when prompted by each of three questions.

./scripts/manage/add\_gke\_account.sh

Example output and values:

Please enter the context you wish to use to manage your GKE resources: gke\_spinnaker-246920\_us-west1-b\_app-cluster

Please enter the id of the project within which the referenced cluster lives: spinnaker-246920

Please enter a name for the new Spinnaker account: app-cluster-acct

This ensures that a Google Cloud service account, named in your Spinnaker properties file, has the proper roles to access the application cluster. Then that service account is used to create Kubernetes credentials named app-cluster-acct in the Spinnaker configuration.

1. Apply this updated configuration to your Spinnaker cluster.
2. kubectl config use-context gke\_${PROJECT\_ID}\_${ZONE}\_spinnaker-1
3. ./scripts/manage/push\_and\_apply.sh

Click *Check my progress* to verify the objective.

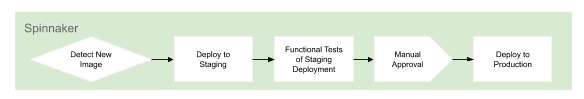
Creating a second GKE cluster for application deployments

Check my progress

**Configuring your deployment pipelines**

Deploy two Spinnaker pipelines that respond when new application images are available. The first pipeline is for a **staging** environment for integration testing of changes.

After the integration tests pass, you must **manually** approve the changes to allow the updated image to deploy to the **production** environment.



Create the application and deployment pipelines in Spinnaker

1. Use spin to register the helloworldwebapp in Spinnaker.
2. cd ./samples/helloworldwebapp/
3. ~/spin app save --application-name helloworldwebapp \
4. --cloud-providers kubernetes --owner-email $IAP\_USER
5. Use spin to create a "Deploy to Staging" Spinnaker pipeline.
6. cat templates/pipelines/deploystaging\_json.template | envsubst > templates/pipelines/deploystaging.json
7. ~/spin pi save -f templates/pipelines/deploystaging.json
8. Save the created staging pipeline id.

The staging pipeline id is used by the production pipeline as a trigger reference. By saving this id the production pipeline will automatically start work when it notices the staging pipeline has completed successfully.

export DEPLOY\_STAGING\_PIPELINE\_ID=$(~/spin pi get -a helloworldwebapp -n 'Deploy to Staging' | jq -r '.id')

1. Use spin to create a "Deploy to Prod" Spinnaker pipeline.
2. cat templates/pipelines/deployprod\_json.template | envsubst > templates/pipelines/deployprod.json
3. ~/spin pi save -f templates/pipelines/deployprod.json

You've created your application and pipelines in Spinnaker. You'll explore your pipelines in a later step.

Click *Check my progress* to verify the objective.

Configuring your deployment pipelines

Check my progress

**Creating your sample application code**

In this section, you copy a sample application, create a Git repository for it in Cloud Source Repositories, then push the application source to the repo.

You then configure Cloud Build to detect changes to your application source code, build a Docker image, and then push it to Container Registry.

Create your application source and config files

1. In Cloud Shell, create a new directory for your application source.
2. mkdir -p ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/
3. Copy the application source files.
4. cp -r templates/repo/src ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/
5. Copy the application config files.
6. cp -r templates/repo/config ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/
7. cp templates/repo/Dockerfile ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/
8. cat templates/repo/cloudbuild\_yaml.template | envsubst '$BUCKET\_NAME' > ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/cloudbuild.yaml
9. cat ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/config/staging/replicaset\_yaml.template | envsubst > ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/config/staging/replicaset.yaml
10. rm ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/config/staging/replicaset\_yaml.template
11. cat ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/config/prod/replicaset\_yaml.template | envsubst > ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/config/prod/replicaset.yaml
12. rm ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp/config/prod/replicaset\_yaml.template

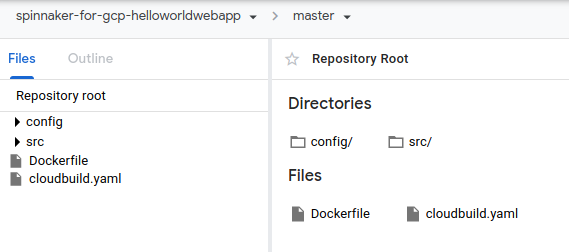
Note the config files include your Kubernetes application manifests. Spinnaker needs access to your Kubernetes manifests in order to deploy the application to your clusters during pipeline execution.

