SWE 645 : Assignment 2 README File

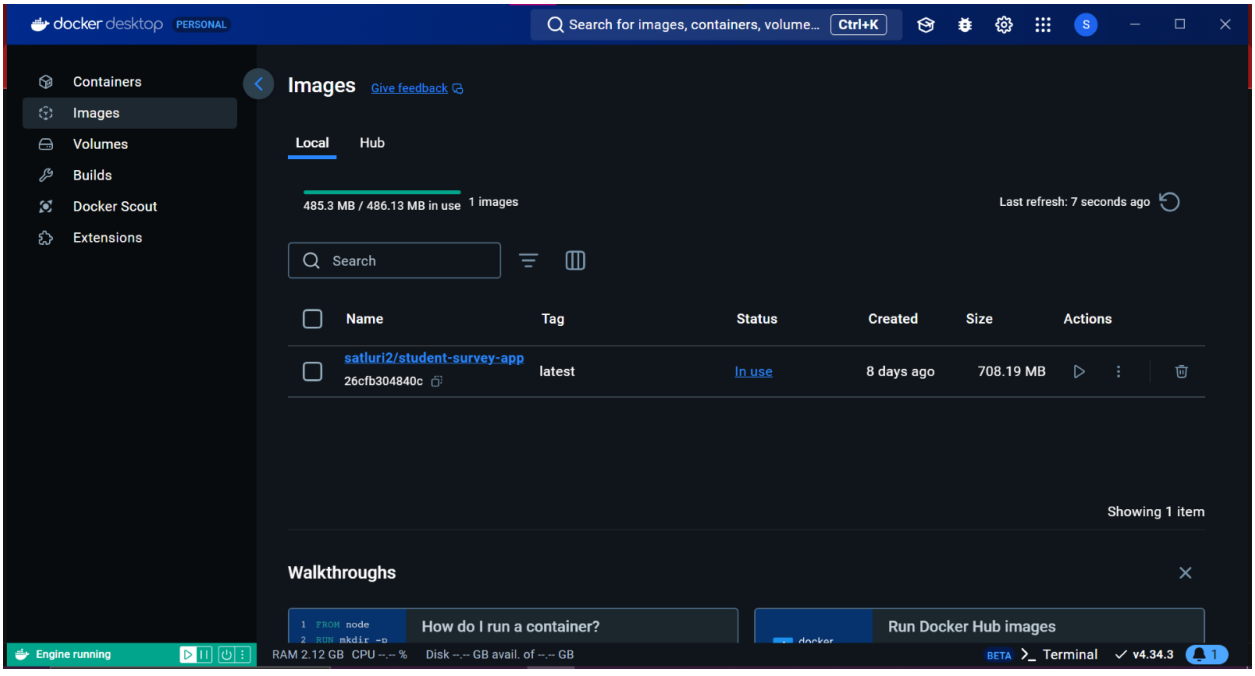
Welcome to the Readme File of SWE645 Assignment 2. We are required to follow a set of procedures in order to complete this assignment. This Readme file covers all the steps and procedures of the tasks in detail along with the screenshots. The Links/URLs used and achieved at various parts of the assignment are also enclosed in this file.

The overview of the steps are mentioned below:

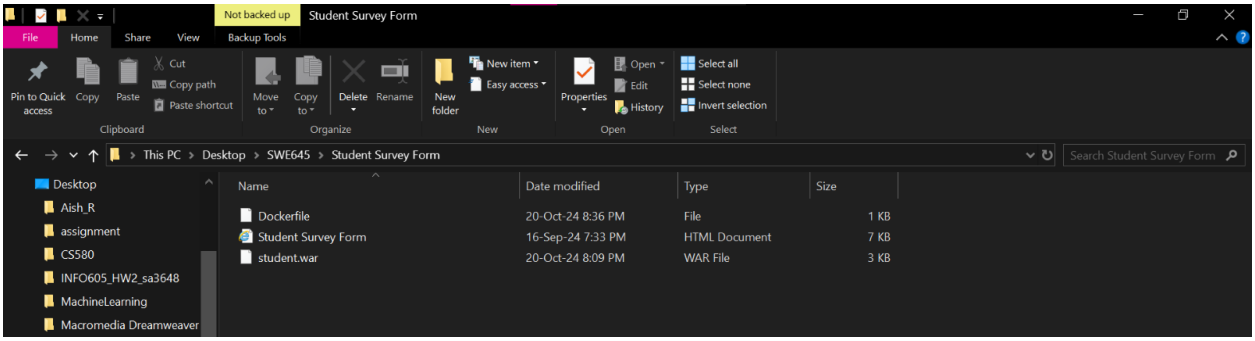
1. Using Dockerhub, we create an image of the application and push it into the dockerhub.
2. Setting up AWS EC2 instances for deploying the application on a Kubernetes cluster. We setup instances on AWS EC2 for deploying the application on a Kubernetes cluster.
3. A similar EC2 instance is created called Rancher Master in order to set up the rancher.
4. Creating a Kubernetes cluster and deploying the application.
5. We setup the Jenkins instance in order to install and run jenkins.
6. Setting up GitHub repository for deploying the webpages.
7. Creating a CI/CD pipeline and running it.

Using Dockerhub, we create an image of the application and push it into the dockerhub

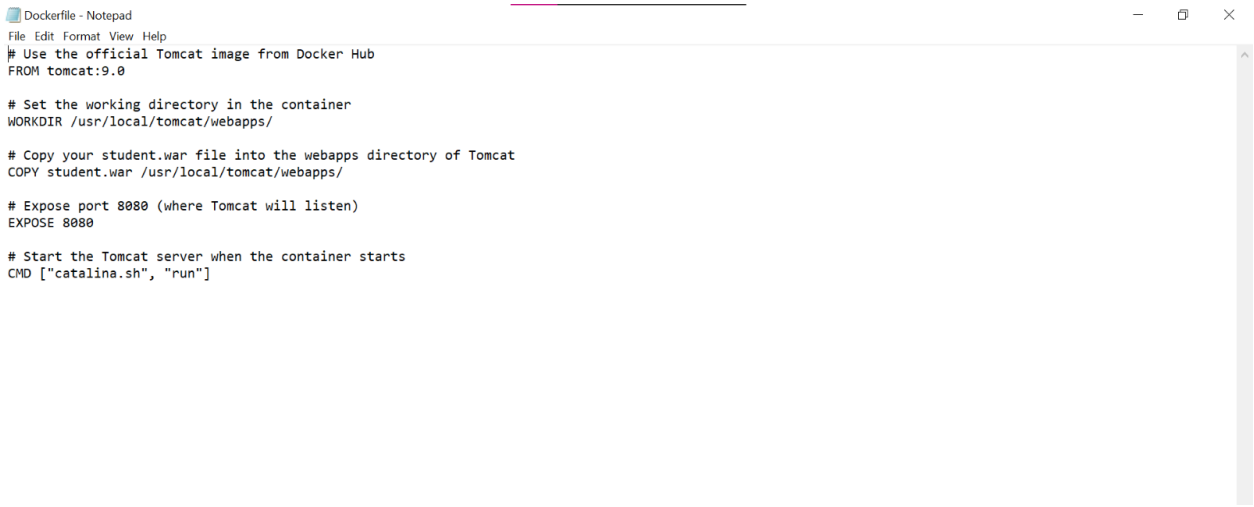
- Go to https://hub.docker.com/ and create an account. You can also download the docker desktop which is optional.
- Login to the docker desktop/ docker hub on the browser using the account created, or open the docker desktop.



- Now create a “Dockerfile” in the same directory where you have the “.war” file. In our assignment, you can check the screenshot below, where we have the “Dockerfile” and the “student.war” file in the same directory. We have obtained the .war file using Eclipse IDE.

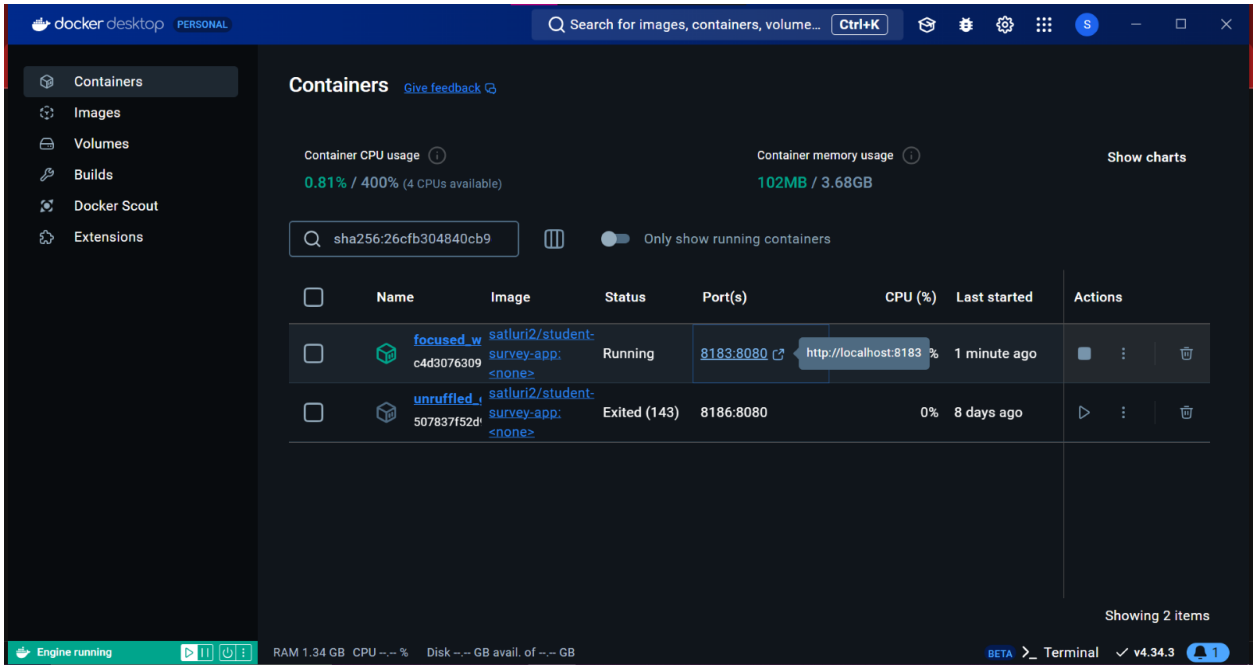


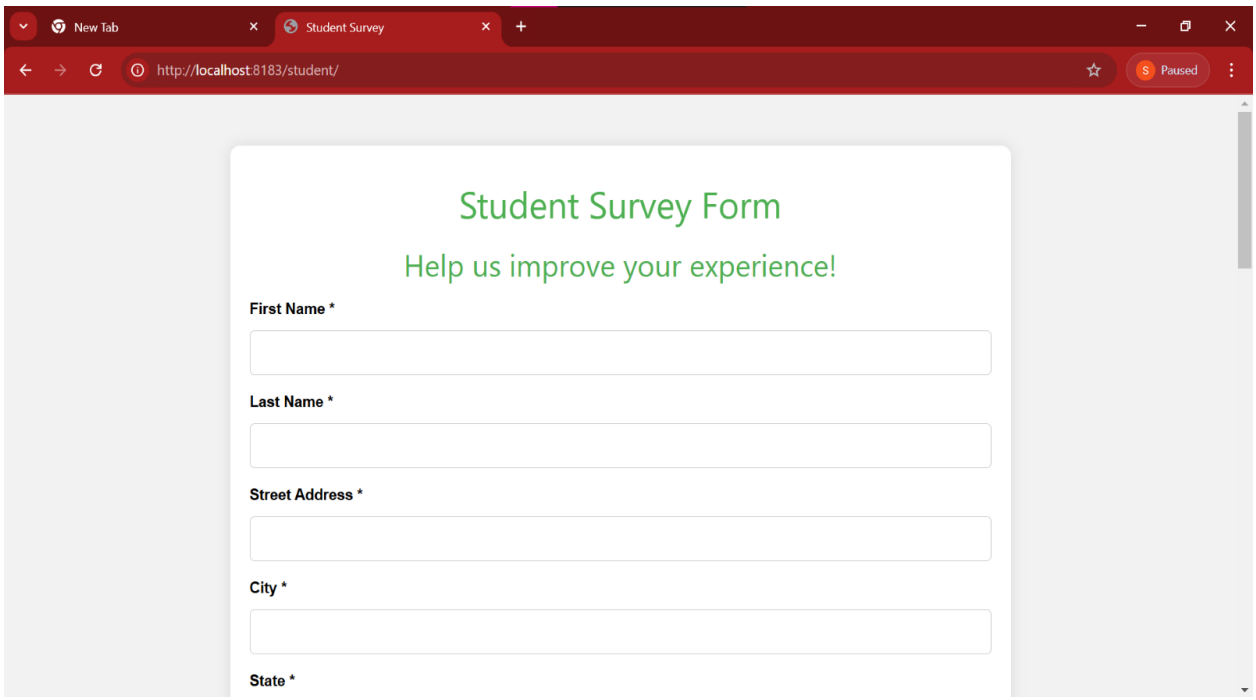
- Now, open the “Dockerfile” in an editor and write the following commands in it, as shown in the screenshot below.



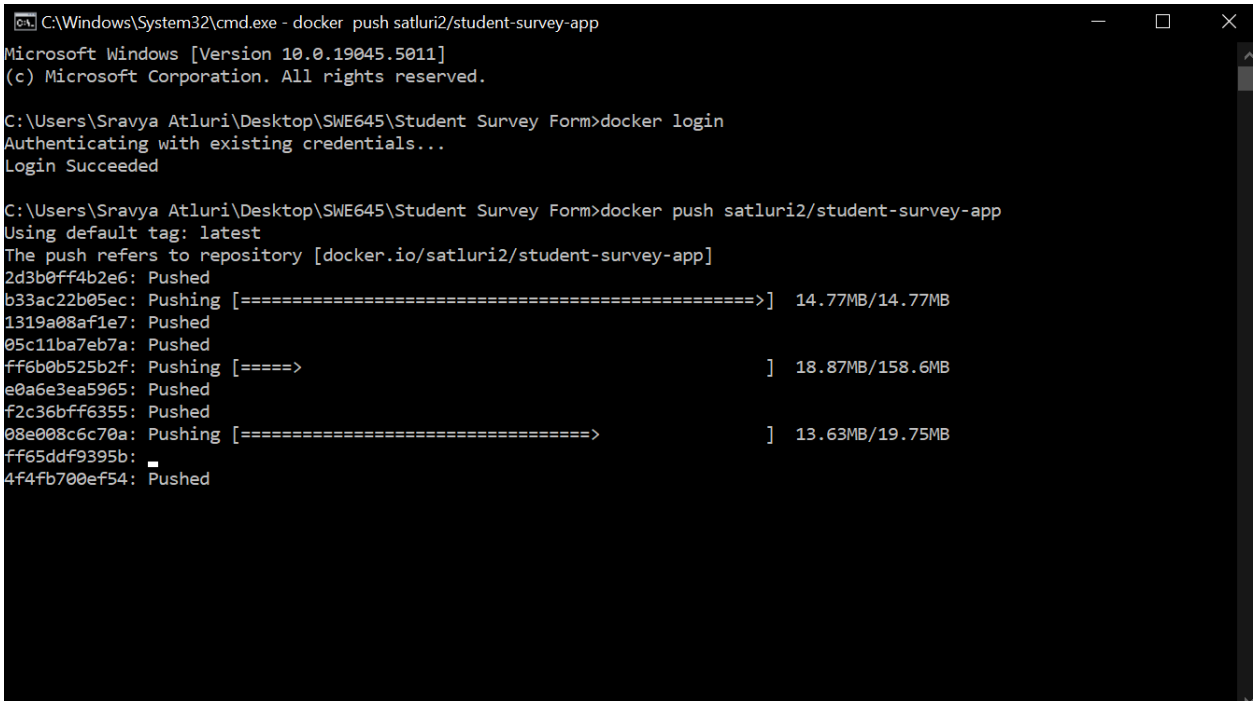
Now open command prompt from the directory where the war file and Dockerfile are present.

