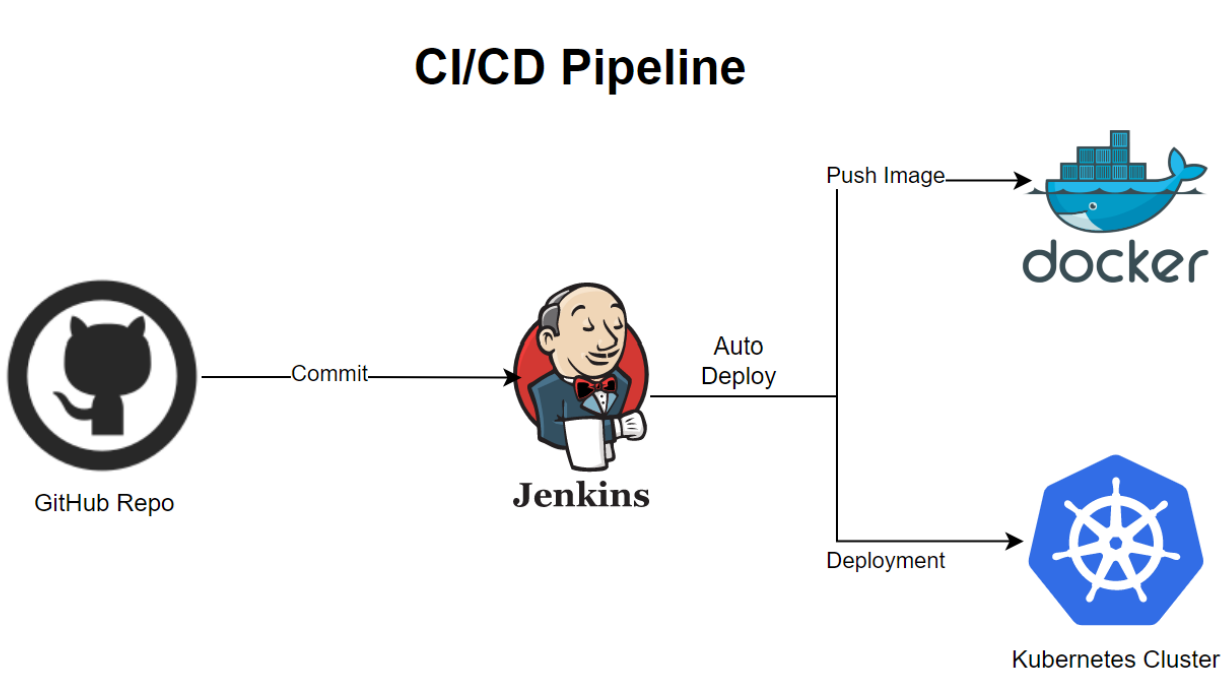
645 HW2 Survey Web Application using CI/CD Pipeline

GitHub URL : https://github.com/sandeep-varma8029/645_HW2_SurveyApp

This Assignment invloves establishing a CI/CD pipeline that includes a Git source code repository of the survey application developed in HW1 Assignment , and Jenkins for automated build and for the automated deployment of your application on Kubernetes. This is achieved by building and deploying a Docker Image to GCP Kubernetes cluster with the help of a Jenkins Pipeline. The Git Repo Folder has the source code of the Web App from eclipse, Dockerfile, Jenkinsfile, deployment.yaml and the HW1 Survey Application's war file. The below image shows an overview of the implementation



The basic Requirements Include:

1. A Google Cloud Platform (GCP) Account
2. Jenkins
3. Docker
4. Docker Hub Account
5. GCP Kubernetes Cluster

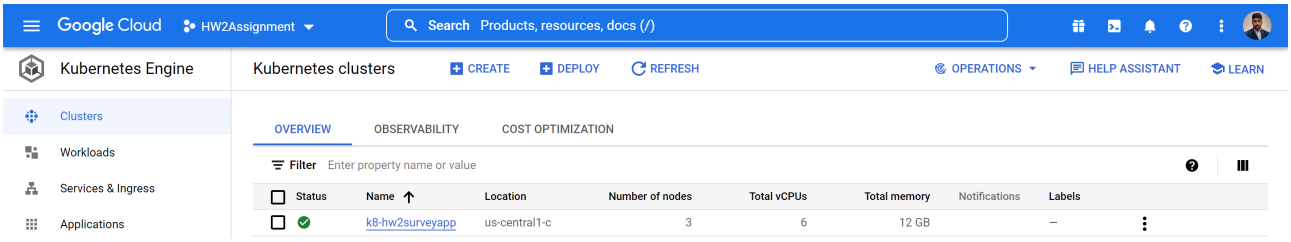
Step 1: GCP Setup

First, we need to setup our GCP to have Jenkins and Kubernetes.To get started:

- Create a GCP account
- Create a new project
- Enter your project details such as name and location
- Click on create, you should see the created project on the dashboard

