**SUMMER INTERNSHIP PROJECT**

**FINAL REPORT**

**UPES - Continuous Delivery**

**Submitted by**

**Team Members**: Rishabh Saxena, Nimish Panday, Akash Kumar das

***Under the guidance of***

**Mr. Kamalpreet Singh**

Asst. Professor, Department of Systemics

School of Computer Science & Engineering



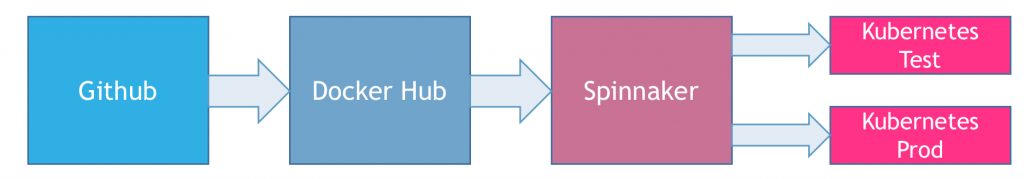
**School of Computer Science and Engineering, UNIVERSITY OF PETROLEUM AND ENERGY STUDIES**

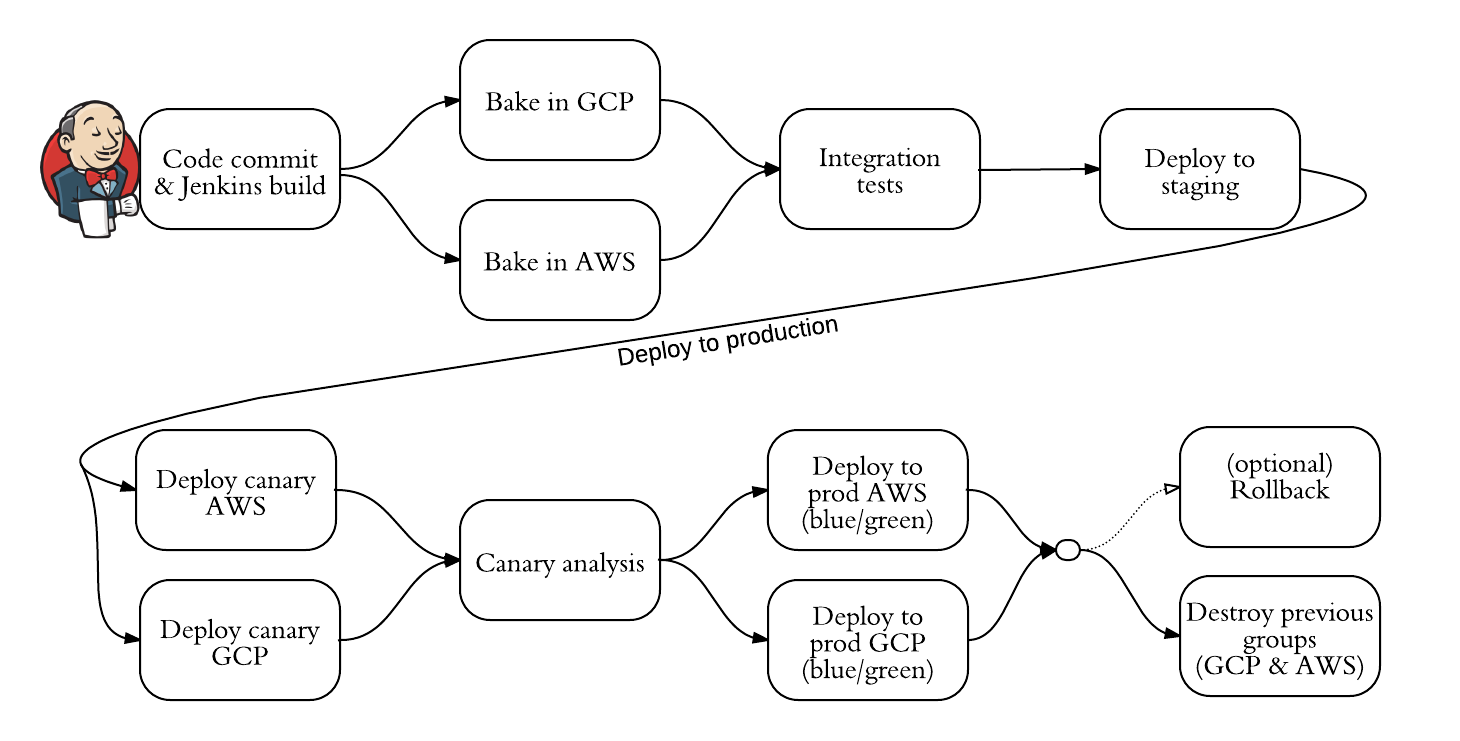
**Dehradun-248007**

**Project Overview:**

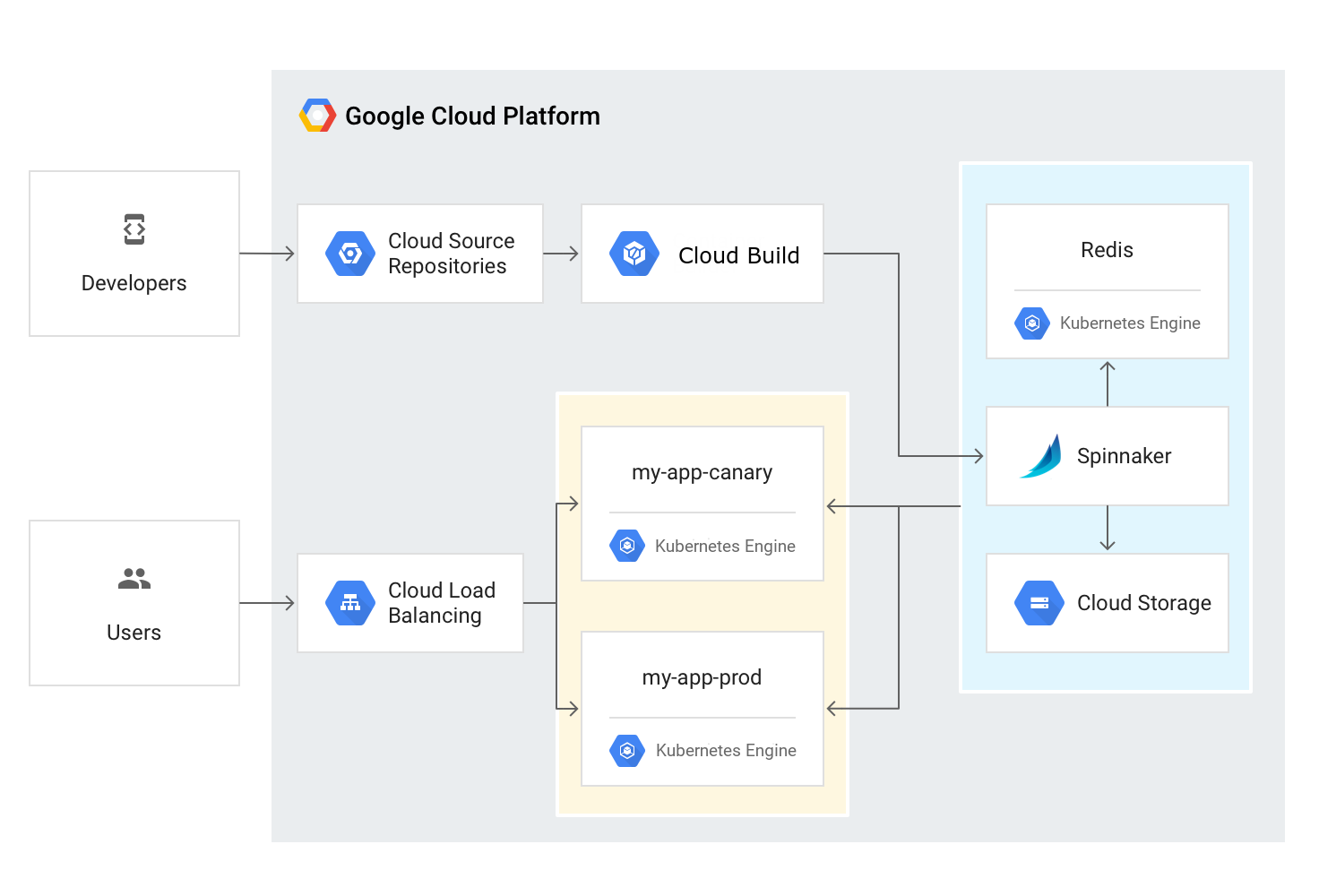
Spinnaker is an open source, multi-cloud continuous delivery platform for releasing software changes, when we change anything in application it is automatically triggered by spinnaker and it checks if the change is applicable then Spinnaker deploys it automatically. We use Spinnakers application management