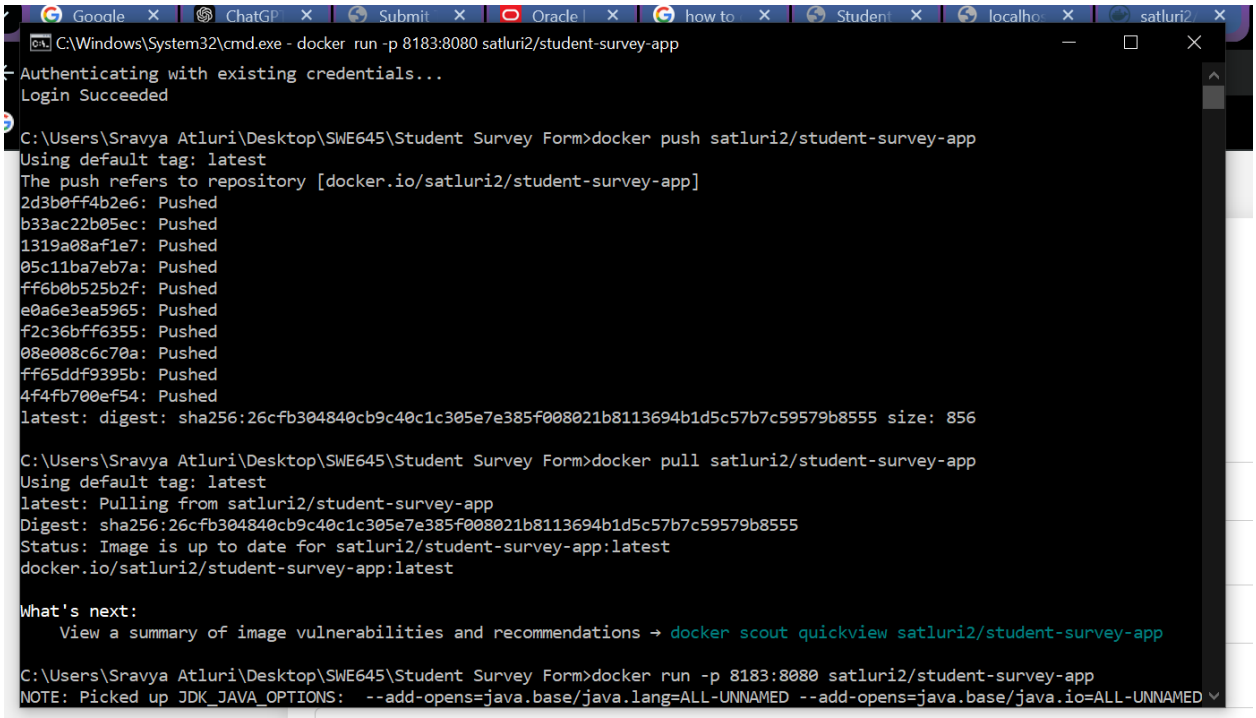
- Run the following commands to build a docker image : “ docker build -t student-survey-app”
- Now an image of the application is created on the local system. To run this application on the localhost, the following command must be run on the command prompt : “ docker run -it -p 8183:8080 student-survey-app ”.
- The application must be deployed on localhost:8183 and the server must be up and running.
- Now go to a browser and open the localhost:8183 and then append “/student” to the URL and the application webpage should be visible on the browser.
- URL : http://localhost:8183/student/



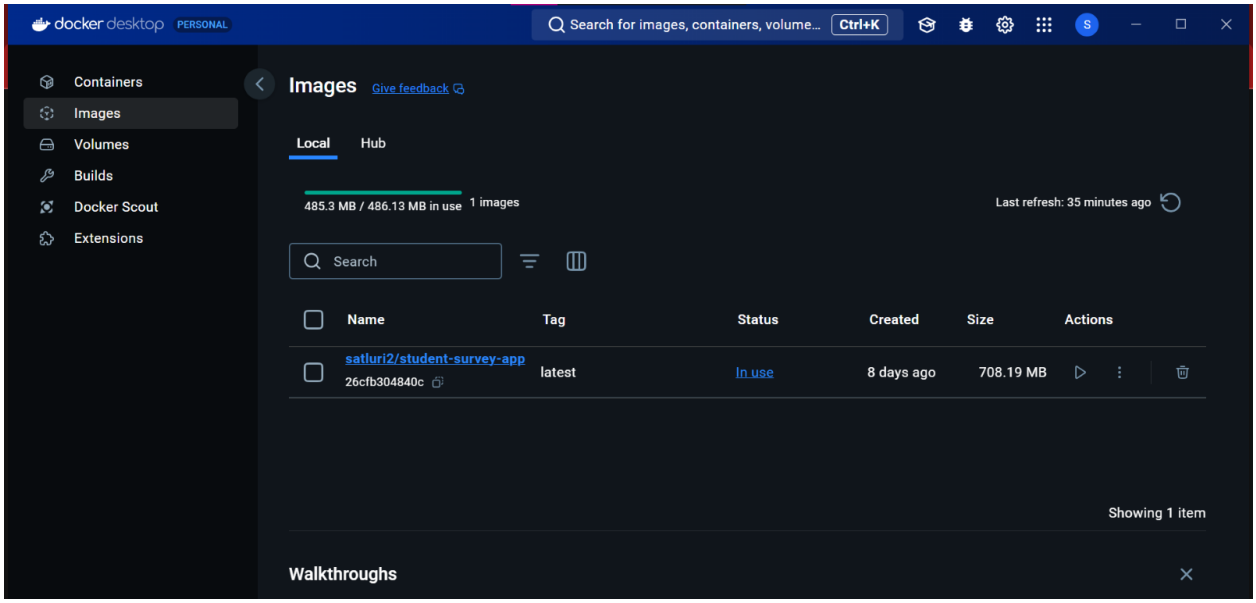


- Now, we have the application image on our local system, and we must push it into the docker hub. We will run the following commands on the command prompt from the same directory, where we have run the commands previously.
- First step is to login to dockerhub using the credentials of the account created earlier : “docker login”. Enter the password when prompted.
- Next, tag the image to be pushed and push it into the hub using the following commands : “docker tag student-survey-app satluri2/student-survey-app” followed by “docker push satluri2/student-survey-app”.



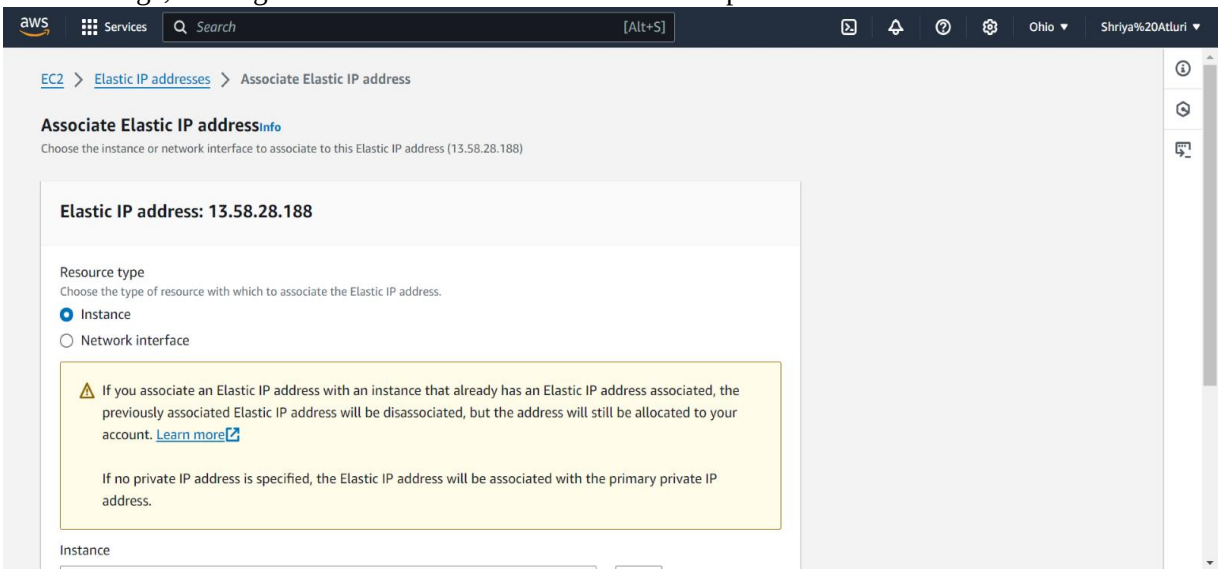


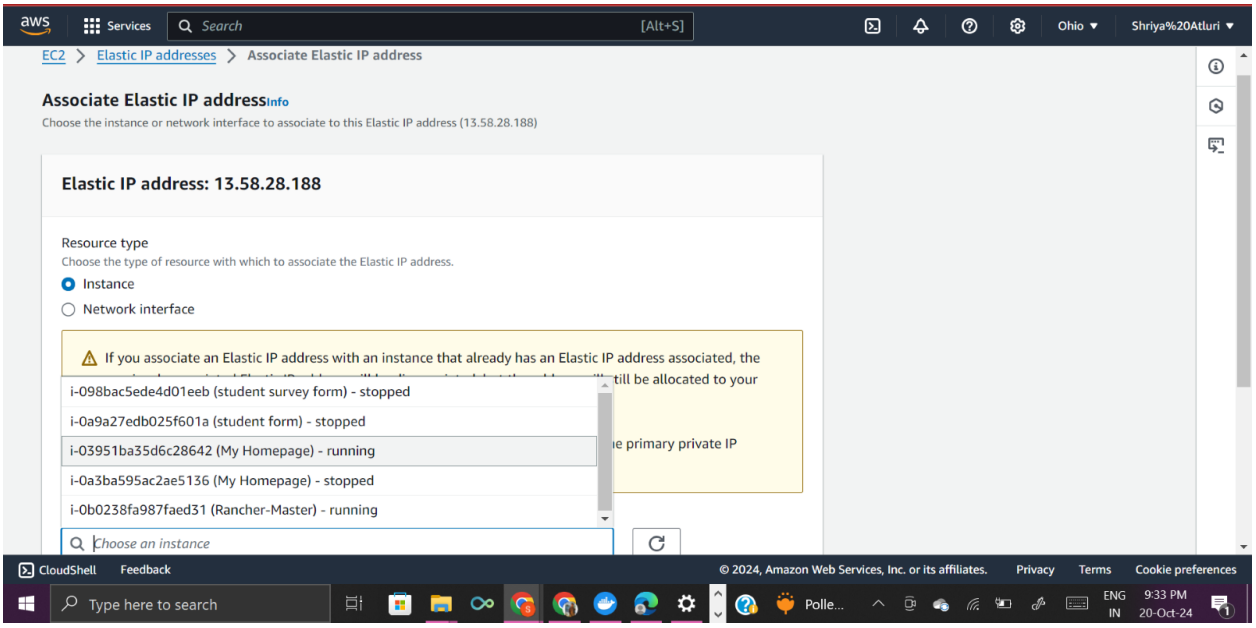
- Now the image should be pushed into the docker hub. You can go to the docker desktop and check it accordingly.



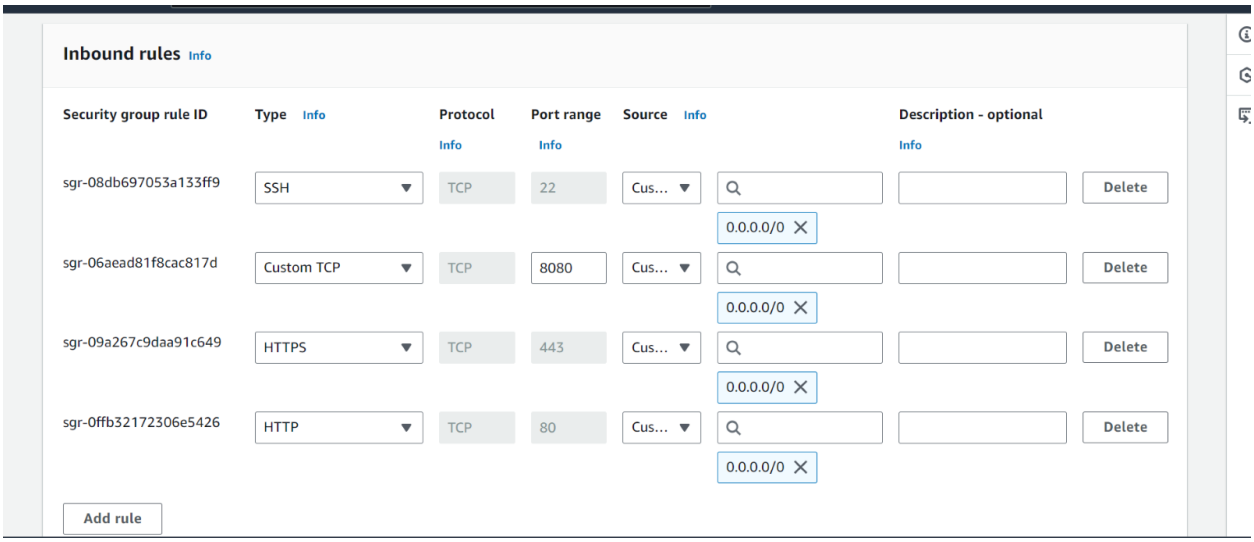
Setting up the AWS EC2 instances for deploying the application on a Kubernetes cluster

- For the rest of the assignment, we need two AWS EC2 instances.
- Login to the AWS Academy and then open AWS. Navigate to the EC2 service and open the dashboard.
- Here, click on “Launch Instance” to create the instance.
- Give the name of the instance as “Rancher-Master”, select AMI : “Ubuntu Server 22.04 LTS (HVM) SSD volume Type”, Instance type as : “t3.large”. Select a key-pair you already have, in this case, it is : “645”, Click on the checkboxes to allow traffic from HTTP and HTTPS from the internet, and finally increase storage from 8Gibs (default) to 30Gibs. And finally click on “Launch instance” button.
- Now, once the instance is created and running, associate it to an Elastic IP address. To do this, go to the “Elastic IPs” in the EC2 menu on the left pane of the screen. Create an Elastic IP using the default settings and associate it to this instance. We do this because whenever the AWS lab is restarted, it restarts the instances on EC2 and the IP addresses change, causing issues to our Rancher and Jenkins setup.