Step 2: Setup Kubernetes on GCP

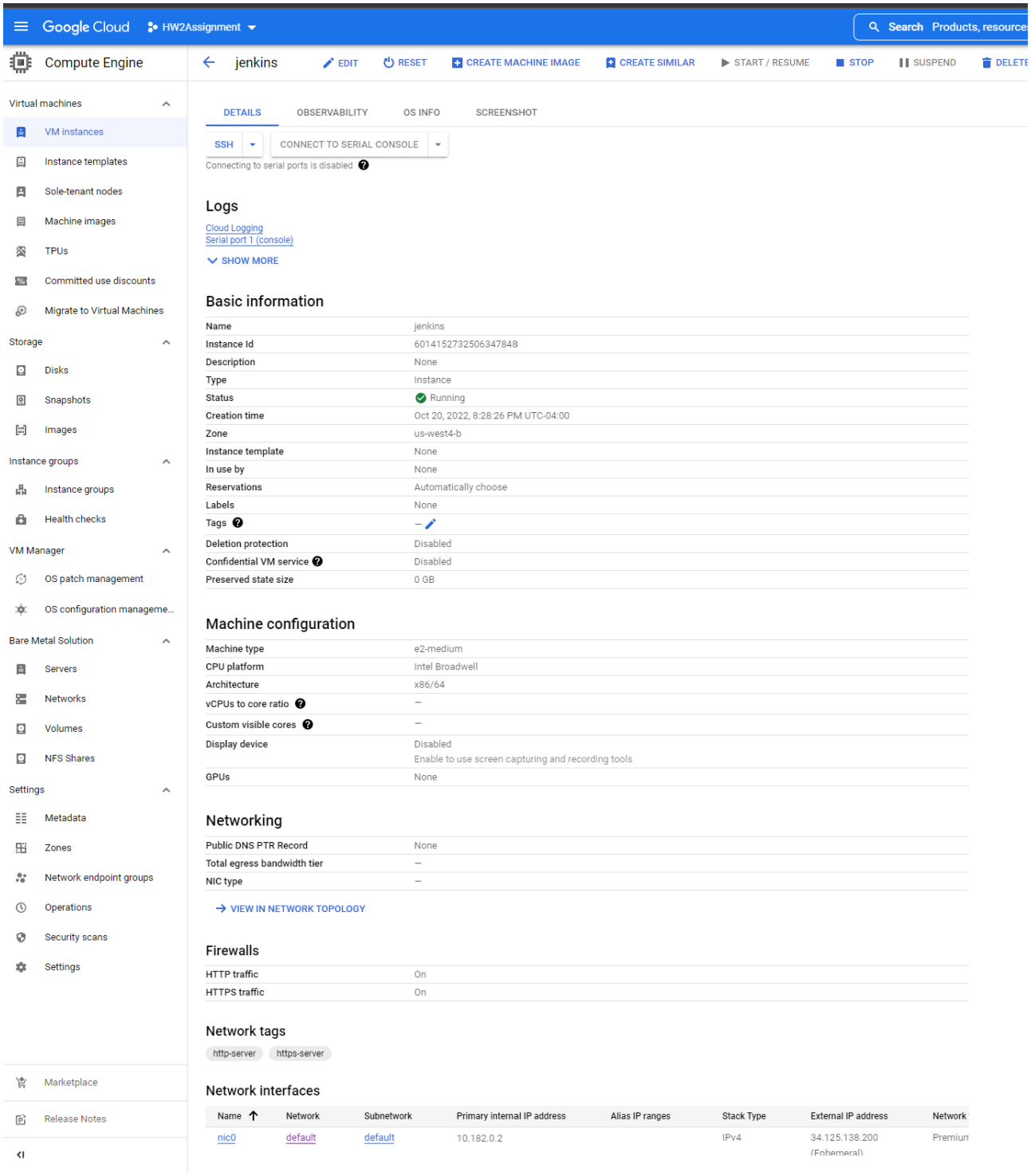
- Click on Kubernetes Engine
- Click on cluster, then click on create
- Click on configure under standard mode
- Enter your cluster name and click on create
. Enable Kubernetes Engine API



Step 3: Setting up Jenkins on GCP

The next step is to create Jenkins's instance on GCP.

- Click on the hamburger menu
- Click on compute engine
- Click on VM instance
- Click on create instance
. Enter your Jenkins instance details
. Enter the name of the instance as Jenkins
. Enable firewall to allow HTTP and HTTPS traffic
. You can configure the Jenkins instance depending on your needs
. Click on create,You should see the Jenkins instance in the list of VM instances

