# SWE 645 – ASSIGNMENT 2

**TEAM MEMBERS**: (team of 2)

1. SANJANA GOVINDU | G01380306 | sgovindu@gmu.edu
2. ADITEE GADDAM | G01363937 | agaddam@gmu.edu

**URLS for Assignment:**

- Github Link - https://github.com/sanjana-govindu/SWE645/
- Video Submission: https://drive.google.com/file/d/1hjEHWNLjMZAaNpfT83fDcz8S3qm09xmW/view?usp=sharing
- Containerized Web Application link: https://hub.docker.com/repository/docker/sanjanagovindu/swe645-a2/general

**Prerequisites/ Requirements:**

1. Github and google accorunt for GCP - google cloud program
2. Tomcat installed on local machine
3. Docker Desktop installed on local machine to build the initial docker image.
4. Account on Docker Hub.
5. AWS running on AWS Academy Learner Lab (link provided by the professor)
6. Jenkins installed on local machine

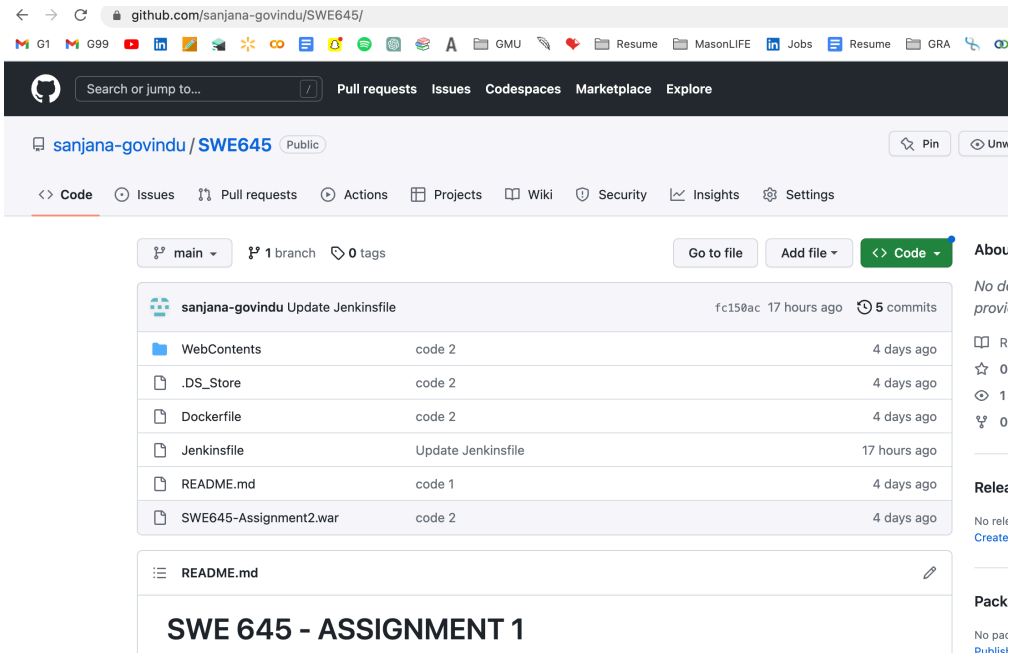
**ASSIGNMENT INSTRUCTIONS**:

The requirements are as follows:

- Containerize the Web application you developed in Homework 1 – Part 2, using Docker container technology.
- Deploy the containerized application on the open-source container orchestration platform Kubernetes to enable scalability and resiliency of the application. Your baseline configuration includes at least three pods running all the time. You can use Rancher (https://rancher.com/docs) to setup Kubernetes cluster. You also have an option to use managed Kubernetes services, such as EKS on Amazon Web Services (AWS) or GKE on Google Cloud Platform (GCP).
- Establish a CI/CD pipeline that includes a Git source code repository at GitHub, and Jenkins for automated build and for the automated deployment of your application on Kubernetes.

## PART 1 – PUSH THE CODE ON GITHUB

1. If you already have a github account then create a github repository with with the assignment 1 submission code in WebContents folder and all readme files outside that folder.
2. Use Github Desktop application to commit and push the changes from your local machine into github.
3. This Github link will be useful later for Jenkins part.




## PART 2 - CONTAINERIZE WEB APPLICATION

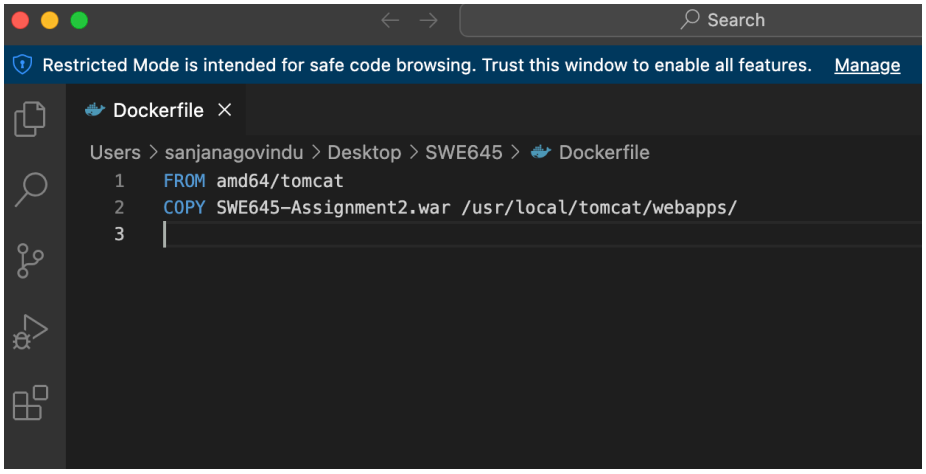
1. To containerize the web application using docker, we need to have docker desktop installed in our local machine i.e., Mac using this link - https://www.docker.com/products/docker-desktop/
2. Also, we need to create an account in Docker Hub - https://hub.docker.com/ with a username (sanjanagovindu) and password of our choice to make our image globally accessible.
3. Create a folder 645 HW 2 in eclipse by creating a dynamic web project and put that WAR file in that folder. To create a war file of the code we can use the below command: **'jar -cvf SWE645-Assignment2.war -C WebContent/ .'**
4. We used Eclipse IDE to create a file without extension called 'DockerFile' as docker requires the file name to be called in that specific way. The DockerFile should be in the

same folder of that WAR file. It installs tomcat in the container and copies the generated war file into the /usr/local/tomcat/webapps dilrectory.

5. Paste the below in DockerFile:

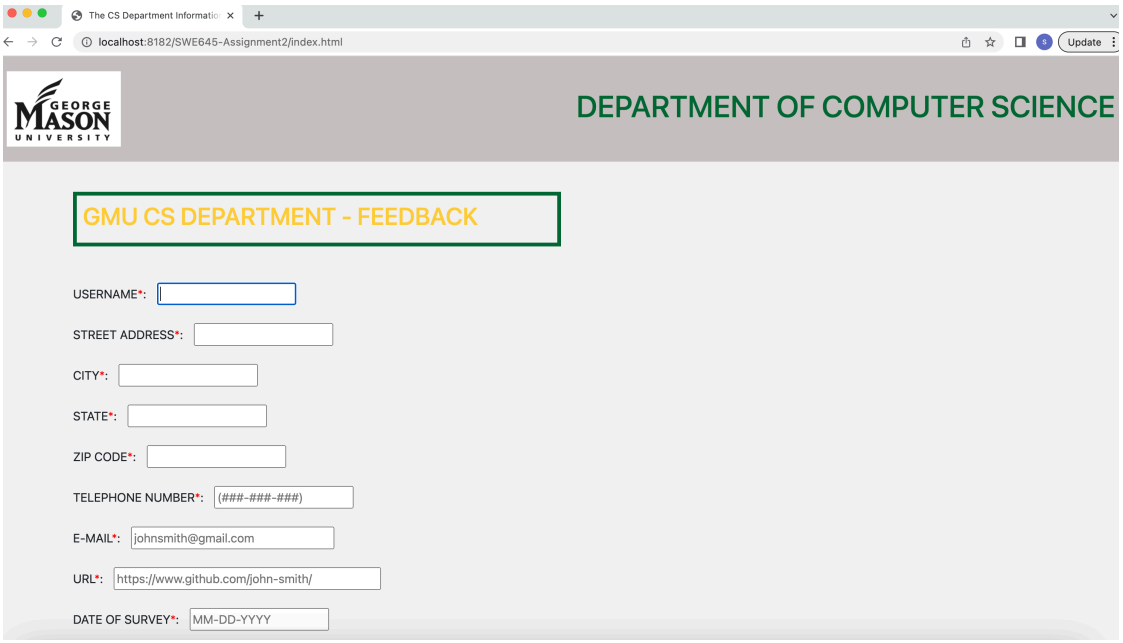
```
FROM amd64/tomcat
COPY SWE645-Assignment2.war /usr/local/tomcat/webapps/
```

6. In that DockerFile, we used the FROM command to get the initial image required for the docker build.



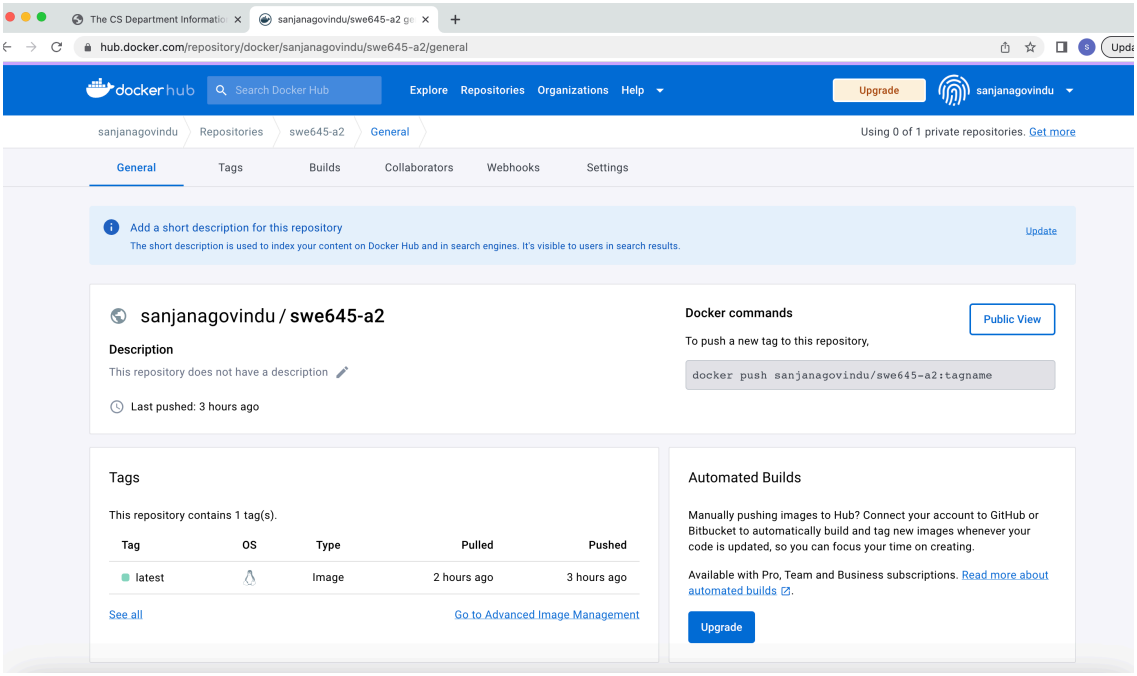

## PART 3 – CREATE IMAGE LOCALLY AND PUSH IT TO DOCKER HUB

1. To access our image globally we need to push it to Docker hub.
2. So, for this we need to create an account in Docker Hub - https://hub.docker.com/ with a username (sanjanagovindu) and password of our choice.
3. Then, in the locally installed docker desktop we need to login into our account that has been created directly or from command line using the command – '**docker login -u <DockerHub_Username>'.**
4. On the command line, use the command: 'docker build --tag swe645-a2 .' to build a docker image with that name in our local (We can use the name of our own choice).
5. Then we can verify if the image is built properly by running the command "**docker run -itd -p 8182:8080 --name swe645-assignment2 swe645-assignment2**"
6. Then run if the containerized application is working by opening http://localhost:8182/SWE645-Assignment2/index.html in your browser like the image below:




7. Then tag the image using the command – "**docker tag swe645-a2 sanjanagovindu/swe645-a2"** and push it to docker hub using the command – **"docker push sanjanagovindu/swe645-a2"**

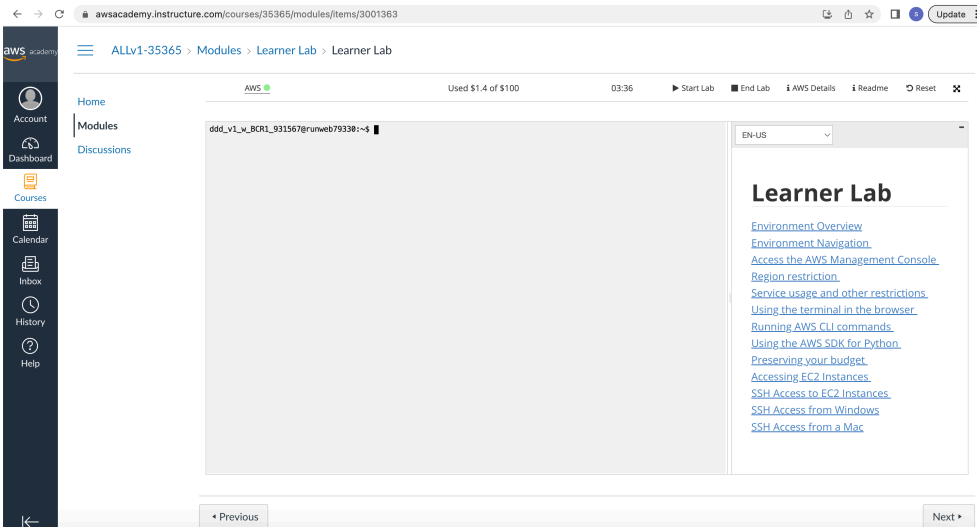
8. Verify that the image is on Docker Hub as we know that the image is accessible from the internet by logging into Docker hub with our credentials.



