Assignment – Google Cloud Platform

By : Karan Walia

**User Story 1 : Creation of Infrastructure as a Code**

Tools used : Terraform, Visual Studio Code

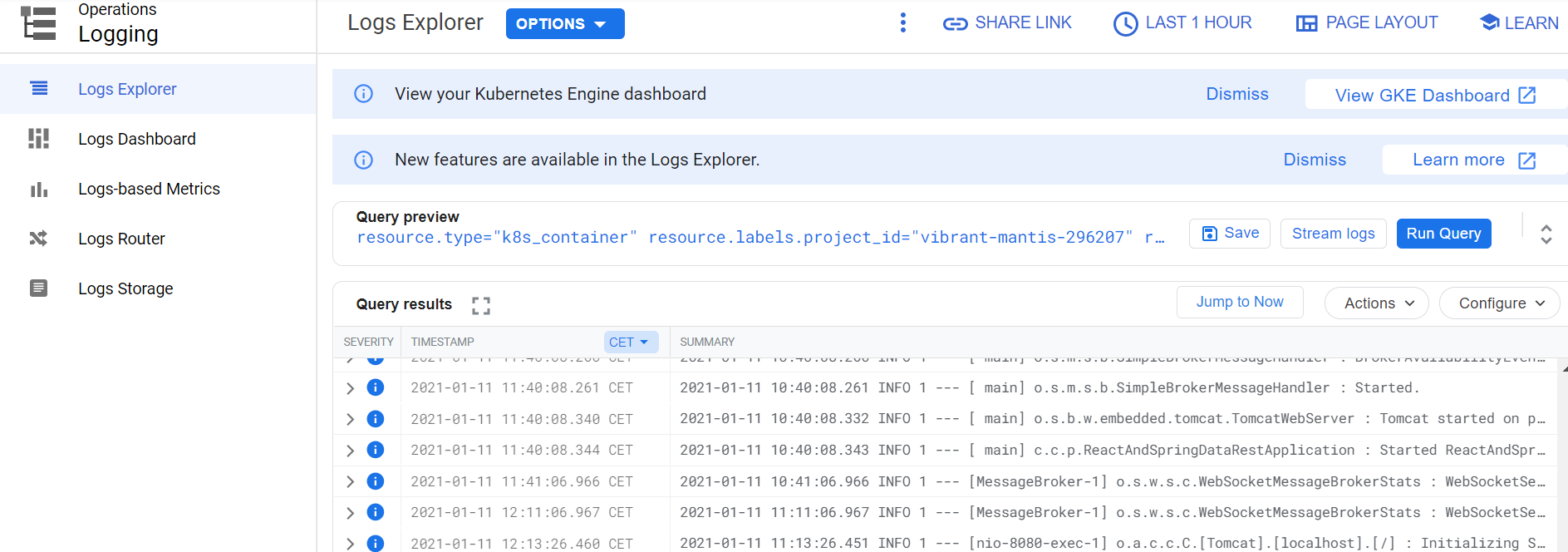
Terraform has been used to create the Kubernetes cluster which will be deployed on the Google cloud platform. Terraform has been used because its platform-agnostic and can be easily integrated with multiple cloud providers. The code can be easily validated and maintained.

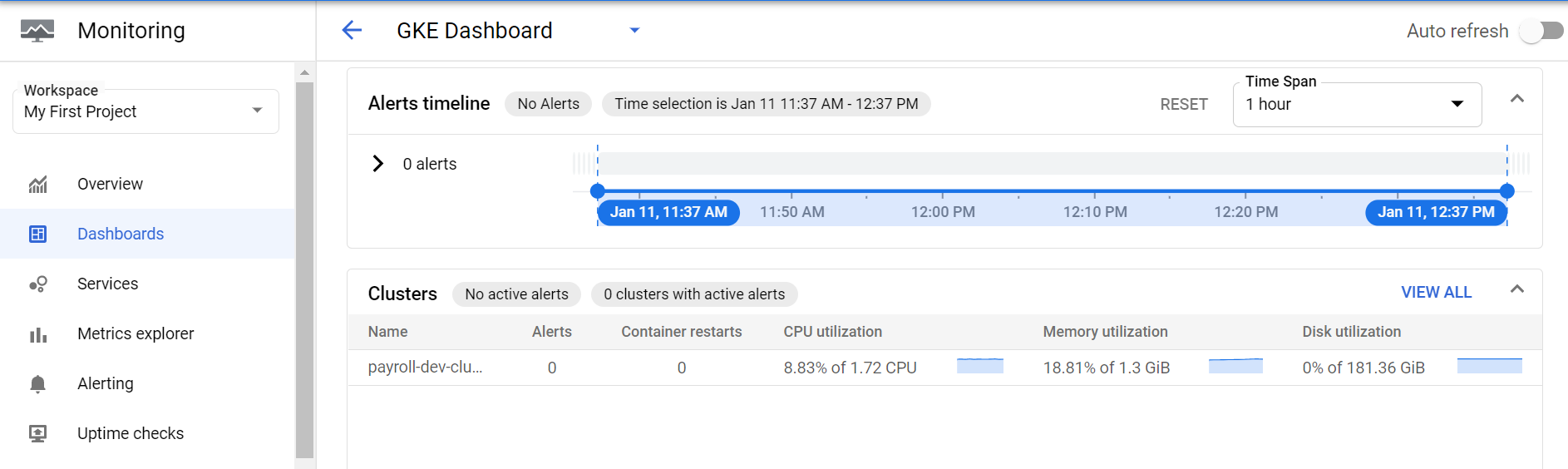
The application will be deployed in the GKE as Kubernetes provides the automation in deploying, scaling and managing the application on clusters. Also many tasks like networking, storage, monitoring and logs are automatically handled using this service.

All the Terraform code files are also shared, which are :

1. Provider file : This file contains the providers which can be then installed by the terraform and used in the project
2. Main file : This file contains all the resources and their properties.
3. Variables file : This file contains all the variables
4. dev and test property file : These files contains the environment specific values for the cloud resources.

Monitoring and logging is taken care automatically in the Kubernetes clusters. We can also set the alerts by creating the different policies.





Autoscaling can be easily achieved with the help of GKE. The cluster autoscaler works on the outer Infrastructure layer on the Nodes whereas HPA(Horizontal pod autoscaler)/VPA(Verical pod autosclaer) works at the Pods. It provides the ability to scale resources in reaction to varying traffic.

The autoscaling is enabled On and seen in the Terraform file, The min count and max count has been given. Also, the policy has been setup on the basis of CPU uitlization.

**User Story 2 : Creation of CICD pipeline**

Tools used : Jenkins, Docker, Helm charts, GIT/GitHub

Jenkins has been used to create the CICD pipeline as it is an open source tool, easy to maintain and plugin for every functionality.