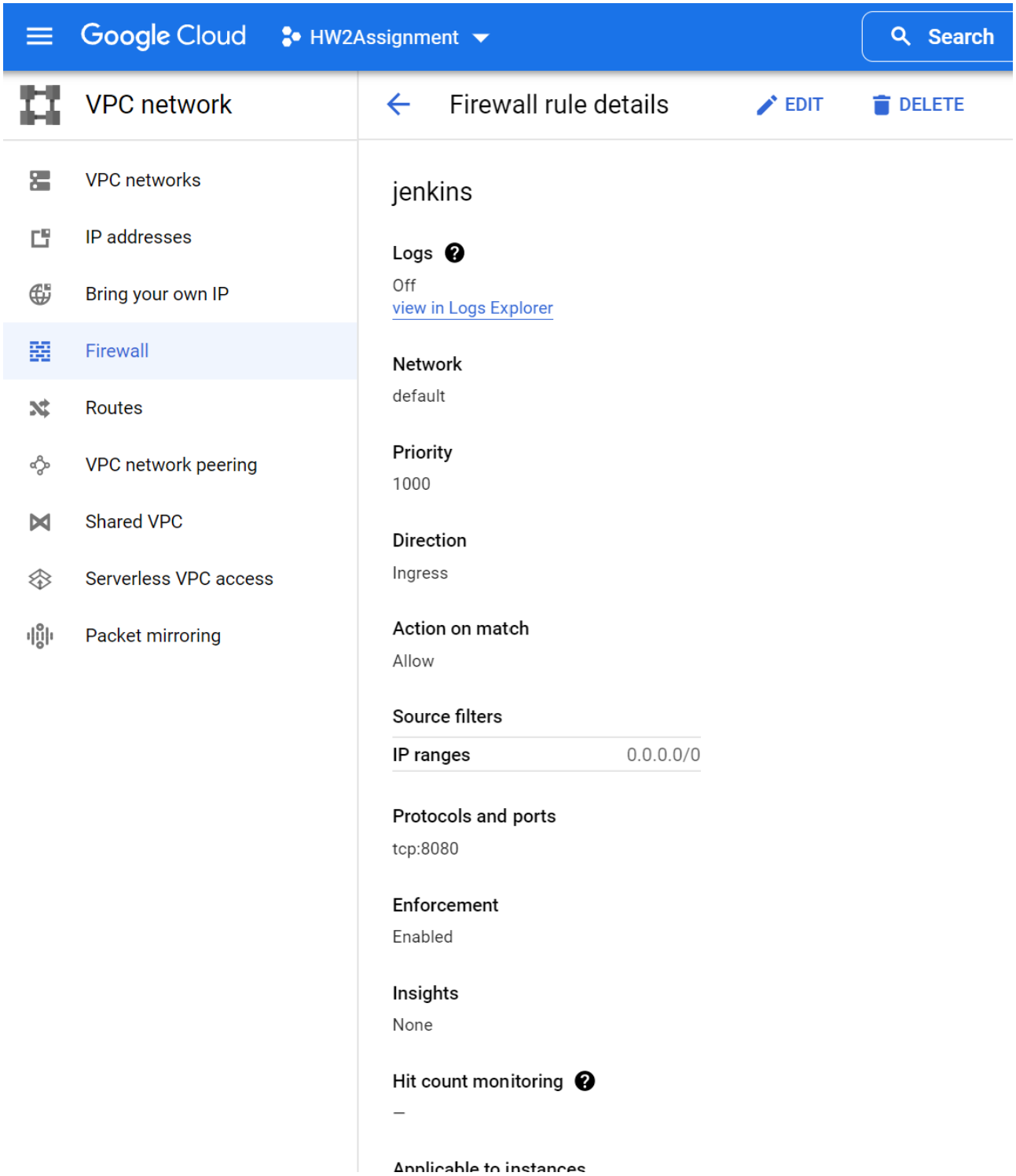


There are various ways in which we can access the Jenkins instance. For this application, we are using browser window

Install Jenkins using the browser window. Check Jenkins Installation on Debian at https://www.jenkins.io/doc/book/installing/linux/#debianubuntu. The external IP is the accessible IP address for the Jenkins instance. For example, http://<jenkins_external_ip>:8080

The above URL will not be available, we need to add a firewall rule that will allow port 8080. To do this

- Click on Jenkins instance
- Scroll down to network tag, click on default under network header
- Click on firewall rules
- Click on add firewall rule
- Enter details of the firewall rule
. Type Jenkins as name
. Type any preferred description
. Select All instances in the network as target
. Type 0.0.0.0/0 as Source IPv4 ranges
. Select Specified protocols and ports
. Check tcp and enter 8080 as the value
. Click on save



http://<jenkins_external_ip>:8080 will be accessible, install the necessary plugins and you will be able to login to Jenkins The simplest and most common way of installing plugins is through the Manage Jenkins > Manage Plugins view, available to administrators of a Jenkins environment. Under the Available tab, plugins available for download from the configured Update Center can be searched and considered.Important plugins to be installed:

```
Google Kubernetes Engine Plugin
Kubernetes
```

```
        Kubernetes CLI Plugin
        Kubernetes Credentials Provider
        Google Container Registry Auth Plugin
        Google Container Registry Auth Plugin
        Docker Pipeline
        Git client plugin
        Git plugin
        GitHub
        GitHub API Plugin
        Github App Kubernetes Credentials
        Github App Kubernetes Credentials
        GitHub Branch Source Plugin
        Pipeline: GitHub Groovy Libraries
```

Welcome to Jenkins!