Create your application source/config Git repository

1. Change directories to the source/config code directory.
2. cd ~/$PROJECT\_ID/spinnaker-for-gcp-helloworldwebapp
3. Make the initial commit to your source code repository.
4. git init
5. git add .
6. git commit -m "Initial commit"
7. Create a repository to host your code:
8. gcloud source repos create spinnaker-for-gcp-helloworldwebapp
9. git config credential.helper gcloud.sh
10. Add your newly created repository as remote:
11. git remote add origin https://source.developers.google.com/p/$PROJECT\_ID/r/spinnaker-for-gcp-helloworldwebapp
12. Push your code to the new repository's master branch:
13. git push origin master
14. If you want, review your source code using Cloud Console by clicking **Navigation menu**, then in the **Tools** section, click **Source Repositories**.

Click **View All repositories** then **spinnaker-for-gcp-helloworldapp**.

You can look through the files.



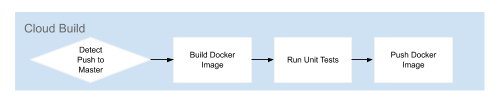
Click *Check my progress* to verify the objective.

Create your application source/config Git repository

Check my progress

**Configuring your build triggers**

In this section, you configure Cloud Build to build and push your Docker image every time you push an update to the **master** branch of your source. 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. Review your Cloud Build steps.
2. cat ./cloudbuild.yaml

Cloud Build will build your application, generate Kubernetes config manifests for multiple environments, and produce an image file.

1. Create the Cloud Build trigger.
2. gcloud beta builds triggers create cloud-source-repositories \
3. --repo spinnaker-for-gcp-helloworldwebapp \
4. --branch-pattern master \
5. --build-config cloudbuild.yaml \
6. --included-files "src/\*\*,config/\*\*"
7. Verify the Cloud Build trigger.
8. gcloud beta builds triggers list

Output **(do not copy)**

---

createTime: '2020-03-03T04:51:57.768831415Z'

filename: cloudbuild.yaml

id: 7e8cbeb3-8678-4bbc-b848-d3b6035f6c3f

includedFiles:

- src/\*\*

- config/\*\*

name: trigger

triggerTemplate:

branchName: master

projectId: qwiklabs-gcp-01-3367ef948325

repoName: spinnaker-for-gcp-helloworldwebapp

From now on, whenever you push to the master branch of your source code repository, Cloud Build automatically builds and pushes your app as a Docker image to Container Registry.

Click *Check my progress* to verify the objective.

Configuring your build triggers

Check my progress

**Building your image**

Push your first image to Container Registry using the following steps.

1. In Cloud Shell, from your source code directory, create a small change.
2. sed -i 's/Hello World/Hello World 1.0/g' ./src/main.go
3. git diff
4. git commit -a -m "Change to 1.0"
5. Push the change which will trigger a build.
6. git push origin master
7. Check Cloud Build in Cloud Console for the triggered build.

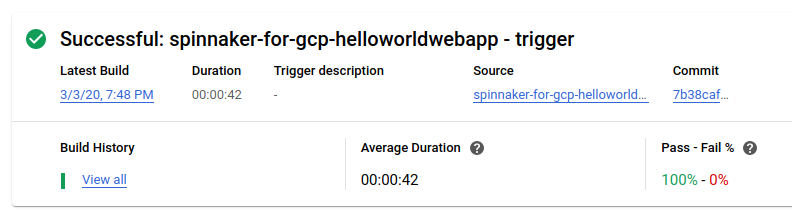
Click **Navigation menu**, then in the **Tools** section, click **Cloud Build** > **Dashboard**.