features to view and manage cloud resources. Applications, clusters and server groups are the key concepts Spinnaker uses to describe your services. Load balancers and firewalls describe how your services are exposed to users.

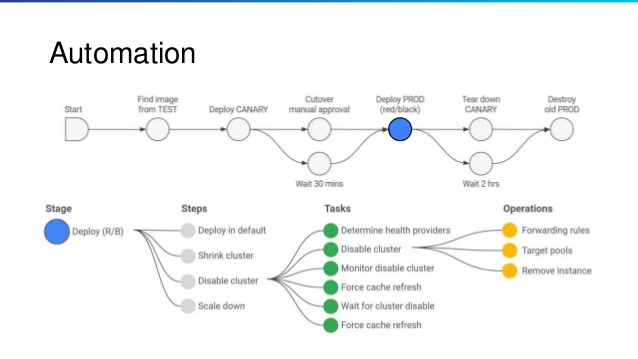
**High Level Design/Overview Diagram:**

****

****

**Detailed Design:**

****

****

**Application Code:**

|  |
| --- |
| <!DOCTYPE html> |
|  |

|  |
| --- |
| <html> |
|  |

|  |
| --- |
| <head> |
|  |

|  |
| --- |
| <meta charset="utf-8"> |
|  |

|  |
| --- |
| <title>Sample Deployment</title> |
|  |

|  |
| --- |
| <style> |
|  |

|  |
| --- |
| body { |
|  |

|  |
| --- |
| color: #ffffff; |
|  |

|  |
| --- |
| background-color: blue; |
|  |

|  |
| --- |
| font-family: Arial, sans-serif; |
|  |

|  |
| --- |
| font-size: 14px; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| h1 { |
|  |