Username

Password

Keep me signed in

Sign in

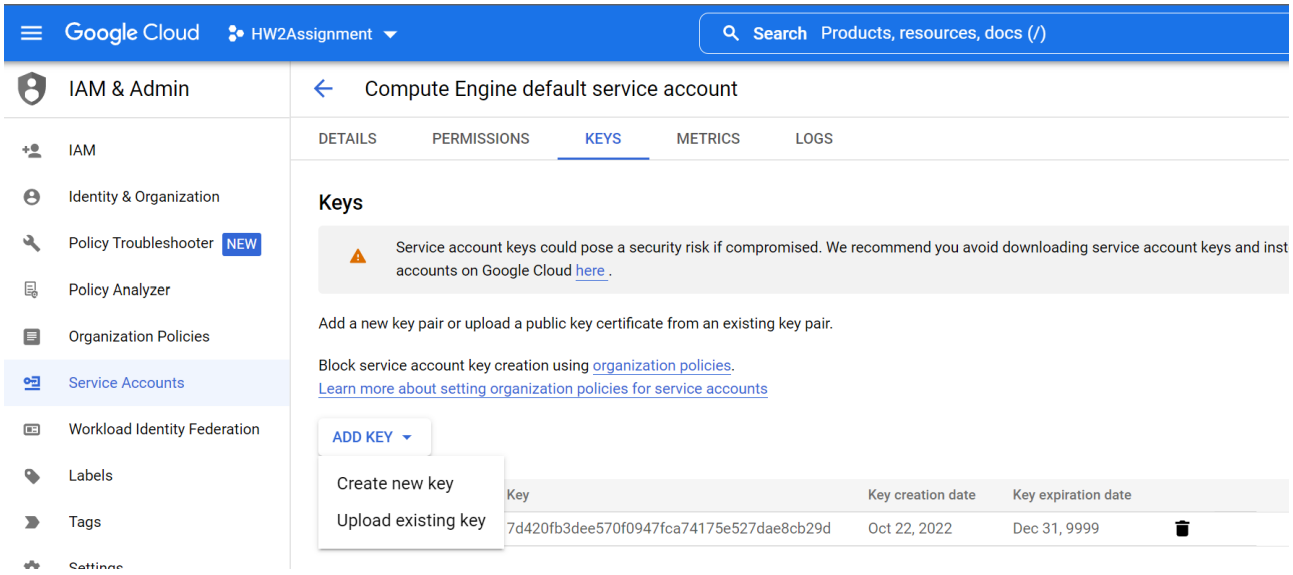
Storing Credentials in Jenkins: We will use the Jenkins credentials store for the pipeline to communicate with the Kubernetes cluster and the Docker Hub registry

```
Navigate to the "Dashboard > Manage Jenkins > Manage Credentials" menu item.
Select the "System" sub-menu" and the "Global credentials" domain.
Click the "Add credentials" link.
Select the "Username with password" credential type and enter your Docker Hub
```

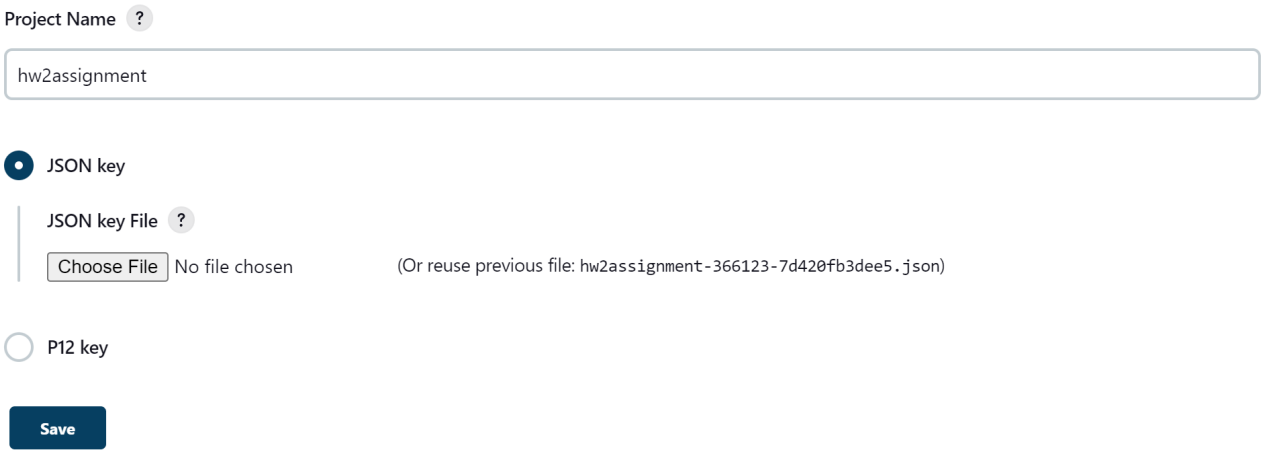
```
username and password in the corresponding fields.
Set the "ID" field to dockerID.
Click "OK" to save the changes.
```

To Add GCP Kubernetes Cluster Credentials:

```
Navigate to the "IAM & admin -> Service accounts" page and create a new service
account.
For this Assignment, I have created a service account with the name "Compute
Engine default service account".
Assign it the "Kubernetes Engine Admin" role.
Next, create a new JSON key for the service account.
Download and save this key.
```