Look for a build with status **Running**.

1. Stay on this page and **wait** for the build to complete before going on to the next step.

After ~1 minute you should see **Successful**.

If you don't see **Successful**, wait a bit and refresh. Verify the trigger was configured properly in the previous instructions.



Click *Check my progress* to verify the objective.

Building the Docker image

Check my progress

**Viewing your pipelines**

The Spinnaker configuration you created uses notifications of new images being pushed to trigger Spinnaker pipelines.

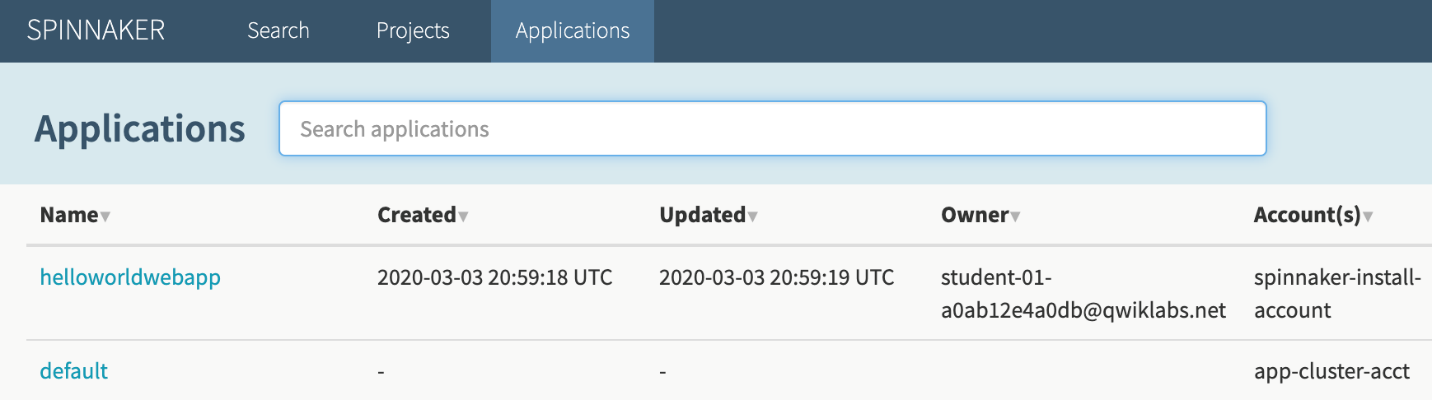
You pushed a change to Cloud Source Repositories which triggered Cloud Build to build and push your image to Container Registry.

Review the Deploy to Staging pipeline

1. Open your application in the Spinnaker UI.

Return to the Spinnaker UI and click **Applications** at the top of the screen to see your list of managed applications.

