Deploy to Kubernetes in Google Cloud Challenge Lab

Author: Vedant Kakde | **GitHub Profile:** github.com/vedant-kakde | **LinkedIn Profile:** linkedin.com/in/vedant-kakde/

Task 1: Create a Docker image and store the Dockerfile

1. First of all, you have to run the following command in Cloud Shell.

```
source <(gsutil cat gs://cloud-training/gsp318/marking/setup_marking.sh)
```

   It installs the marking scripts, which use to check your progress.

2. Then, run the commands below to clone the valkyrie-app source code repository to the Cloud Shell. (*Remember to **replace** `YOUR_PROJECT_ID` with your Project ID*)

```
export PROJECT=$YOUR_PROJECT_ID

gcloud source repos clone valkyrie-app --project=$PROJECT
```

3. Create a `Dockerfile` under the `valkyrie-app` directory and add the configuration to the file. Copy the given codes from the lab page to the following snippet, and then run the commands in the Cloud Shell.

```
cd valkyrie-app

cat > Dockerfile <<EOF

// COPY TO HERE

EOF
```

4. Build the image with the following command:

```
docker build -t valkyrie-app:v0.0.1 .
```

5. Run `docker images` to look at the images you built.

Before clicking **Check my progress** on the lab page, don't forget to run the following commands to execute the marking script:

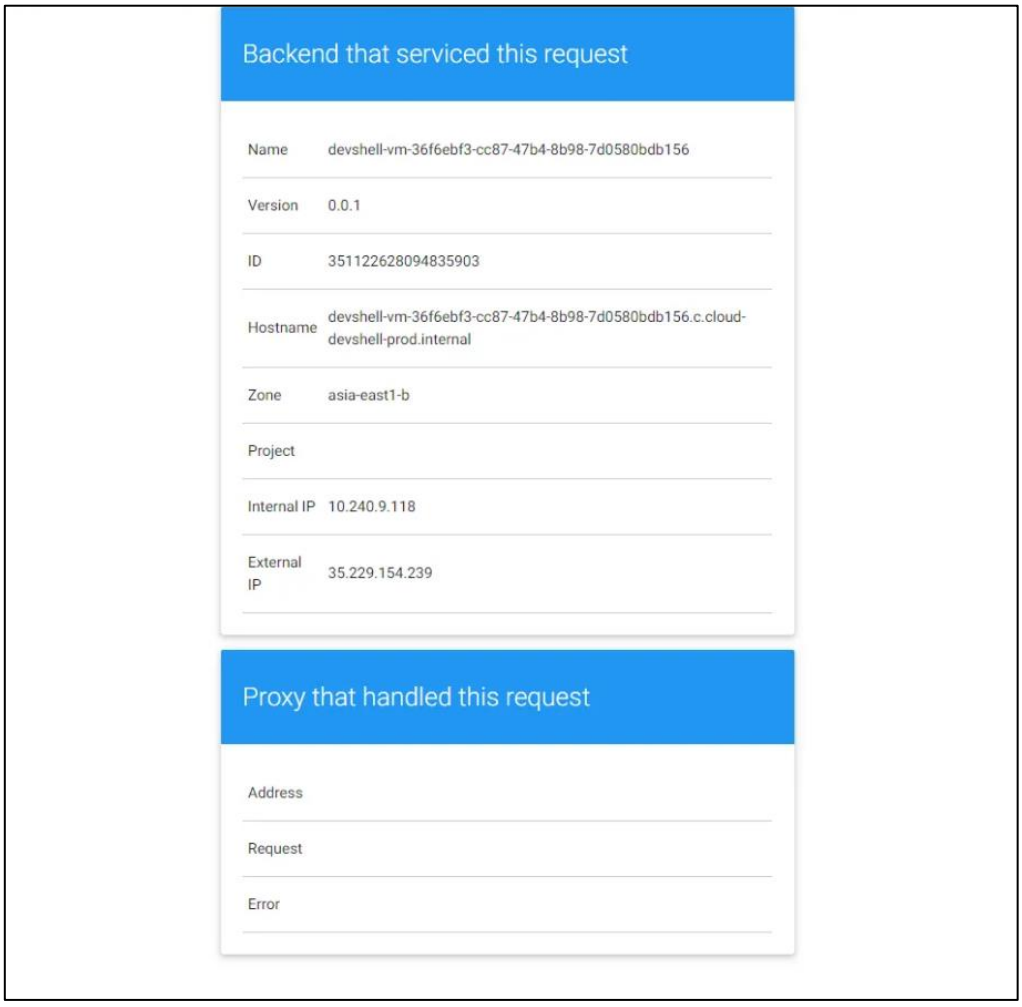
```
cd ~/marking

./step1.sh
```

Task 2: Test the created Docker image

The lab instruction requires you to run the docker image built in Task 1 and show the running application by **Web Preview** on port 8080. Based on the requirements, the docker command will be:

```
docker run -p 8080:8080 --name valkyrie-app valkyrie-app:v0.0.1 &
```

1. In the Cloud Shell, go back to the `valkyrie-app` directory, and run the above command.
2. Click **Web Preview** to see the running app.



After that, open a new Cloud Shell to run the `step2.sh` marking script.

```
cd ~/marking

./step2.sh
```

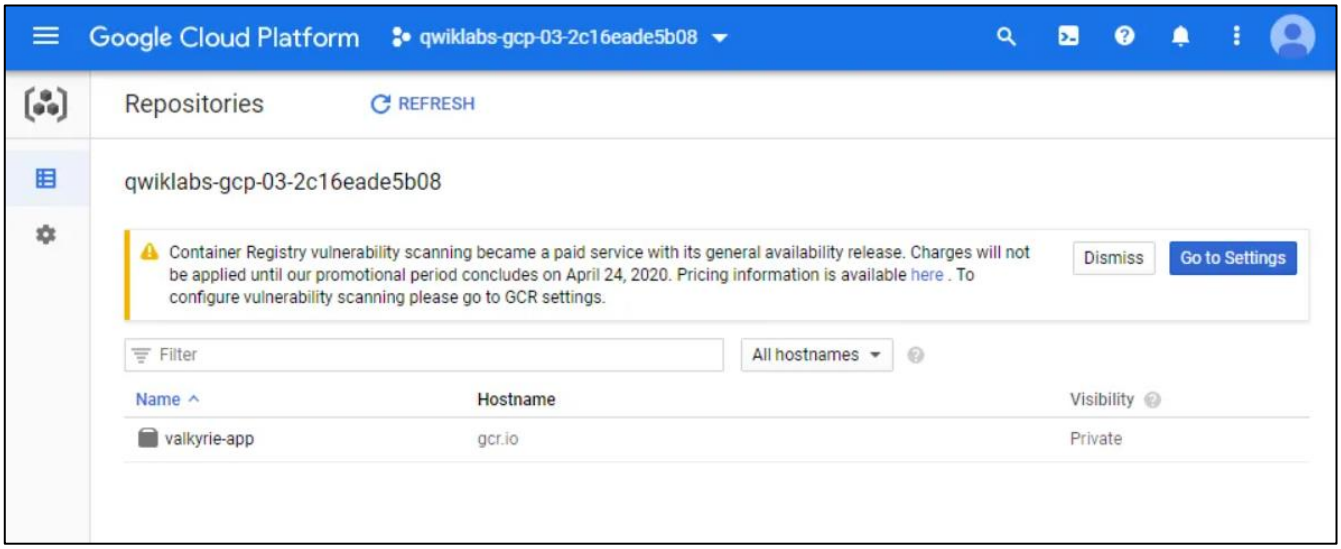
Task 3: Push the Docker image in the Container Repository

In this task, you will push the Docker image valkyrie-app:v0.0.1 into the Container Registry with a tag `gcr.io/YOUR_PROJECT_ID/valkyrie-app:v0.0.1`.