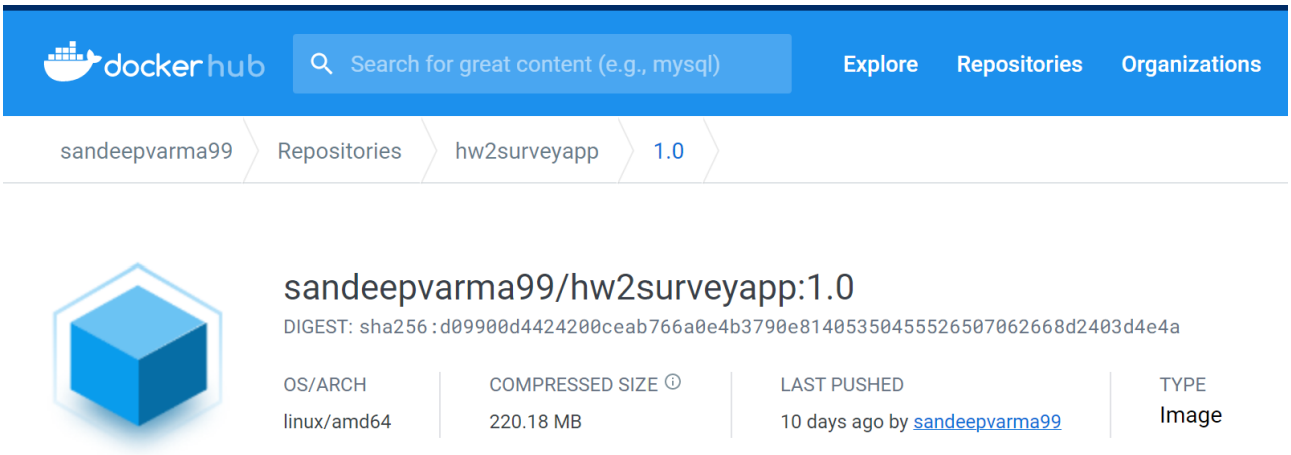
On Jenkins Click the "Add credentials" link.
Select the "Google Service Account from private key" credential type and set the project name (which doubles as the credential identifier) to my project name.
Select the "JSON key" radio button and upload the JSON key obtained in the previous step in Setting up the GCP Account.
Click "OK" to save the changes.



Step 4: Creating a Docker Container

. Make sure you have docker installed on your machine, install from https://docs.docker.com/engine/install/ubuntu/
. Create a file called Dockerfile. Docker requires the file to be called ‘Dockerfile’
. Build the HW1 Survey Web App project on eclipse and put the war file in the same folder as the Dockerfile,Note that ‘HW1_Survey_from’ is the display name of the Tomcat application
. In the DockerFile, use the FROM command to get the initial image for the build,

We want to run the war file in Tomcat so we should use the tomcat image:
  FROM tomcat:8.5.47-jdk8-openjdk
. Next, we need to drop the war file in the webapps folder:
  COPY ./HW1_Survey_form.war /usr/local/tomcat/webapps
. On the command line, use this command: ‘docker build -t hw2surveyapp:1.0’ You can use whatever name and tag you want
. Verify that the image is properly working by running ‘docker run -it -p 8080 hw2surveyapp:1.0’ and open a browser at http://localhost:8080/HW1_Survey_from
. On the command line, login to docker using ‘docker login -u <your username>’
. Change the name of you image to be <your username on dockerhub>/<name of the app>:<image tag> using the docker tag command. In my case it is: ‘docker tag hw2surveyapp:1.0 sandeepvarma99/hw2surveyapp:1.0’
. Verify that your image is on Docker Hub,Your image should be accessible from the internet



Step 5: Create deployment file for Kubernetes and JenkinsFile for the Jenkins pipeline

A deployment.yaml file in the repository is added which defines how the built container should be deployed on Kubernetes. The definition pulls the built container from Docker Hub and creates a new deployment with it in your Kubernetes cluster running on 3 pods. It also creates a LoadBalancer service so that the deployment can be accessed from outside the cluster.

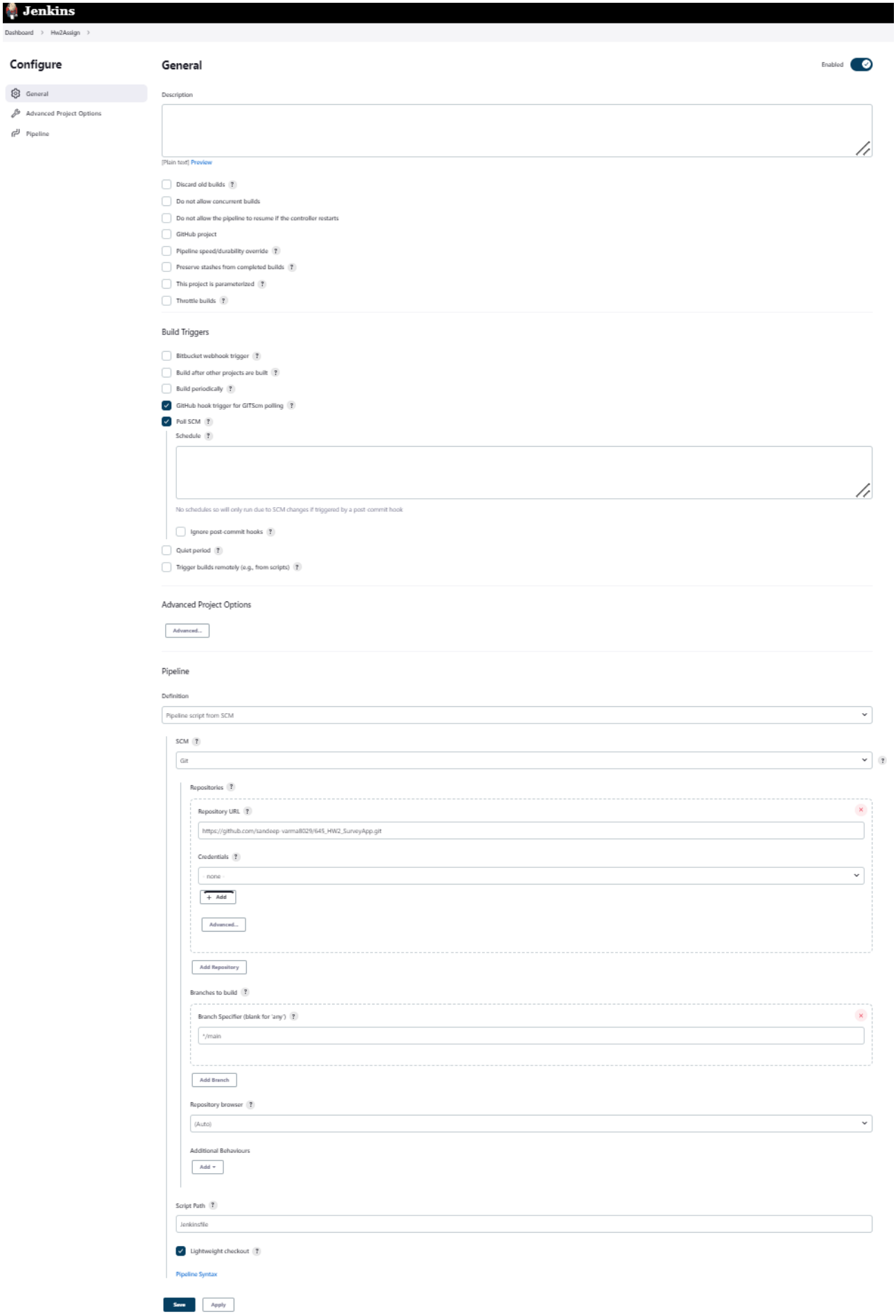
Finally, create a pipeline script named Jenkinsfile (also added in the repo). This is the script Jenkins will use to build and deploy the application. What the Jenkins pipeline file does are as follows;