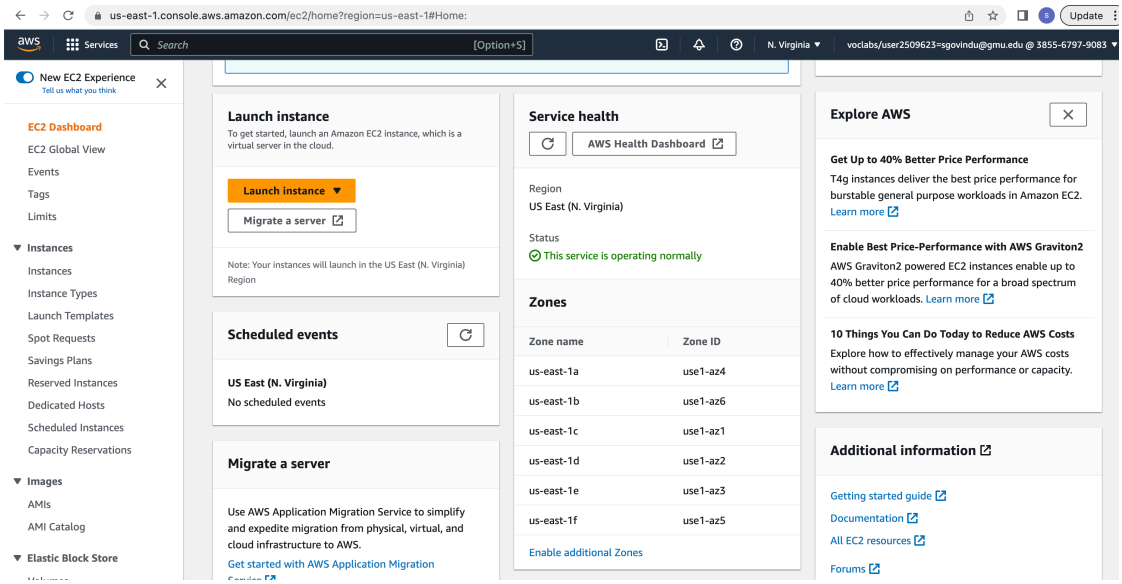
## PART 4 – SETTING UP RANCHER & CREATING A CLUSTER USING RANCHER

### Setup Rancher:

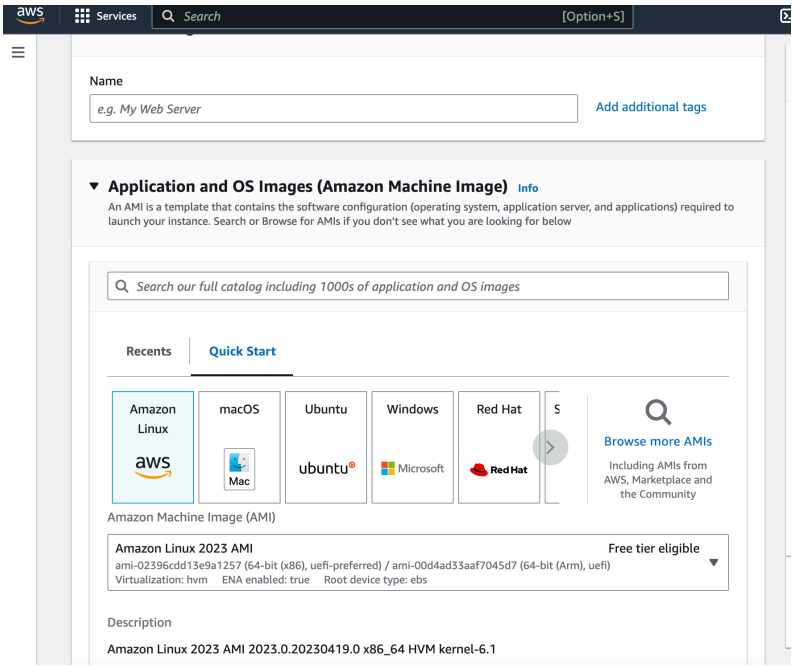
1. For this, we need to log on to AWS console https://aws.amazon.com/ using our credentials or create an account on AWS. In this case we need to use AWS Learner lab for our requirement. https://awsacademy.instructure.com/courses/35365?invitation=JMhpNmcZxRQt4I1LnoQuolHzB8cdmYgRtpvwlGJA




2. Click on Start Lab which will take some time and click on AWS after it is up and navigate to EC2.
3. We need to run that rancher on an Ubuntu EC2 instance with that docker running.
4. To launch that EC2 instance in the AWS Console, click on Services > EC2 and click Instance and ‘Launch instance’.



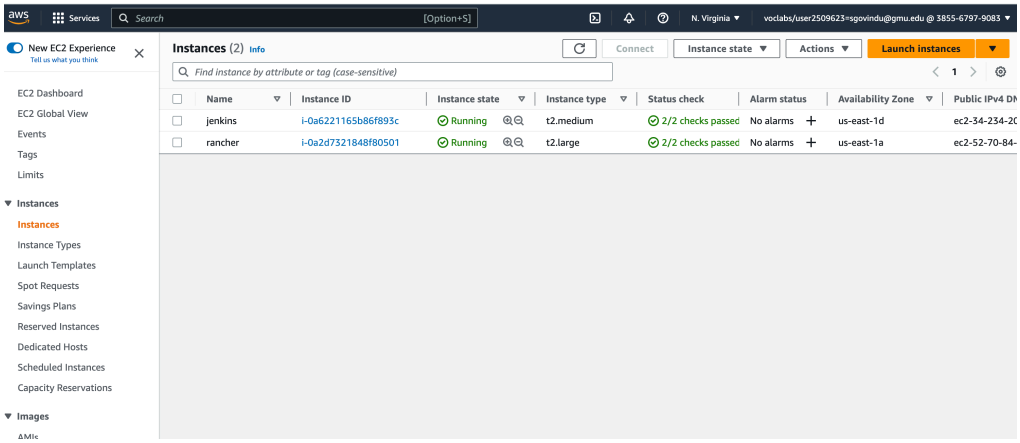

5. Launch the ubuntu instance with a unique name and t2.large as instance type.



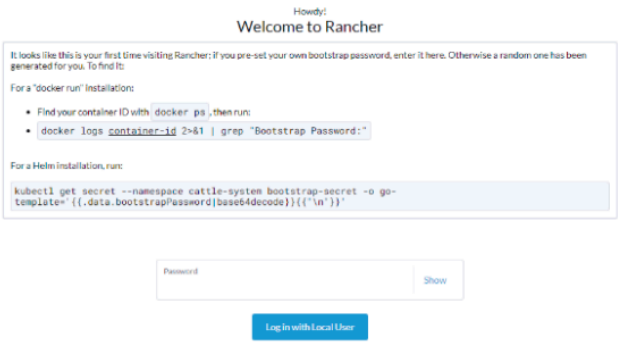

6. Save this key value pair or we can also use the pem file that we have used for our previous assignment. Then allow HTTPS and HTTP traffic from the internet and configure the storage upto 50 for root volume.
7. Now we need to Launch the instance.
8. After the instance is running, we need to get the public IPv4 DNS. Then, open putty by login in with the hostname of that DNS address and authenticating with the PPK key. ssh

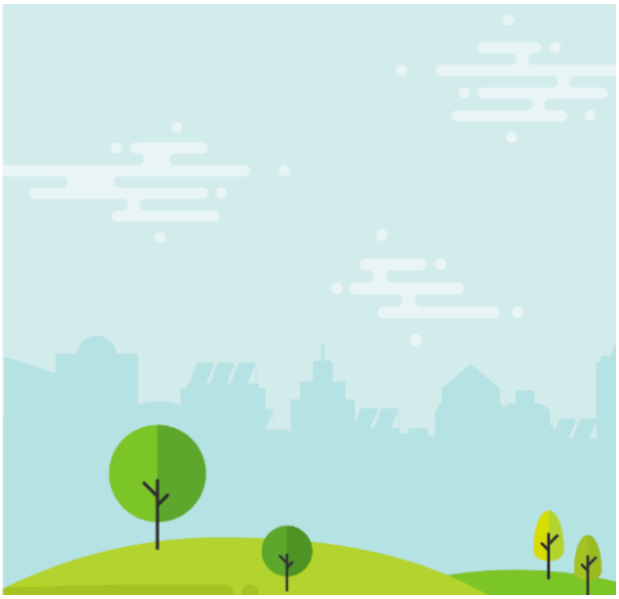
-i <YOUR KEY LOCATION> ubuntu@<YOUR PUBLIC IP>. (We might have to change the permission on your key to be read-only.)

PUBLIC IPV4 DNS - ec2-52-70-84-129.compute-1.amazonaws.com



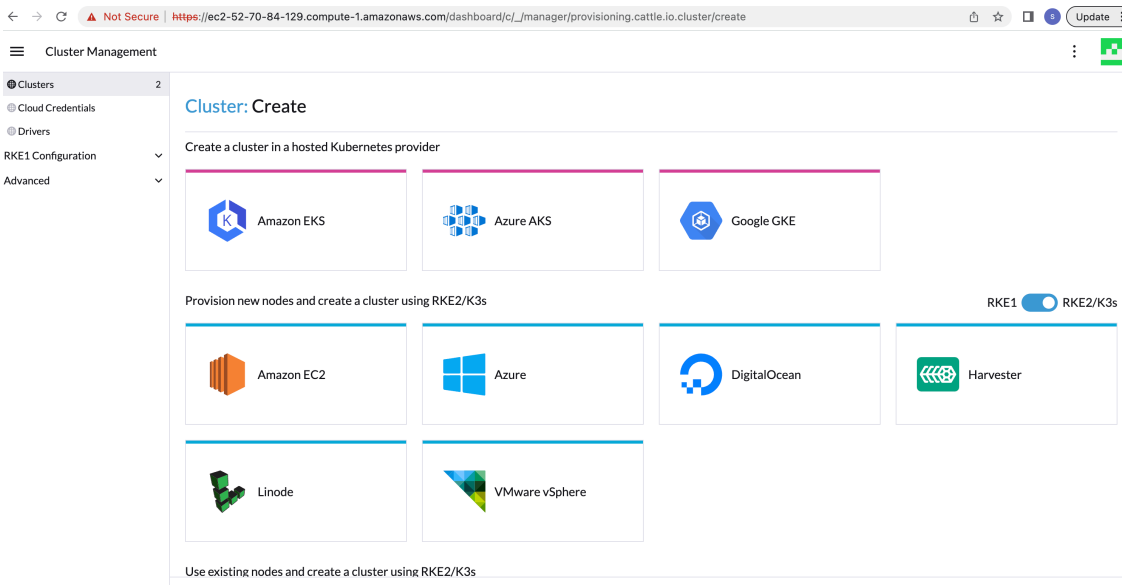

9. Now, we need to update the instance using the command - **sudo apt-get update**
10. Then we need to install Docker using the command - **sudo apt install docker.io**
11. Verify that docker is installed by running the command - **docker -v**
12. Then we need to run the docker command to install rancher - **sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher**
13. After waiting for some time we need to open the public IPv4 address and we will be redirected to the rancher page. We will be given few instructions to get our password. We can set our own password after logging in.



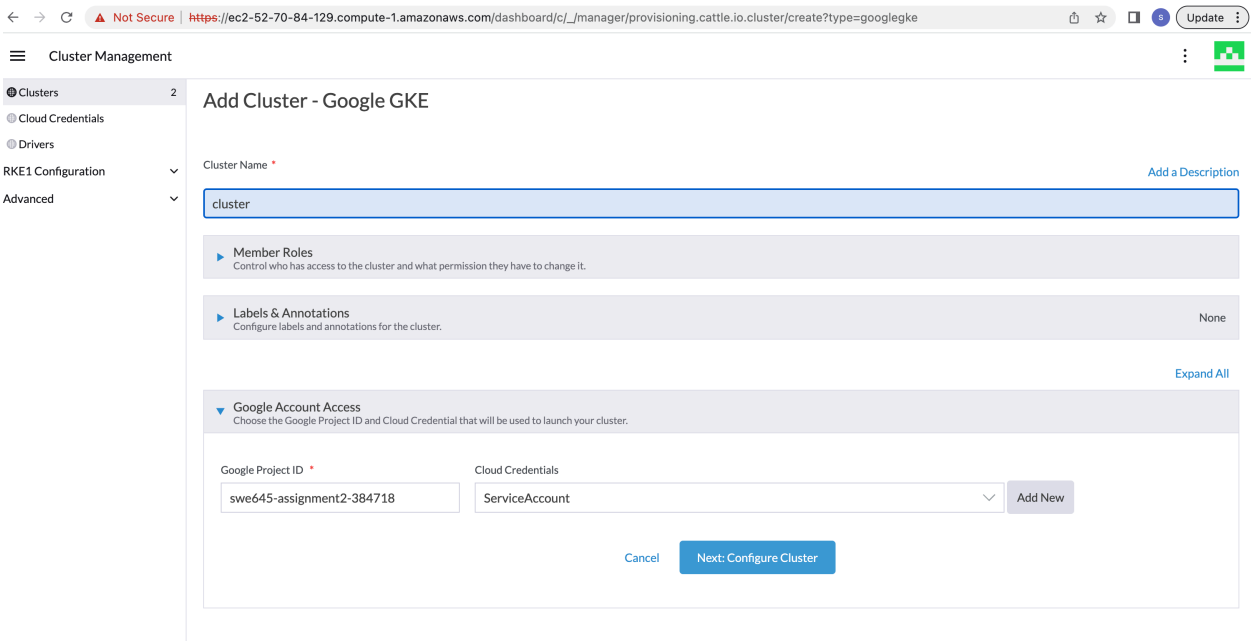



**PART 5 - Create a cluster using rancher:**

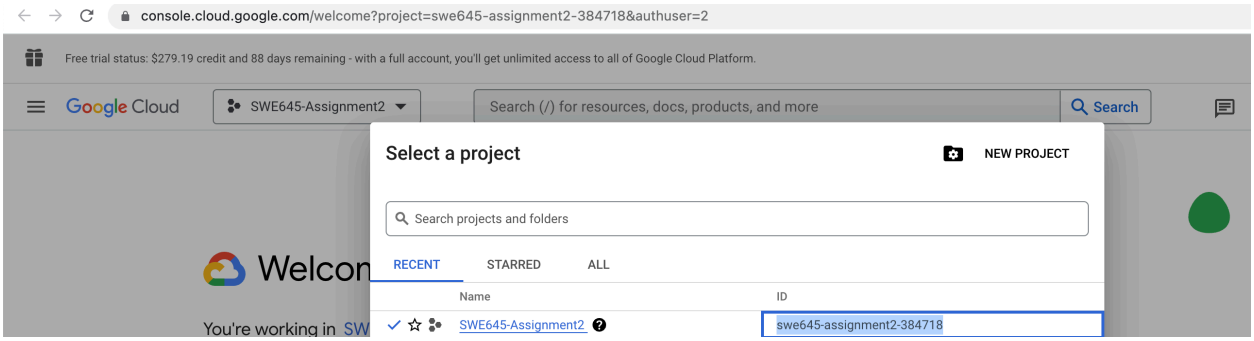
1. After we come up with the rancher page we need to login into rancher and for that we need to set up our own password.
2. We need to run ‘sudo docker ps’ command to get the container id and use the command in the above image to extract the Bootstrap password and then login successfully using our password.
3. After login we can see that a local cluster is already available in homepage.
4. Then we need to click on Create new cluster and choose Google GKE as an option.