|  |
| --- |
| font-size: 500%; |
|  |

|  |
| --- |
| font-weight: normal; |
|  |

|  |
| --- |
| margin-bottom: 0; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| h2 { |
|  |

|  |
| --- |
| font-size: 200%; |
|  |

|  |
| --- |
| font-weight: normal; |
|  |

|  |
| --- |
| margin-bottom: 0; |
|  |

|  |
| --- |
| } |
|  |

|  |
| --- |
| </style> |
|  |

|  |
| --- |
| </head> |
|  |

|  |
| --- |
| <body> |
|  |

|  |
| --- |
| <div align="center"> |
|  |

|  |
| --- |
| <h1>Welcome to V1 of the web application</h1> |
|  |

|  |
| --- |
| <h2>This application will be deployed on Kubernetes.</h2> |
|  |

|  |
| --- |
| </div> |
|  |

|  |
| --- |
| </body> |
|  |

</html>

**Url of Project Github Repository**: <https://github.com/nimish14>

**Setup Instructions:**

Steps involved in Installing a Spinnaker:

**Install Halyard**

Halyard manages the lifecycle of your Spinnaker deployment, including writing & validating your deployment’s configuration, deploying each of Spinnaker’s micro services, and updating the deployment.

All production-capable deployments of Spinnaker require Halyard in order to install, configure, and update Spinnaker. Though it’s possible to install Spinnaker without Halyard, we don’t recommend it, and if you get stuck we’re just going to tell you to use Halyard.

There are two ways you can install Halyard:

* locally on Debian/Ubuntu or macOS
* on Docker

1. **Get the latest version of Halyard:**

**For Debian/Ubuntu**:

curl –O https://raw.githubusercontent.com/spinnaker/halyard/master/install/debian/InstallHalyard.sh

**For macOS:**

curl -O https://raw.githubusercontent.com/spinnaker/halyard/master/install/macos/InstallHalyard.sh

1. **Install it:**

sudo bash InstallHalyard.sh

If you’re prompted for any information, default answers are usually suitable.

1. **Check whether Halyard was installed properly:**

hal -v

If this command fails, make sure hal is in your $PATH, and check the logs under /var/log/spinnaker/halyard/halyard.log.

1. **Run** . ~/.bashrc to enable command completion.

**Choose a cloud provider**

In Spinnaker, Providers are integrations to the Cloud platforms you deploy your applications to.

In this section, you’ll register credentials for your Cloud platforms. Those credentials are known as Accounts in Spinnaker, and Spinnaker deploys your applications via those accounts.

Choose any one of the cloud provider and follow as your cloud provider instructions but below instructions are for google compute engine.

**Add an account:-**

First, make sure that the provider is enabled:

hal config provider google enable

All that’s required are the following values (we’ve provided defaults for you):

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

SERVICE\_ACCOUNT\_DEST=# see Prerequisites section above

Finally, add your new google account:

hal config provider google account add my-gce-account --project $PROJECT \

--json-path $SERVICE\_ACCOUNT\_DEST

**Choose an environment**

In this step, you tell Halyard in what type of environment to install Spinnaker.

The recommended path is a distributed installation onto a Kubernetes cluster, but all of these methods are supported:

* Distributed installation on Kubernetes

Halyard deploys each of Spinnaker’s microservices separately

* Local installations of Debian packages

Spinnaker is deployed on a single machine. This is ok for smaller Spinnaker deployments, but Spinnaker will be unavailable when it’s being updated.

* Local git installations from github

**Choose Distributed installation:-**

Run the following command, using the $ACCOUNT name you created when you configured the provider:

hal config deploy edit --type distributed --account-name $ACCOUNT

**External Storage:-**

There are so many technologies outside there like Minio, Azure Storage, Google cloud etc.

Below instructions are for minio Technology:-

1. kubectl create -f minio-deployment.yaml
2. Spinnaker.s3.versioning :false
3. Ubuntu Installions;-

echo $MINIO\_SECRET\_KEY | hal config storage s3 edit --endpoint $ENDPOINT \

--access-key-id $MINIO\_ACCESS\_KEY \

--secret-access-key # will be read on STDIN to avoid polluting your

# ~/.bash\_history with a secret

hal config storage edit --type s3

**Deployment of Spinnaker:-**

**Pick a version**

1.List the available versions:

2. hal version list

3. Set the version you want to use:

4. hal config version edit --version $VERSION

**Deploy Spinnaker**

Connect to the Spinnaker UI

1. Run the following command:

2. hal deploy connect

If necessary, set up an SSH tunnel to the host running Halyard.This command automatically forwards ports 9000 (Deck UI) and 8084 (Gate API service).

3. Navigate to localhost:9000.

Now that Spinnaker is deployed and capable managing your cloud provider, you can either visit the Guides pages to learn how to use Spinnaker, or continue with additional configuration, such as your image bakery.