- Pull from your repository
- Dockerize the application
- Push the image to docker hub
- Update the deployment YAML file with the build number
- Deploy to GKE

Step 6: Jenkins Pipeline Setup

We are using Jenkins as our CI/CD

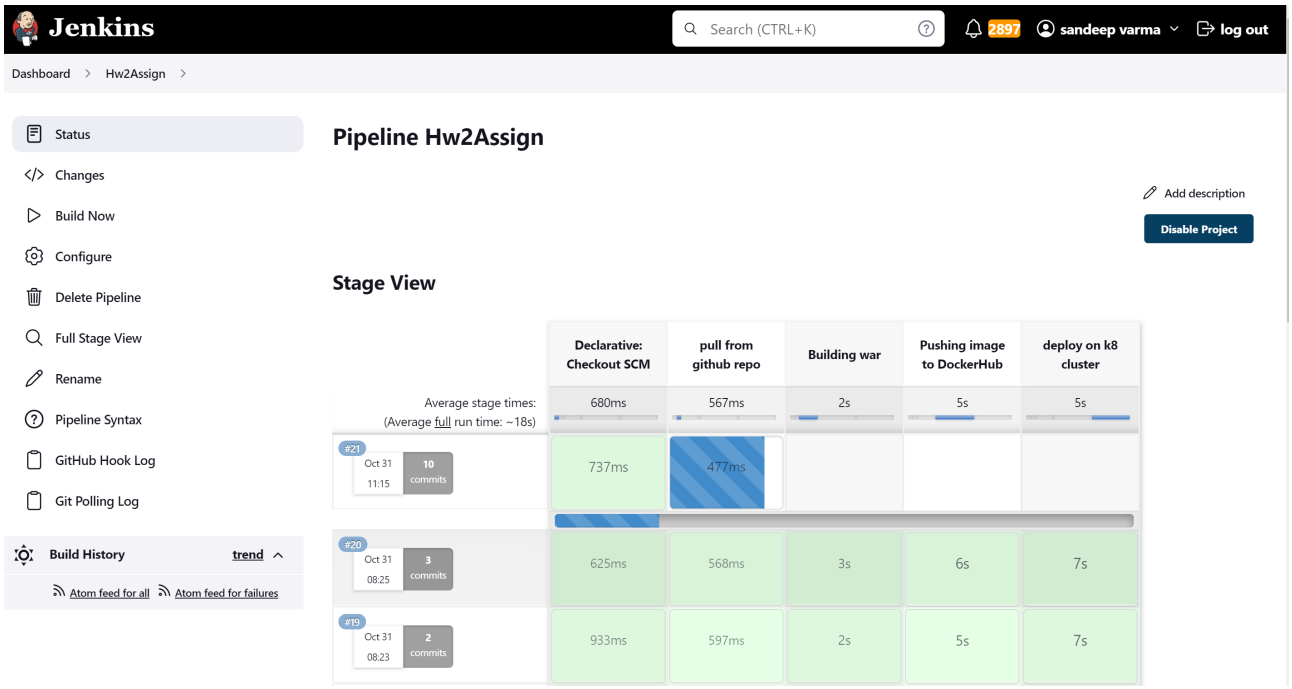
- Open Jenkins on GCP
- Click on New Item
- Enter Jenkins job name
. Under 'Build Triggers' select 'GitHub hook trigger for GITScm polling' and 'Poll SCM'
- Select Pipeline and ok
- Click on pipeline
- Select Pipline script from SCM as definition
- Select git as SCM
- Enter your repository url
- Enter Jenkinsfile in the script path
- Click on apply and save