**Docker**

Docker has been used to create the image of the application. The docker file is shared in the attached documents. The commands user are :

FROM openjdk:8-jdk-alpine

ARG JAR\_FILE=target/\*.jar

COPY ${JAR\_FILE} app.jar

EXPOSE 8080

ENTRYPOINT ["java","-jar","/app.jar"]

**GIT/GitHub** – This has been used for the code repository.

The webhook was created and the URL of the Jenkins has been linked for the connection. Also the triggering options were selected.

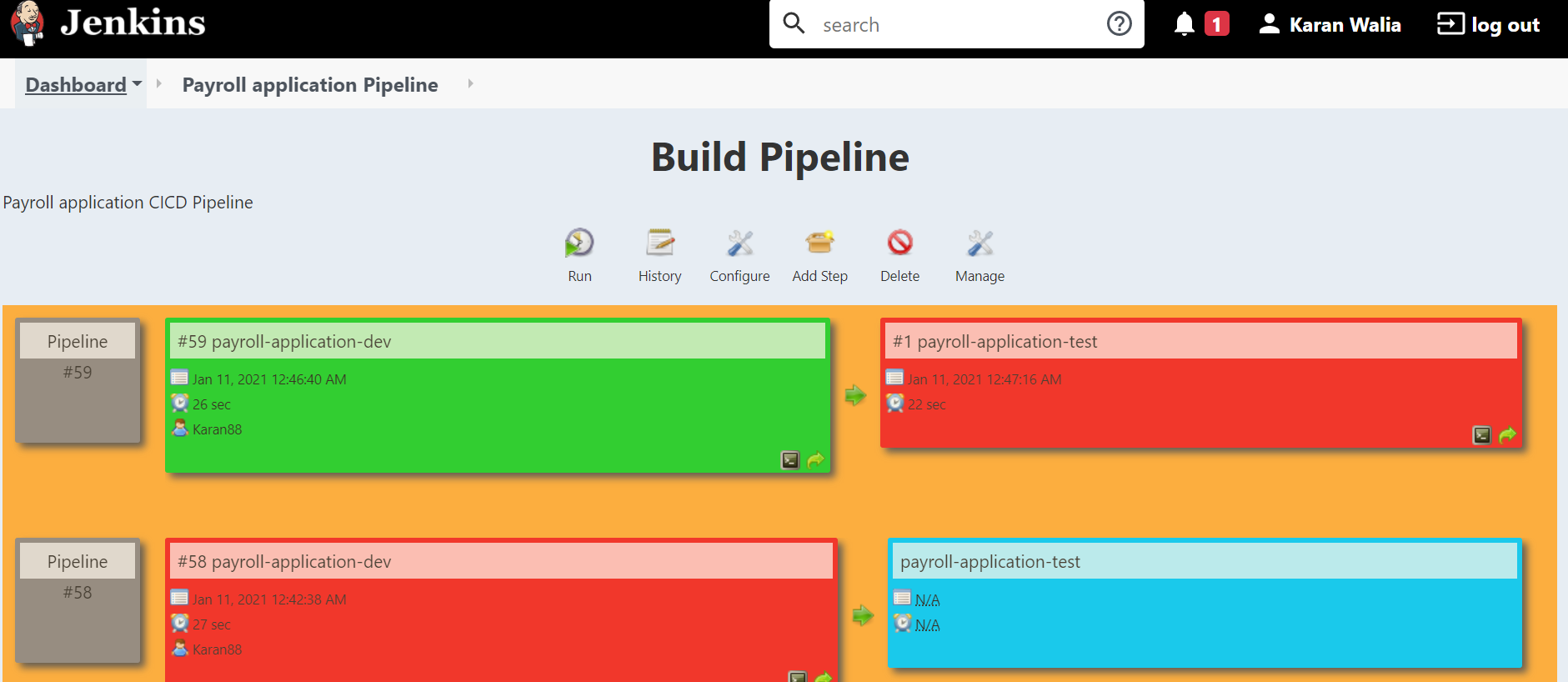
**Helm charts** – The helm charts works best in managing the Kubernetes applications. It is having pre-built configurations and the deployments are fast and smooth.

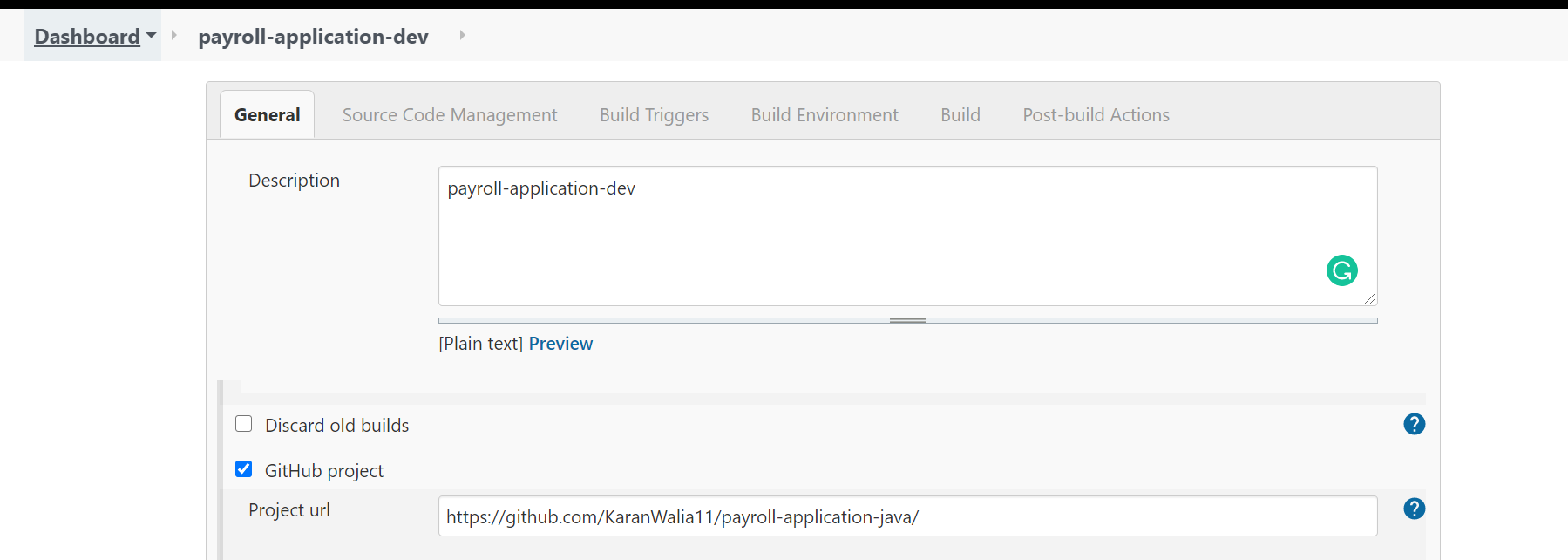
The CICD pipeline contains the 2 stages for development and testing environment as shown below. The