**How to Configure spinnaker if you have selected gcloud as a cloud provider :-**

## 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 application 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-storage-account \
3. --display-name spinnaker-storage-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=$(gcloud iam service-accounts list \
6. --filter="displayName:spinnaker-storage-account" \
7. --format='value(email)')
8. export PROJECT=$(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

## 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 applications](http://kubeapps.com/).

### Install Helm

1. Download and install the helm binary:
2. wget https://storage.googleapis.com/kubernetes-helm/helm-v2.7.2-linux-amd64.tar.gz
3. Unzip the file to your local system:
4. tar zxfv helm-v2.7.2-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, v2.7.2 appears for both client and server.

./helm version

Client: &version.Version{SemVer:"v2.7.2",

GitCommit:"8478fb4fc723885b155c924d1c8c410b7a9444e6",

GitTreeState:"clean"}Server: &version.Version{SemVer:"v2.7.2",

GitCommit:"8478fb4fc723885b155c924d1c8c410b7a9444e6",

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 configuration file:
2. export SA\_JSON=$(cat spinnaker-sa.json)
3. export PROJECT=$(gcloud info --format='value(config.project)')
4. export BUCKET=$PROJECT-spinnaker-config
5. cat > spinnaker-config.yaml <<EOF
6. storageBucket: $BUCKET
7. gcs:
8. enabled: true
9. project: $PROJECT

jsonKey: '$SA\_JSON'

# Disable minio as the default

minio:

enabled: false

# Configure your Docker registries here

accounts:

- name: gcr

address: https://gcr.io

username: \_json\_key

password: '$SA\_JSON'

email: 1234@5678.com

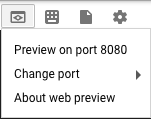
EOF

### Deploy the Spinnaker chart

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 0.3.1

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 "component=deck" \
3. -o jsonpath="{.items[0].metadata.name}")
4. kubectl port-forward --namespace default $DECK\_POD 8080:9000 >> /dev/null &
5. To open the Spinnaker user interface, click **Web Preview** in Cloud Shell and click **Preview on port 8080**.



## Building the Docker image

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

### Create your source code repository

1. Download the source code:
2. wget https://gke-spinnaker.storage.googleapis.com/sample-app.tgz
3. Unpack the source code:
4. tar xzfv sample-app.tgz
5. Change directories to 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)

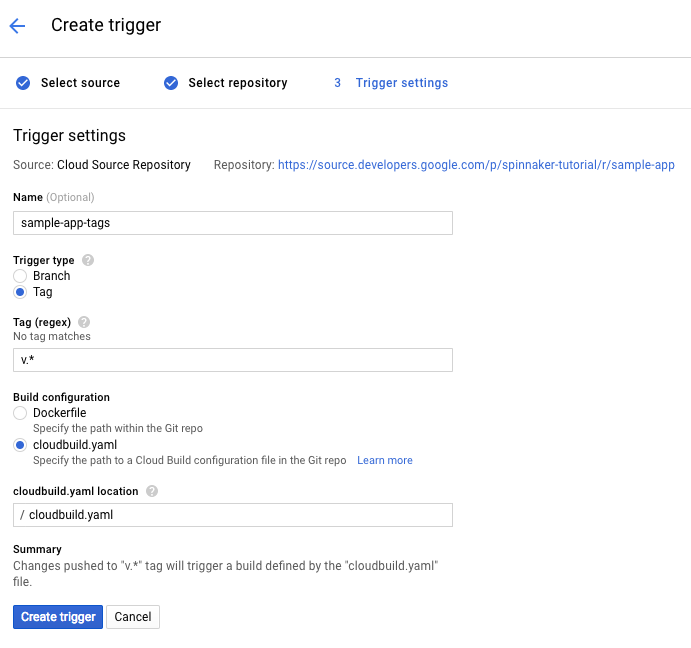
### Configure your build triggers

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, click **Build Triggers** in the Container Registry section.

[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 application as a Docker image to Container Registry.

### 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 **Container Registry**, 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/gcr/builds)

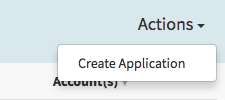
## Configuring your deployment pipelines

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

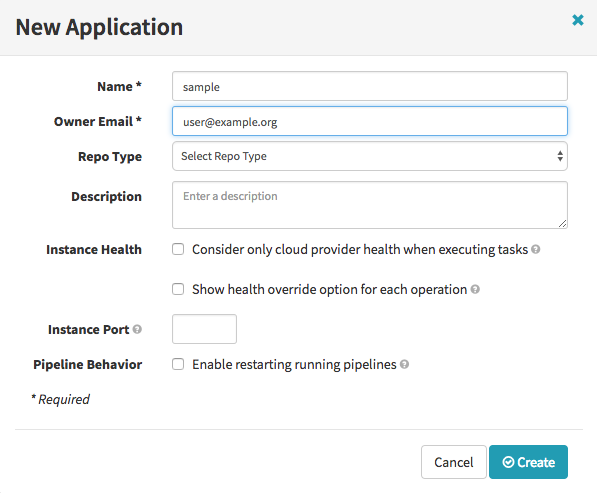
You deploy to a scaled-down environment for integration testing. After the integration tests pass, you must manually approve the changes to deploy the code to production services.