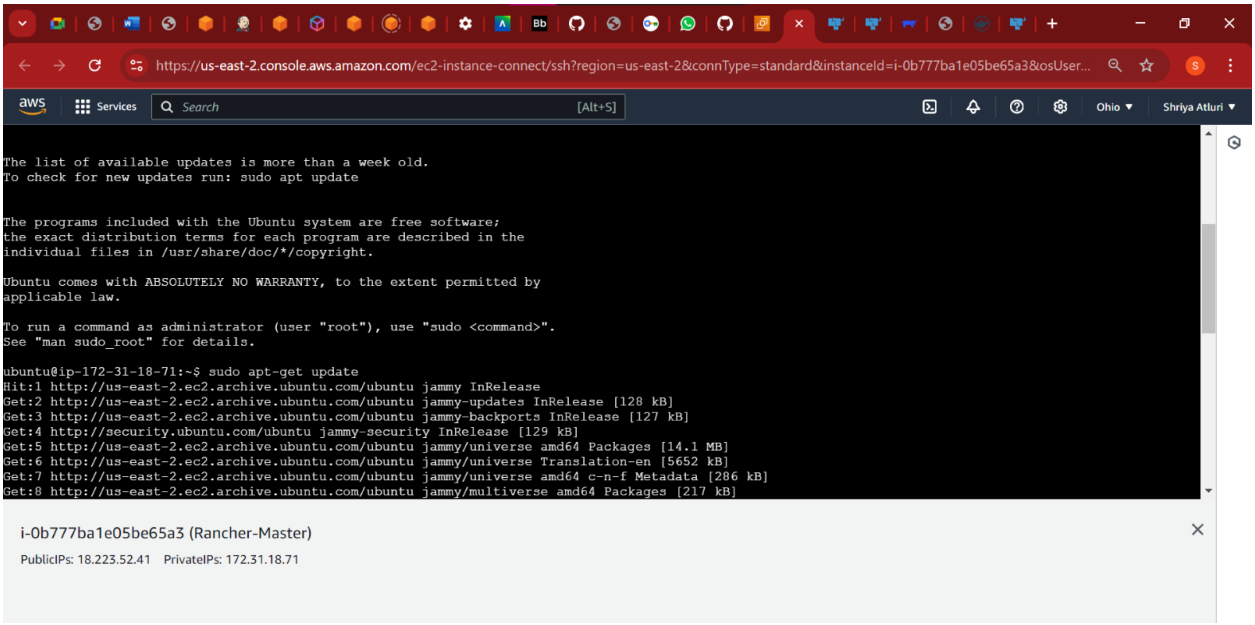


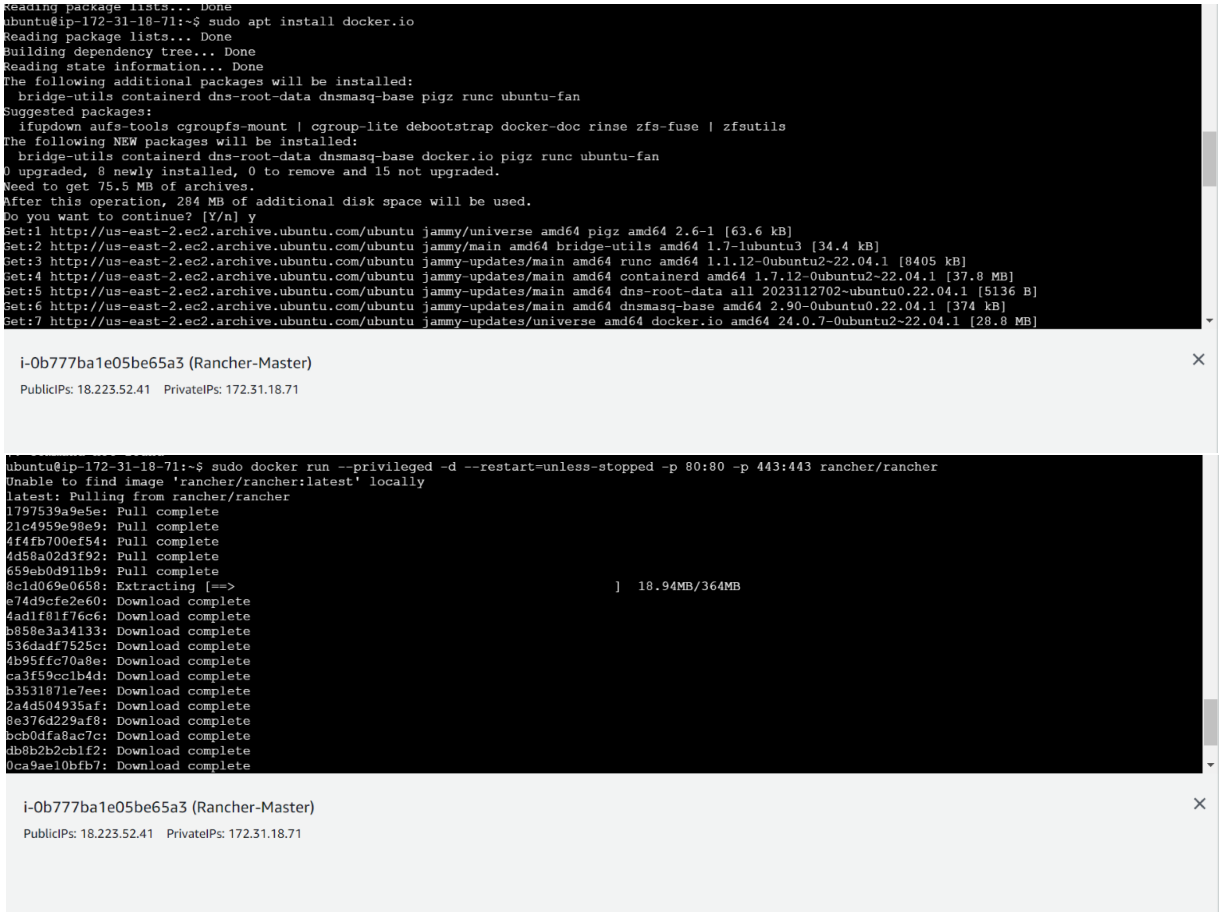


- Now, click on the instance and go to the Security tab. Here, in the “inbound rules”/”Outbound rules” section, click on the security group wizard.
- Scroll down and from either of the “inbound”/”outbound” tab click on edit rule option.
- Here, add a rule using the following configuration, Type : “Custom TCP”, Port range : “8080”, Source : “custom” and “0.0.0.0/0” and click on “Save rules”.



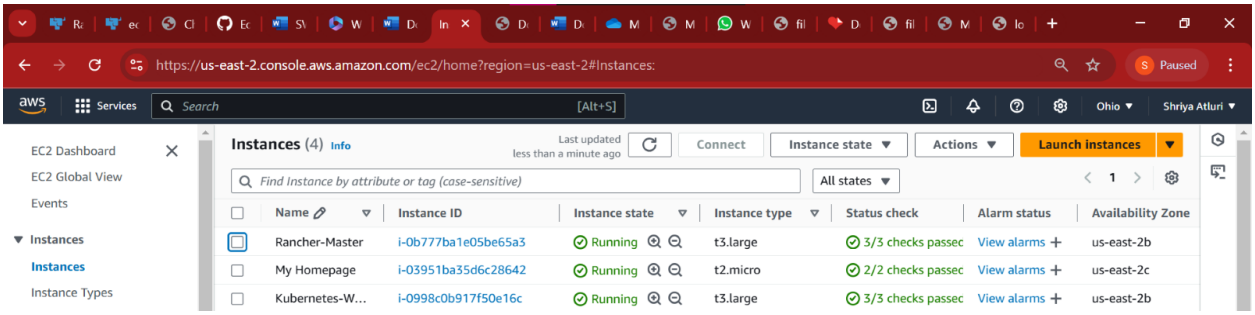
- Now, click on the EC2 instance and click on “connect”. Go to “EC2 instance connect” tab, give the username as “ubuntu” and click on connect. A shell opens in a new tab.
- Here, we will give commands to install the docker on the instances.
- Commands to install docker : “ sudo apt-get update” followed by “sudo apt install docker.io”, Give permission when prompted: “Y”.





- After successfully running of the commands, docker should be successfully installed.
- Repeat all the steps mentioned above, to create the second instance with the name: “Kubernetes-Worker”. And proceed to install docker as completed for the first instance.

- This second instance will be our Worker node and the first instance will be the Master node.
- At the end make sure that both the instances are in “running” state.



Setting up the Rancher

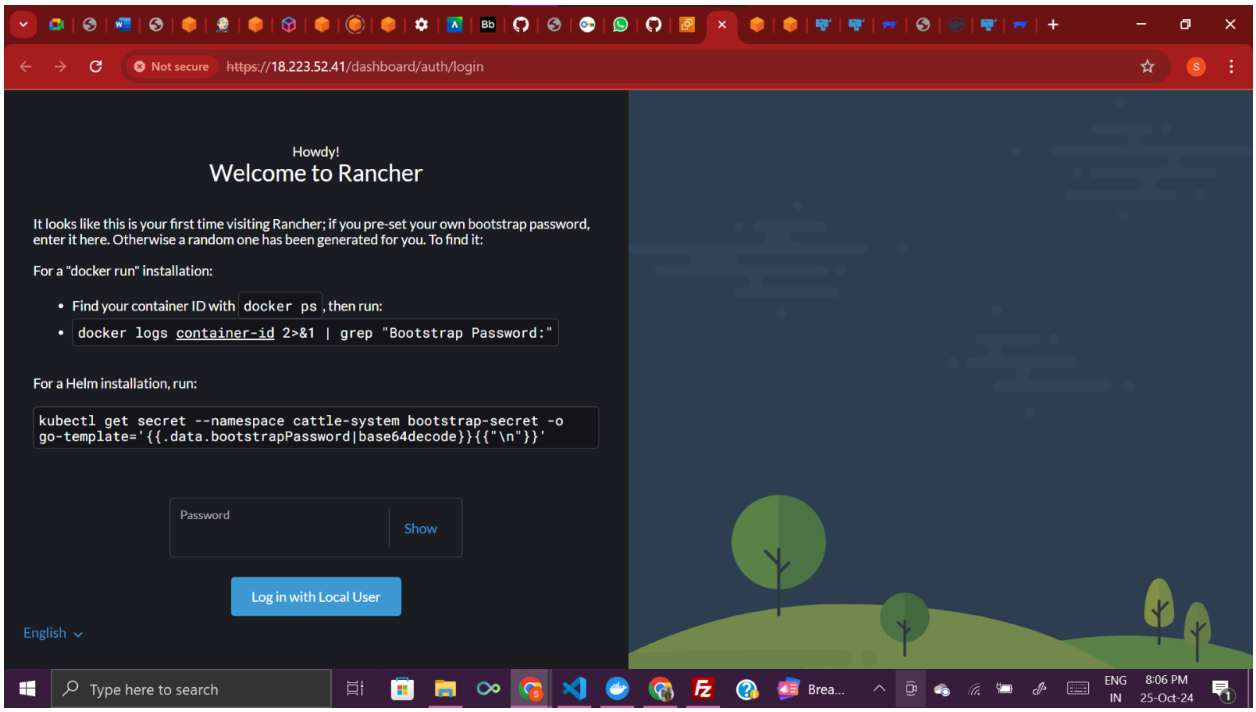
- Now that we have both the instances up and in “running” state, we will proceed to set up “rancher” on the master node i.e., “Rancher-Master” instance.
- Open the browser and go to the following website: https://www.rancher.com/quick-start . Here, scroll down to the “Start the Server” section under “Deploy Rancher”.
- Copy the command present there, which is : “ $ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher “.

- Now, connect to the “Rancher-Master” instance in the same way as described in the earlier section.

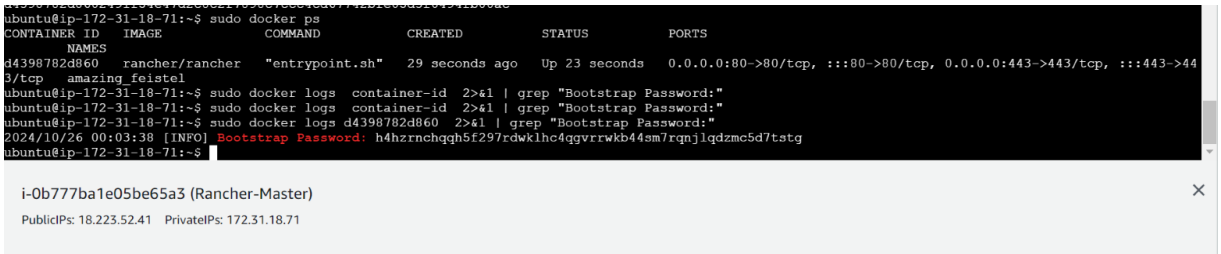
- Here, run the copied command. After it has run successfully, rancher will have been installed on this instance and can be accessed using the public IPv4 DNS address of the instance.

- Now, on the instance, run the following command to get the details of the container running on the instance: “ sudo docker ps “. Store the result obtained somewhere (in notepad preferably), we will need this information later.