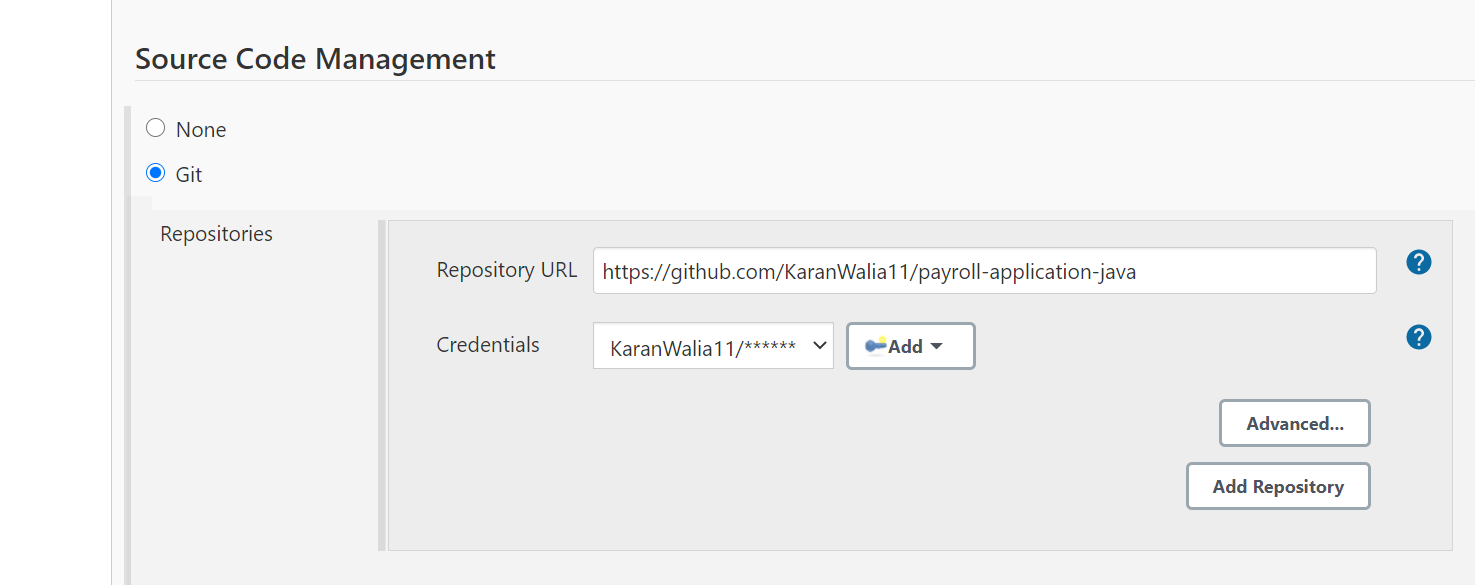
Workflow is like this :

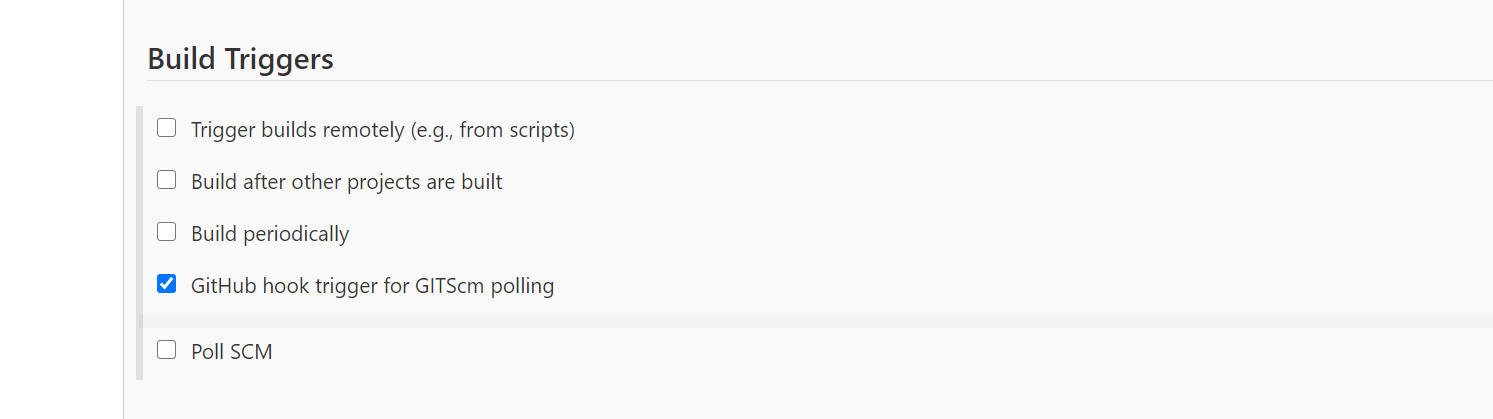
GitHub changes -> Build trigger -> Application Image creation -> Container registery -> Kubernetes Cluster.

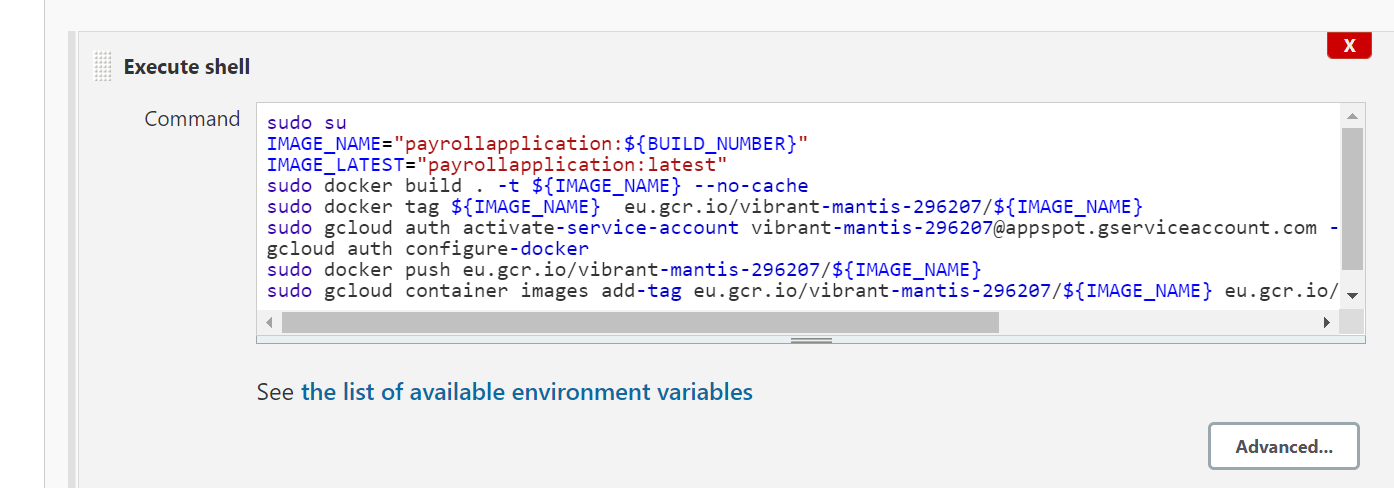
The screenshots of Jenkins configurations are shown below :











