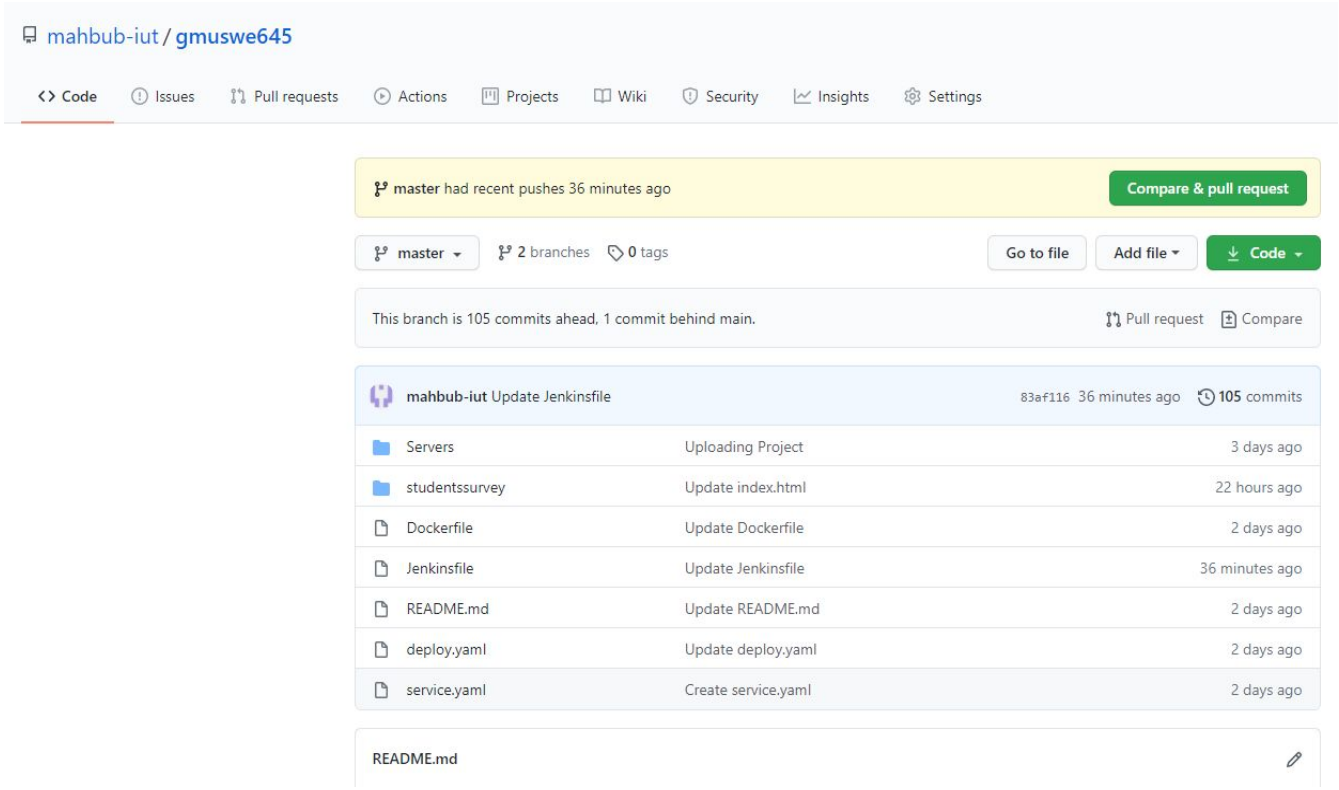
SWE 645 Homework 2 (Installation and Setup instruction)

Submitted by: (Mahbubul Alam Palash & Taseef Rahman)