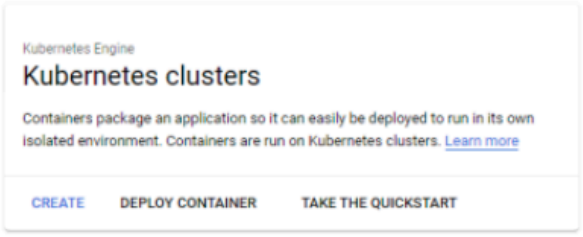



5. After that give a unique name to the cluster and assign a project ID to it using GCP - https://console.cloud.google.com/ and also create a Service Account using GCP and download the json file to create cluster.

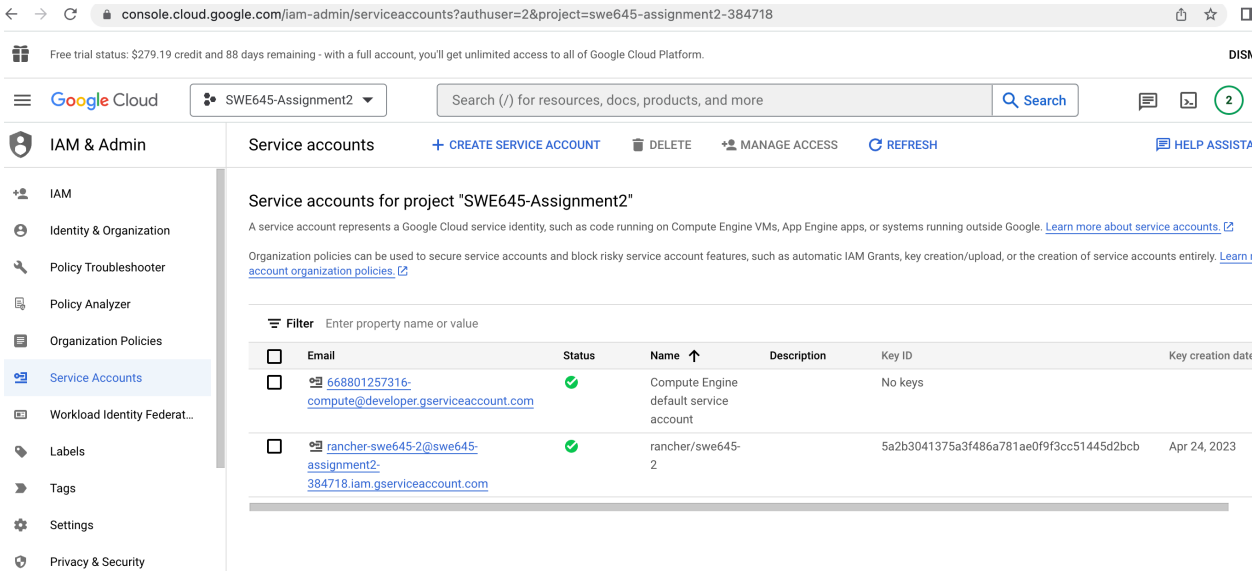


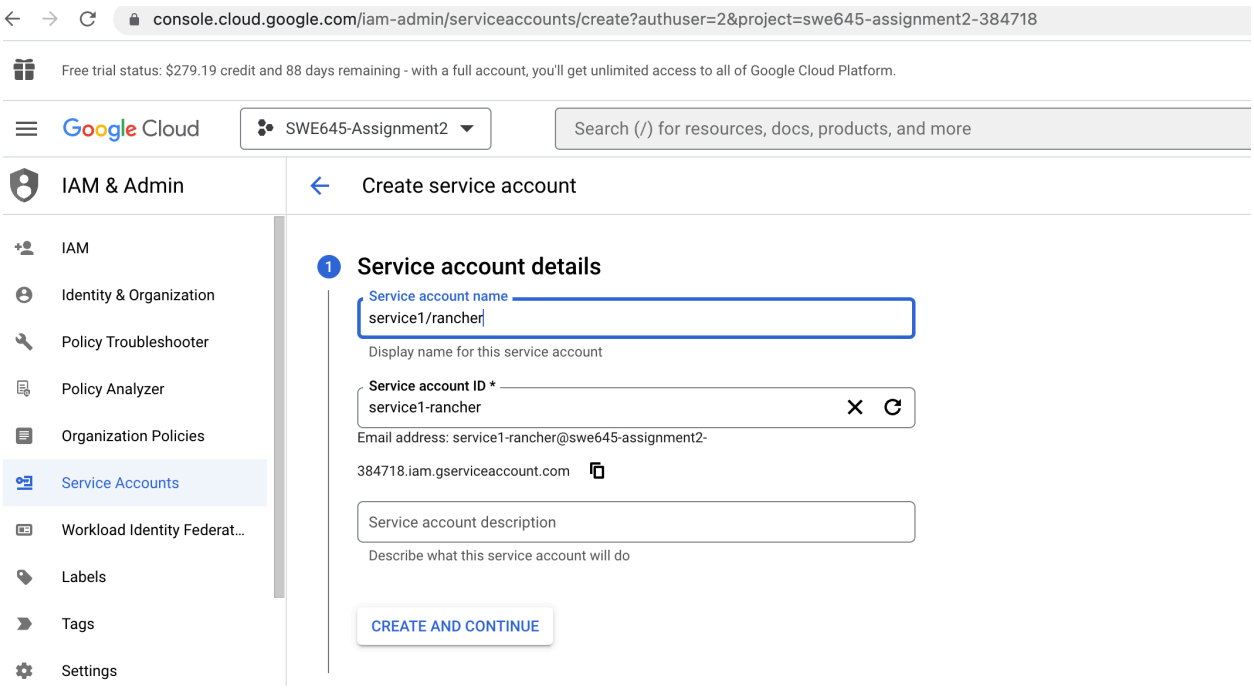

6. Create a project in GCP using free access and navigate to Kubernetes cluster and enable them.

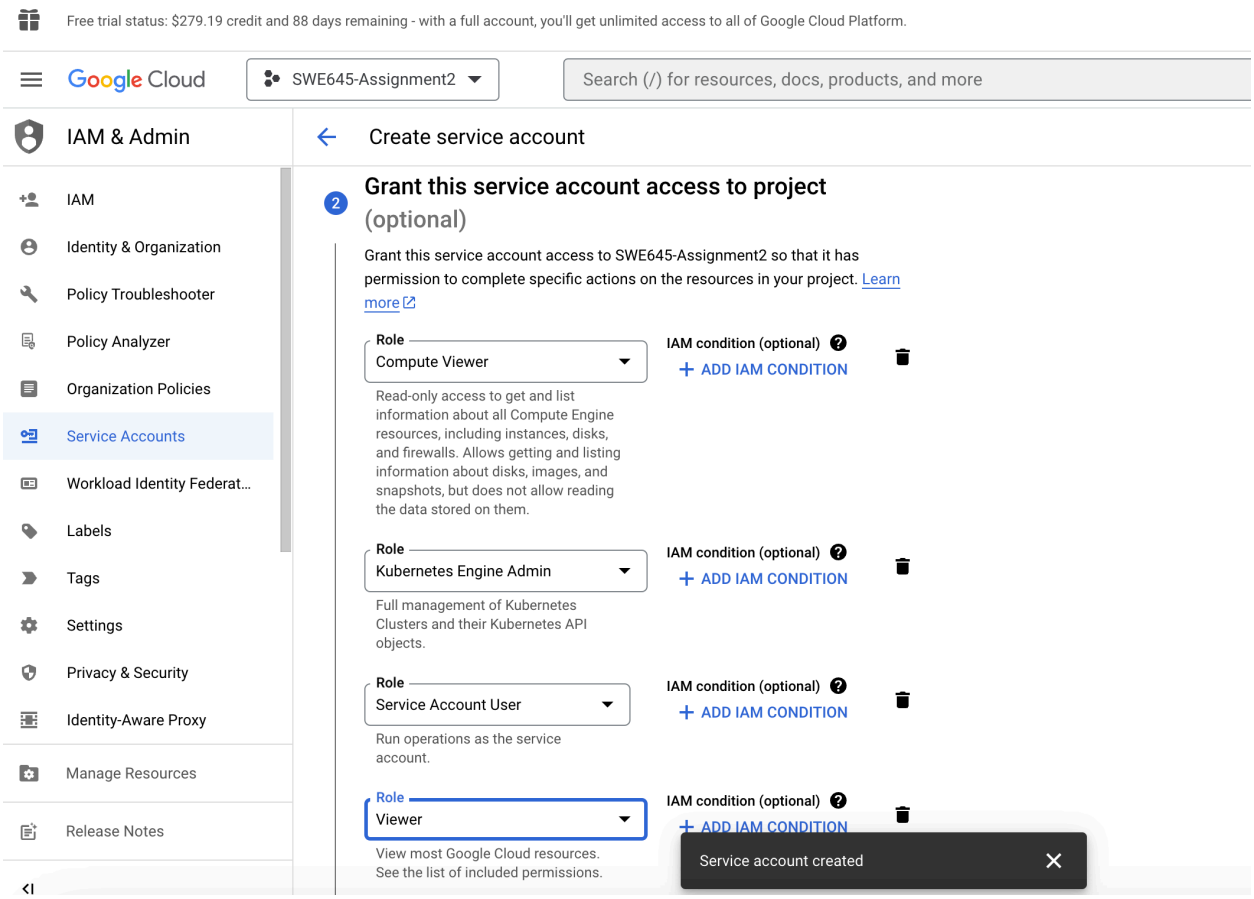


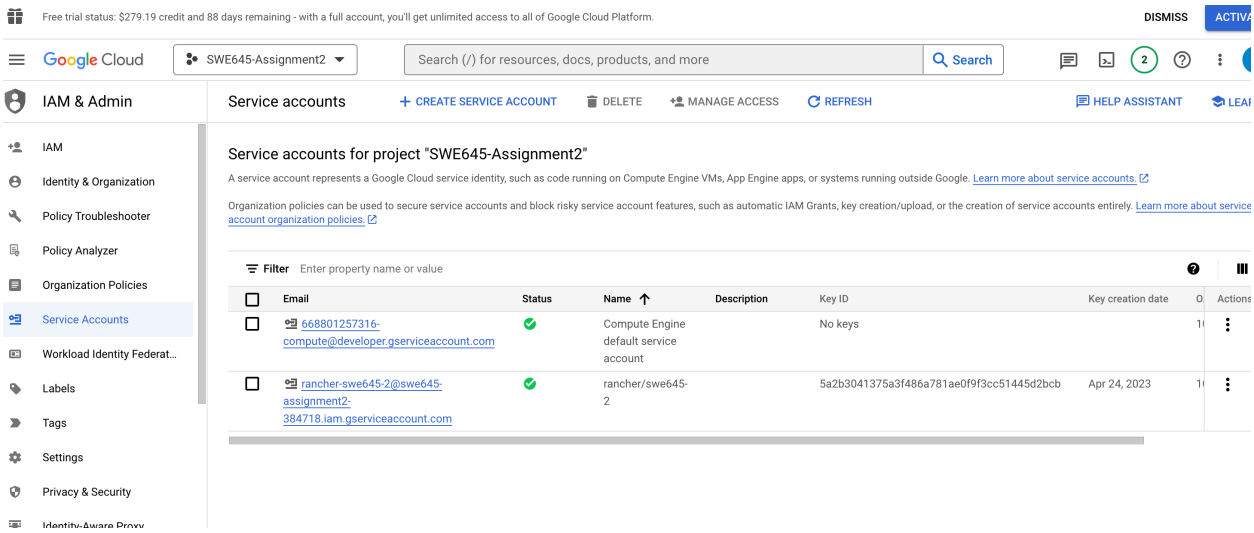
7. Now create a Service Account User required to create a cluster like this and give it a unique name and we need to assign roles like – Viewer, Kubernetes Engine admin, Compute Viewer and Service Account User and then click continue and done.