**Commands used in the first build are as below :**

sudo su

IMAGE\_NAME="payrollapplication:${BUILD\_NUMBER}"

IMAGE\_LATEST="payrollapplication:latest"

sudo docker build . -t ${IMAGE\_NAME} --no-cache

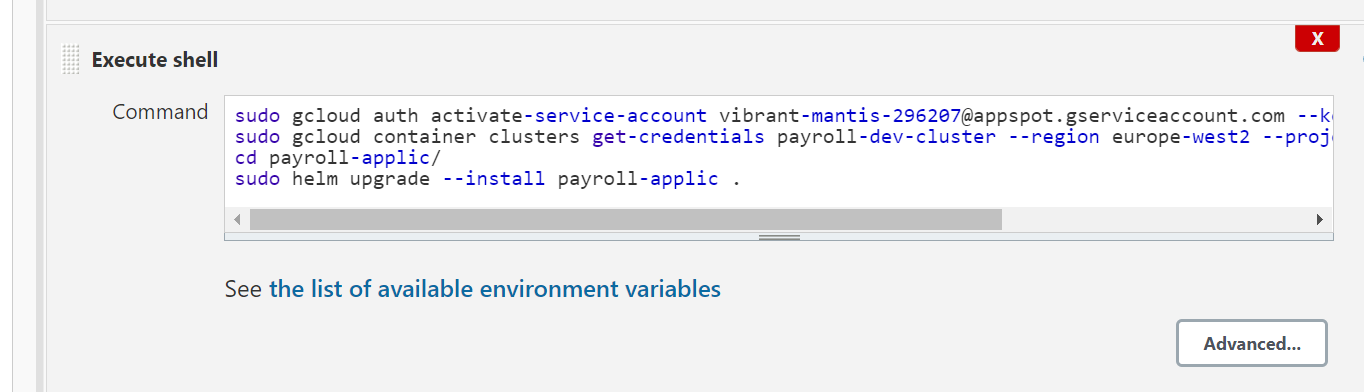
sudo docker tag ${IMAGE\_NAME} eu.gcr.io/vibrant-mantis-296207/${IMAGE\_NAME}

sudo gcloud auth activate-service-account vibrant-mantis-296207@appspot.gserviceaccount.com --key-file=/home/walia\_kapil3/secret.json

gcloud auth configure-docker

sudo docker push eu.gcr.io/vibrant-mantis-296207/${IMAGE\_NAME}

sudo gcloud container images add-tag eu.gcr.io/vibrant-mantis-296207/${IMAGE\_NAME} eu.gcr.io/vibrant-mantis-296207/${IMAGE\_LATEST}



**Commands used in the second build are as below :**

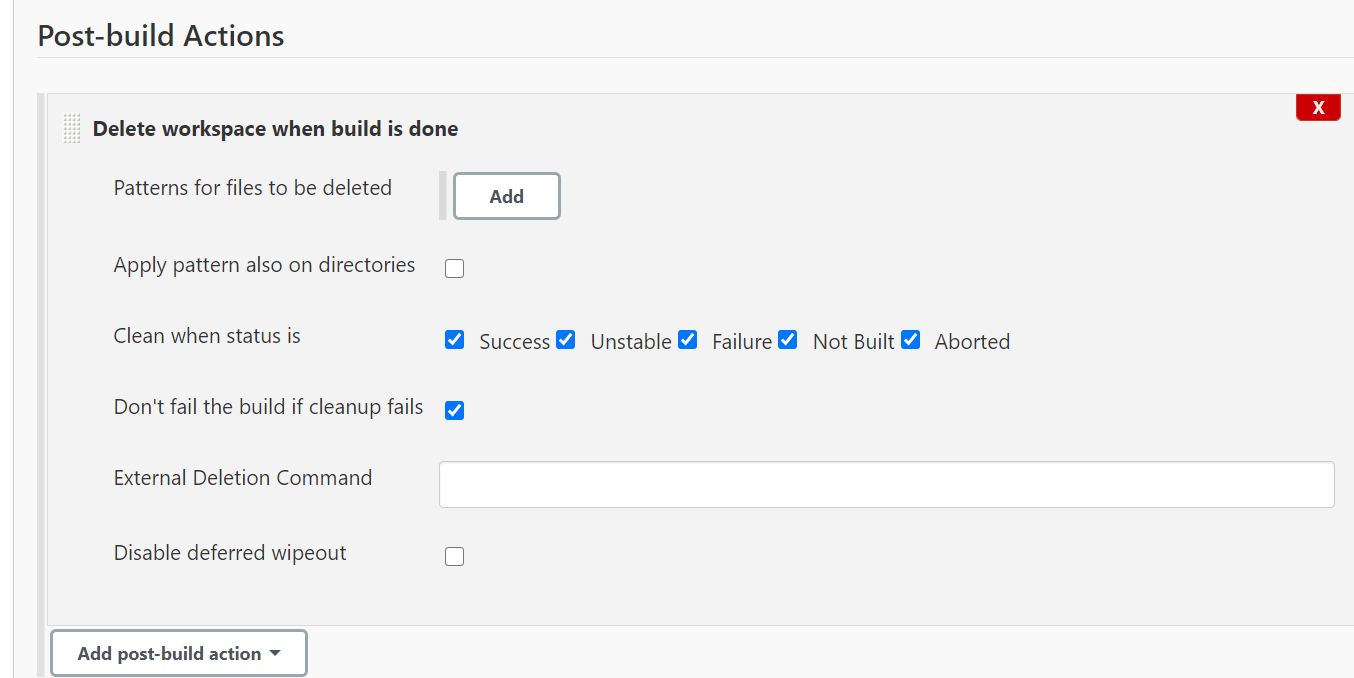
sudo gcloud auth activate-service-account vibrant-mantis-296207@appspot.gserviceaccount.com --key-file=/home/walia\_kapil3/secret.json

sudo gcloud container clusters get-credentials payroll-dev-cluster --region europe-west2 --project vibrant-mantis-296207