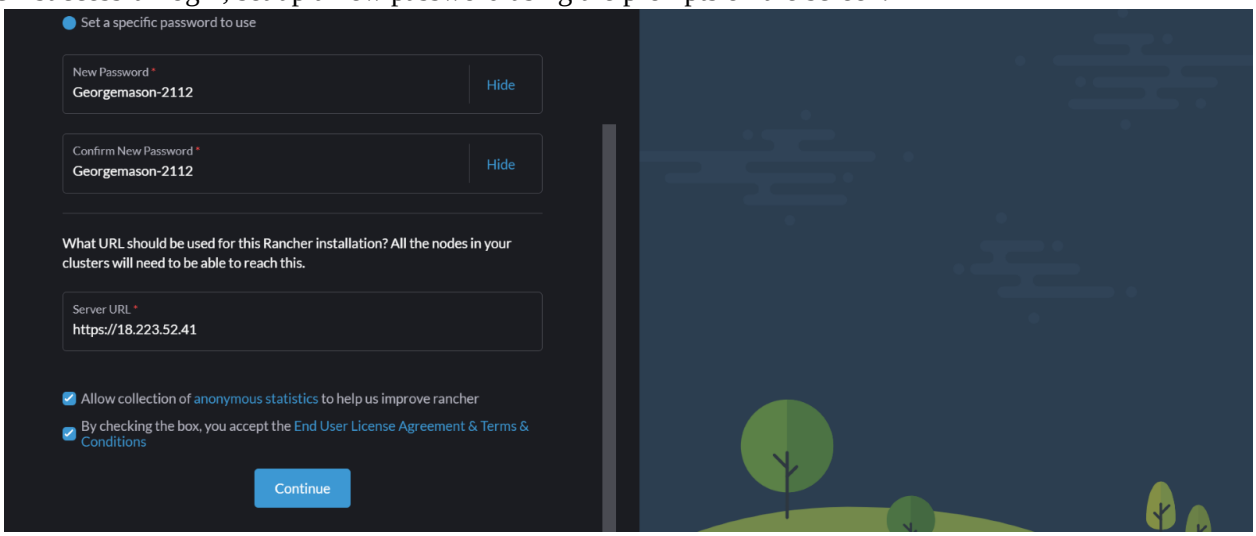
Now, in the EC2 instance page, open the “Public IPv4 DNS” URL in the new tab. Give permission to proceed to the URL, even though it is unsafe. Upon loading, the rancher login page should be visible.



- Here, on this page it asks for a password. To get the password, copy the command displayed above on the same webpage and edit the container id. The container id is retrieved from the earlier stored output and run it on the instance console. The command should look like this: “sudo docker logs d4398782d860 2>&1 | grep “Bootstrap Password:”. The console will display the Password on the screen.



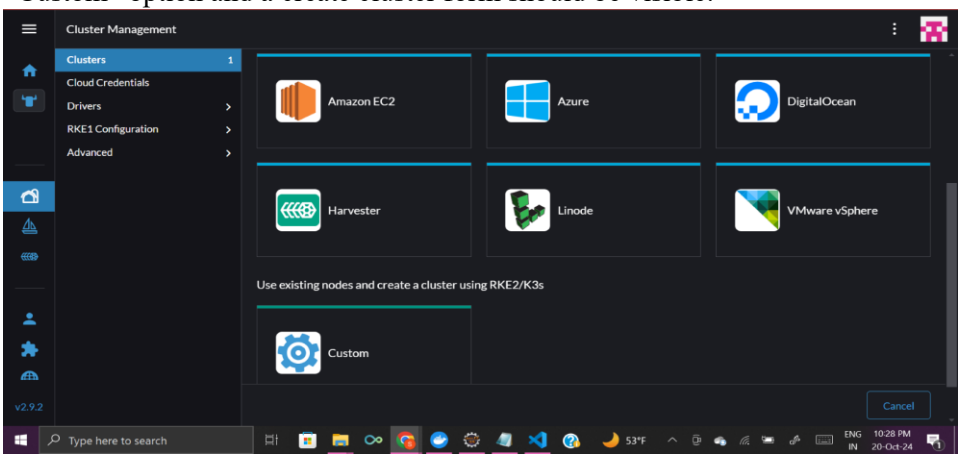
- Copy the password and paste it on the rancher login screen to login.
- On successful login, set up a new password using the prompts on the screen.



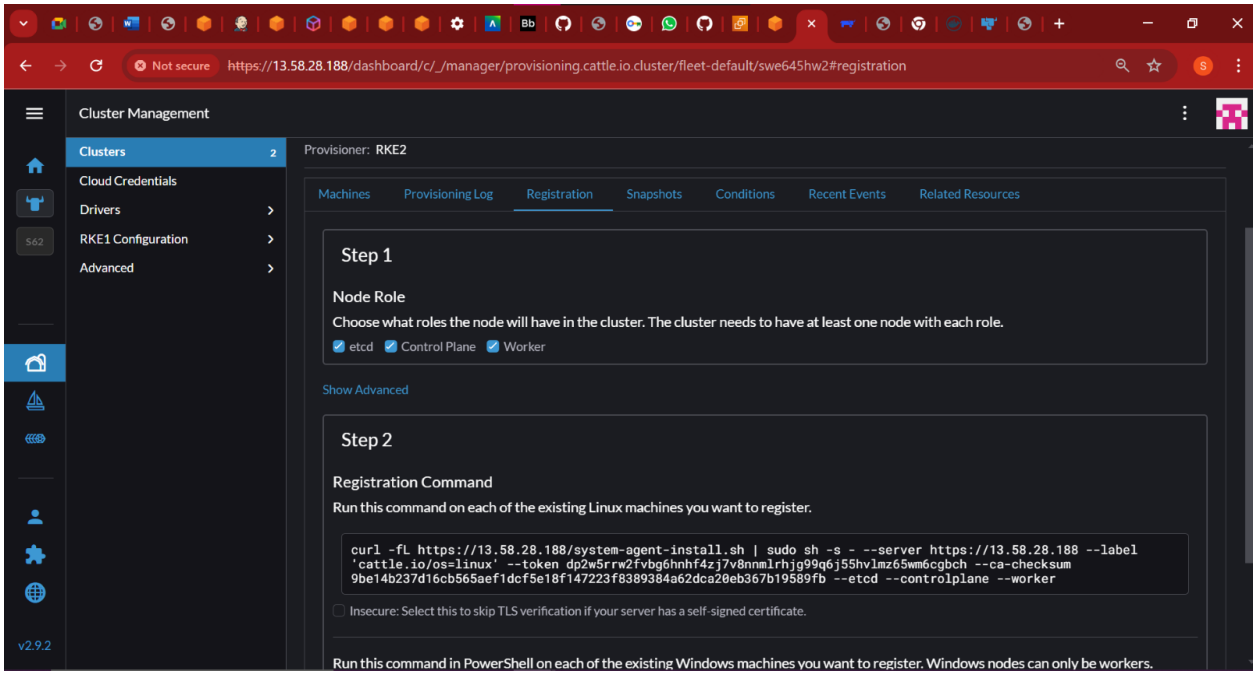
- Now the rancher dashboard will be displayed, where we can proceed with creating a cluster and deploying the application.

Creating a Kubernetes cluster and deploying the application.

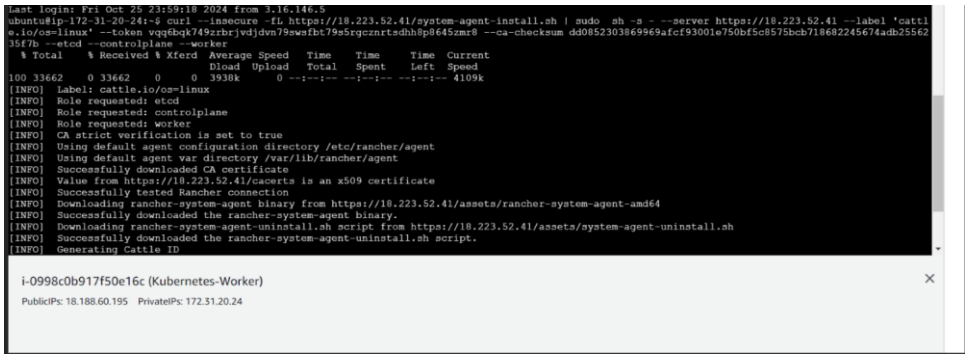
- Now, as we have the rancher up and running, we will be creating a cluster and deploying our application on the cluster using the Rancher UI.
- On the dashboard, click on “Create Cluster” button. It should show various cluster options, select the “Custom” option and a create cluster form should be visible.



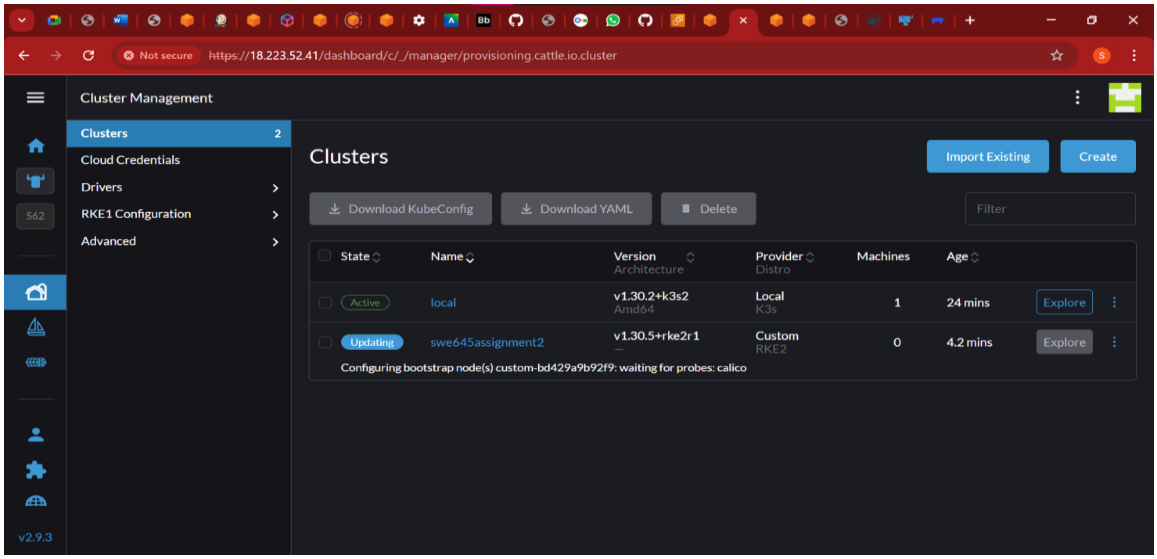
- Here, fill the form by giving the cluster name : “swe645assignment2” and leave everything else as default and click on the “Create” button.
- On the next page, go to “Registration” tab and under Step1, make sure that the 3 checkboxes of “etcd”, “control pane” and “worker” are checked.
- Now, copy the command present below in Step2.



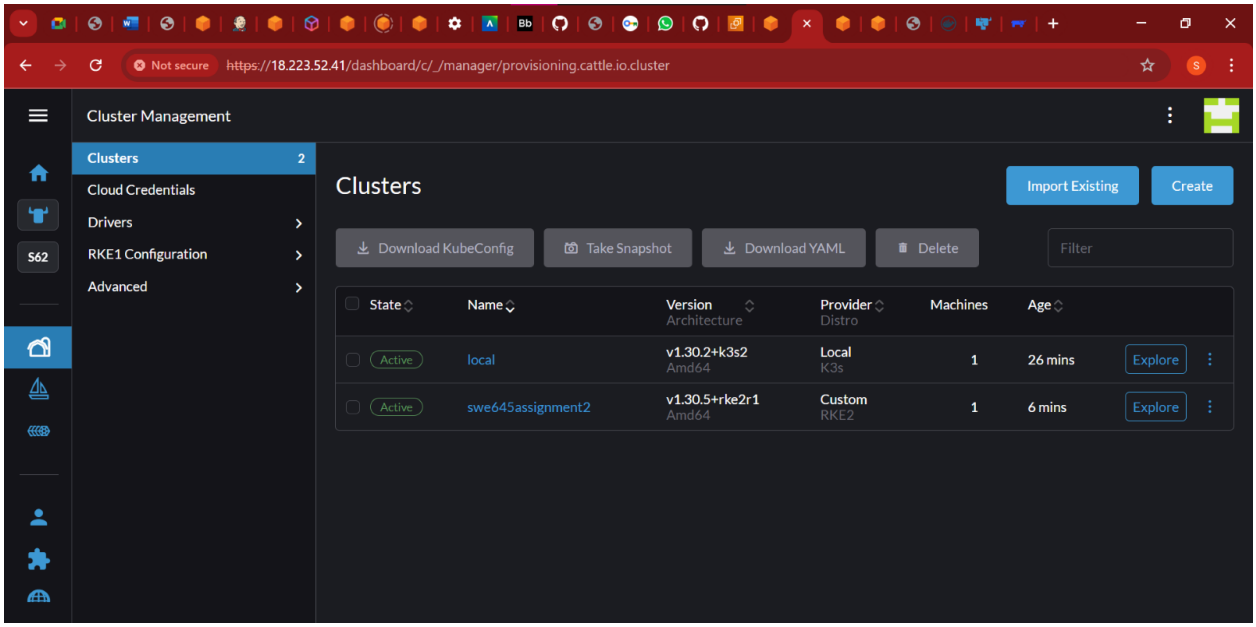
- Open the AWS EC2 instance and connect to the “Kubernetes-Worker” node.
- Here, Run the copied command but, append the “—insecure" after the “curl” word in the command, and later run it.



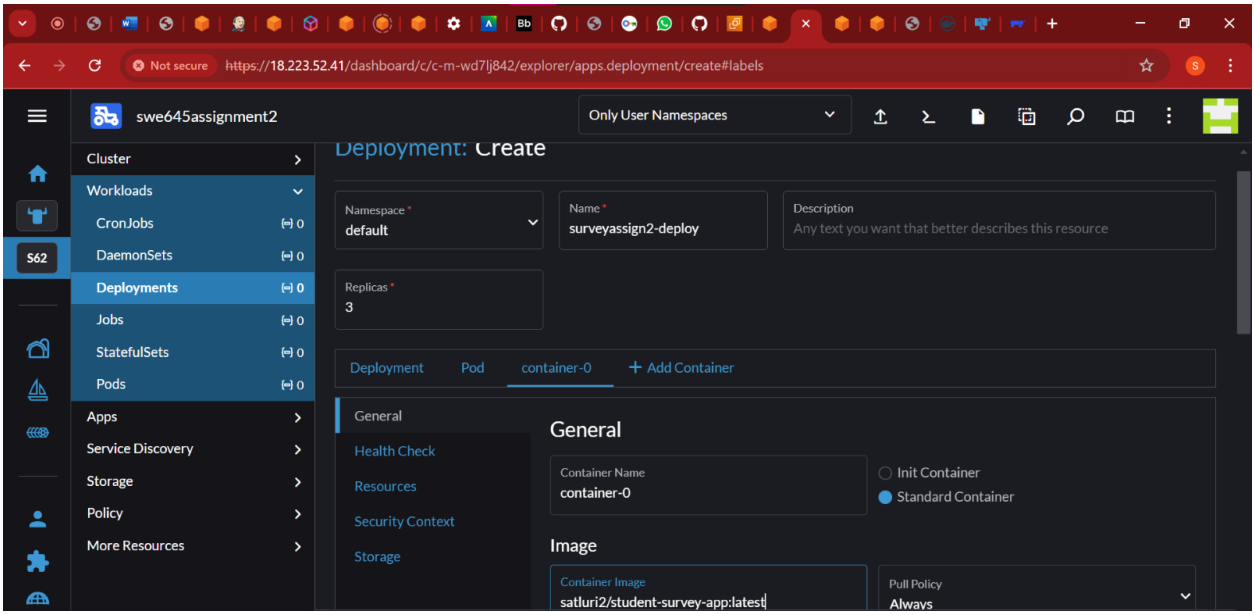
- After the command is successfully executed, the cluster should be in the “Updating” state and after a few minutes it will change to “Active” state after which the cluster will be ready for deployment. This can be viewed under the “Clusters” option in “Cluster Management” option on the Rancher UI.