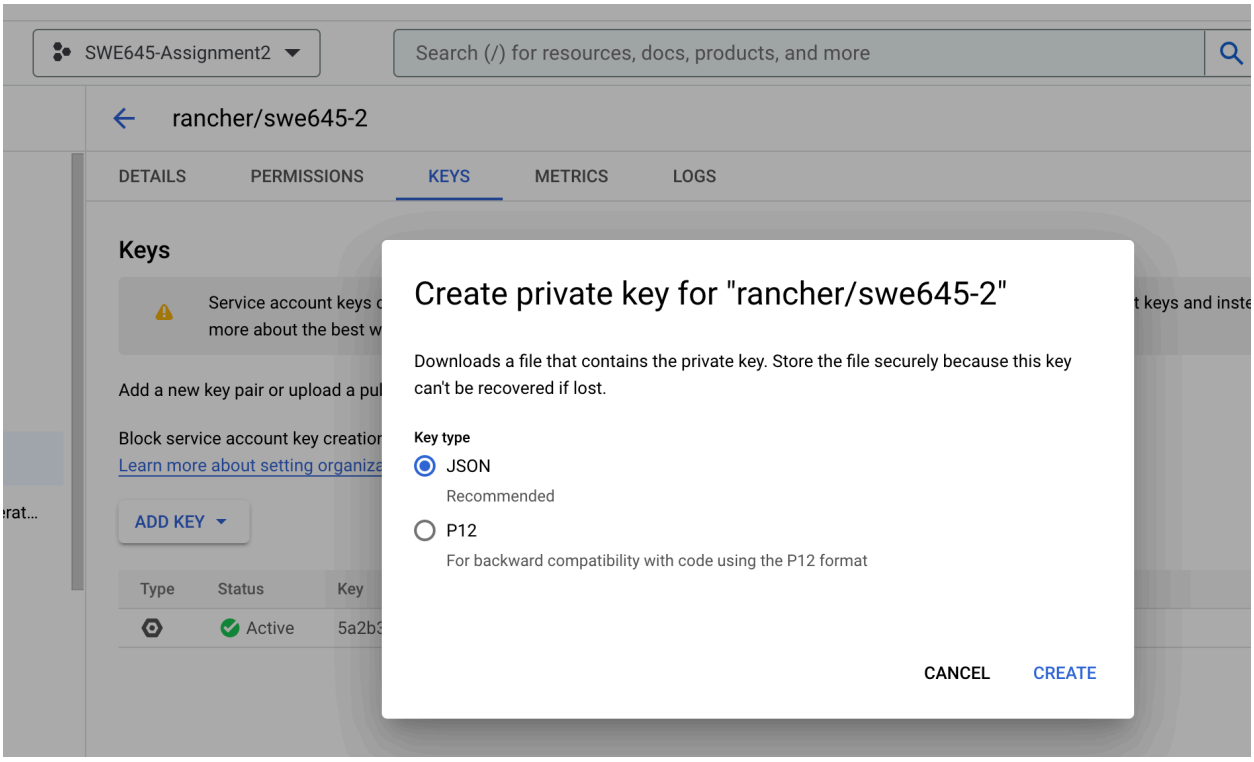



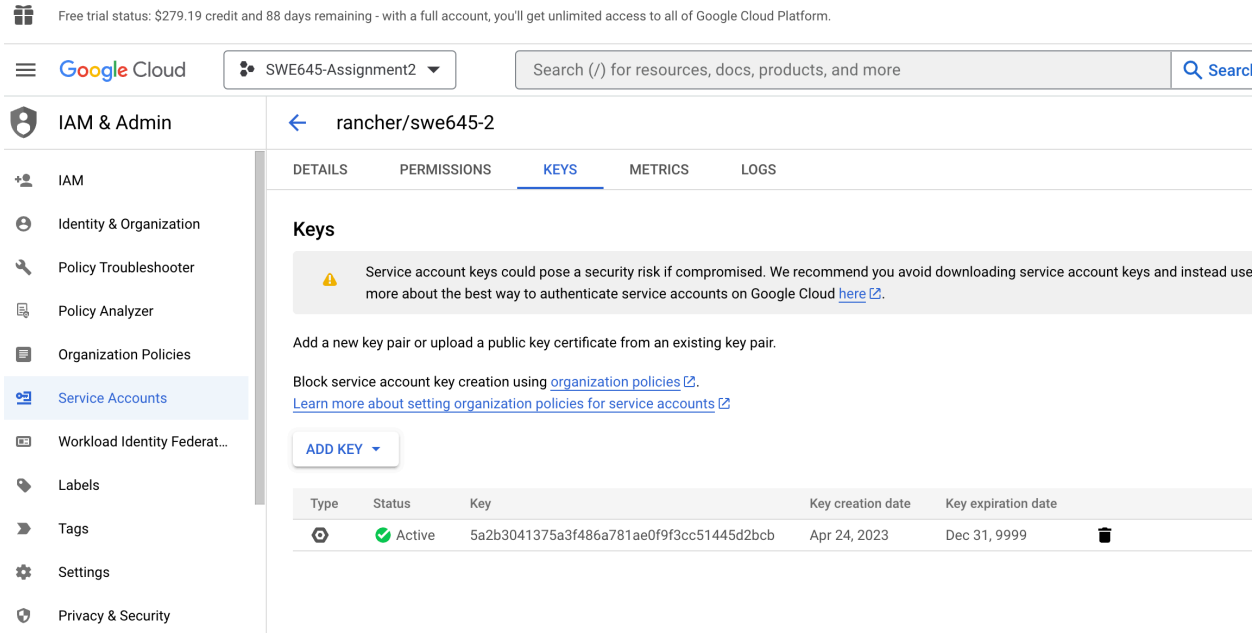



8. Now we can see that the service account got created and now we need to click on Manage keys from the three dot icon beside the service.

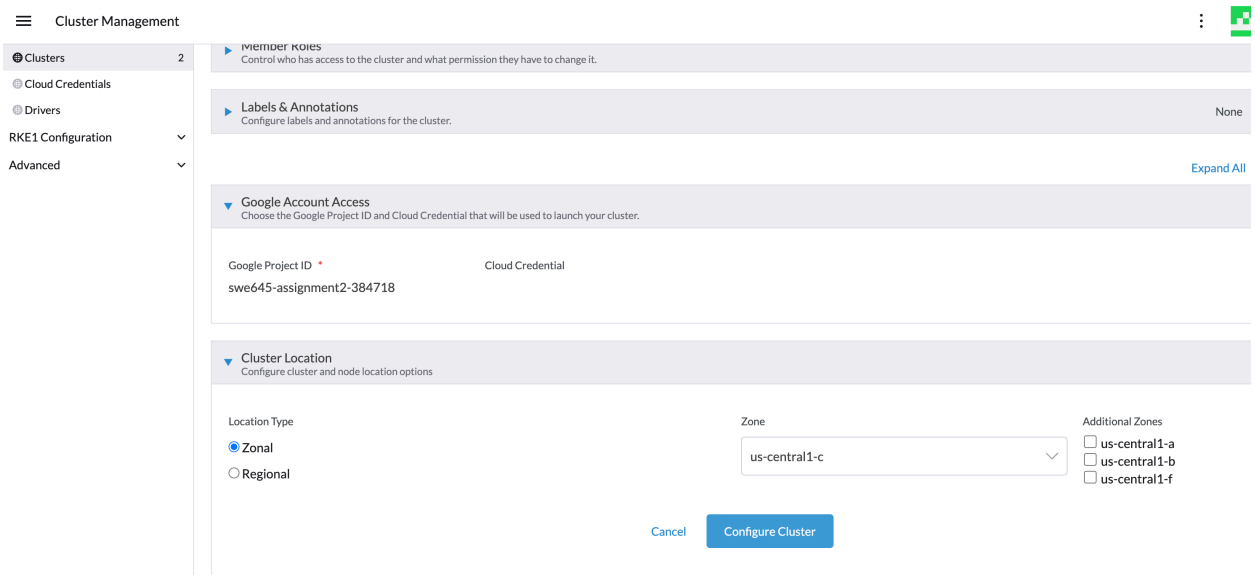


9. Now click on Add key and create a JSON key.

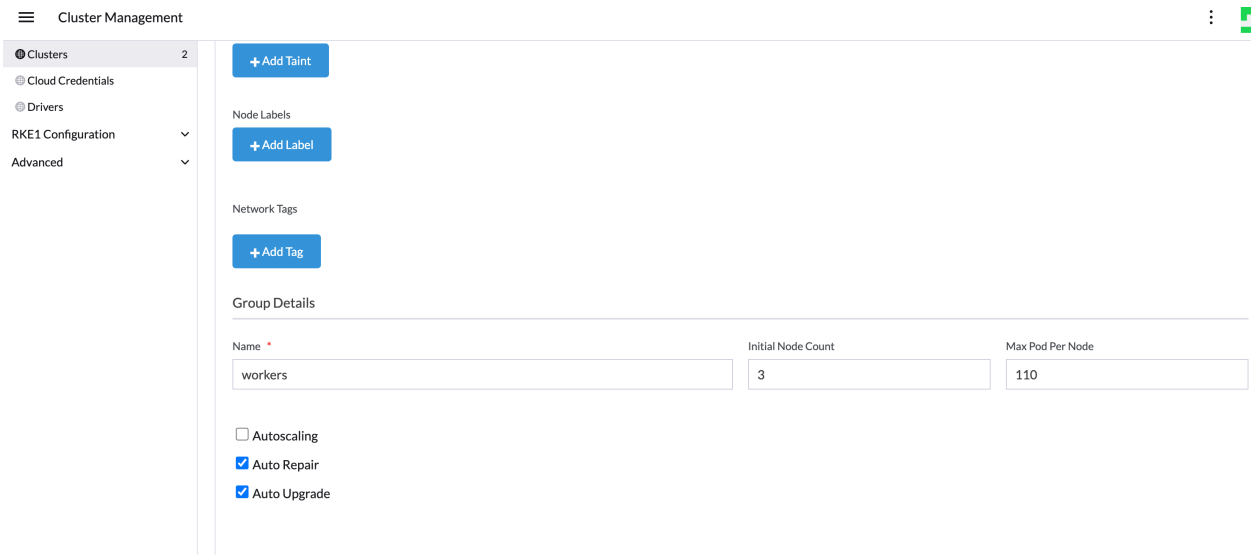


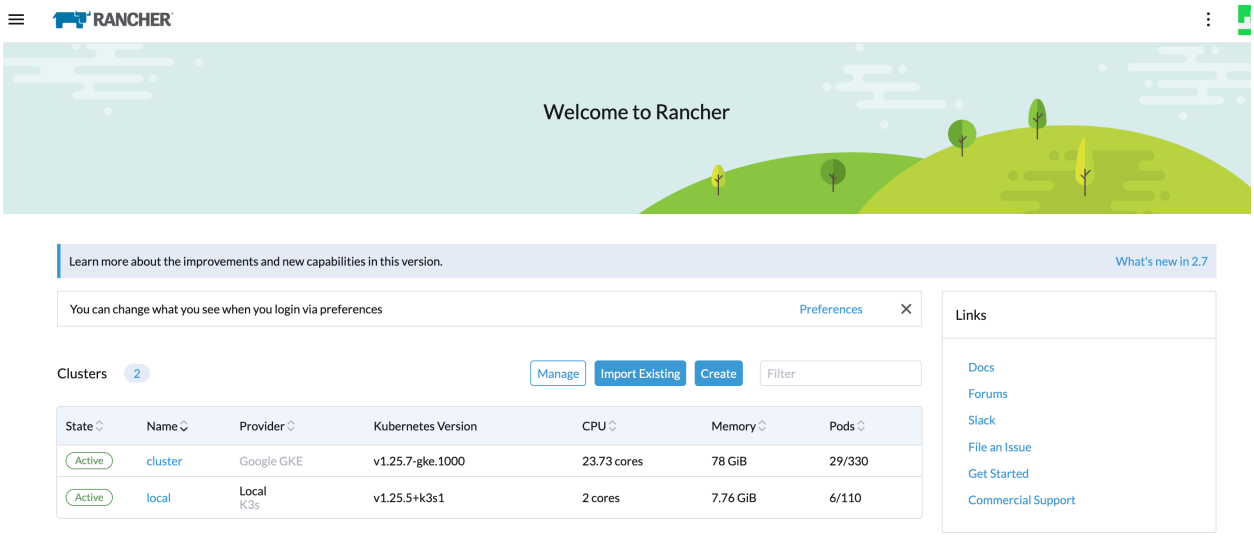

10. Now add the service account to the cluster in rancher by selecting “Read from a file” in rancher and select the downloaded JSON file from our local machine and lick on next.
11. Then choose and confirm the local zones and lick on Configure cluster.




12. Then under Node pools, name the node pool as “worker” and set the initial count to 3 and leave the remaining settings as default.

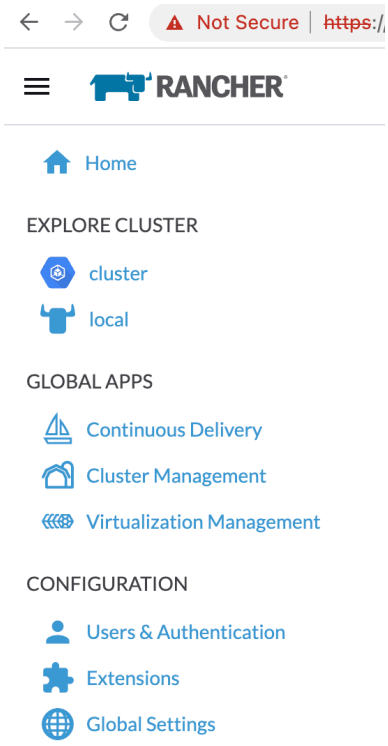



13. It will take 5-10 min to create a cluster from provisioning state and in case of error view the logs and create it again.