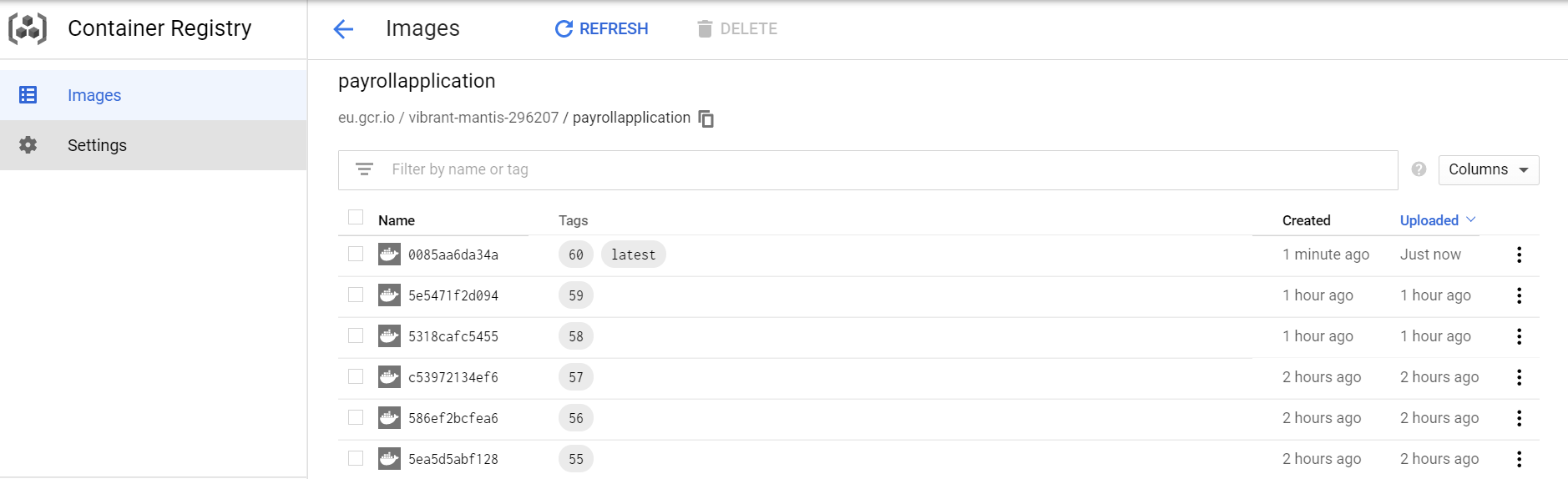
cd payroll-applic/

sudo helm upgrade --install payroll-applic .

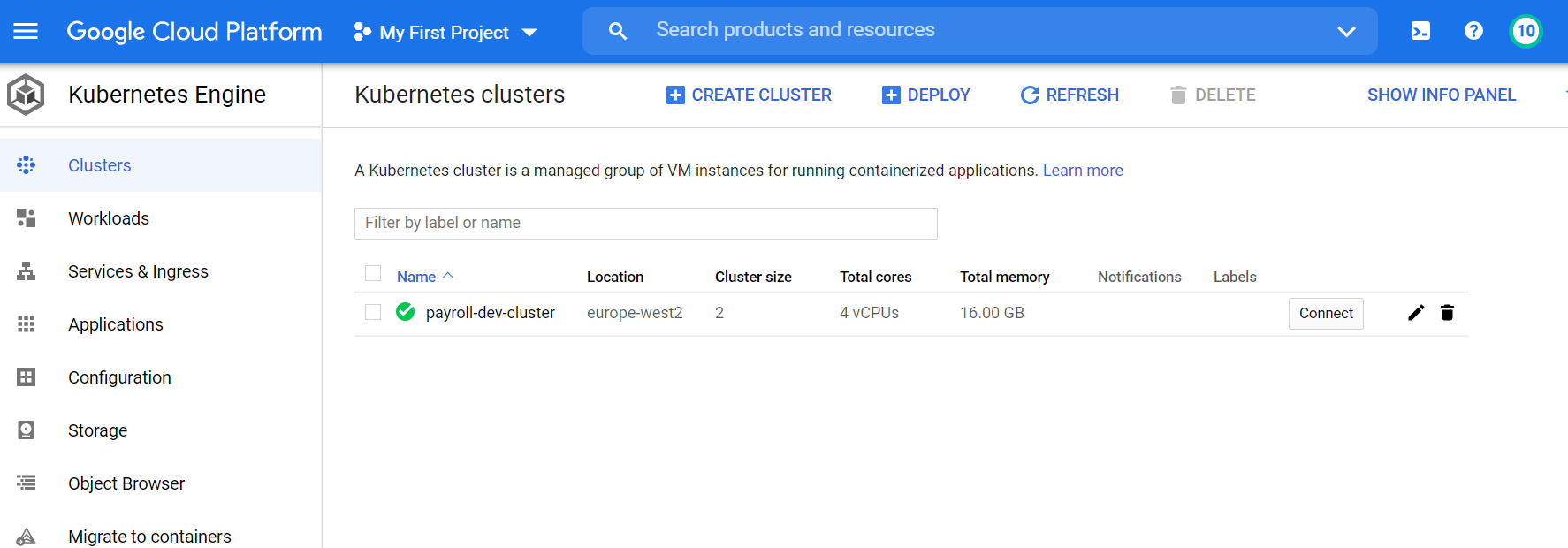
**Configuration for Clean up functionality :**