### Create the application

1. In the Spinnaker UI, click **Actions**, then click **Create Application**.



1. In the **New Application** dialog, enter the following fields:
   * **Name:** sample
   * **Owner Email:** [your email address]
2. Click **Create.**



### Create service load balancers

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

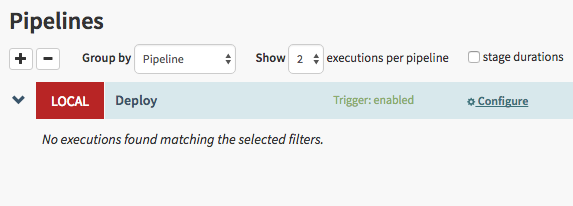
In Cloud Shell, run the following command from the sample-app root directory:

kubectl apply -f k8s/services

### Create the deployment pipeline

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.

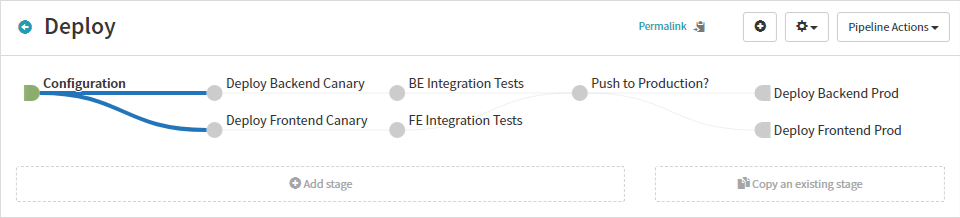
1. In a new tab of Cloud Shell, run the following command in the source code directory to upload an example pipeline to your Spinnaker instance:
2. export PROJECT=$(gcloud info --format='value(config.project)')
3. sed s/PROJECT/$PROJECT/g spinnaker/pipeline-deploy.json | curl -d@- -X \
4. POST --header "Content-Type: application/json" --header \
5. "Accept: /" http://localhost:8080/gate/pipelines
6. In the Spinnaker UI, click **Pipelines** on the top navigation bar.



1. Click **Configure** in the Deploy pipeline.

Configure pipeline

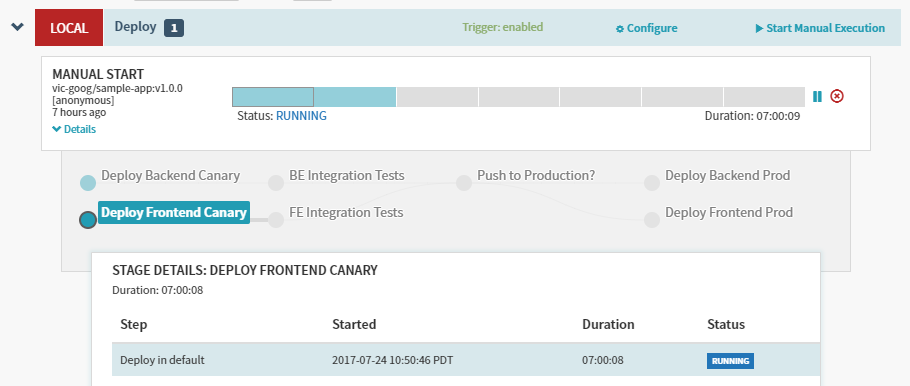
The continuous delivery pipeline configuration appears in the UI:



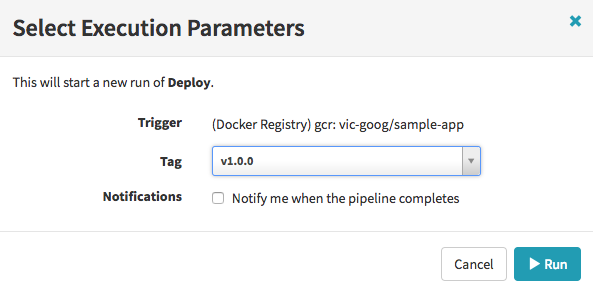
### Run your pipeline manually

The configuration you just created contains a trigger to start the pipeline when you push a new Git tag containing the prefix "v". In this section of the tutorial, you test the pipeline by running it manually. In the next section, you test it by pushing a Git tag and watching the pipeline run automatically.

1. Return to the Pipelines page by clicking **Pipelines**.
2. Click **Start Manual Execution**.



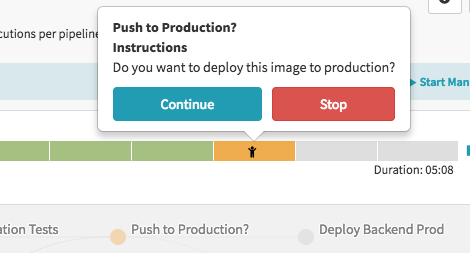
1. Select the v1.0.0 tag from the **Tag** drop-down list, then click **Run**.



1. After the pipeline starts, click **Details** to see more information about the build'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**.