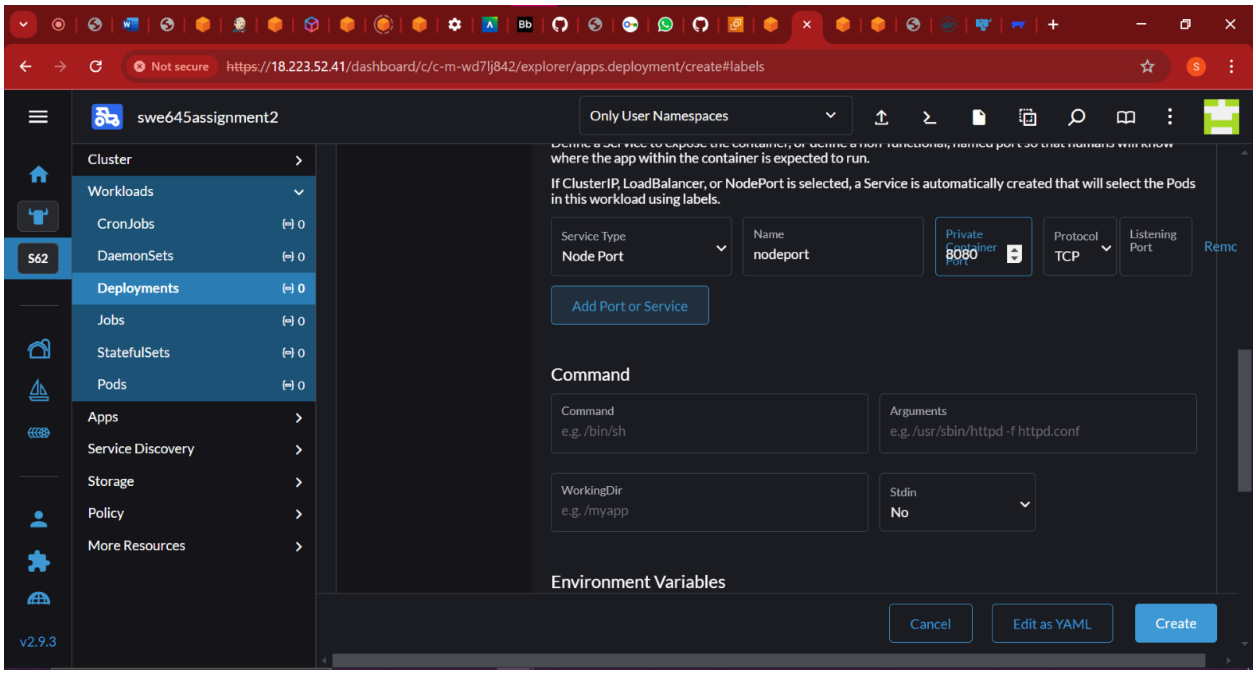
- Once the cluster is in the “Active” state, click on the “Explore” button which is next to the cluster.



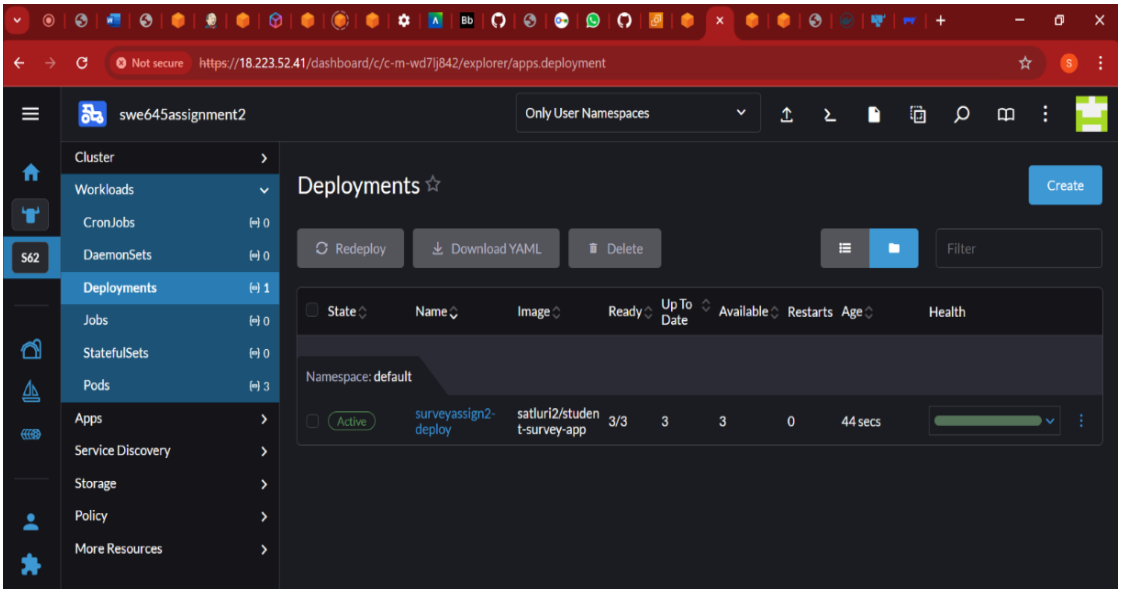
- On the left pane, under “Workloads”, select “Deployments”.
- Click on the “Create” button and a form should be visible on the screen.
- Fill in the form using the following details. Namespace : “default”, Name : “surveyassign2-deploy”, Replicas : “3” (no. of pods), Container image : “satluri2/student-survey-app: latest” (same as the name of the image on the docker hub).



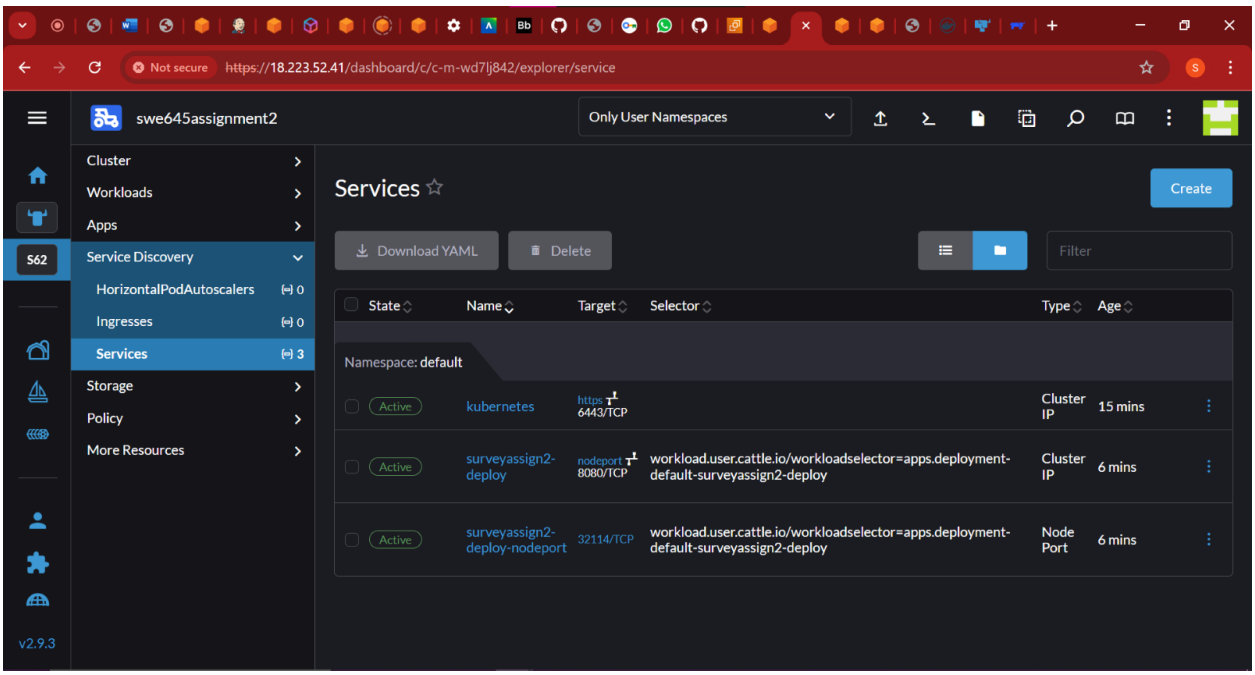
- Under Networking, click on “Add port or service” button.
- Give the following details, Service type : “Node Port”, name : “nodeport”, private container port : “8080”, protocol : “TCP”.
- Leave everything else as default and click on create.



- Now the deployment is complete, and it should be in “Updating” state. Give it a few minutes and it should be in “Active” state.

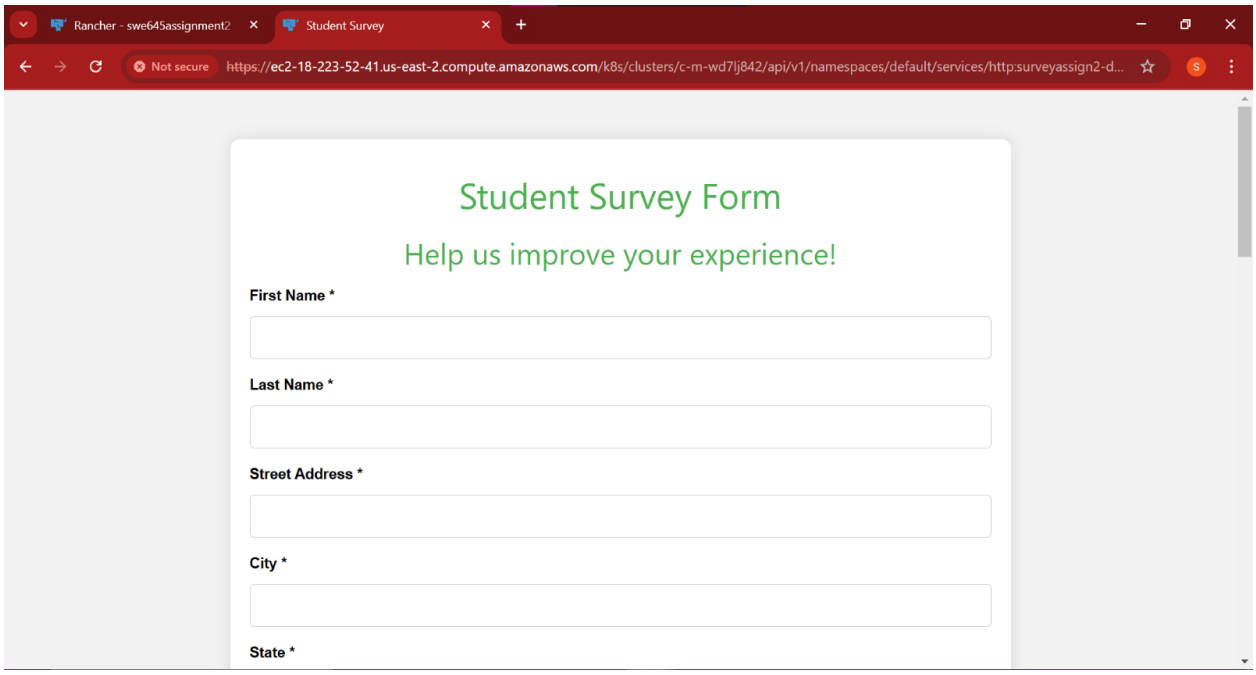


- Once it is in the “Active" state, click on the deployment and navigate to the “Services” tab.
- Here, find the deployment created and click on the “nodeport” to open it in a new tab, under the target option.

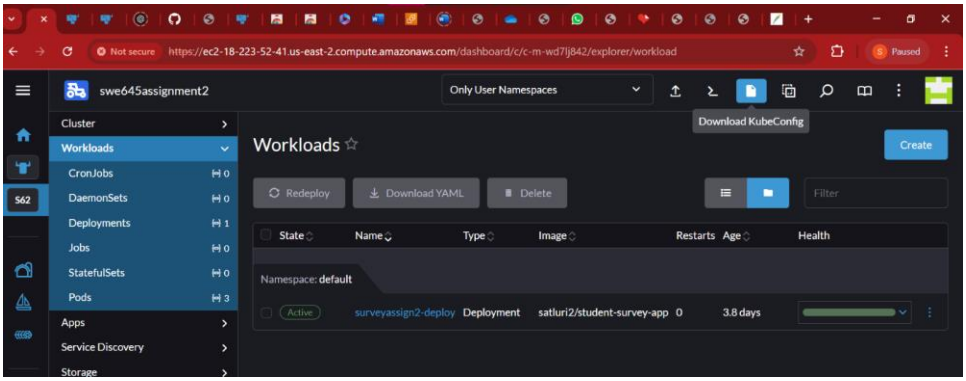


- Now to the URL, append “/student” and open it. You should be able to see the application running.
- URL : https://ec2-18-223-52-41.us-east-2.compute.amazonaws.com/k8s/clusters/c-m-wd7lj842/api/v1/namespaces/default/services/http:surveyassign2-deploy:8080/proxy/student/

- With this, we will have successfully deployed the application on a Kubernetes cluster.