Thus, you should format the docker commands as below.

```
docker tag valkyrie-app:v0.0.1 gcr.io/$PROJECT/valkyrie-app:v0.0.1

docker images

docker push gcr.io/$PROJECT/valkyrie-app:v0.0.1
```

After pushing the container, the `valkyrie-app` repository will appear in the Cloud Console as shown in the image below.



Task 4: Create and expose a deployment in Kubernetes

1. In the Cloud Shell, go to the `valkyrie-app/k8s` subdirectory.
2. Get authentication credentials for the cluster

```
3. gcloud container clusters get-credentials valkyrie-dev --region us-east
```

4. Use a text editor to modify `deployment.yaml` and replace `IMAGE_HERE` with `gcr.io/YOUR_PROJECT_ID/valkyrie-app:v0.0.1`
5. Use `kubectl create -f <filename>` command to deploy **deployment.yaml** and **service.yaml**

Task 5: Update the deployment with a new version of valkyrie-app

Step 5-1 Increase the replicas from 1 to 3

Step 5-2 Update the deployment with a new version of valkyrie-app

1. Go back to the `valkyrie-app` directory in the Cloud Shell.
2. Merge the branch called `kurt-dev` into master using the following `git` command:

```
git merge origin/kurt-dev
```

3. Build and push the new version with tagged v0.0.2:

```
docker build -t valkyrie-app:v0.0.2 .

docker tag valkyrie-app:v0.0.2 gcr.io/$PROJECT/valkyrie-app:v0.0.2

docker images

docker push gcr.io/$PROJECT/valkyrie-app:v0.0.2
```

4. Trigger a rolling update by running the following command:

```
kubectl edit deployment valkyrie-dev
```

   Change the image tag from `v0.0.1` to `v0.0.2`. then save and exit.

Task 6: Create a pipeline in Jenkins to deploy your app

In this task, you will need to:

1. Connect to Jenkins
2. Adding your service account credentials
3. Creating a Jenkins job
4. Modifying the pipeline definition
5. Modify the site
6. Kick off Deployment

6.1 To connect the Jenkines

1. Get the password with the following command:

```
printf $(kubectl get secret cd-jenkins -o jsonpath="{.data.jenkins-admin-password}" |
base64 --decode);echo
```

2. Connect to the Jenkins console using the commands below:

```
export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/component=jenkins-master" -l "app.kubernetes.io/instance=cd" -o
jsonpath="{.items[0].metadata.name}")
```

```
kubectl port-forward $POD_NAME 8080:8080 >> /dev/null &
```

If there is another running container, use the `docker` commands below to kill it:

```
docker ps


docker container kill $(docker ps -q)
```

3. Click on the **Web Preview** button in cloud shell, then click “Preview on port 8080” to connect to the Jenkins console.
   Username: admin