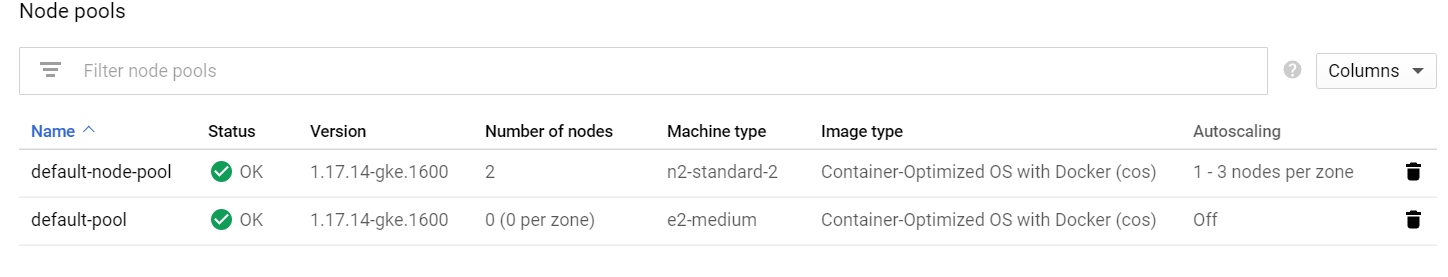


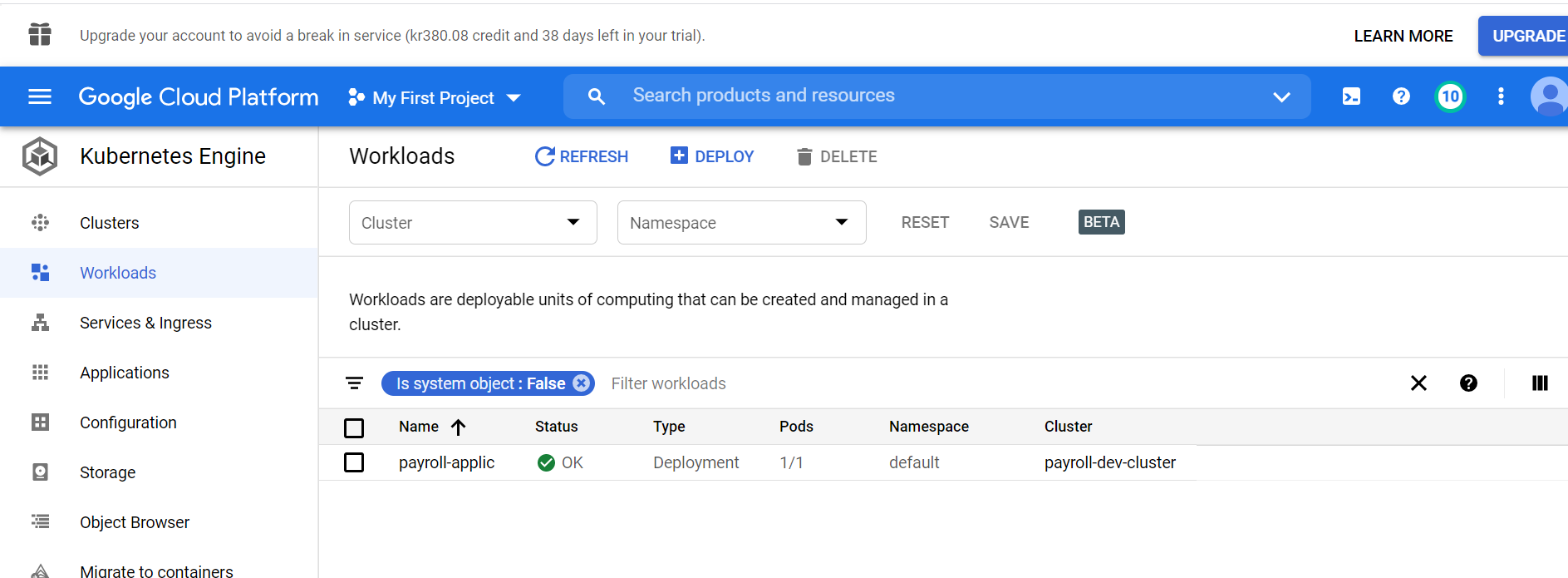
Application images in the Container registery with different versions :



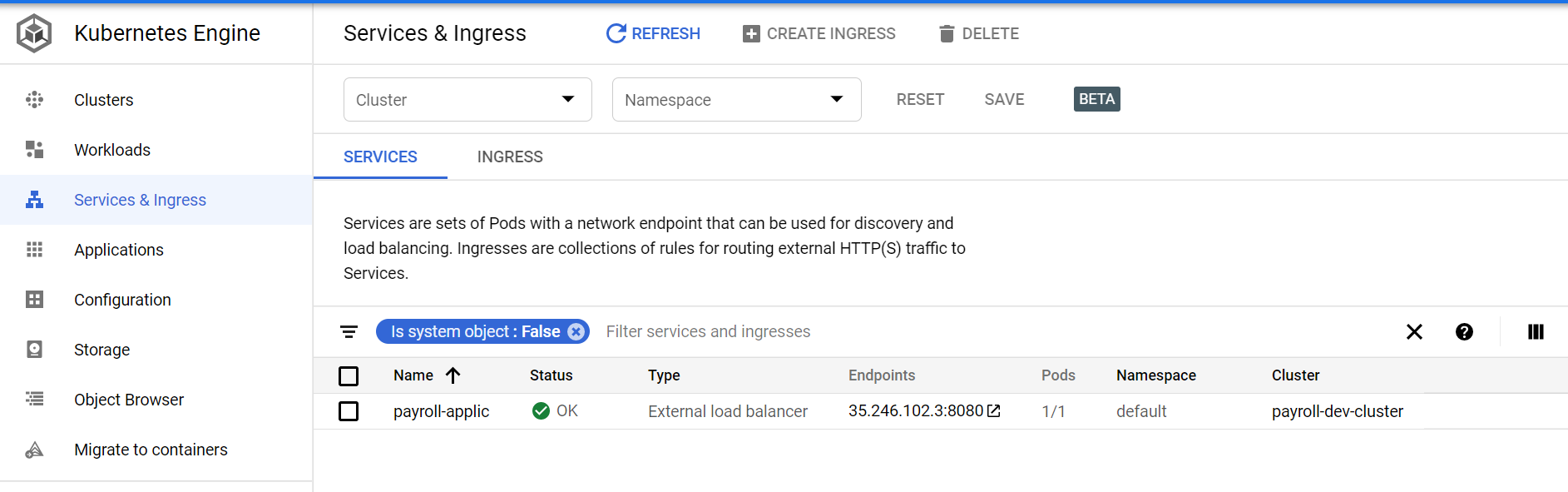
Kubernetes cluster in the GCP :