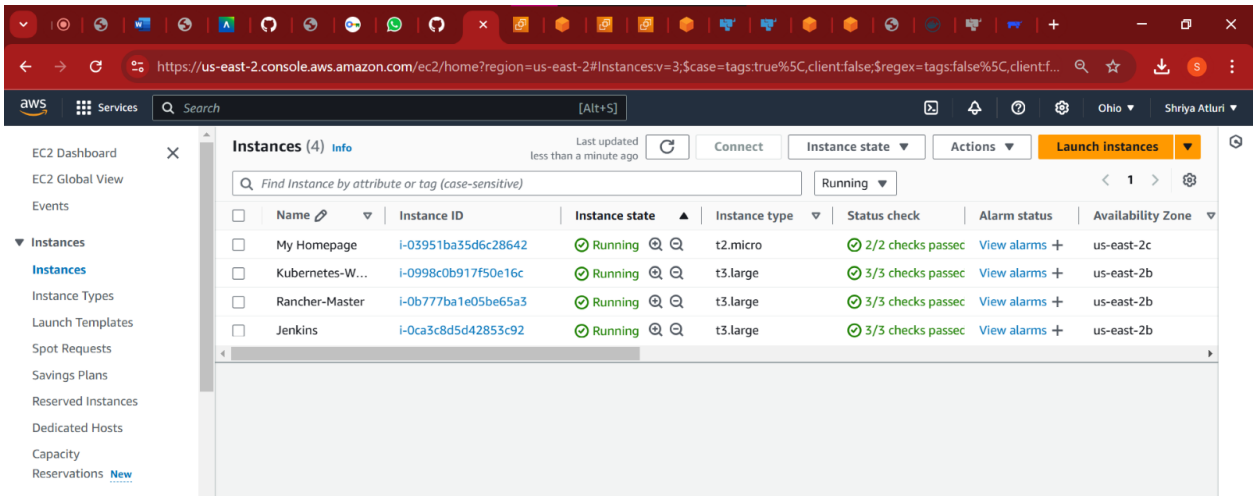


- For the next step we will need the “KubeConfig” file. To download it, go to our cluster page. On the top right, from the menu displayed, click on the “Download KubeConfig file” and save it in your local machine.

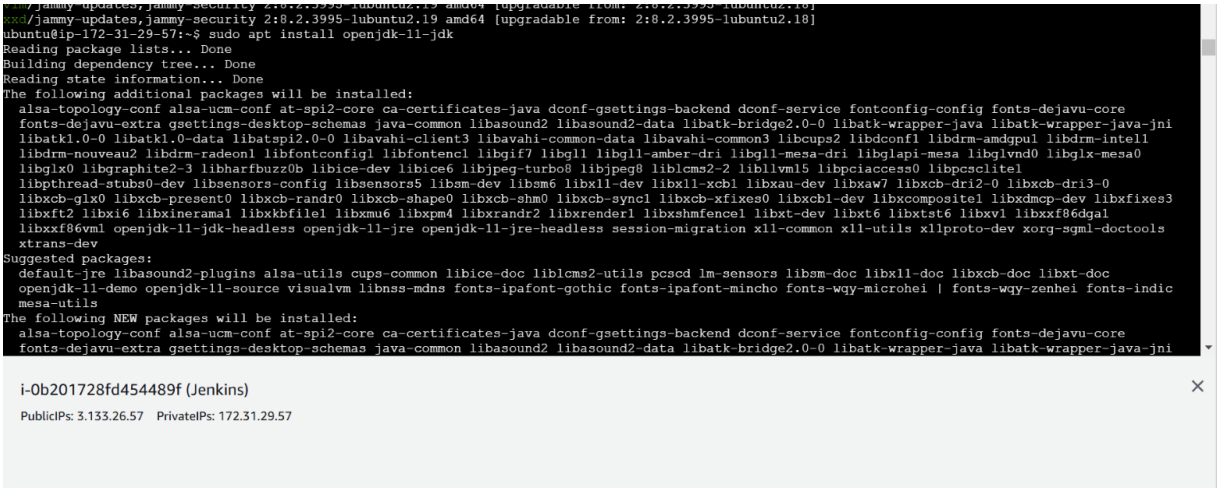


<u>We setup the Jenkins instance in order to install and run jenkins</u>

- Now, we will focus on creating a CI/CD pipeline using Jenkins, to automatically build and reflect any changes in our source code.
- For this, we need to have Jenkins installed on our machine. So we will start off by creating an AWS EC2 instance in our AWS lab.
- We will create the instance in the same manner as we did to create our both the EC2 instances to deploy our application on the Kubernetes cluster.
- Give the name of the EC2 instance as “Jenkins”, keeping all the configurations same, give an elastic IP and edit the Security group.

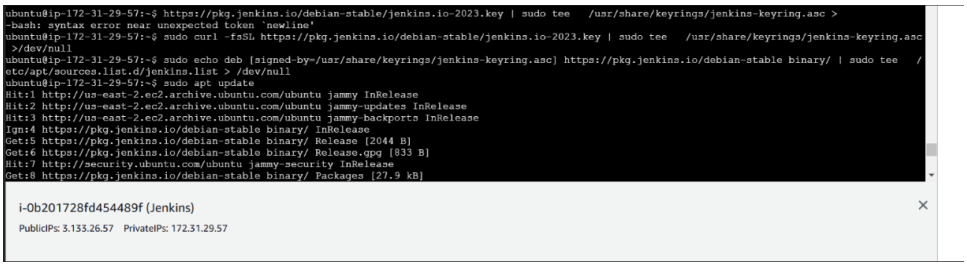


- Connect to the instance using “EC2 instance connect”. Make sure that the instance is in the “running” state.
- First, we need to install java before proceeding with Jenkins.
- Now, we run the following commands : “sudo apt-get update” followed by “sudo apt update”.
- Now run “ sudo apt install openjdk-11-jdk” and give permission when prompted as “Y”. This should install java jdk-11.



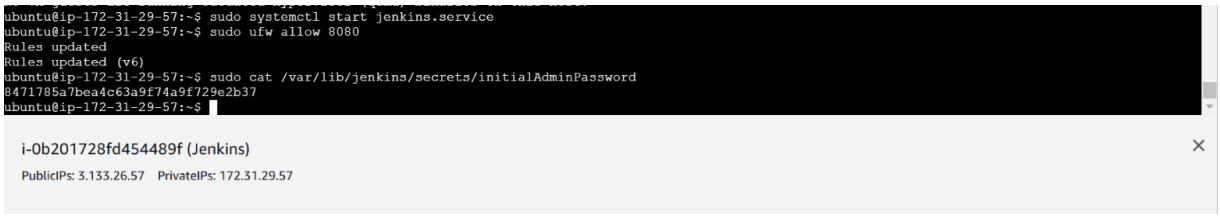
- Now Jenkins can be installed by running the following commands: “sudo curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null” to get the Jenkins package.

- Next run “ sudo echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null “ to unzip Jenkins.



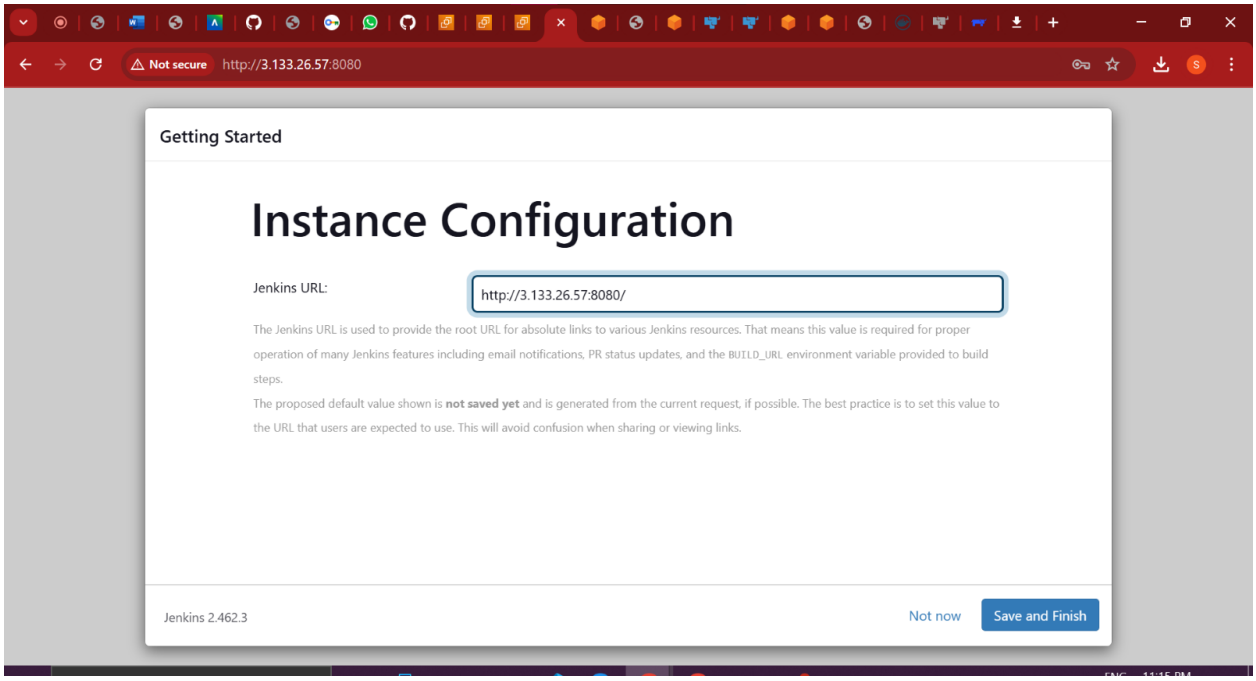
- Now update apt and install Jenkins using the commands : " sudo apt update" followed by "sudo apt install jenkins" . Give permission as "Y" when prompted and jenkins should be installed successfully.

- Now, start jenkins using the command : "sudo systemctl start jenkins.service".
- Expose the port 8080 using the command : "sudo ufw allow 8080".
- Now, run the following command to get admin password : "sudo cat /var/lib/jenkins/secrets/initialAdminPassword " . It will display the password. Store It safely.



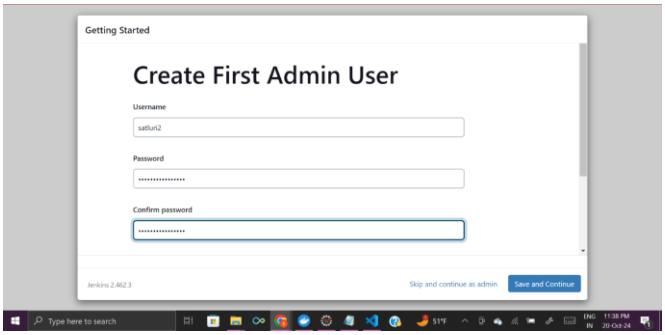
- Run the following commands to install snapd : "sudo apt install snapd".
- Now using snap, install kubectl : " sudo snap install kubectl –classic".

- Go to the AWS EC2 instance "Jenkins" and open the "Public IPv4 address" in a new tab.
- To the URL, append the port 8080 and edit "https" to "http". It should look like the following: " http://3.133.26.57:8080".
- Enter the admin password obtained earlier to unlock jenkins.



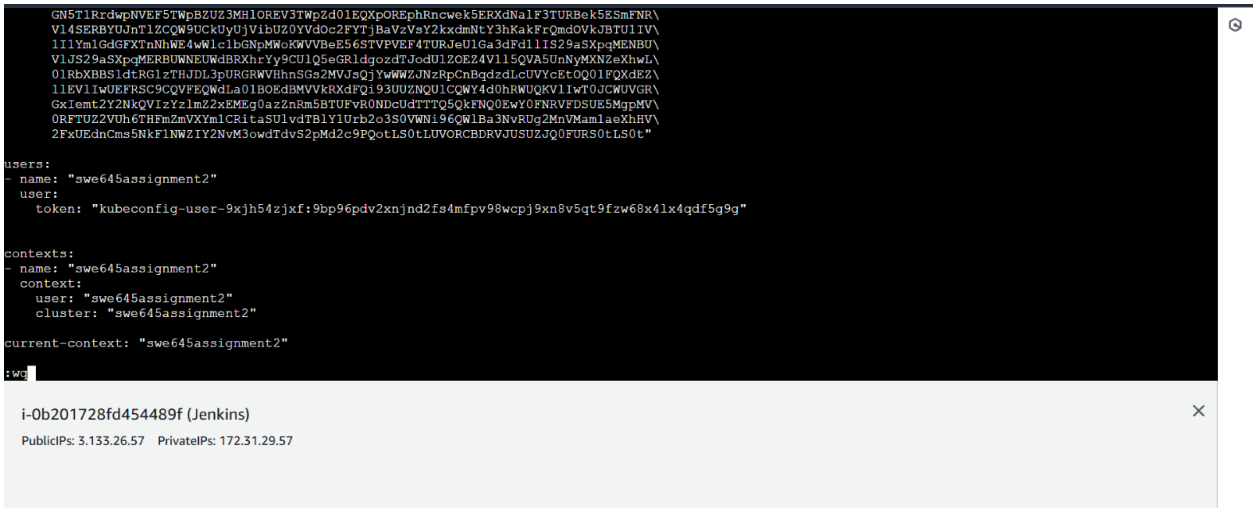
- Now proceed to install the suggested plugins.

- Next create an admin user. Click on “Save and create user” followed by “Save and Finish” and lastly “Start using jenkins”.

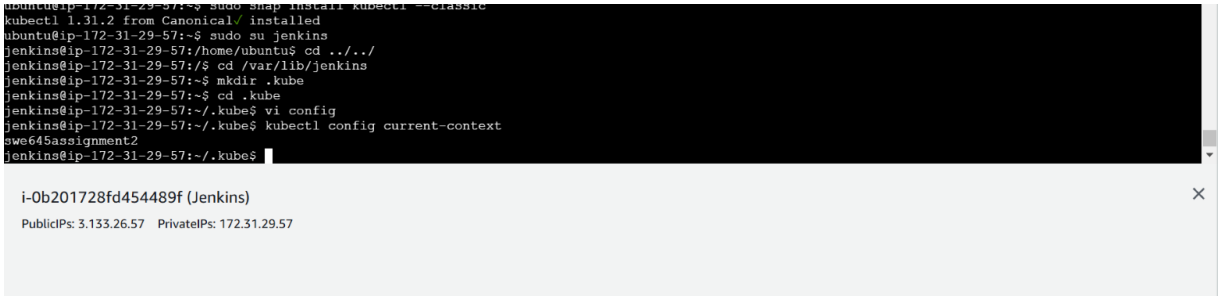


- You should be able to see the jenkins dashboard.

- Now, go back to the EC2 console. We need to create a config file.
- Run the following commands : “sudo su jenkins” to go to jenkins home.
- Next, go to root directory : “ cd ../../”.
- Go to the directory : “cd /var/lib/jenkins”
- Create a directory and enter this : “mkdir .kube” followed by “cd .kube”.
- Create the file and open it : “vi config”
- Now copy the contents from the “KubeConfig” file downloaded earlier and paste it here.
- Save the file using “:wq”.



- Now to verify, run the command : “kubectl config current-context”. It should display the cluster name as the output.