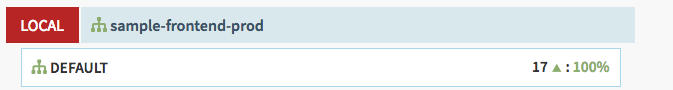


Your rollout continues to the production frontend and backend deployments. It completes after a few minutes.

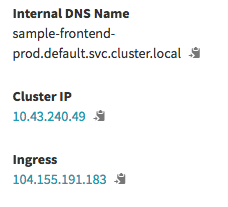
1. To view the app, click **Load Balancers** in the top right of the Spinnaker UI.

load balancers

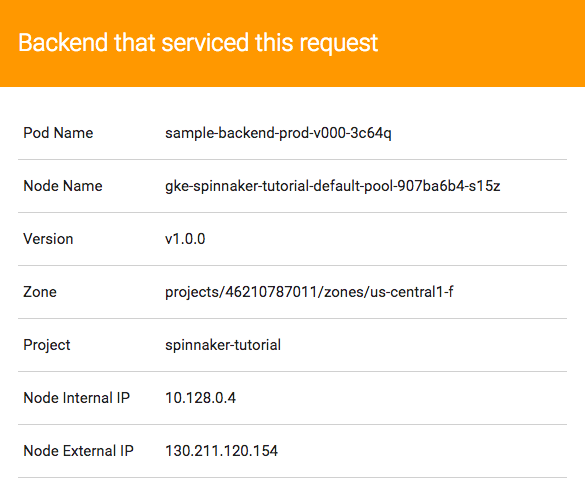
1. Scroll down the list of load balancers and click **Default**, under **sample-frontend-prod**.



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



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

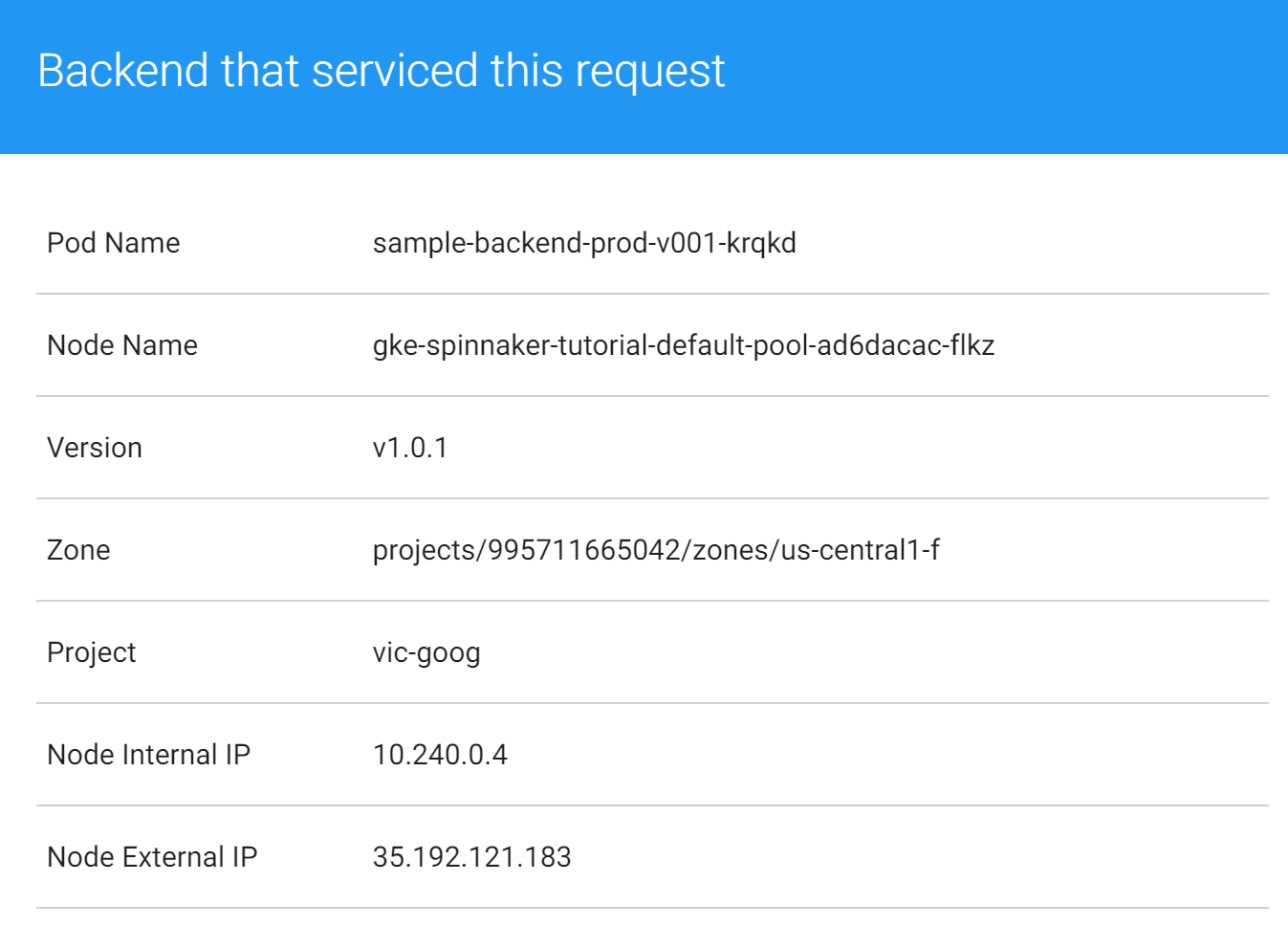


You have now manually triggered the pipeline to build, test, and deploy your application.

