SWE 645 : Homework 3 README File

To finish Homework-3, several steps need to be completed. This README file provides a detailed overview of the tasks and procedures, along with accompanying screenshots. It also includes links/URLs to various elements of the assignment for reference.

1. Building an image of the application and uploading it to Docker Hub using Docker Desktop.

2. Configuring AWS EC2 instances to deploy the application on a Kubernetes cluster using Rancher.

3. Setting up Rancher.

4. Using the Rancher UI to deploy the application and set up a Kubernetes cluster.

5. Setting up an AWS EC2 instance for Jenkins installation and operation.

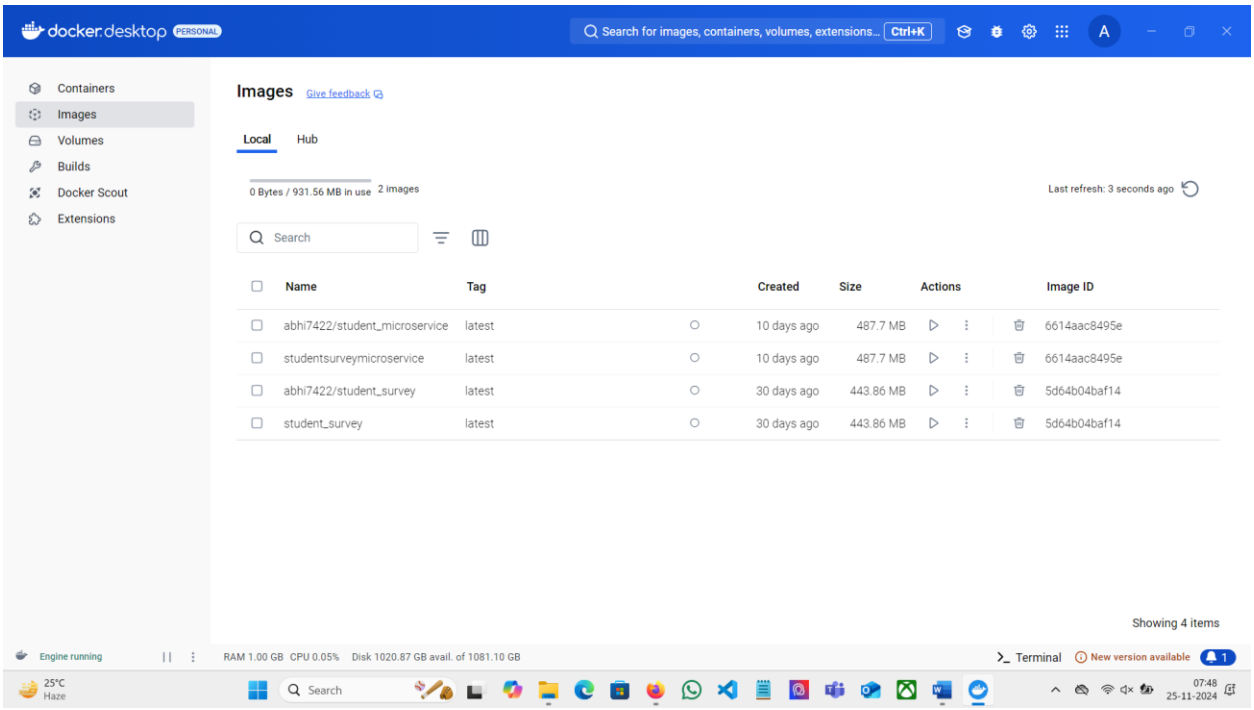
6. Setting up the project's GitHub repository.

7. Constructing a CI/CD pipeline and running it fully configured on Jenkins.

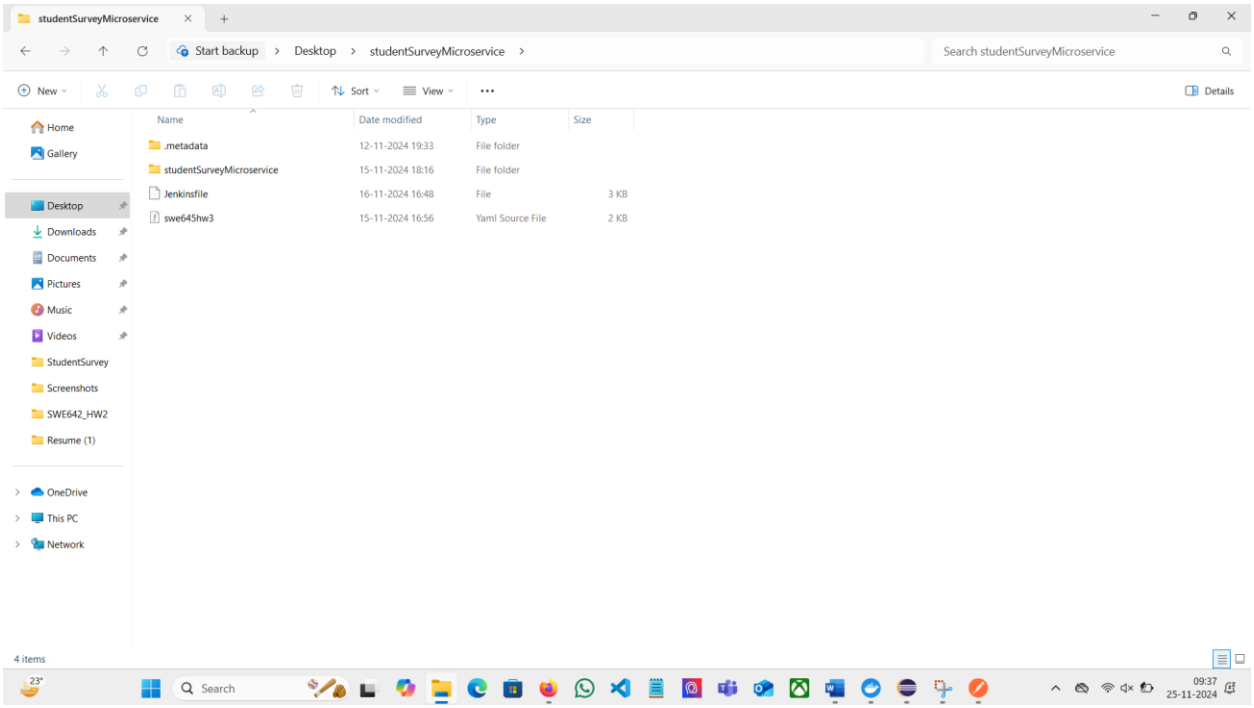
1. Building an image of the application and uploading it to Docker Hub using Docker Desktop.