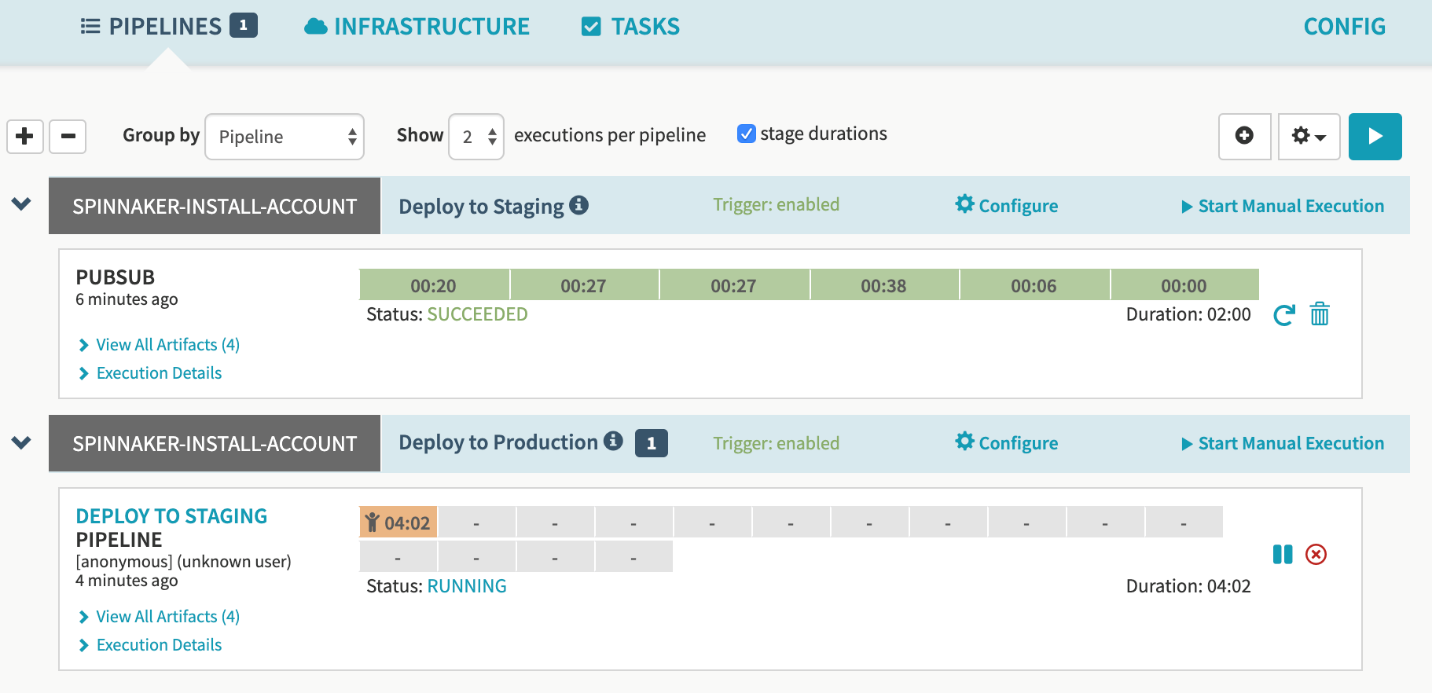
**helloworldwebapp** is your application. If you don't see helloworldwebapp, try refreshing the Spinnaker Applications tab.



1. Click **helloworldwebapp** to view your application deployment.
2. Click **Pipelines** at the top to view your application pipeline status.

You should notice two pipelines: **Deploy to Staging** and **Deploy to Production**.