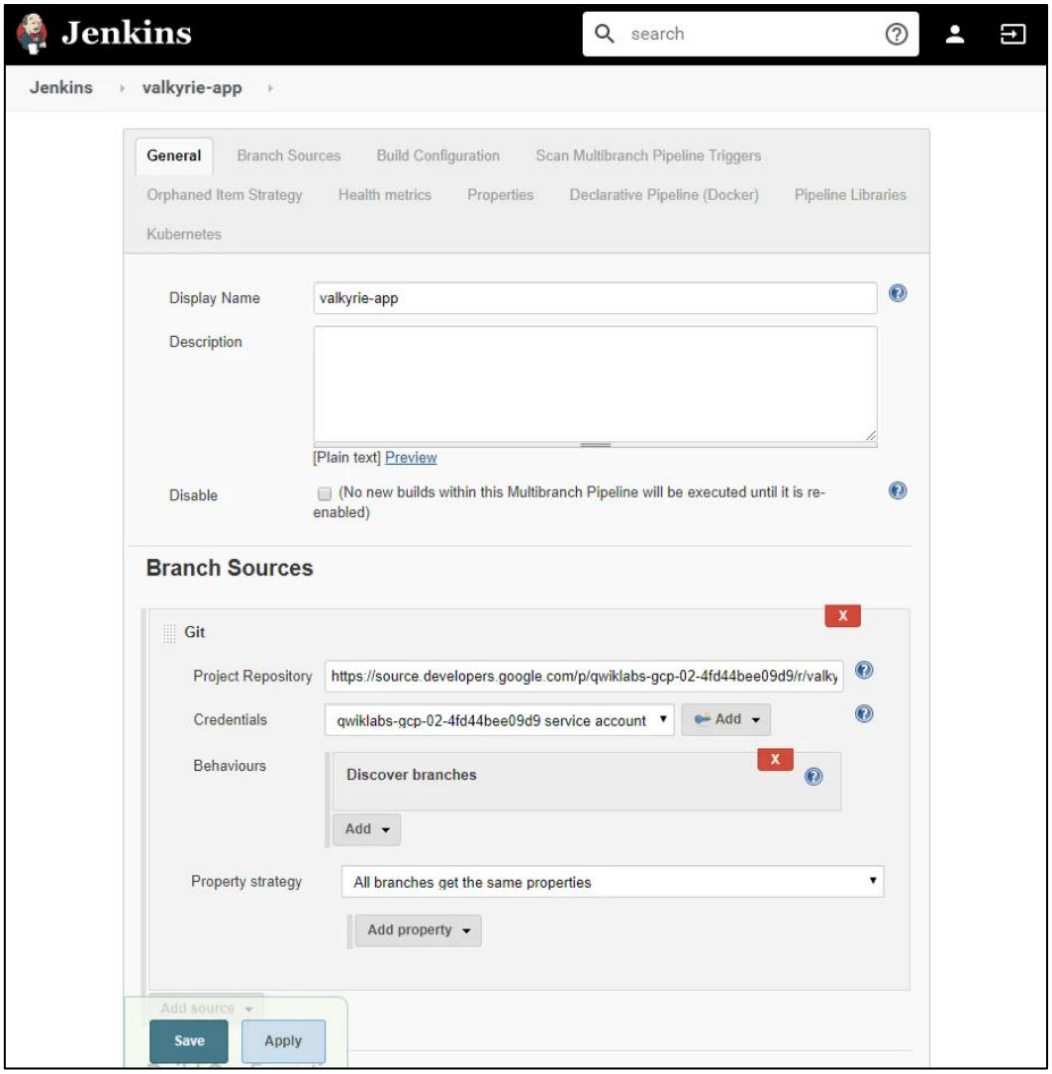
6.2 Adding your service account credentials

1. In the Jenkins user interface, click **Credentials** in the left navigation.
2. Click **Jenkins**
3. Click **Global credentials (unrestricted)**.
4. Click **Add Credentials** in the left navigation.
5. Select **Google Service Account from metadata** from the **Kind** drop-down and click **OK**.

6.3 Creating the Jenkins job

Create a pipeline job that points to your */master branch on your source code.

1. Click **Jenkins** > **New Item** in the left navigation:
2. Name the project **valkyrie-app**, then choose the **Multibranch Pipeline** option and click **OK**.
3. On the next page, in the Branch Sources section, click **Add Source** and select **git**.
4. Paste the HTTPS clone URL of your sample-app repo in Cloud Source Repositories `https://source.developers.google.com/p/YOUR_PROJECT_ID/r/valkyrie-app` into the Project Repository field. Remember to replace **YOUR_PROJECT_ID** with your GCP Project ID.
5. From the **Credentials** drop-down, select the name of the credentials you created when adding your service account in the previous steps.
6. Under **Scan Multibranch Pipeline Triggers** section, check the **Periodically if not otherwise run** box and set the **Interval** value to 1 minute.
7. Your job configuration should look like this:



6.4 Modifying the pipeline definition

Open `Jenkinsfile` file in a text editor, and replace **YOUR_PROJECT** with your GCP project ID.

6.5 Modify the site

Open `source/html.go` file in a text editor, and change the color of headings from green to orange.