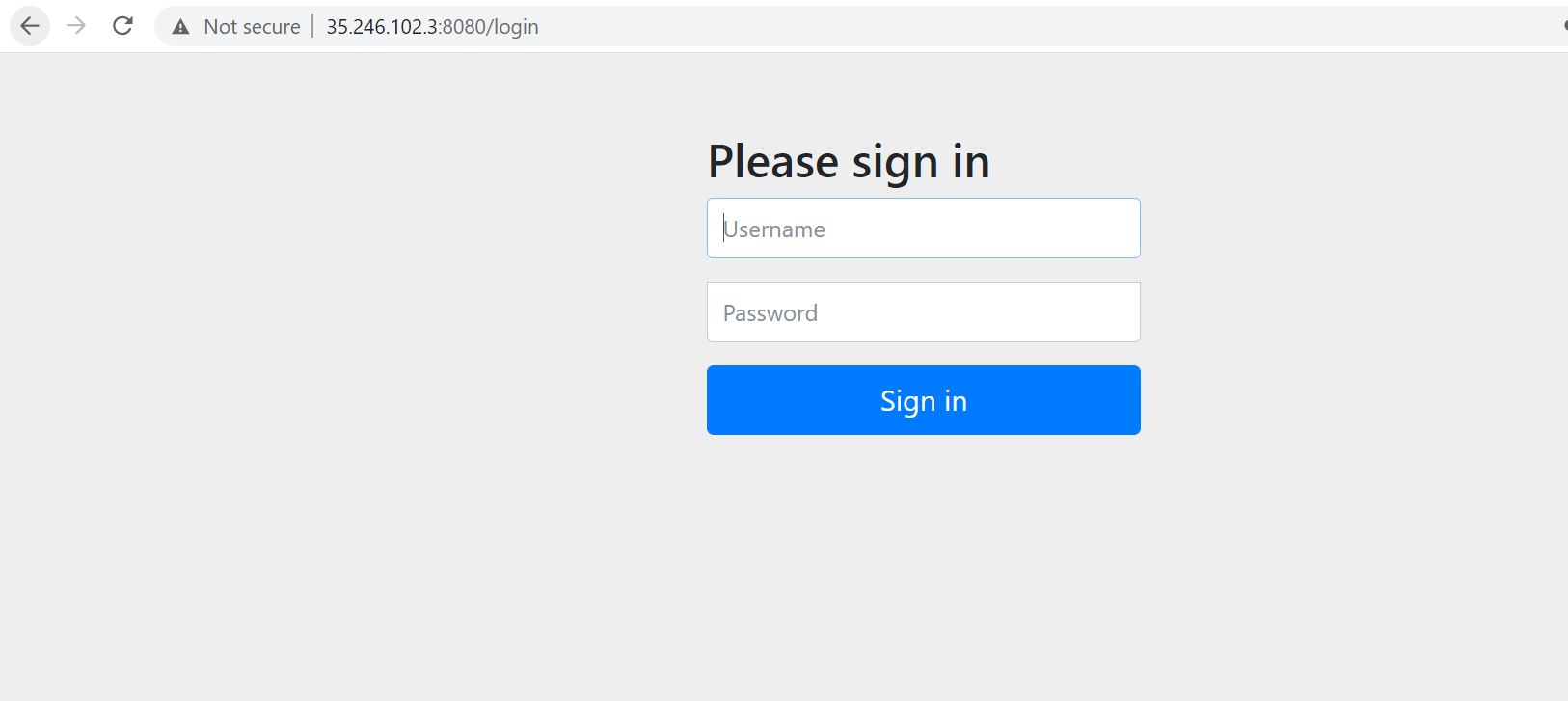


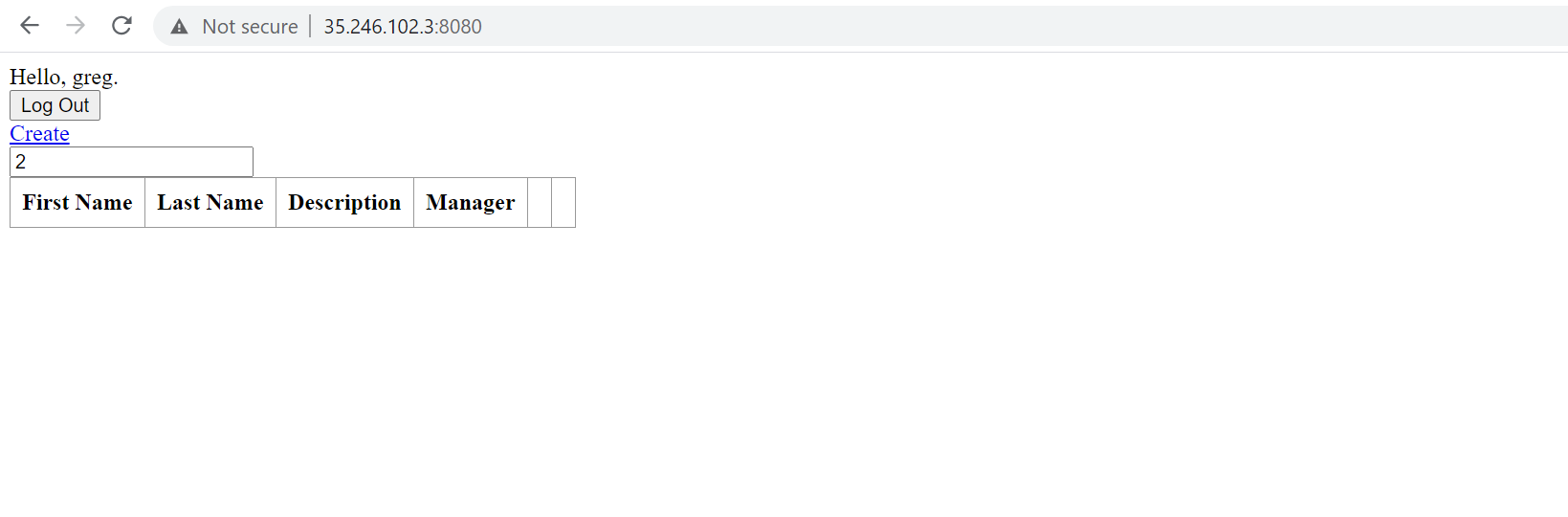


The application has been exposed with load balancer service. The external IP address is as shown below :



Application deployed on the Kubernetes cluster, up and running as shown below :





**Creating of PostgreSQL db in the GCP :**

CREATE DATABASE payroll;

Three tables created by using the following commands :