**Deploy to Staging** should complete with ~2 minutes.

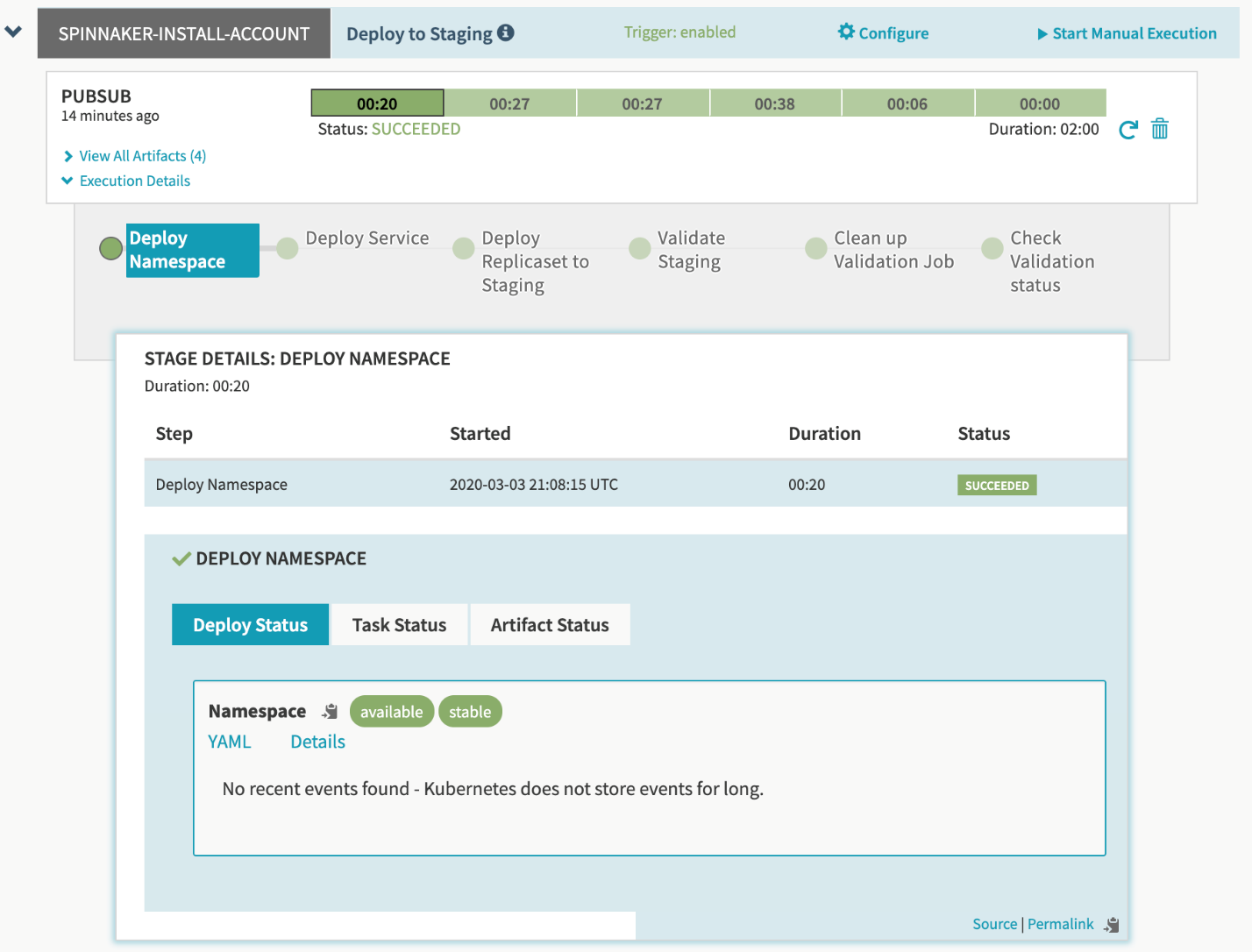


1. Review staging execution details.

When the pipeline is completed, all of the stage boxes should be colored green to signify success running each.