Setting up Github repository

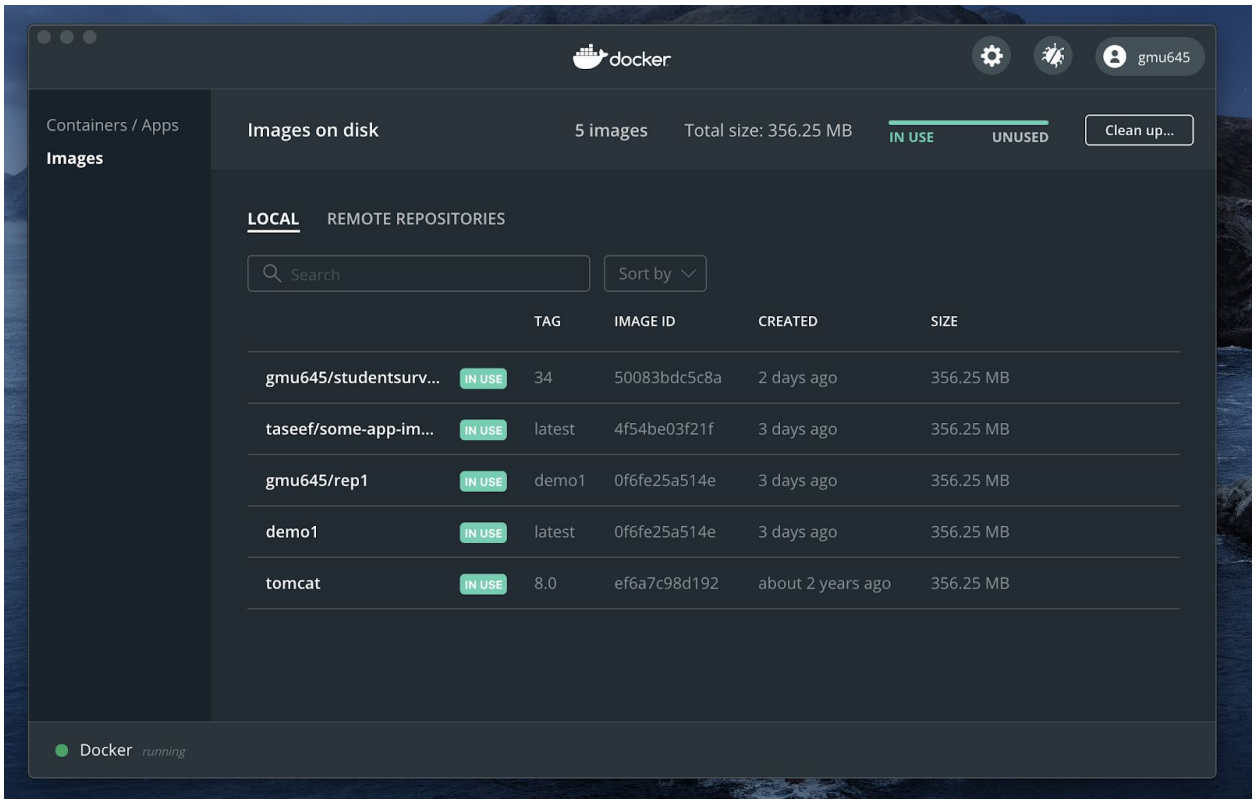
Simply a place to store the related files which we can later link up using git webhook. We created a github repository and We uploaded the project from Eclipse to the github repository using eclipse Team command.



Docker Hub and image creation-

Involves docker hub account creation and later, docker image from the .war file. One helpful reference is this one- *https://medium.com/@pra4mesh/deploy-war-in-docker-tomcat-container-b52a3baea448*.

One important caveat is that we have to name our docker image following the convention repository_name/image_name while pushing the image to DockerHub. Below is a sample of docker dashboard after a few images are deployed on a local machine.



Deploying Project in Google Kubernete Engine:

At the beginning a cluster named SWE645 was created in Google kubernete engine. To give access to google kubernetes cluster access for kubectl was configured. The kube configuration file **config** for the cluster was downloaded and placed to the **.kube** directory of Jenkins home directory. The **google_application_credentials** from google kubernete engine was also downloaded and the location was set as an environment variable.

Then for initial deployment a yaml file named deployment.yaml for deployment was created and used that using the command

kubectl apply -f deployment.yaml

19 lines (19 sloc) | 309 Bytes

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: swe645final
  labels:
    app: survey
spec:
  replicas: 3
  selector:
    matchLabels:
      app: survey
  template:
    metadata:
      labels:
        app: survey
    spec:
      containers:
      - name: student
        image: gmu645/studentsurvey:40
```

Figure: ***deployment.yaml***

A load balancer service yaml file service.yaml for exposing the application to outside world was written and service was created using the command:

kubectl apply -f service.yaml

master gmuswe645 / service.yaml

mahbub-iut Create service.yaml

1 contributor

12 lines (12 sloc) | 195 Bytes