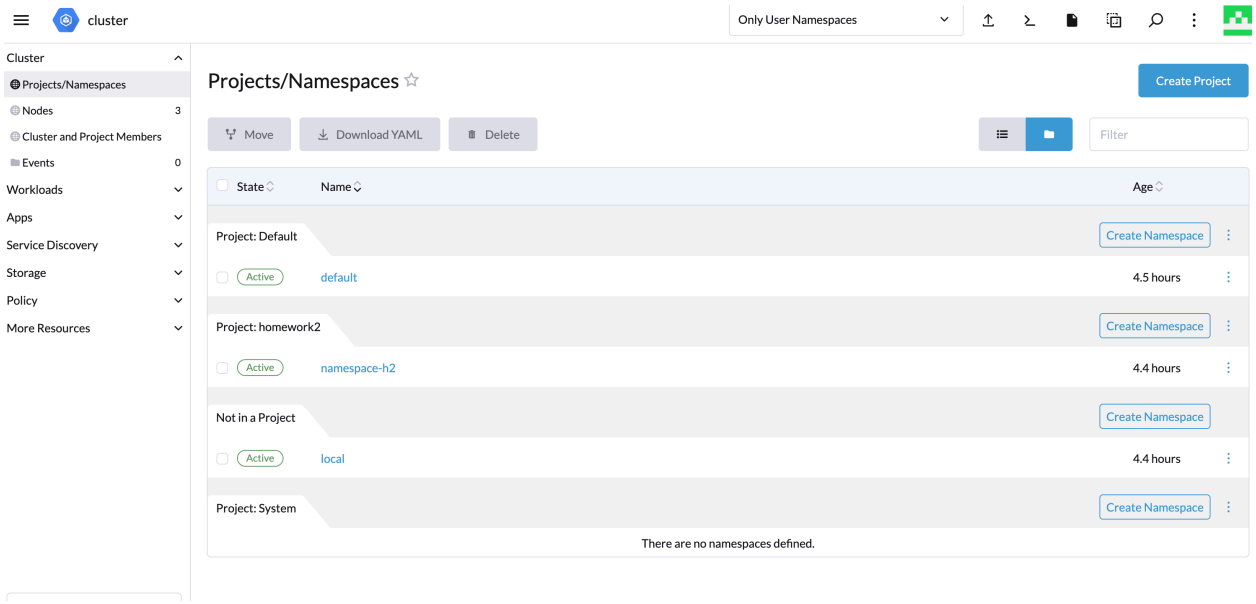

## PART 6 – DEPLOYING DOCKER IMAGE IN RANCHER/ CLUSTER

1. After creation of cluster, navigate to the hamburger menu in rancher and select the cluster that you created.

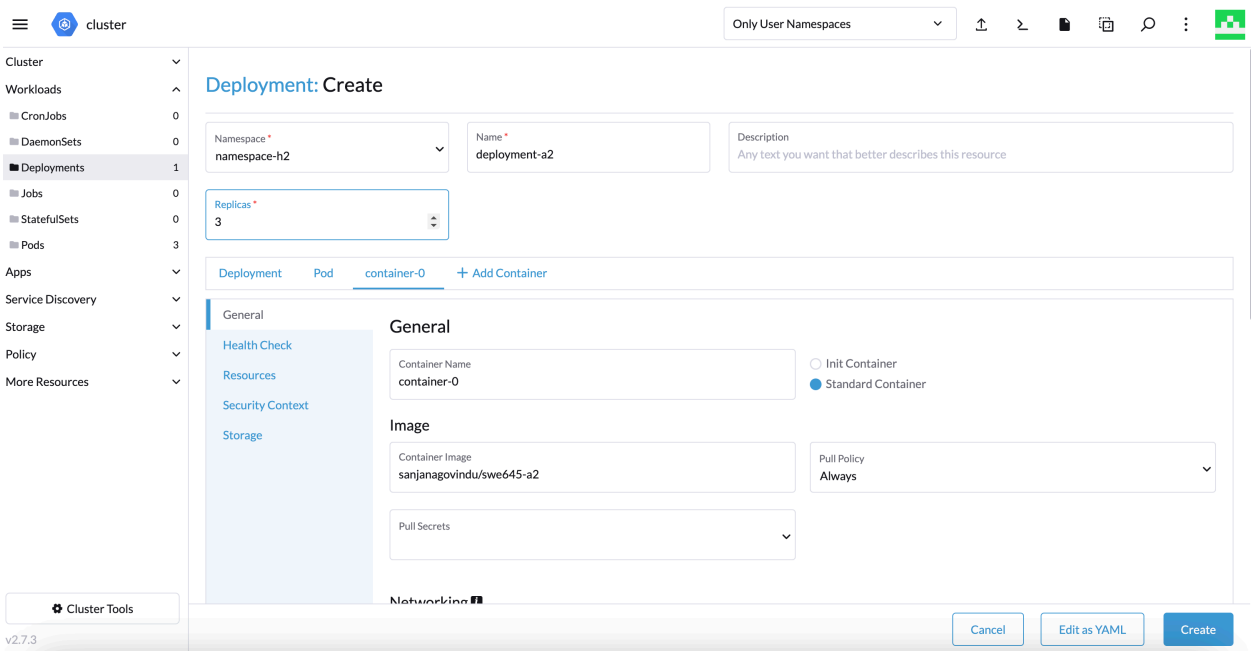



2. Select the cluster from the hamburger menu and navigate to “Projects/namespace” in rancher.

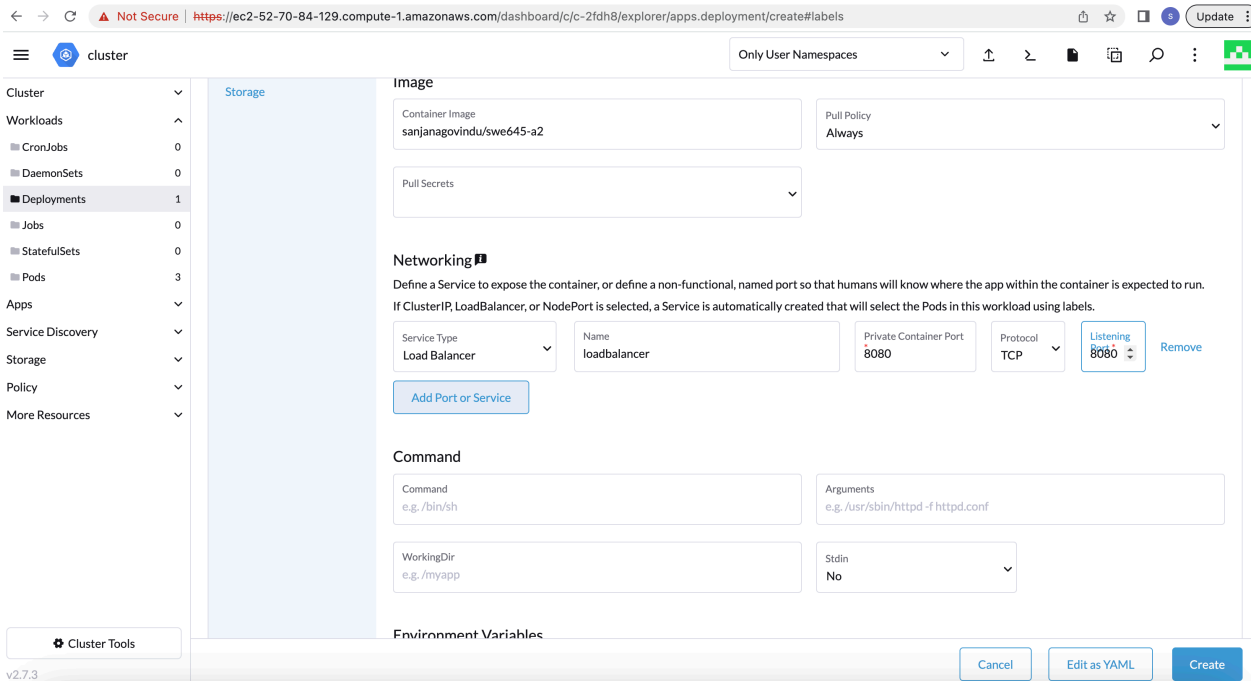
3. Now create a project and also a namespace under that project.



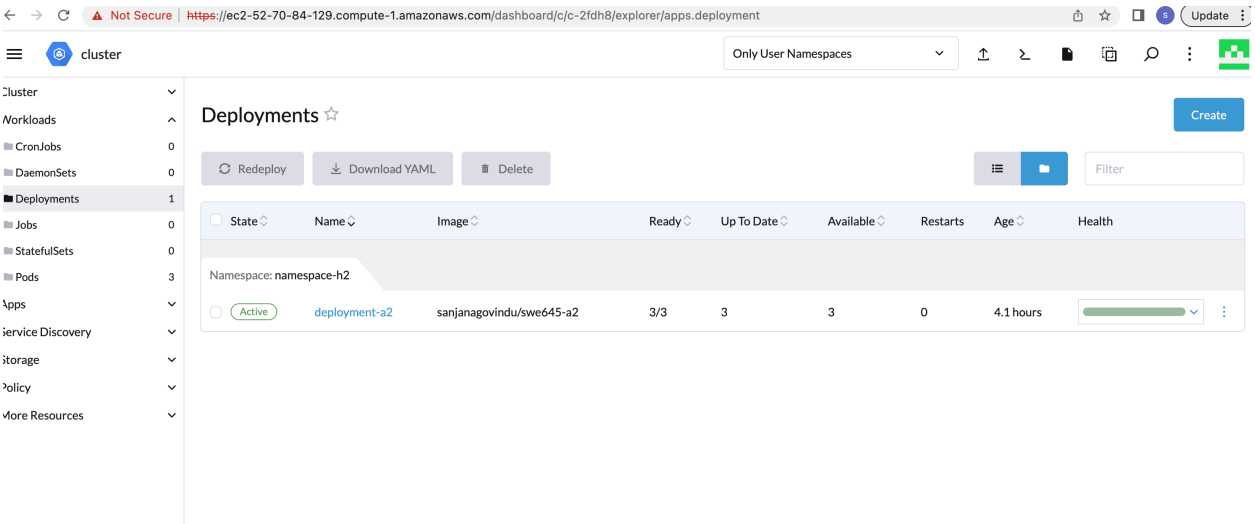

4. Now navigate to workloads tab and click on deployments to create a new deployment.
5. Add a new deployment name and select the namespace that we created and add 3 replicas to the deployment.
6. Then add the container image name to the deployment and add port or service.

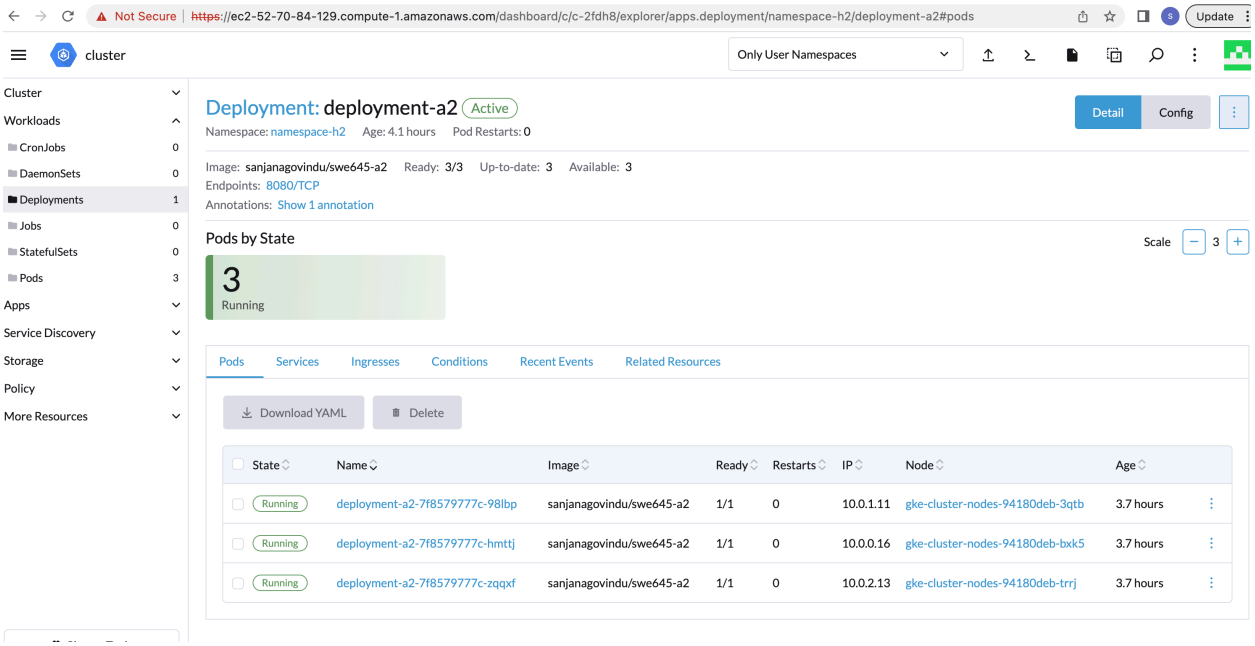


7. Then under the ‘Networking’ tab add the Service Type as ‘Load Balancer’ and give it a name and add the private container port and listening port as 8080 by leaving the protocol as TCP.

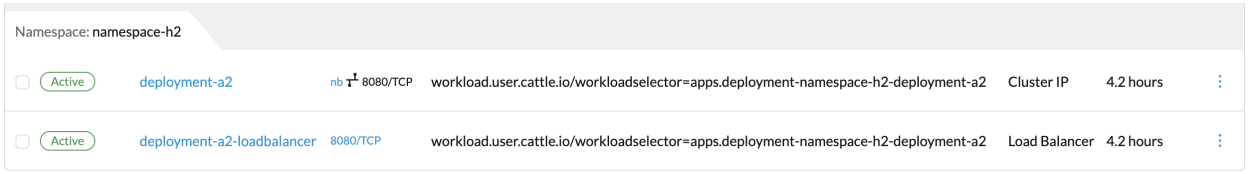


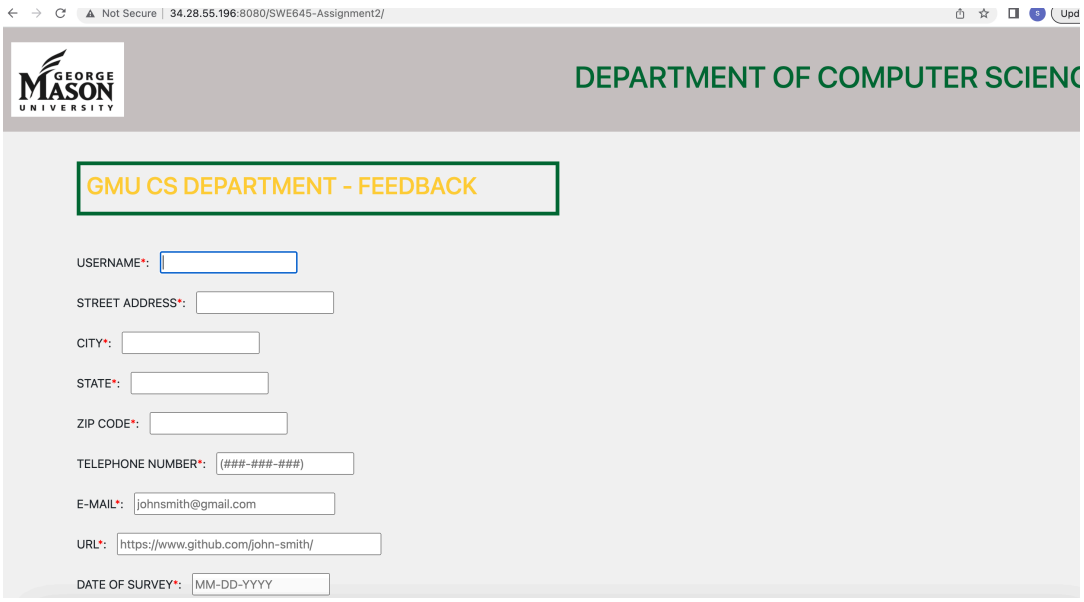

8. Leave the Deployment and Pod settings to default and create a deployment. It will usually take 5 minutes time to get created.