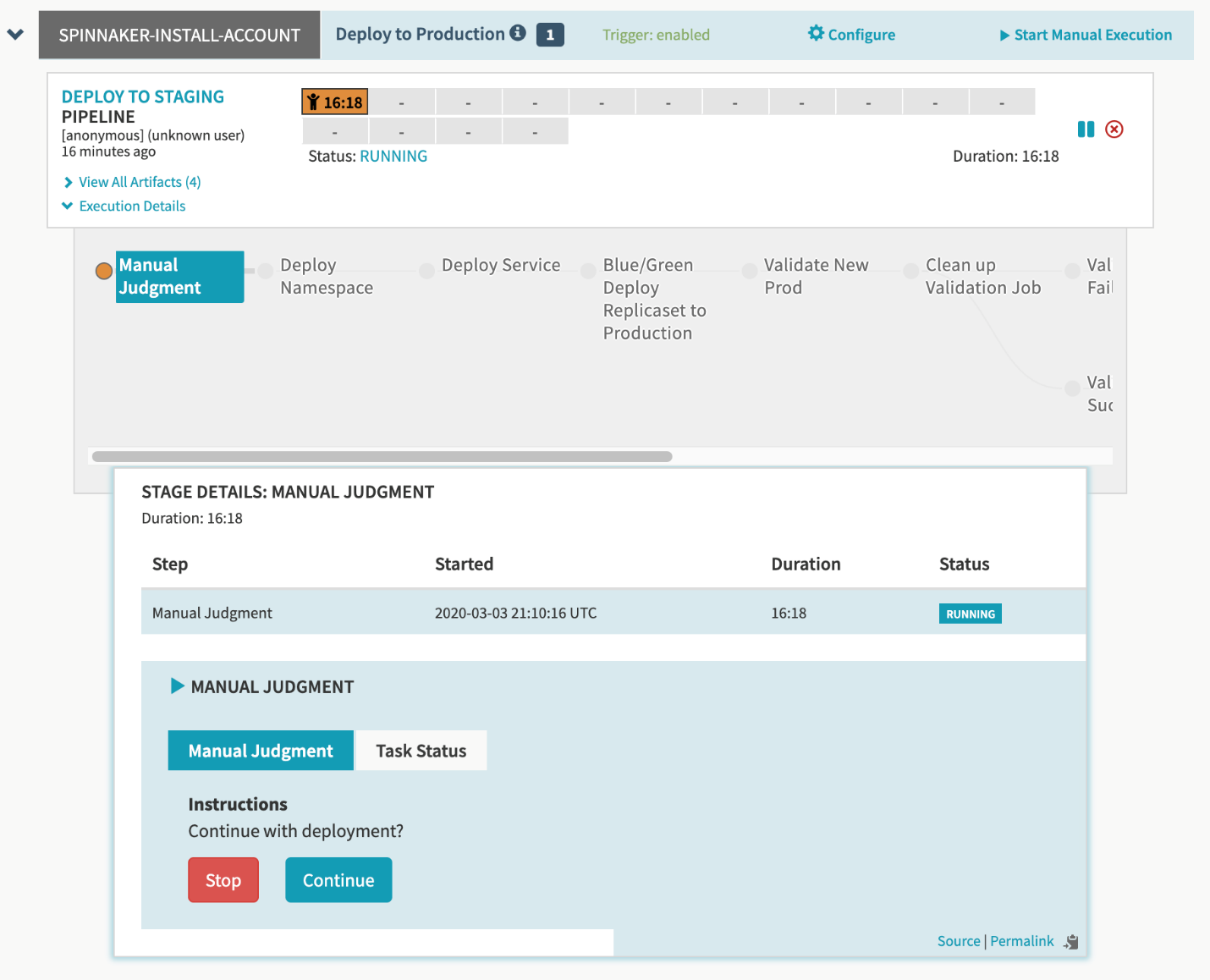
Click **Execution Details** on the **Deploy to Staging** pipeline.

Look for Status **SUCCEEDED**.



Review the Deploy to Production pipeline

1. In Pipelines, close **Execution Details** on **Deploy to Staging** and click **Execution Details** on **the Deploy to Production** pipeline.



Notice the production pipeline is paused. It's set to await a **Manual Judgement** before proceeding.

1. Allow the production deployment to proceed.

Below the question **Continue with deployment?**, click **Continue**.

This unblocks the pipeline. You can follow the pipeline executing while it has status of **RUNNING**.

1. Click a stage to see details about it.

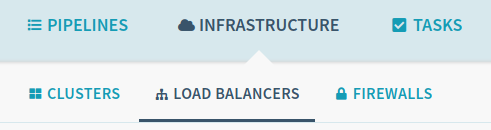
Click **Deploy Namespace** to see up to date status of the stage. Click **Task Status** and **Artifact Status** too.

1. Scroll the pipeline right to see all possible stages.

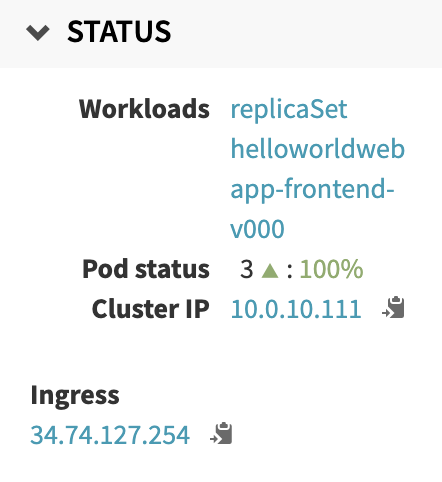
Notice **Validation Succeeded** becomes green when your pipeline is done.