Jenkins
Dashboard › Hw2Assign ›
Configure
General
Advanced Project Options
Pipeline
General
Enabled
Description
[Plain text] Preview
Discard old builds
Do not allow concurrent builds
Do not allow the pipeline to resume if the controller restarts
GitHub project
Pipeline speed/durability override
Preserve stashes from completed builds
This project is parameterized
Throttle builds
Build Triggers
Bitbucket webhook trigger
Build after other projects are built
Build periodically
GitHub hook trigger for GITScm polling
Poll SCM
Schedule
No schedules so will only run due to SCM changes if triggered by a post-commit hook
Ignore post-commit hooks
Quiet period
Trigger builds remotely (e.g., from scripts)
Advanced Project Options
Advanced...
Pipeline
Definition
Pipeline script from SCM
SCM
Git
Repositories
Repository URL
https://github.com/sandeep-varma8029/645_HW2_SurveyApp.git
Credentials
- none -
Add
Advanced...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/main
Add Branch
Repository browser
(Auto)
Additional Behaviours
Add
Script Path
Jenkinsfile
Lightweight checkout
Pipeline Syntax
Save
Apply

The Job will appear on your jenkins dashboard

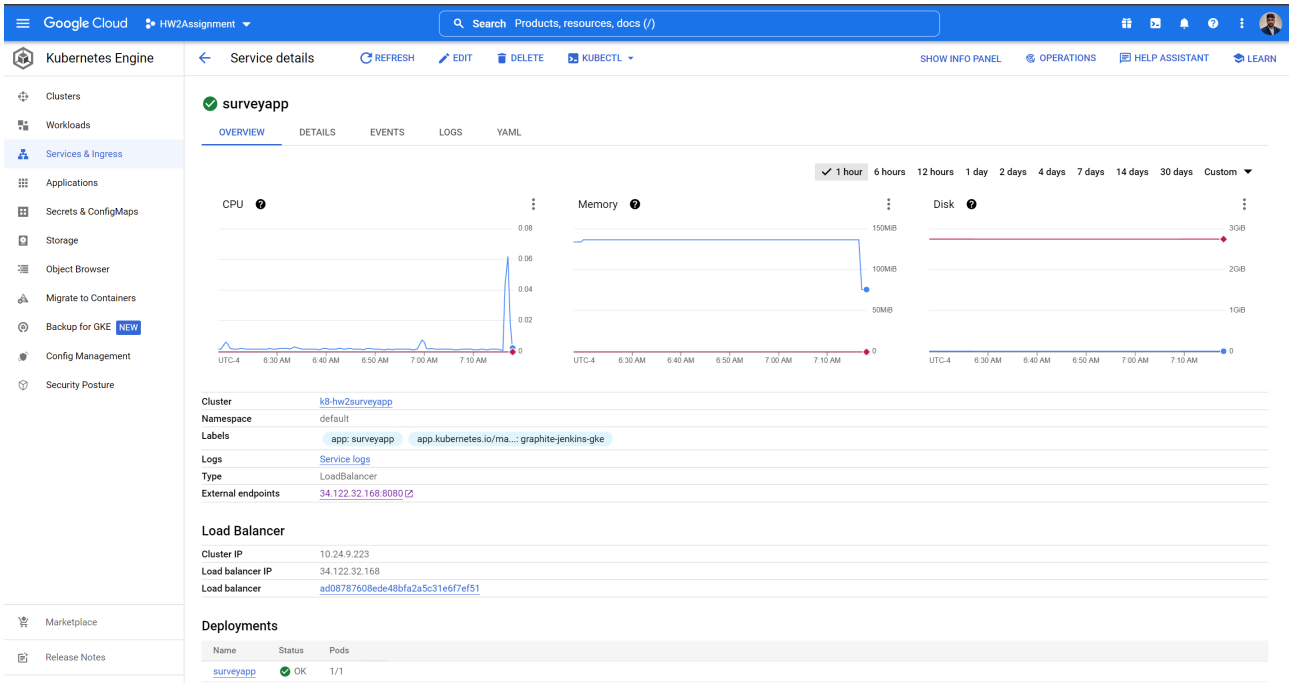
Step 7: Building and Deploying the Application