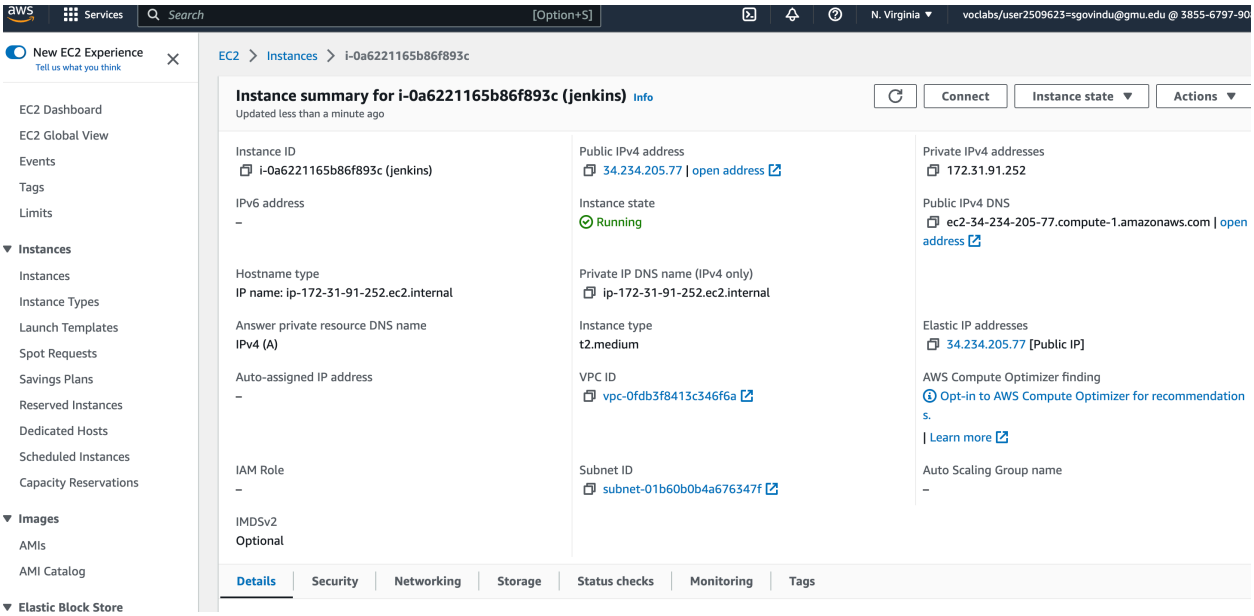

9. Confirm if the deployment works by selecting the deployment-a2 service or by clicking on "8080/TCP" endpoint and add your WAR file name to the link to access your survey form.

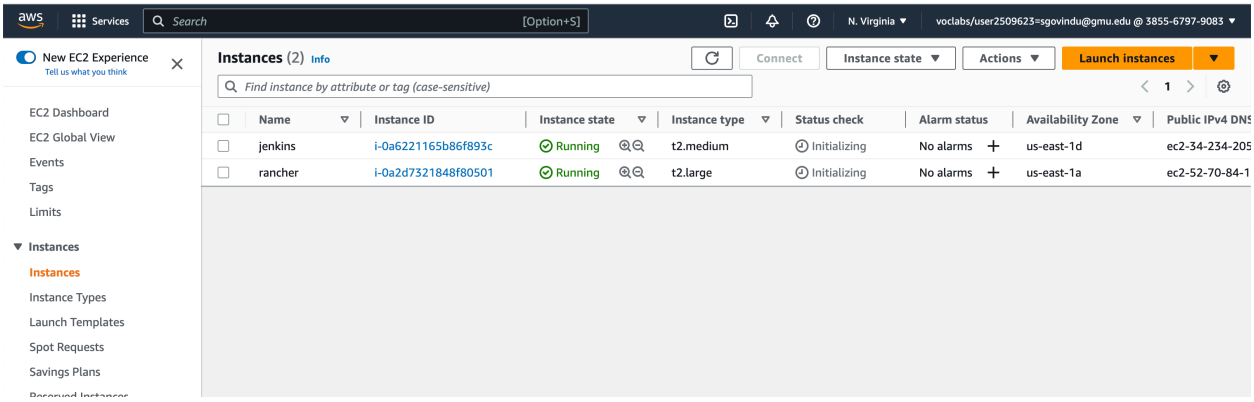


## PART 7 – INSTALL AND SETUP JENKINS

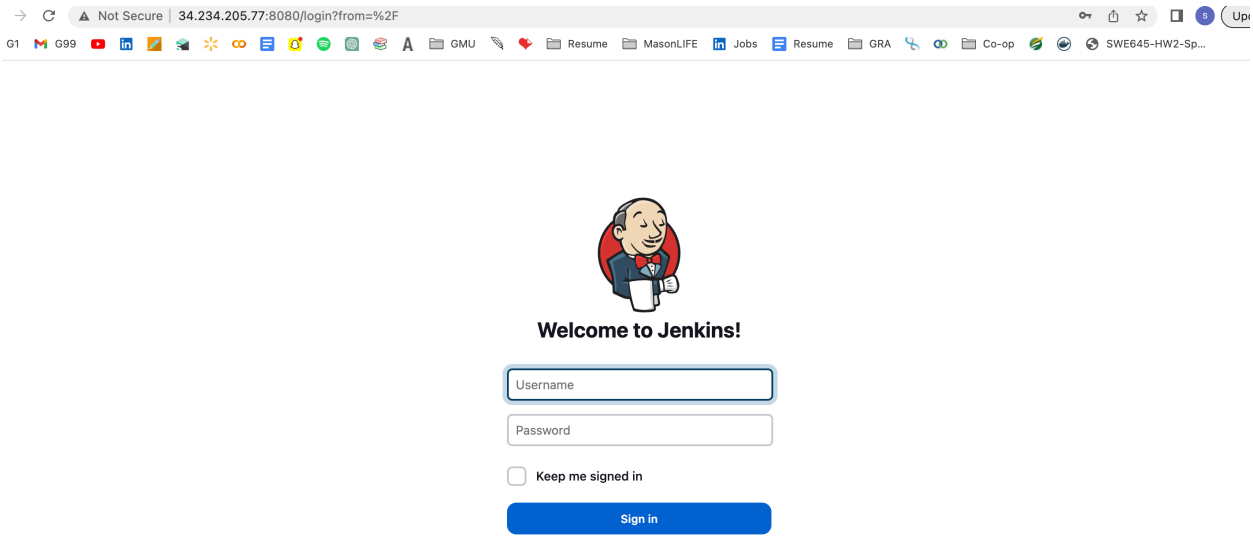
1. For setting up Jenkins we need to create another EC2 instance in AWS Lab
2. EC2 > Launch Instance > Give a name > Select Ubuntu > Add the existing pem file > Allow traffic from HTTP and HTTPs and confirgure volume to 50 > launch instance (basically follow the same steps in Part 3 as we did for rancher)



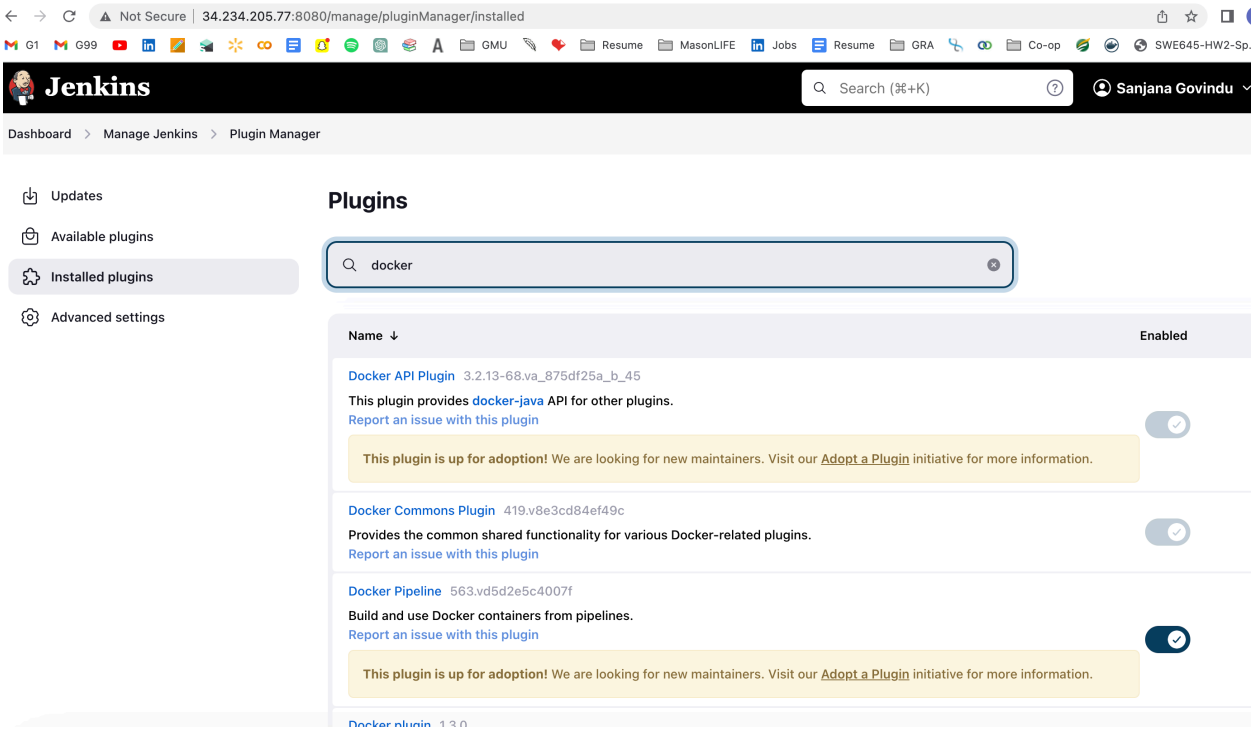



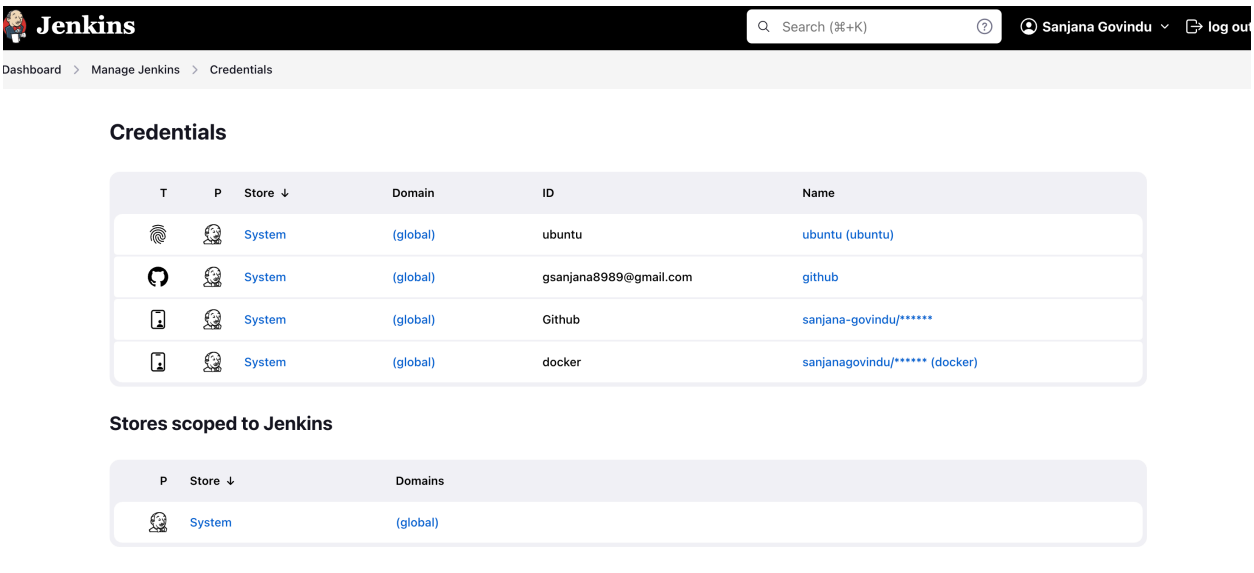

3. Then after instance is launched install docker and add the jenkins user to the docker group using the command which enables using docker commands without using sudo – "sudo usermod -aG docker jenkins"

4. Install JDK 11 on EC2 instance in AWS itself by clicking on SSH connect and run this command if the key is not publicly viewable - chmod 400 swe645-assign2.pem and for jdk use - sudo apt install openjdk-11-jdk" and check the java version installed.
5. Use https://www.jenkins.io/doc/book/installing/linux/#long-term-support-release to install Jenkins on local machine and follow the steps.
6. Now we need to verify if Jenkins is installed in our local machine running "systemctl status jenkins"
7. To enable continuous deployment through Jenkins we need to install kubectl using commands - "sudo apt install snapd" and "sudo snap install kubectl --classic"
8. We need to got to rancher again and download the cluster configuration for Jenkins and switch to Jenkins users using "sudo su jenkins"
9. We need to create a file using vi editor and paste the downloaded cluster configuration YAML which is "$HOME/.kube/config"
10. We need to veryify if kubectl and the configuration is working correctly using command "kubectl config current-context"
11. Output will be the name of the cluster if verified correctly.
12. We need to use the configured elastic IP within the port 8080 to access Jenkins from the browser and follow the steps to setup user name and password for Jenkins.