**CREATE TABLE manager (**

**id SERIAL,**

**name TEXT NOT NULL,**

**password TEXT NOT NULL,**

**roles TEXT [],**

**PRIMARY KEY (id)**

**) ;**

**CREATE TABLE employee (**

**id SERIAL,**

**firstname TEXT NOT NULL,**

**lastname TEXT NOT NULL,**

**description TEXT NOT NULL,**

**version TEXT NOT NULL,**

**manager TEXT NOT NULL,**

**PRIMARY KEY (id)**

**) ;**

**CREATE TABLE manager\_employee (**

**man\_emp\_id SERIAL,**

**man\_id INT NOT NULL,**

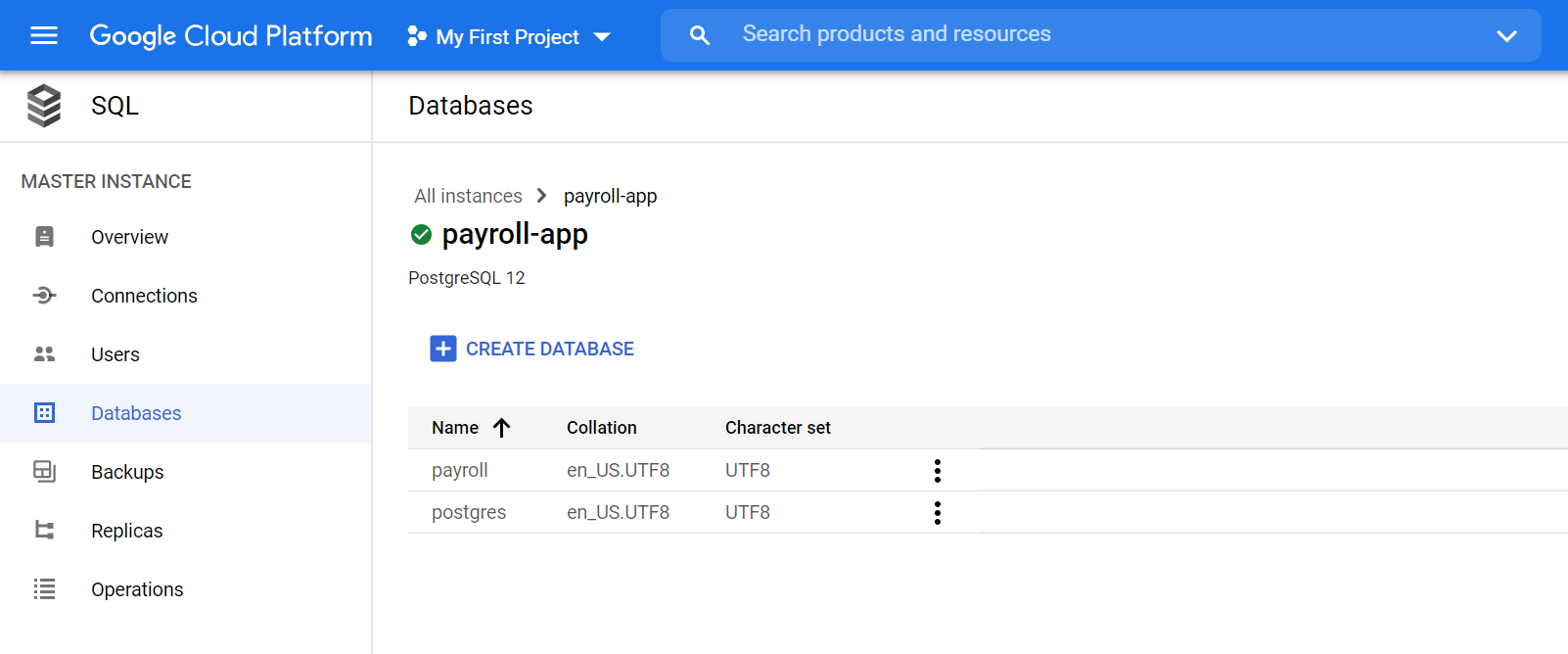
**emp\_id INT NOT NULL,**

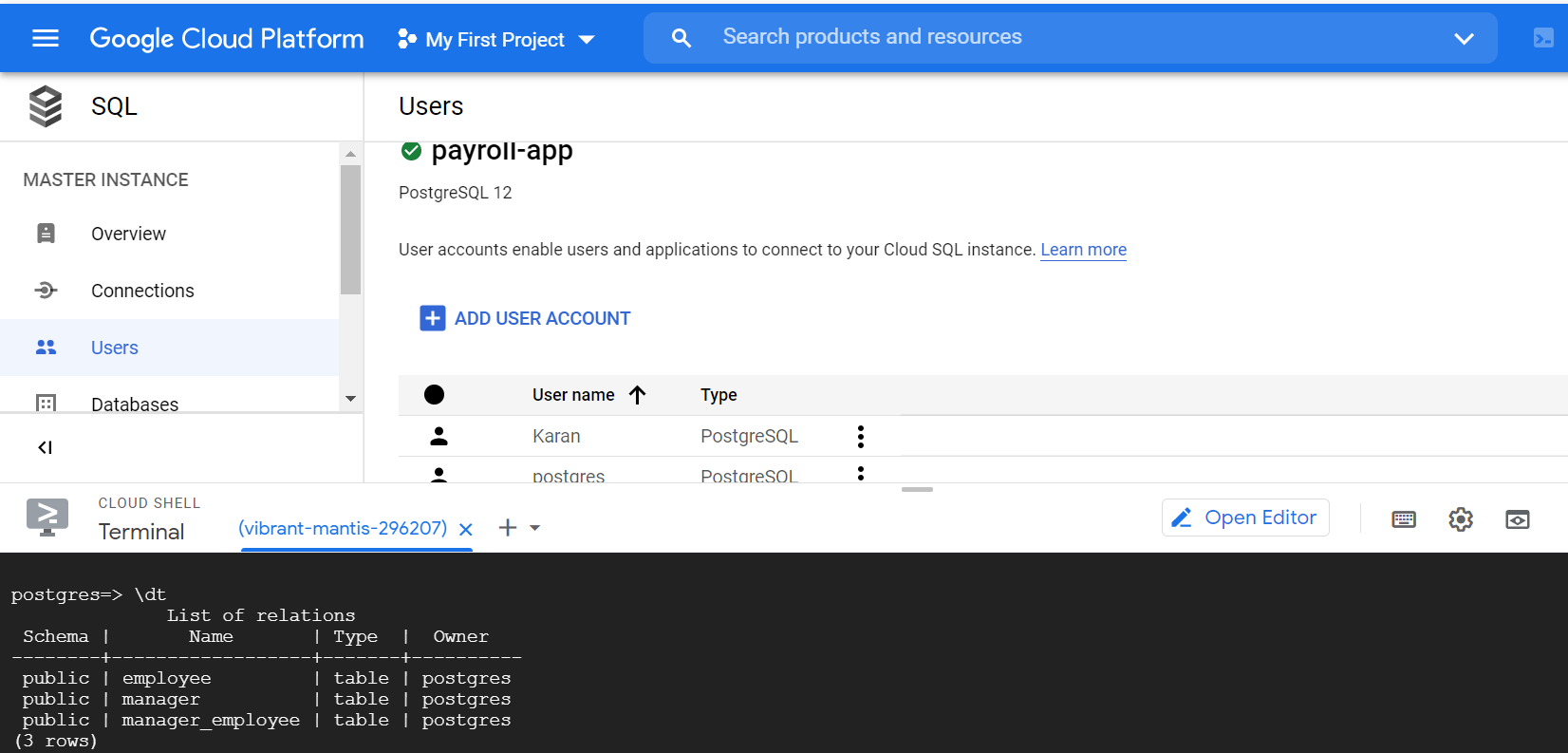
**PRIMARY KEY (man\_emp\_id ),**

**FOREIGN KEY (man\_id) REFERENCES manager (id),**

**FOREIGN KEY (emp\_id) REFERENCES employee (id)**

**) ;**





**Changes in the application :**

**For connection to the DB :**

To connect to the DB, the POM file and Application properties files has been modified.

**For splitting of Frontend and Backend :**

I tried to create a different project for the Frontend which has been also shared by me, but as was not so much clear about how the functionalities should work and due to the time constraints I could not proceed in creating different images for frontend and Backend part.