- Next, exit the jenkins mode using : “exit” command.
- Now proceed to install docker using the commands : “sudo apt-get update” and “sudo apt update” followed by “sudo apt install docker.io”. Give permission as “Y” when asked.

```
14 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-29-57:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 14 not upgraded.
Need to get 75.5 MB of archives.
After this operation, 284 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.12-0ubuntu2~22.04.1 [8405 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.12-0ubuntu2~22.04.1 [37.8 MB]
Get:5 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dns-root-data all 2023112702~ubuntu0.22.04.1 [5136 B]
Get:6 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.90-0ubuntu0.22.04.1 [374 kB]
Get:7 http://us-east-2.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.7-0ubuntu2~22.04.1 [28.8 MB]
```

i-0b201728fd454489f (Jenkins)

PublicIPs: 3.133.26.57 PrivateIPs: 172.31.29.57

- After installing, change the file permissions for socket use, using the command: “sudo chmod 777 /var/run/docker.sock”.

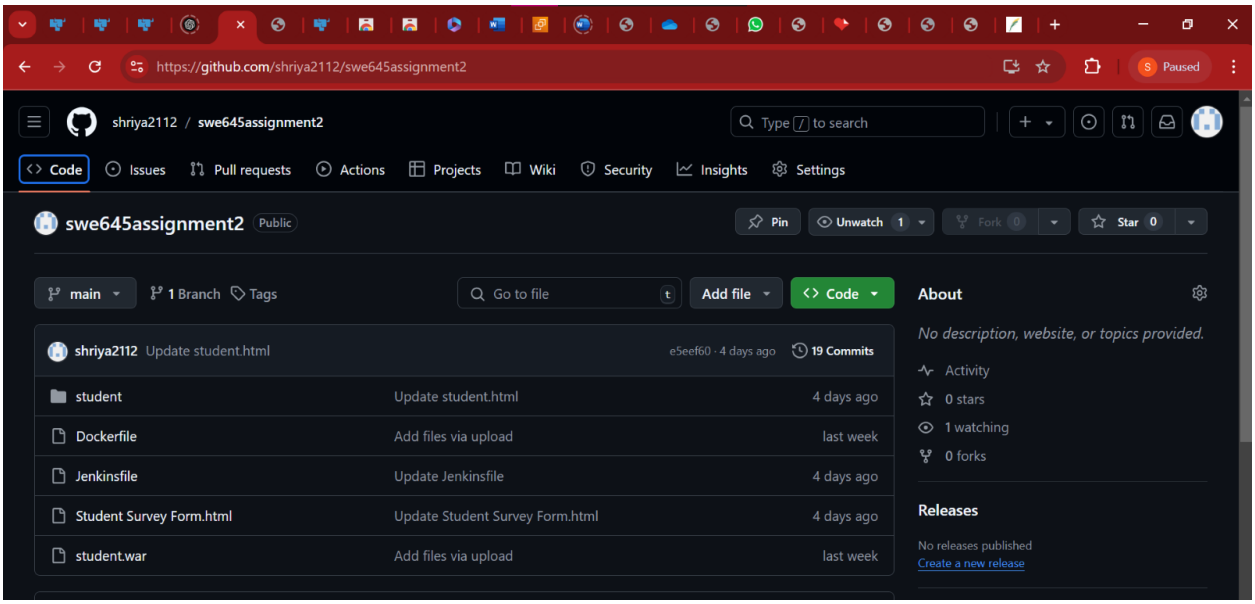
```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-29-57:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-29-57:~$
```

i-0b201728fd454489f (Jenkins)

PublicIPs: 3.133.26.57 PrivateIPs: 172.31.29.57

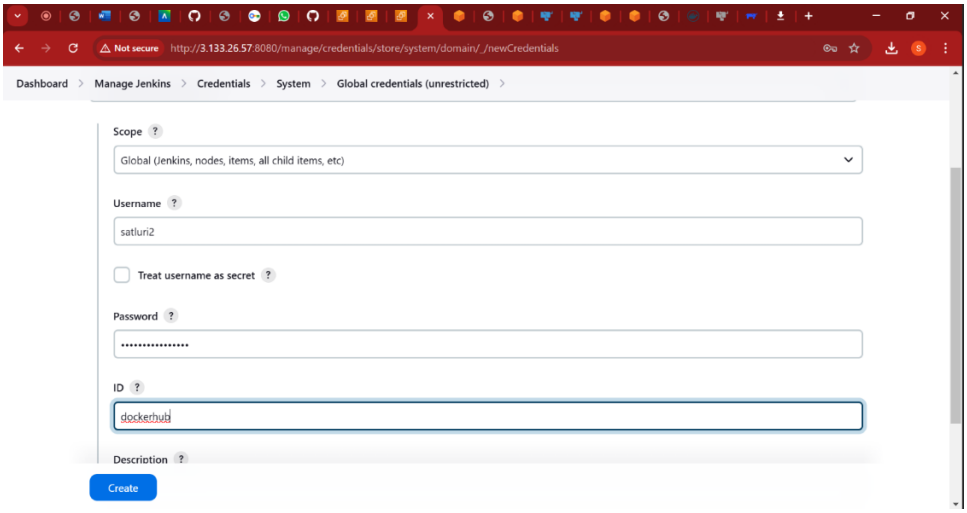
Setting up the GitHub repository for deploying the webpages

- In order to create our pipeline, we need to have the source code saved somewhere from where we can attach it to the pipeline, and it can look for any changes. For this purpose, we will be using GitHub.
- Go to https://github.com/ and create an account and login to it.
- Now, create a new repository : “ swe645assignment2”.
- To that repository : add your source files, Dockerfile and the war file, and then click on commit.
- We will be using this repository for our pipeline.
- Github repository URL : https://github.com/shriya2112/swe645assignment2.git

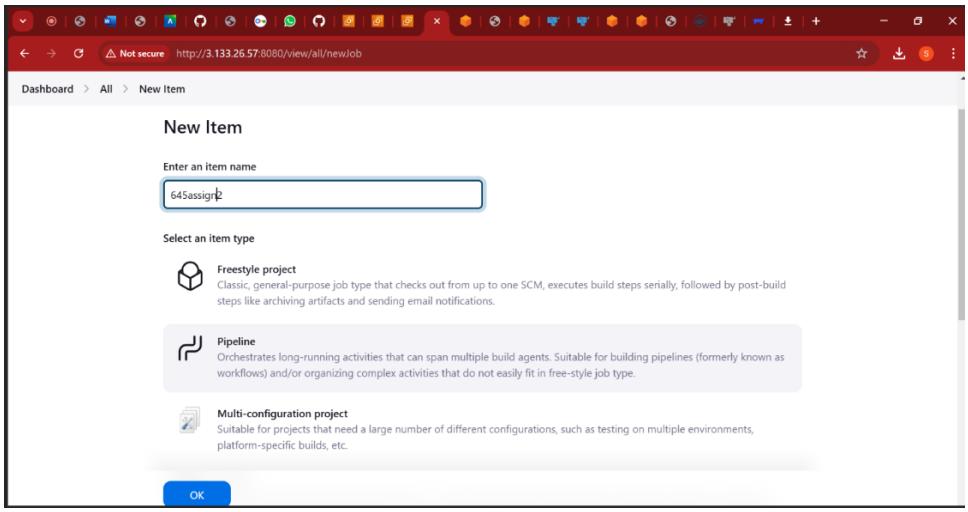


Creating a CI/CD pipeline and running it

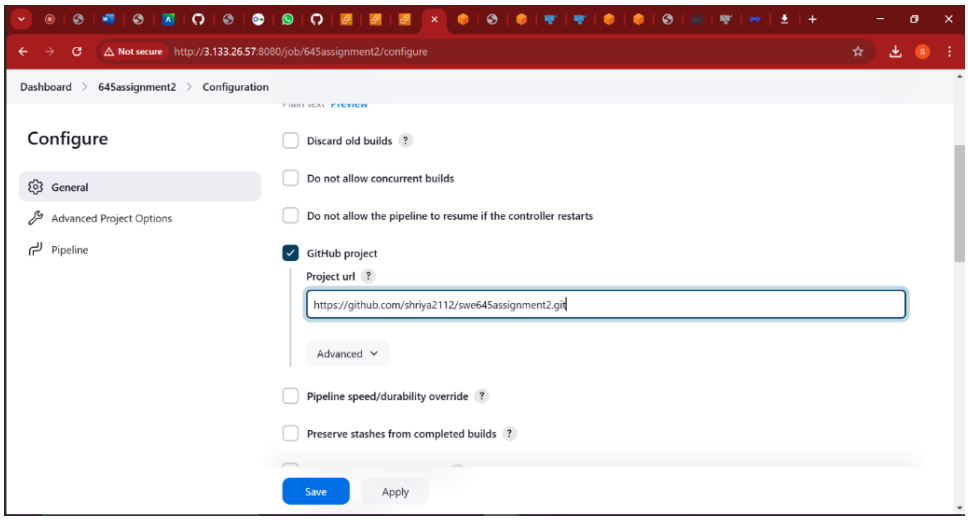
- Now, we have everything set up and ready to create our pipeline.
- Go to the jenkins dashboard, click on “Mange Jenkins”, scroll and select the “Credentials” option.
- Here, select the “global” option and then click on “Add credentials”.
- Select the Kind : “Username and password”, enter the dockerhub username and password and give id as “dockerhub” and click on create.



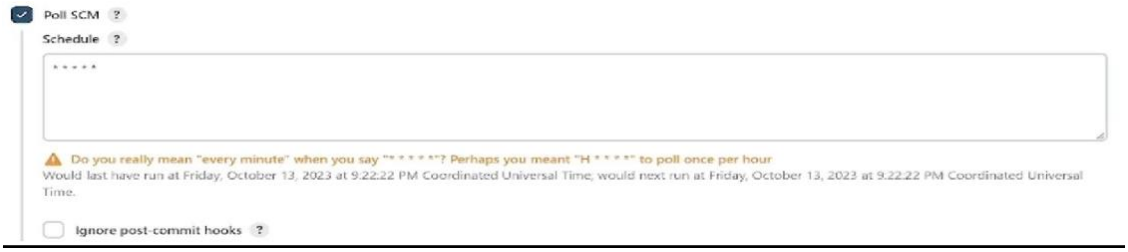
- Repeat the above steps to save GitHub credentials as well, give the id as “github”. For github, we will have to use PAT (Personal Access Tokens) as the password.
- Now go to the dashboard and click on “New Item”.
- Give the name as “645assign2” and click on “Pipeline” option.



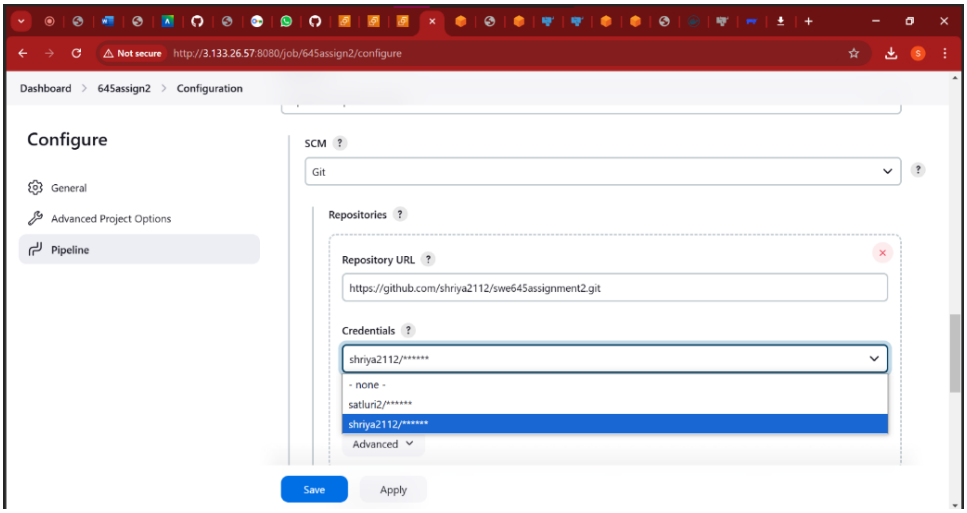
- Now, on the next page, check the “Github Project” checkbox, and in the project URL, paste the URL from Github repository : https://github.com/shriya2112/swe645assignment2.git.



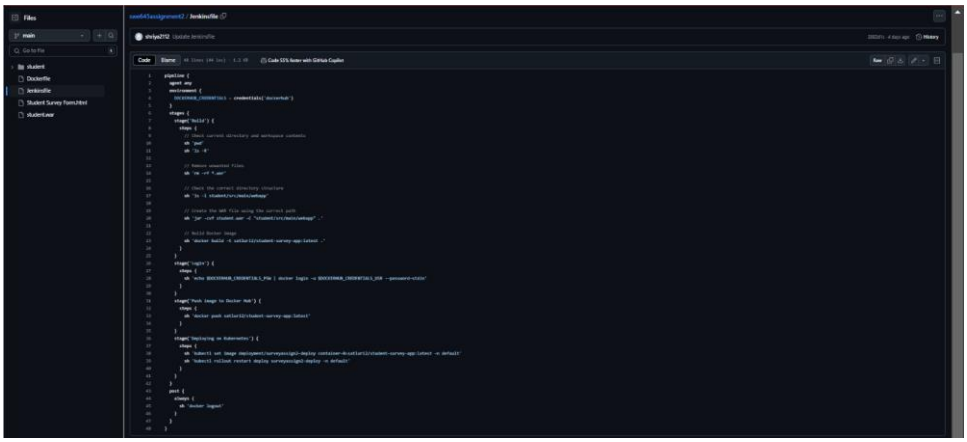
- Scroll down and check the “Poll SCM” checkbox.
- Give the Schedule as : “* * * * *”.