Click on selected job and Click on Build now to run the job

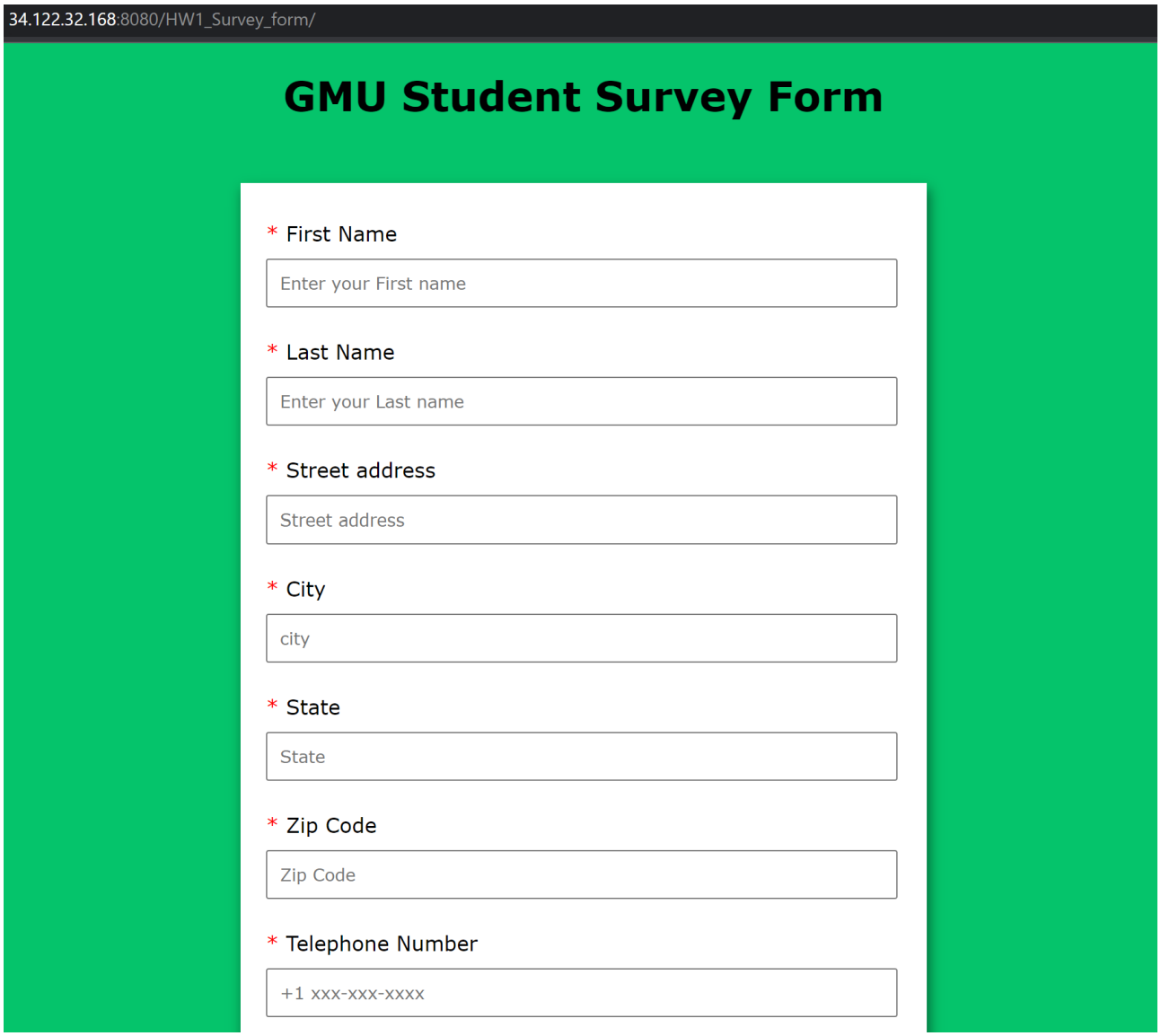


After successful Jenkins job build, an instance of the application(s) is/are created on GKE.So any further commits on github will trigger jenkin build and deploy changes directly on the service deployed in the GKE Cluster runnig on 3 pods.

Name	Status	Type	Endpoints	Pods	Namespace	Clusters
surveyapp	OK	External load balancer	34.122.32.168:8080	3/3	default	k8-hw2surveyapp



The application can be assessed at http://34.122.32.168:8080/HW1_Survey_form/



AWS URL of the SurveyPage : http://studentsurveyapp.s3-website-us-east-1.amazonaws.com/

AWS URL of Sai Sandeep Varma Mudundi HomePage: http://personalwebsitehw1.s3-website-us-east-1.amazonaws.com/

AWS Homepage URL of Sai Anjaneya Sowrab Meduri(G01270421): http://swe645-msa.s3-website-us-east-1.amazonaws.com/

AWS Homepage URL of Rajeev Priyatam Panchadula(G01333080): http://rajeevpartone.s3-website-us-east-1.amazonaws.com/

AWS Homepage URL of Lakshmi Guttikonda(G01334433): http://cs645ha.com.s3-website-us-east-1.amazonaws.com/

Project Demo Link : https://drive.google.com/file/d/1EqZemRzBKUokM3IK3q_2spCHs0Q79_wx/view?usp=share_link

Team Details and Contributions:

```
Sai Sandeep Varma Mudundi (G01352322) - Jenkins CI/CD pipeline creation,
Kubernetes Cluster setup,GitHub setup containerization of web app,Jenkinsfile,
deployment file creation ,readme , demo video and documentation
Sai Anjaneya Sowrab Meduri( G01270421) - Docker setup,Dockerfile creation,
hosting of web application container on kubernetes cluster documentation and
containerization
Rajeev Priyatam Panchadula( G01333080) - Git SCM polling,Virtual Machine instance
configuration and documentation, setting up git repo war file generation
Lakshmi Guttikonda(G01334433) - Documentation, AWS hosting of homepage,
maintaining jenkins plugins and IAM credintials, and web appliction contribution
```

References :


```
[1] : https://www.linkedin.com/pulse/create-cicd-pipeline-jenkins-google-
kubernetes-engine-pralay-debroy?trk=articles_directory
[2] : https://medium.com/@bukunmitanimonure/build-and-deploy-docker-image-to-gcp-
kubernetes-cluster-with-jenkins-pipeline-5405a2966a58
```