## 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 application. Nine of your backends are running the previous version of your application, while only one backend is running the canary. You should see the new, blue version of your application 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 application 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 application 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

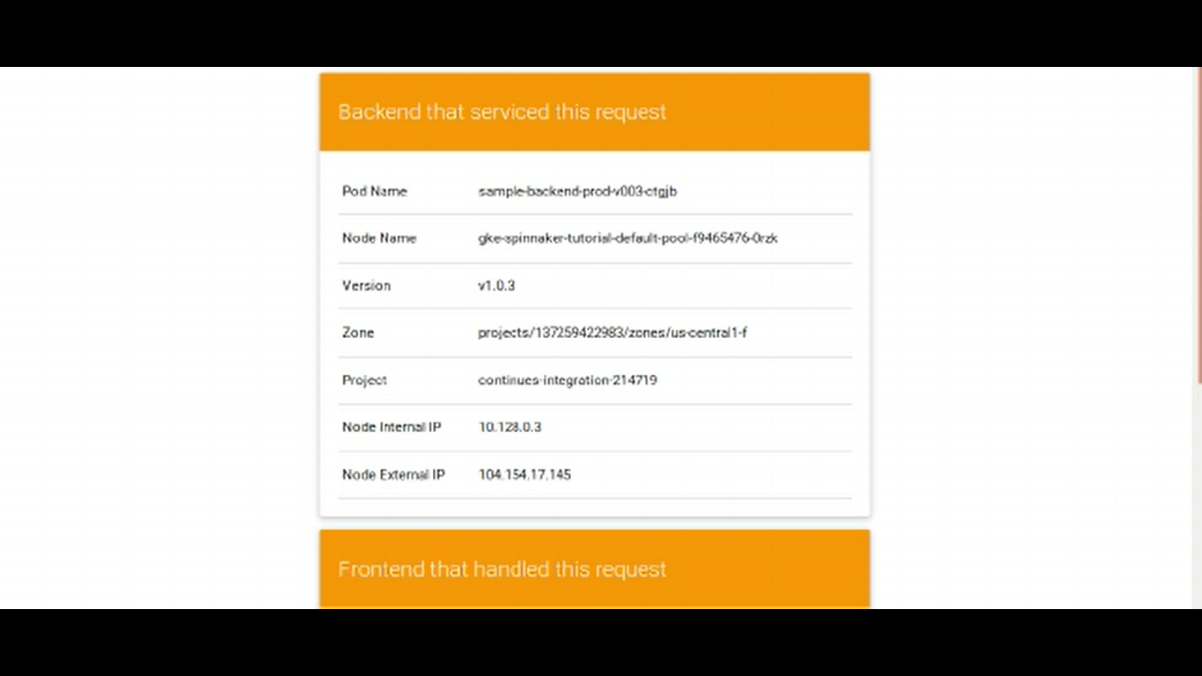
## Cleaning up

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

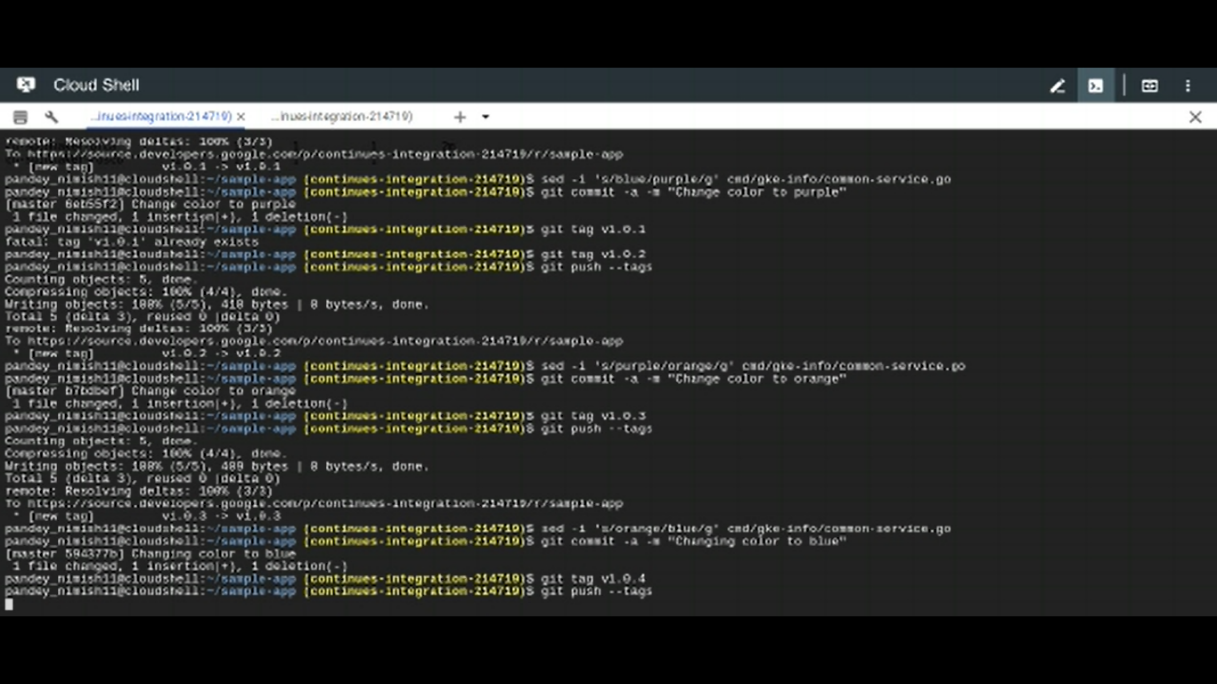
1. Delete the Spinnaker installation:
2. ../helm delete --purge cd
3. Delete the sample app services:
4. kubectl delete -f k8s/services
5. Delete the service account:
6. export SA\_EMAIL=$(gcloud iam service-accounts list \
7. --filter="displayName:spinnaker-storage-account" --format='value(email)')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