View the running production application

1. Click **Infrastructure** > **Load Balancers** at the top of the Spinnaker UI.



1. Under **service-helloworldwebapp-service**, click **HELLOWORLDWEBAPP-PROD**.
2. On the right under **STATUS**, copy the **Ingress** ip-address by clicking **Copy Ingress IP to clipboard**.



The Ingress IP link from the Spinnaker UI uses HTTP.

1. Paste the address into your browser to view the production version of the application. You may need to make sure it is an HTTP address.



You have now verified building, testing, and deploying your application.

Click *Check my progress* to verify the objective.

Viewing your pipelines

Check my progress

**Triggering your pipeline from code changes**

In this section, you test the pipeline end to end by making a code change, and watching the pipeline run in response.

Again, when you push a change, you trigger Cloud Build to build a new Docker image and push it to Container Registry. Spinnaker detects that a new image was created and triggers a pipeline to deploy the image to staging, run tests, and roll out the same image to all pods in the deployment.

1. Change the color of the application background to blue.

In Cloud Shell, from spinnaker-for-gcp-helloworldwebapp, make a small code change.

sed -i 's/green/blue/g' ./src/main.go

git diff

git commit -a -m "Change to blue."

1. Push the change which will trigger a build.
2. git push origin master
3. Check Cloud Build in Cloud Console for the triggered build.

Click **Navigation menu**, then in the **Tools** section, click **Cloud Build** > **Dashboard**.

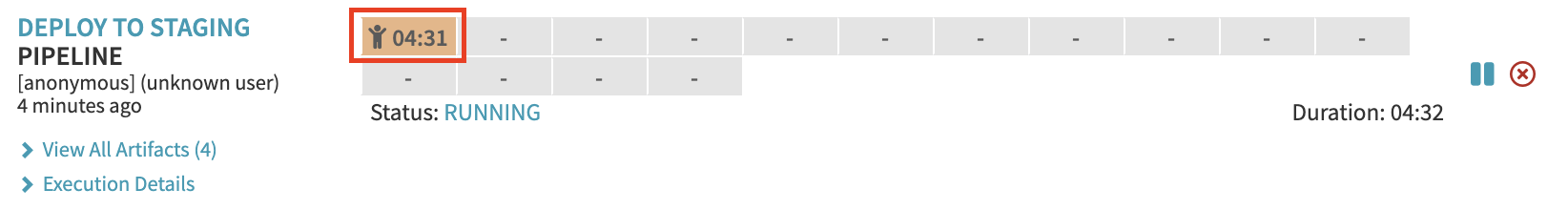
Look for a build with status **Running**.

1. Stay on this page and **wait** for the build to complete before going on to the next step.

After ~1 minute you should see **Successful**.

1. Return to the Spinnaker UI and click **Pipelines** to watch the staging pipeline start to deploy the image.
2. Approve the paused production pipeline.

Hover over the yellow stage indicator then click **Continue**.



1. Wait for all stages to complete.

Continue to the next step once the **Validation Succeeded** stage is completed.

1. Verify your change was deployed.

Refresh your production application page and notice the background is now **blue**.



Click *Check my progress* to verify the objective.

Triggering your pipeline from code changes

Check my progress

**Rolling back a change by reverting your previous commit**

1. Roll back the latest change.
2. git revert --no-edit HEAD

Reverting creates a new commit undoing the latest commit.

1. Push the change which will trigger a build.
2. git push origin master
3. As before, wait a few minutes for the build to complete.
4. As before, use the Spinnaker UI to watch the staging pipeline complete.
5. As before, confirm the prompt to deploy to production.
6. Wait for the reverted changes to be fully deployed.

Wait for the stage **Scale Down Old Prod** to be completed.

1. Verify the roll back by refreshing the application UI.

Refresh your application UI, open to the Ingress IP.

Look for the green background again.



**Congratulations**

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

**End your lab**