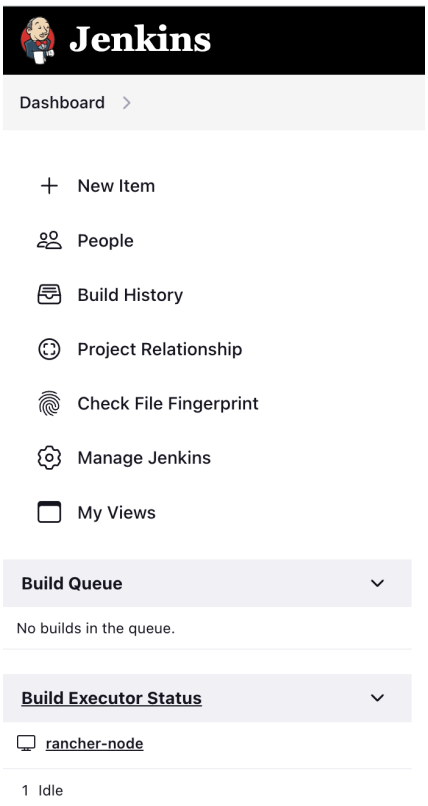


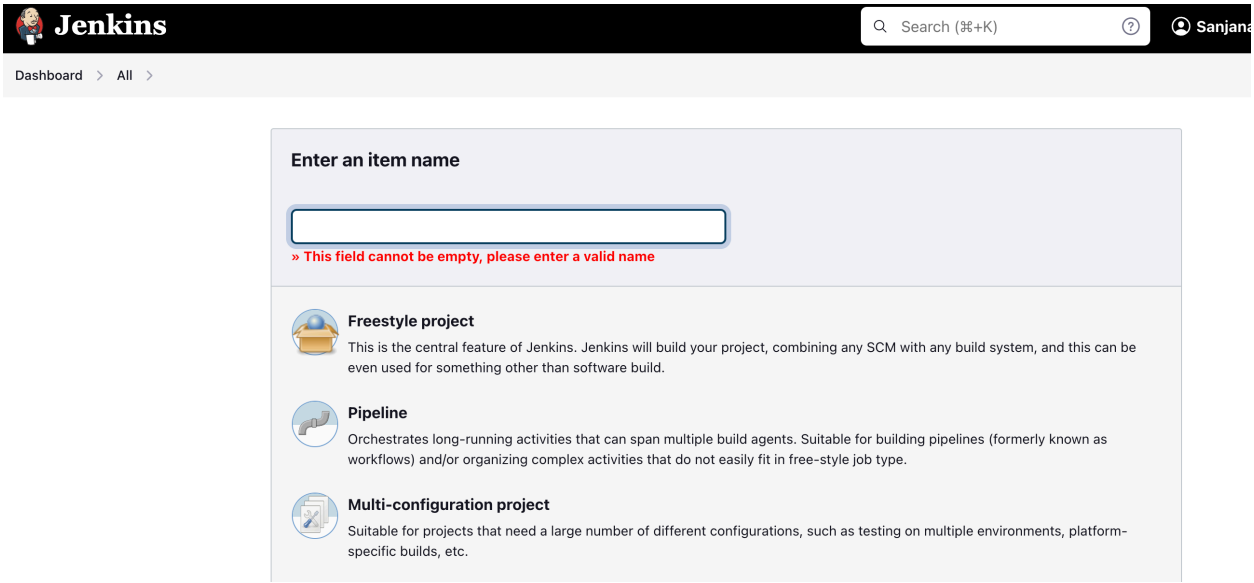

13. Then we need to setup the credentials for docker and github by nagivating to Manage Jenkins > Manage Plugins > Install the docker, git, githib, pipeline plugins then Manage Jenkins > Manage Credentials to add the credentials.



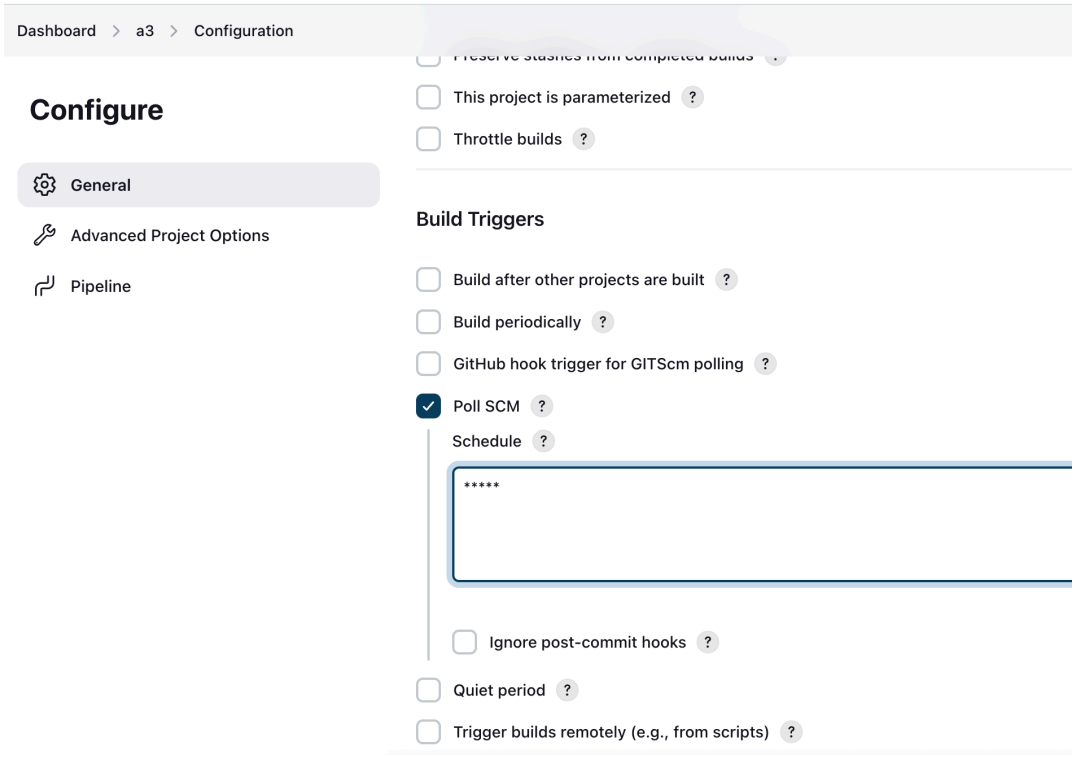




14. Then we go to dashboard and click on ‘New Item’ to create a pipeline and name the pipeline as a2 for this assignment.

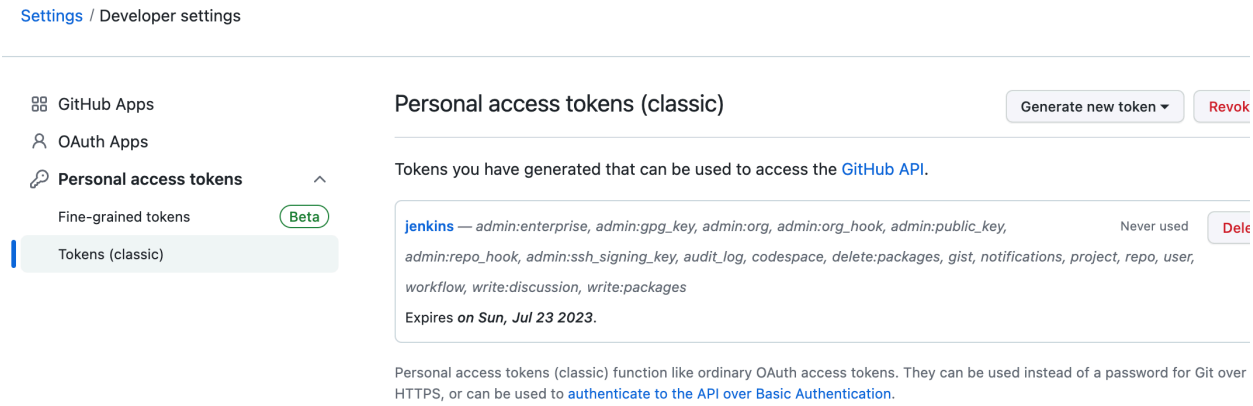


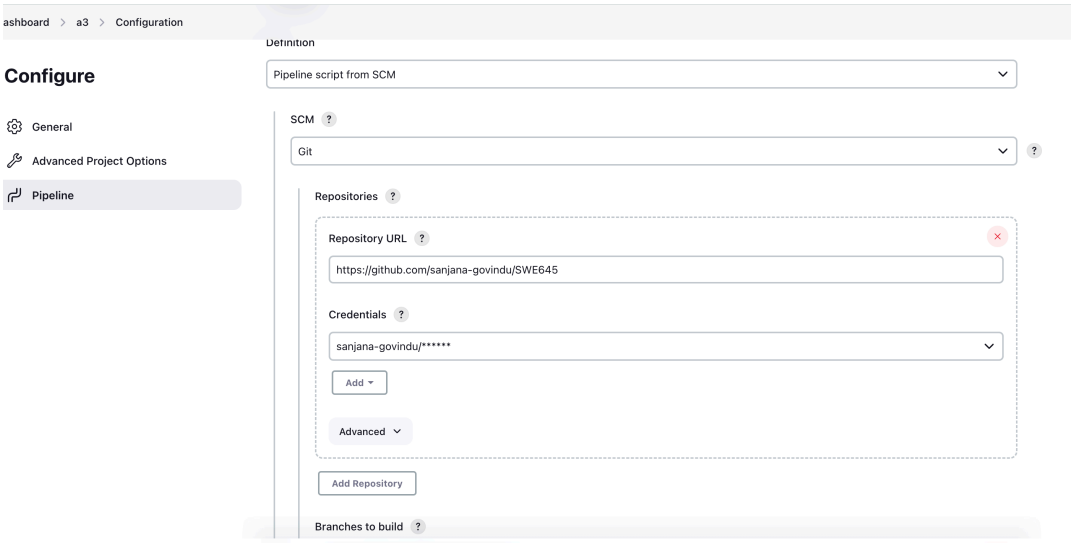




15. In the configuration, set the “Build Triggers” by enabling the “Poll SCM” and add “*****” so that the repository can be checked in every minute.



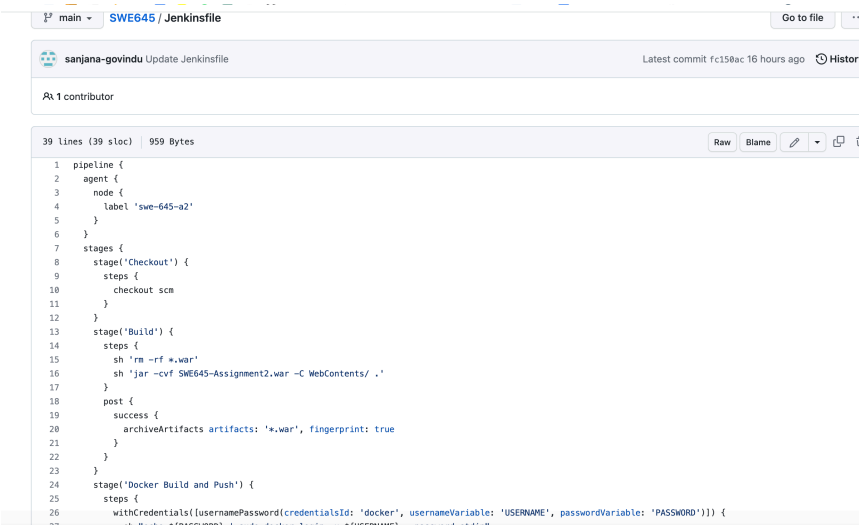

16. Scroll down and in Pipeline set it to “Pipeline script from SCM” and select Git > Provide the Github repository URL and for GitHub personal access token should be used instead of the password and it must have read and write access to access the GitHub repository.