**Output Screenshots:-**

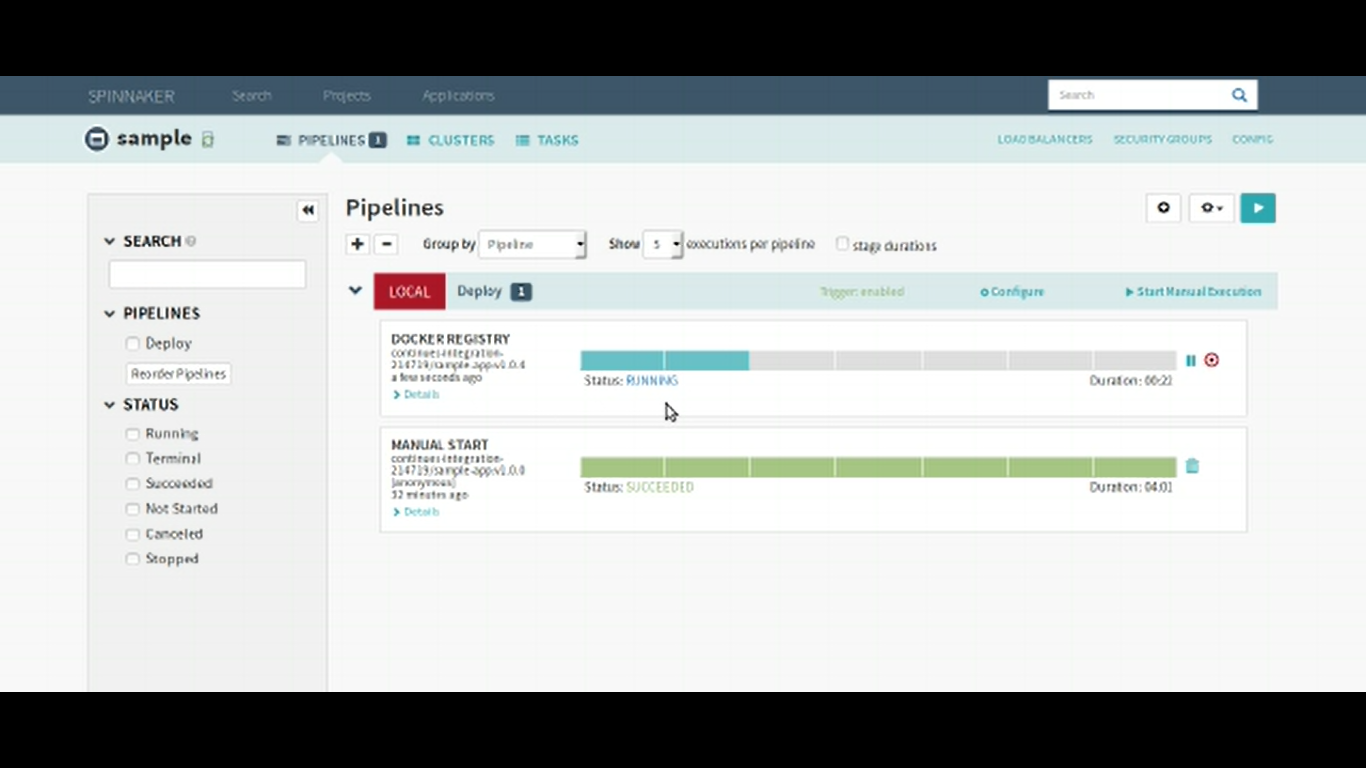
**Screenshot 1: Initial Color of webpage**



**Screenshot 2: Creating an automatic trigger by prefix “v”**



**Screenshot 3: Creation of pipeline**



Screenshot 4: Automatic deployment following the trigger and creation of pipeline