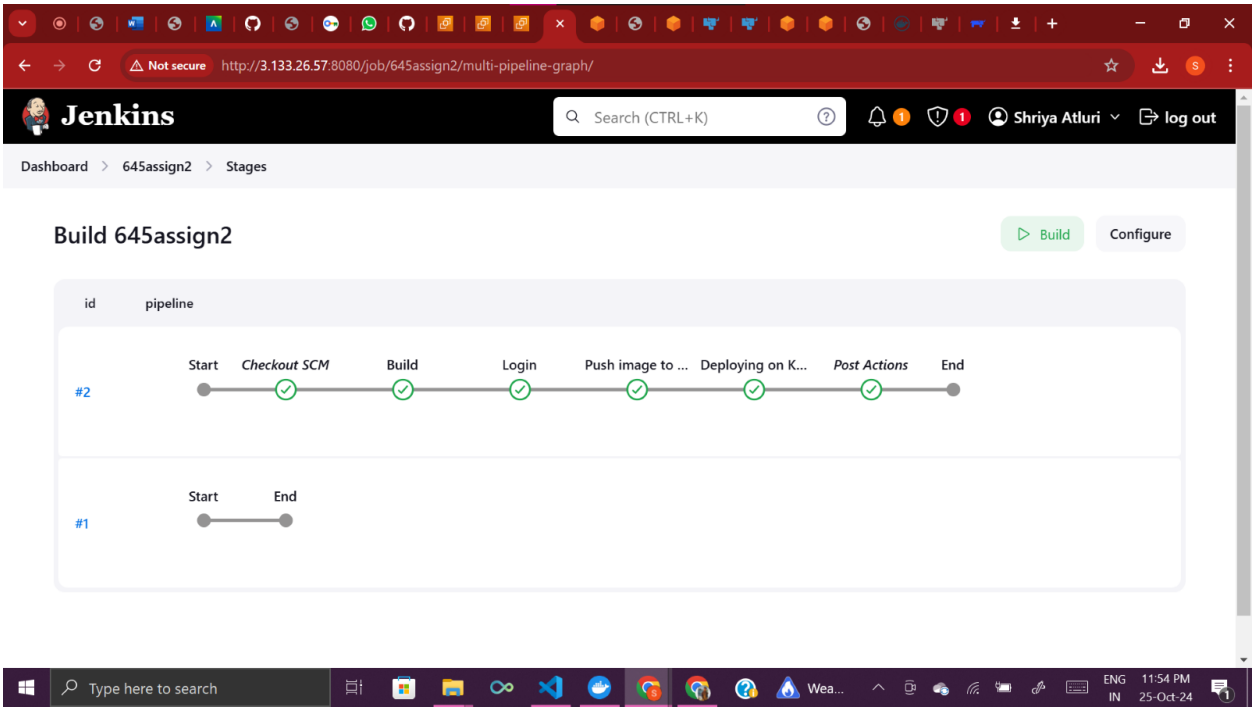
- Scroll down to the “Pipeline” section.
- Select the definition: “Pipeline from SCM” and SCM: “git”.
- Give the Github repository : “https://github.com/shriya2112/swe645assignment2.git”.
- Under Credentials, select the github credentials.
- Then, change the branch specifier : “*/main”.



- Go to the github repository and then create a new file : “Jenkinsfile” and then commit changes.
- Click on Save.
- Now a build will automatically start on this pipeline.
- Now go to the Github repository => “Jenkinsfile” and click on the edit option.
- Enter the commands given in the screenshot below and then, commit the changes. These are the commands that should be executed when any commit or changes occur to the files in the repository. They contain the commands on how to generate a new war file, create a docker image, login to the docker and push image including the deploy image on Kubernetes cluster.

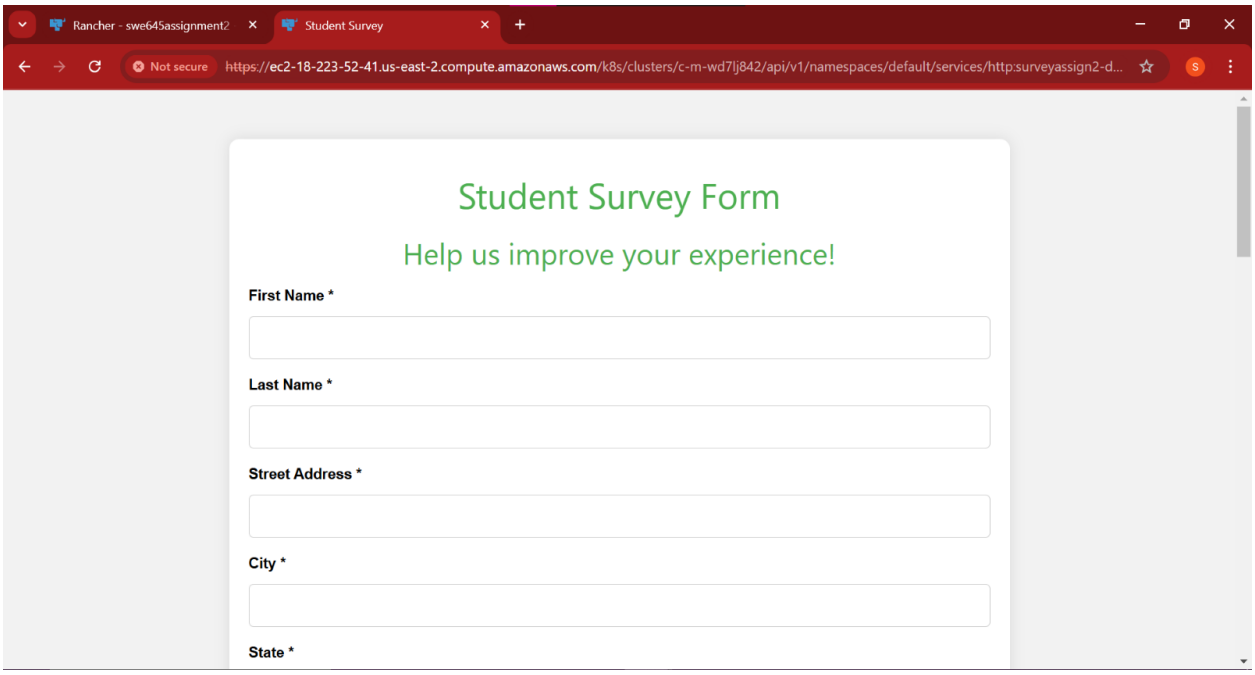


- Now when a build occurs, there might be some errors in the path, depending on the contents of the Jenkins file. Solve them by editing the Jenkinsfile accordingly.

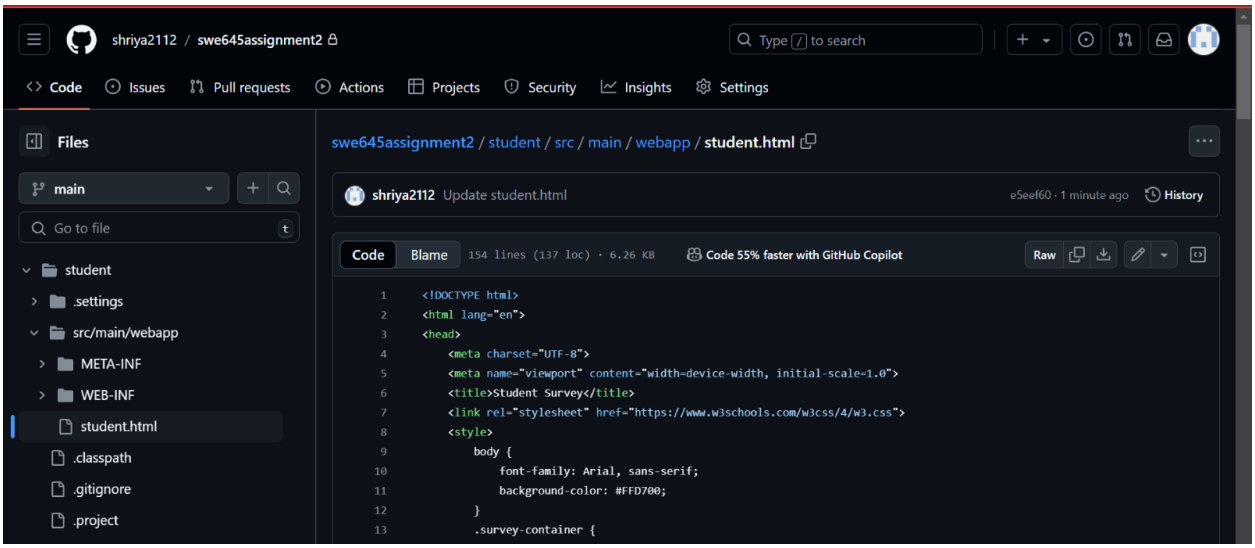


- Now after a successful build, open the application deployed on Kubernetes cluster using the same old URL. Don’t forget to append the “/student/” to the URL again.

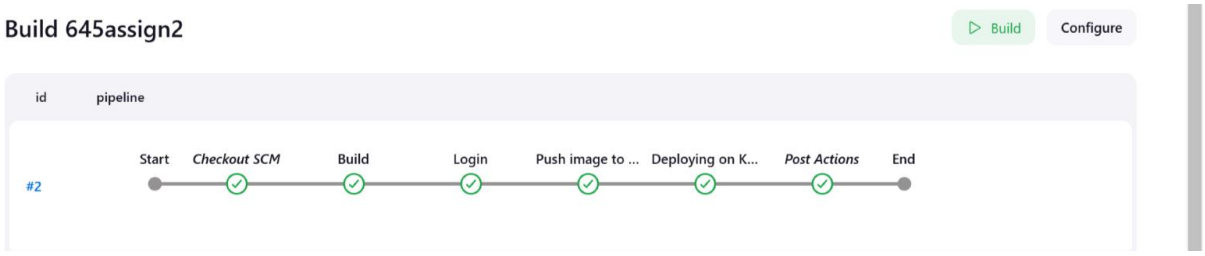
- And you should be able to see the application webpage.



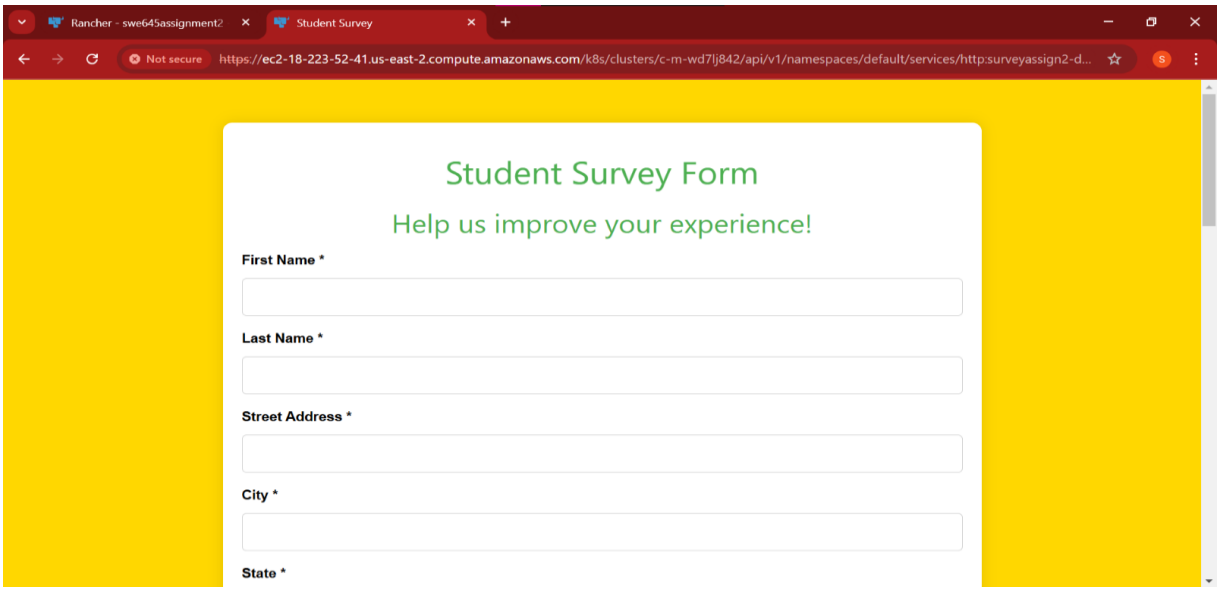
- Now to test the pipeline, we will bring about some changes to the source file. Open your source file in github and perform some changes. In our case, we have decided to change the background color of the survey page on our website to gold color from white, following the GMU color schemes. So, we edit the CSS file and commit the changes. Gold color = #FFD700



- After the commit, the pipeline automatically creates a build. After the build is successful, if we reload the webpage, we will be able to see the changes being reflected and the background color changes to gold.



- URL : https://ec2-18-223-52-41.us-east-2.compute.amazonaws.com/k8s/clusters/c-m-wd7lj842/api/v1/namespaces/default/services/http:surveyassign2-deploy:8080/proxy/student/



URL/Links of the application deployed

1. GitHub URL : https://github.com/shriya2112/swe645assignment2.git
2. Docker hub URL : https://hub.docker.com/r/satluri2/student-survey-app
3. Application deployed on cluster URL : https://ec2-18-223-52-41.us-east-2.compute.amazonaws.com/k8s/clusters/c-m-wd7lj842/api/v1/namespaces/default/services/http:surveyassign2-deploy:8080/proxy/student/
4. Docker URL : http://localhost:8183/student/
5. Rancher start URL : https://www.rancher.com/quick-start

References

1. https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04
2. Lecture 4 and 5, 6 and 7 class notes.
3. https://docs.docker.com/get-started/02_our_app/
4. https://docs.rke2.io/
5. How to Push Eclipse Project into GitHub: https://youtu.be/gO20QGT6aW8?si=IWoe8x6ODSEJItjU
6. Build & Push Docker Image using Jenkins Pipeline : https://youtu.be/PKcGy9oPVXg?si=s6XUUQj7tQAdhE3H

File Description

1. Front End files folder - Contains the HTML and CSS files of the assignment.
2. Student Survey Form – Contains the source code of the assignment.
3. Dockerfile – Docker file used to build the image.
4. Jenkinsfile – Jenkinsfile used to perform the builds after any commits/changes occur to the source code.
5. student.war – War file of the application.
6. student.zip – Zipped folder of all the source code of the application (from eclipse IDE).
7. swe645assignment2.yaml – Kubeconfig file of the cluster.
8. Swe 645 Assignment 2 – Execution video .mp4 – Video recording of the entire assignment development and execution.
9. SWE645_Assignment 2_README_SSP.docx – Word document , README file of the assignment.
10. SWE645_Assignment 2_README_SSP.pdf – PDF version of the README file.

Group Members and their Contributions

1. **Shriya Atluri (G01476041)**: Developed the source file, worked on creating and deploying application on Kubernetes cluster, worked on creating the docker image and the Github repository, Set up the CI/CD pipeline, worked on solving the issues and errors in setting up the jenkins and the CI/CD pipeline, contributed to the recording of the video for this assignment and the entire documentation part of the assignment.

AWS Homepage URL : http://shriya-2112.s3-website.us-east-2.amazonaws.com

2. **Sai Sampreeth Joshi (G01408085)** : Worked on creating and deploying application on Kubernetes cluster, worked on creating the docker image and the Github repository, Set up the CI/CD pipeline, worked on solving the issues and errors in setting up the jenkins and the CI/CD pipeline, contributed to the documentation part of the assignment.

3. **Pragathi Reddy Guduri (G01478980)** : Helped in setting up the jenkins environment and solving the issues and errors in setting up the jekins and CI/CD pipeline, contributed to the recording of the video for this assignment.