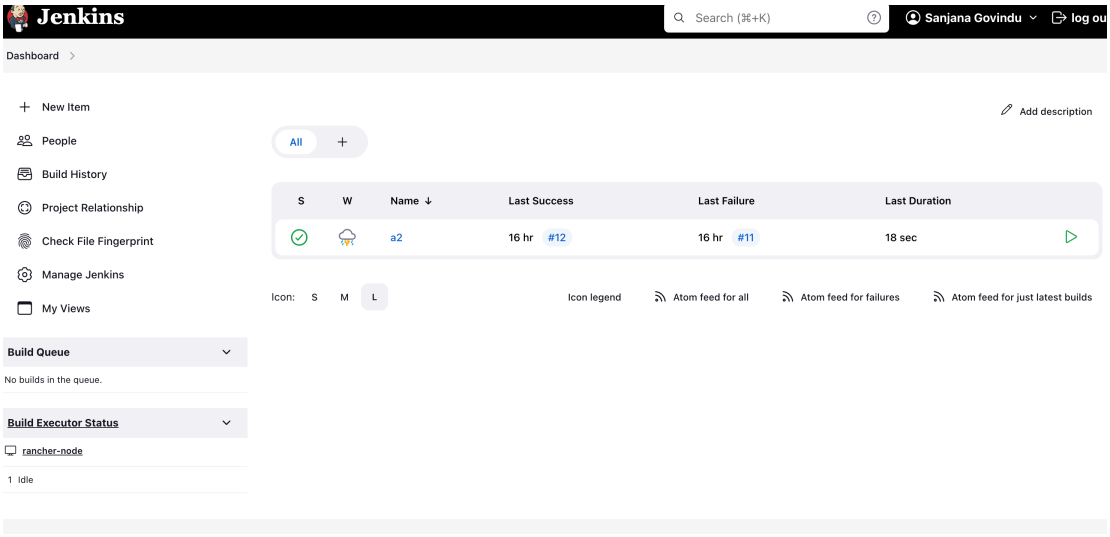





17. Leave the script path in our Jenkins file in the root directory and then uncheck the Lightweight checkout.

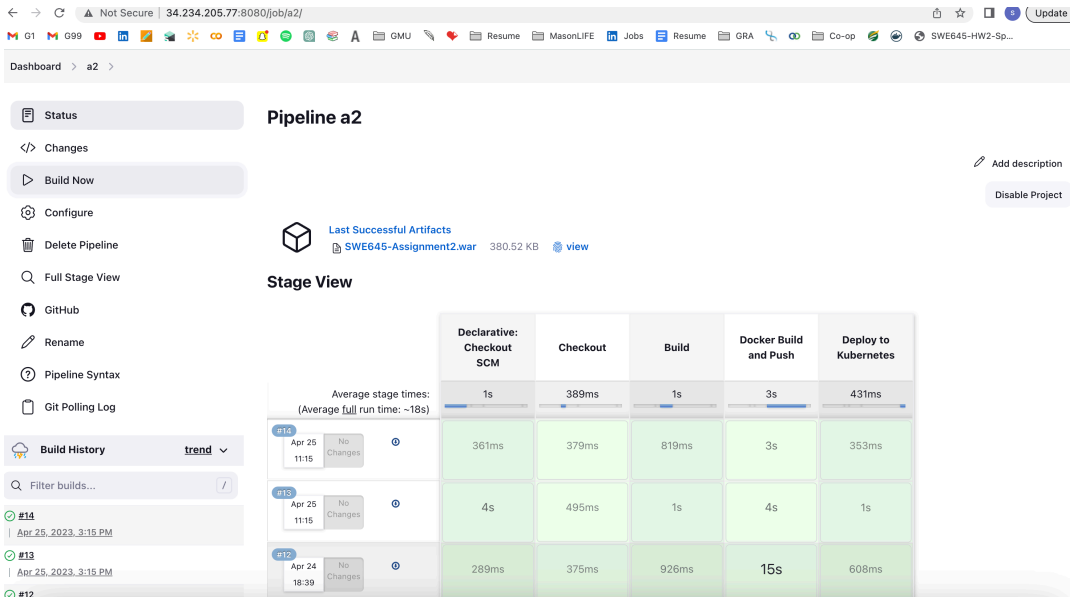

18. Leave everything else to default and create the pipeline.
19. The Jenkins File in Github is like this:





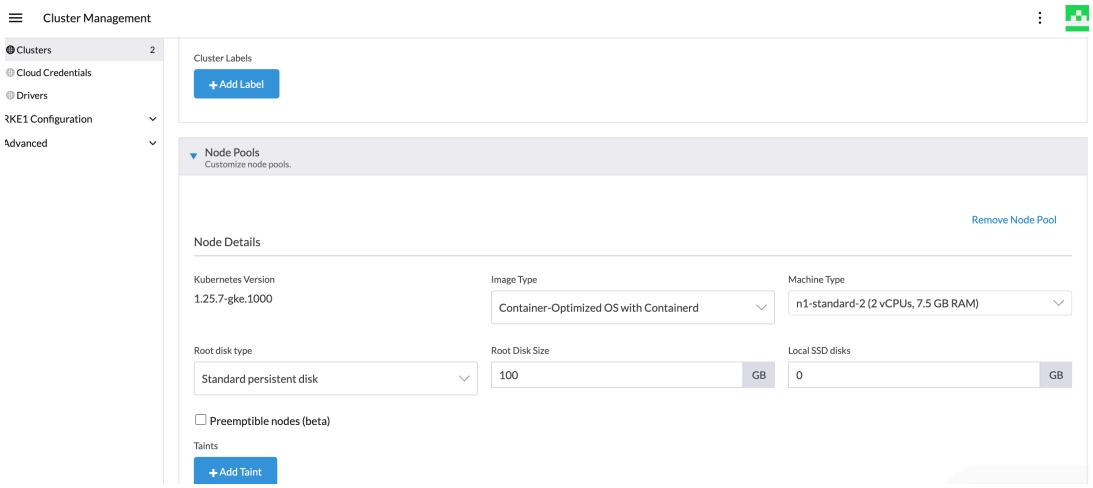

It retrieves the configured configured credentials of docker and in first stage the repository is cloned and previous WAR files are removed and new WAR is generated along with a new build of the docker image and then the docker image is pushed to docker hub and it verifies if the dockerhub credentials are correct here. Then the deployment is restarted and it will check a d pull image from docker hub with latest updates and we can observe the deployment in rancher and can see that the pods are restarted. Finally the pipeline is created here and everything is shown as green.We can see the final pipeline in Jenkins and a check if everything is configured correctly. The pipeline will be built and triggered and chanced will be deployed and we can see a version change in the webpage. If any errors are encountered we need to check the logs and correct them.

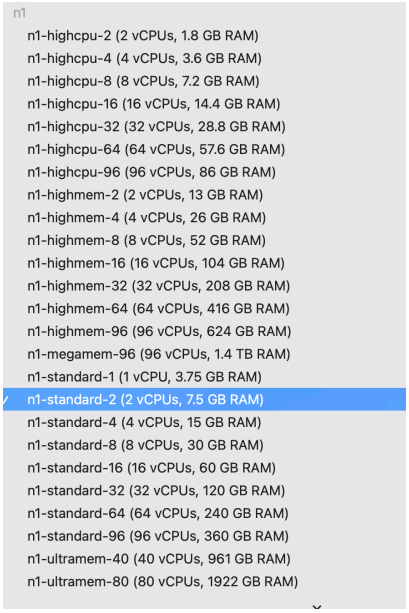
**STAGE VIEW:**



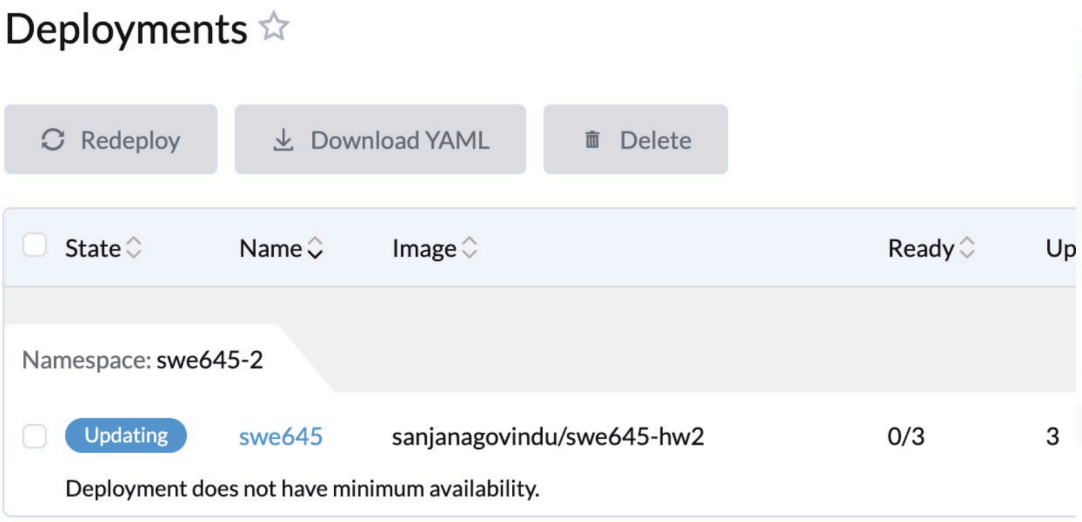
## CHALLENGES FACED DURING THIS ASSIGNMENT/ LESSONS LEARNED:

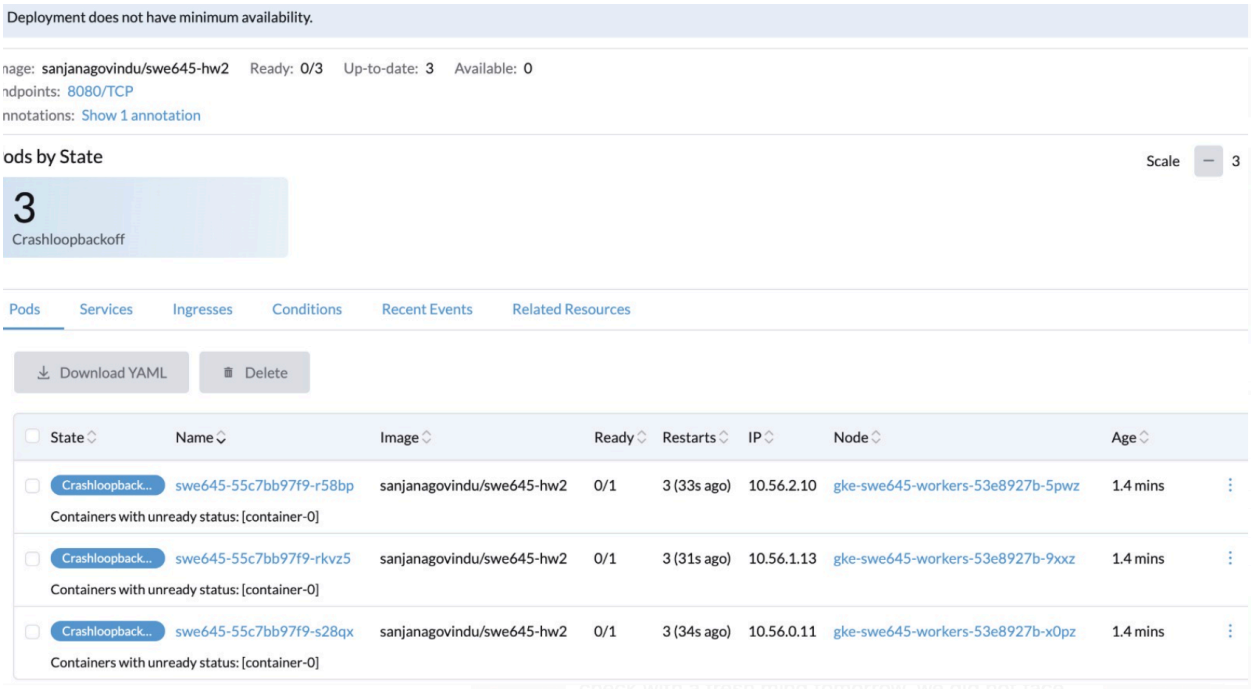
- In this assignment we faced a major issue while pushing the image to docker hub and running it locally. We are unable to build the docker image locally properly if we use FROM tomcat or FROM tomcat-10.4-jdk11 in DockerFile so we need to use FROM amd64/tomcat for sure if we are using MAC OS.
- While creating an EC2 instance, we were facing an issue if the instance type was set to t2.medium so we used t2.large and we also need to configure the volume to greater than 30 here we used 50. If its set to default the instance is not working properly for later use.
- And while creating cluster we need to set the node pools machine type to a higher version so that we don't face an issue and the cluster can handle it.

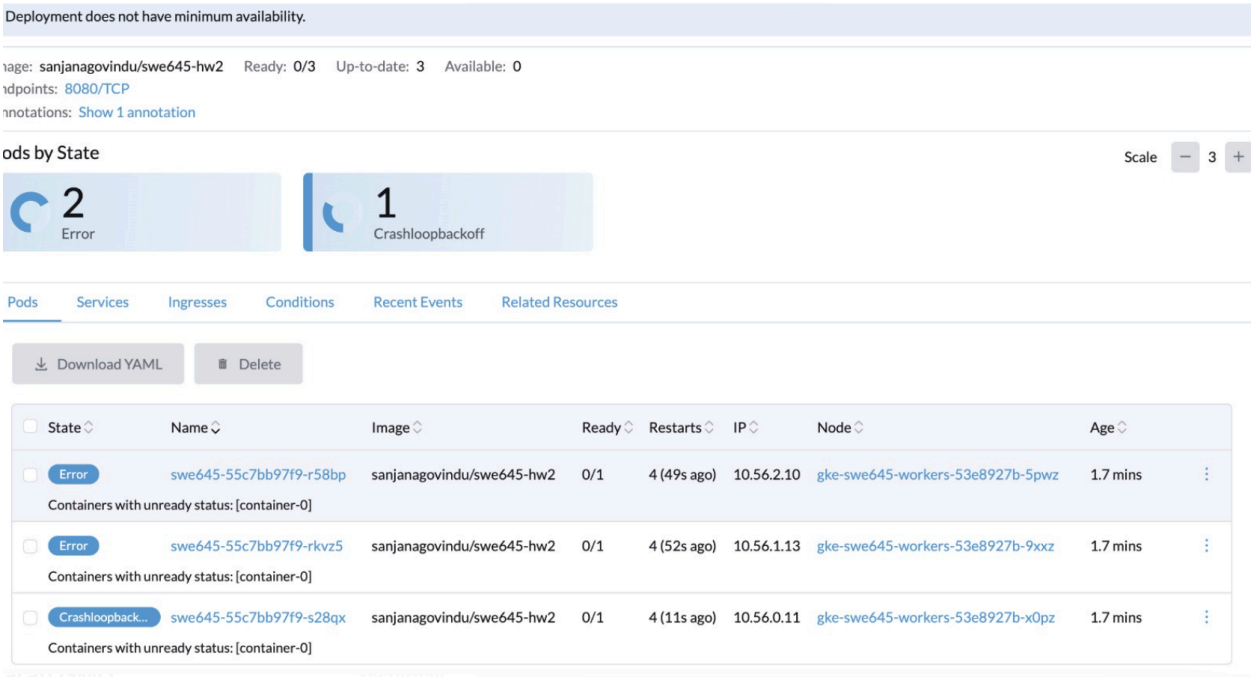


- So if we don't set the machine type to a higher version we can't create a deployment an sot shows cashback loop errors where the pods fail and we need to start everything from the beginning and clear cache. Below are the deployment errors. So we need to choose the machine type carefully so that it will be suitable for our assignment.
- In my case if we use a very high-end machine the google cloud platform doesn't have enough memory to handle the deployment and then it will show quota is not enough for a free account so we had to try creating the cluster multiple times to handle our deployment.






**Role of team members in this assignment:**

- **ADITEE**: Worked on containerizing the web application using docker and then pushed the image into Docker Hub to make it globally accessible for further use.
- **SANJANA**: Worked on setting up Rancher and creating a cluster using rancher. Then deployed the docker image using docker. Made it work using the Kubernetes cluster so that it can be made available on internet. We created a cluster using Kubernetes installed and pods and nodes running and linked it to our docker image to make it available to the internet. Also worked on setting up Jenkins using rancher. Built the pipeline in Jenkins.

**References:**


- Referred the setup instructions uploaded in blackboard by the professor.
  Install docker desktop on Mac: https://docs.docker.com/desktop/install/mac-install/
- https://docs.docker.com/engine/install/linux-postinstall/
- Rancher AWS Quick Start Guide: https://rancher.com/docs/rancher/v2.5/en/quick-start-guide/deployment/amazon-aws-qs/
- Deployment using Kubernetes:
  https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
- https://www.jenkins.io/doc/book/pipeline/docker/