```
1   apiversion: v1
2   kind: Service
3   metadata:
4     name: surveyservice
5   spec:
6     type: LoadBalancer
7     selector:
8       app: survey
9     ports:
10    - protocol: TCP
11      port: 80
12      targetPort: 8080
```

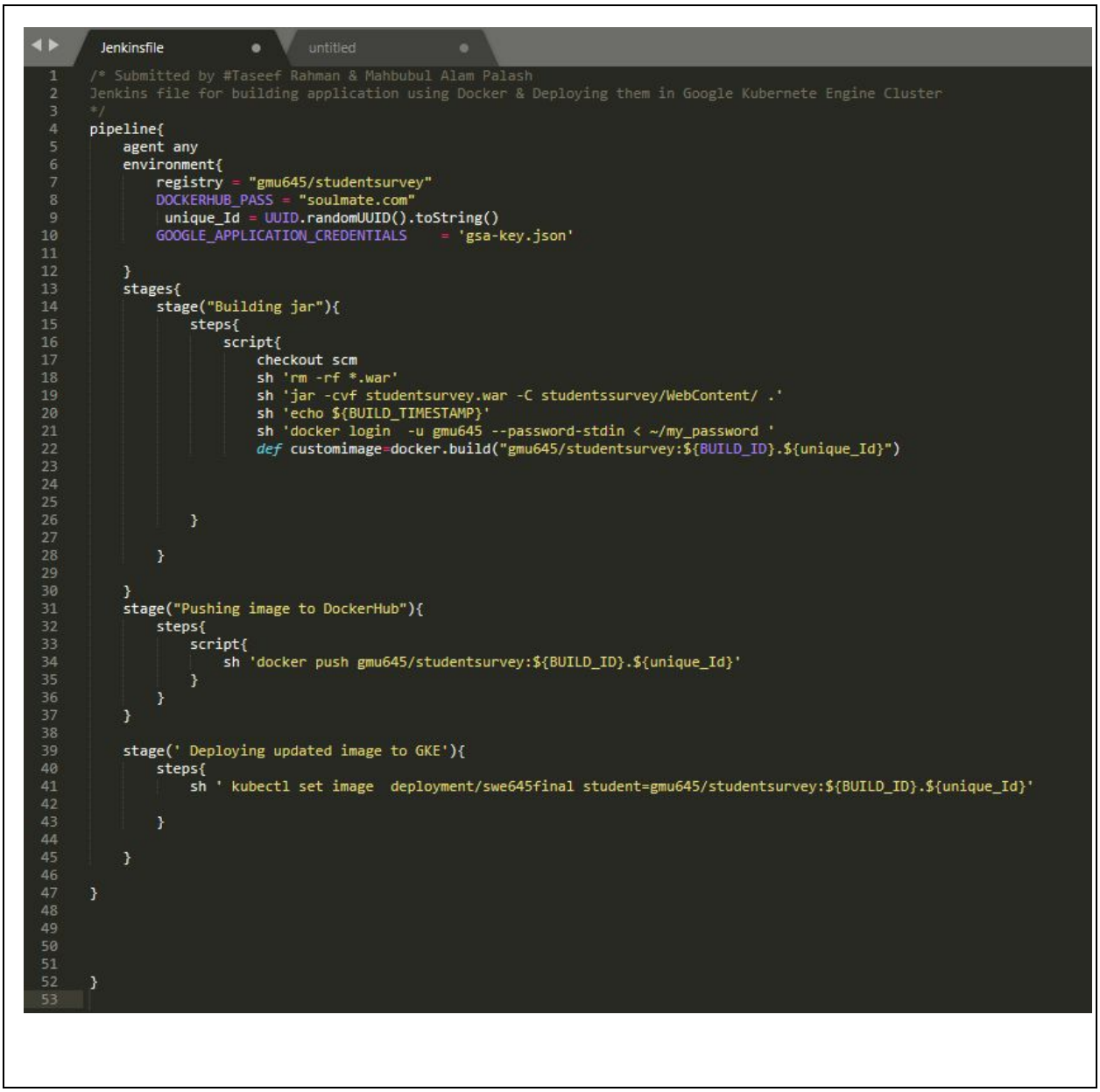
Figure:***service.yaml***

Jenkins-

1. Install Jenkins from **jenkins.io.**
2. Related commands-
   a)wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add –
   b)sudo apt update
   c)sudo apt install jenkins

Later, Jenkins can be opened at the port localhost:8080. Now we have to copy the content of the google Kubconfig file to /home/Jenkins/.kube/config.
We can further verify that it's running with the command *kubectl config current-context.*

3. Setting up Jenkins- After selecting "Pipeline" , we can have cron jobs set up as per the requirement.
4. Then we connect the github repository. Before that we have to set up the github access accordingly so that it doesn't have any errors.
5. We also need to put a file named Jenkinsfile in the related repository.