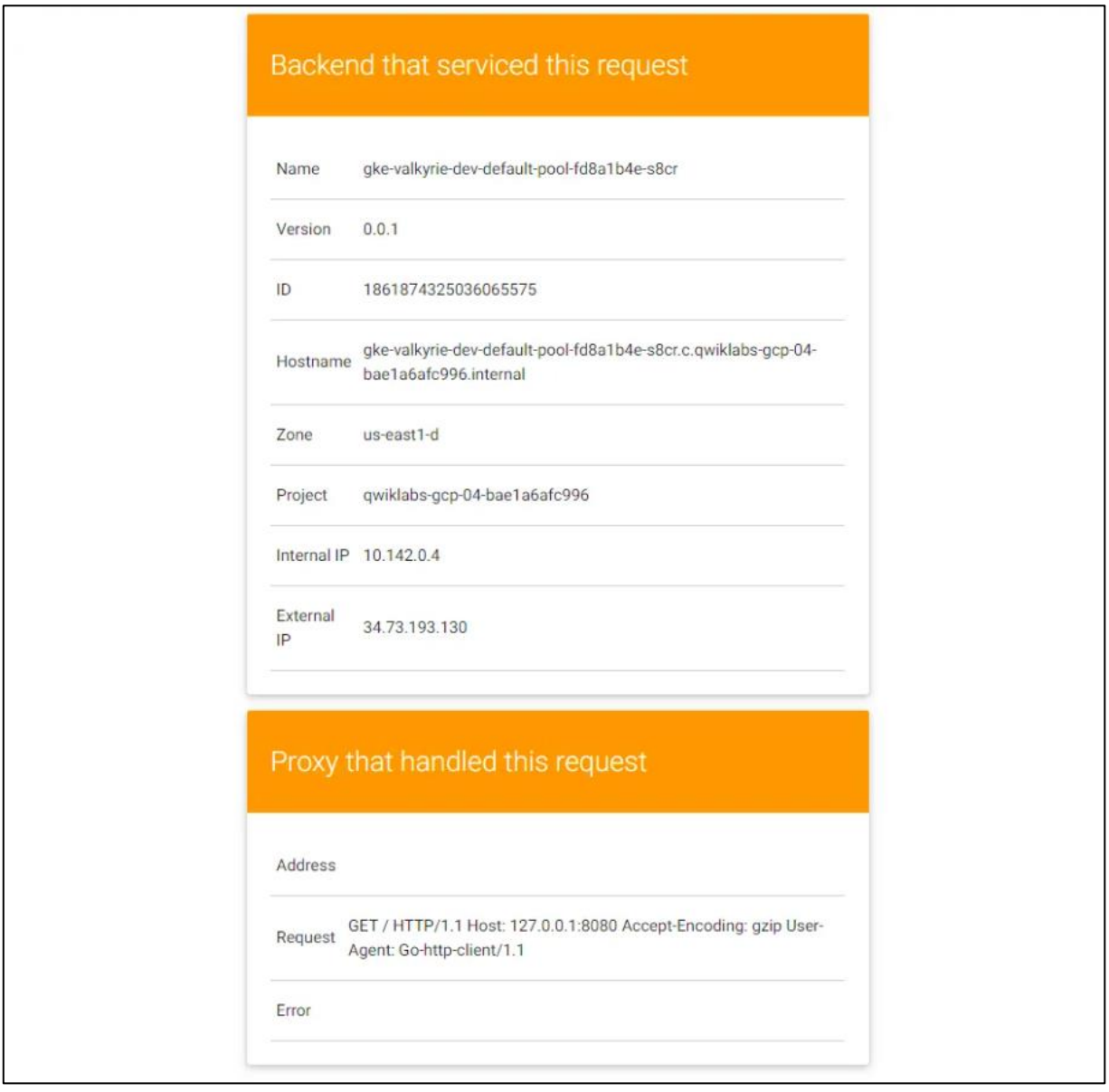
6.6 Kick off Deployment

Commit and push the changes:

```
git config --global user.email $PROJECT

git config --global user.name $PROJECT



git add *

git commit -m 'green to orange'
```

```
git push origin master
```

Finally, manually trigger the build in the Jenkins console



Congratulations! You completed this challenge lab.