6. For the jenkinsfile, we can specify the stages that we want to keep track of, for example in our case the stages were
   a)Building the image
   b) Pushing image to DockerHub
   c) Updating the updated image at GKE Cluster



7. A webhook with the github was created which triggered the jenkins file after every commit and a new image is built, pushed to the docker hub and is deployed on the cluster created in Google Kubernete Engine.

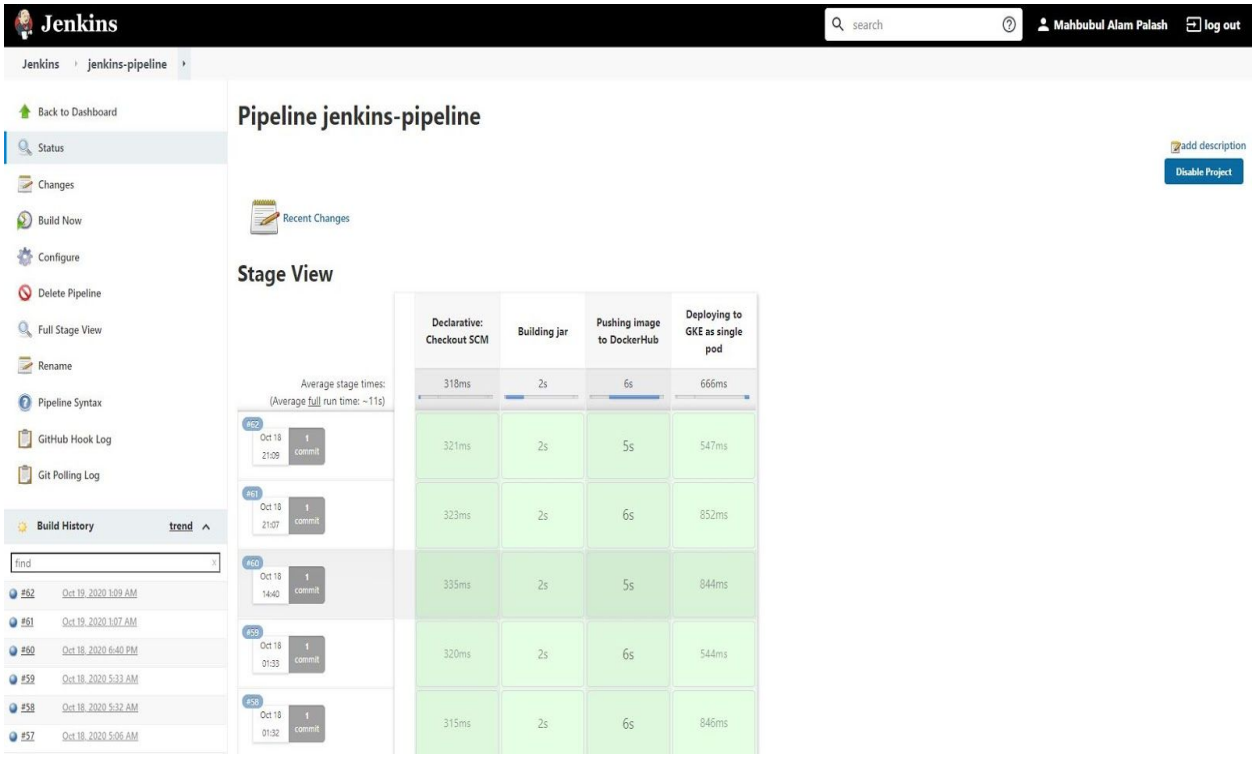


Figure:a git commit triggered a build and deployment in Jenkins

Contributions of teammates:

Taseef Rahman:

1. Creation of Docker Image and also docker hub setup
2. Writing the jenkins file
3. Setting up github webhook and pipeline in jenkins
4. Writing the documentation

Mahbubul Alam Palash:

1. Setting up the Kubernete Cluster in GKE
2. Installing jenkins server and setting up kubectl access to Google Kubernete Cluster
3. Writing the yaml files for deployment and Service Creation

4. Writing the documentation