• Go to https://hub.docker.com/ and create an account. Download Docker Desktop as well for smooth operation.

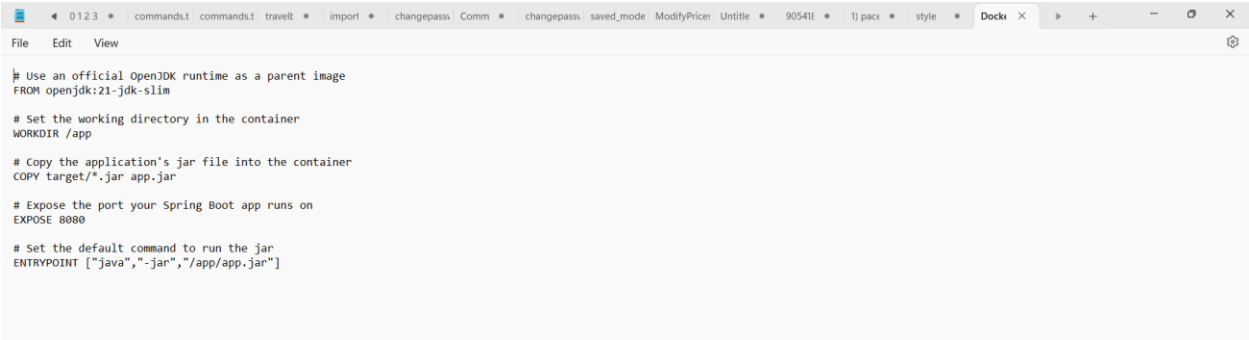
• Next Using the generated account, log in to Docker Desktop on a laptop and Docker Hub on a web browser.



• In the same directory as your ".jar" file, create a "dockerfile" now. Here, the "dockerfile" and the "StudentSurveyMicroservice.jar" files are located in the same directory.

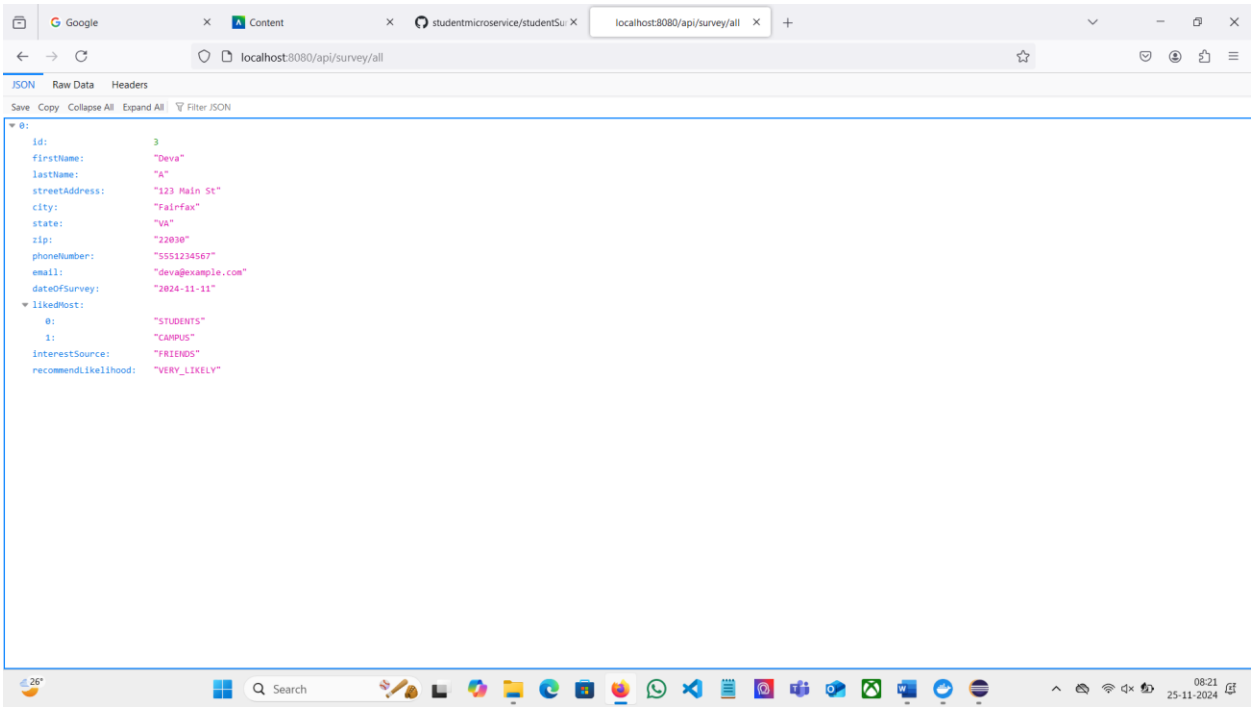


• As indicated in the image below, open the "dockerfile" in an editor and enter the following commands.

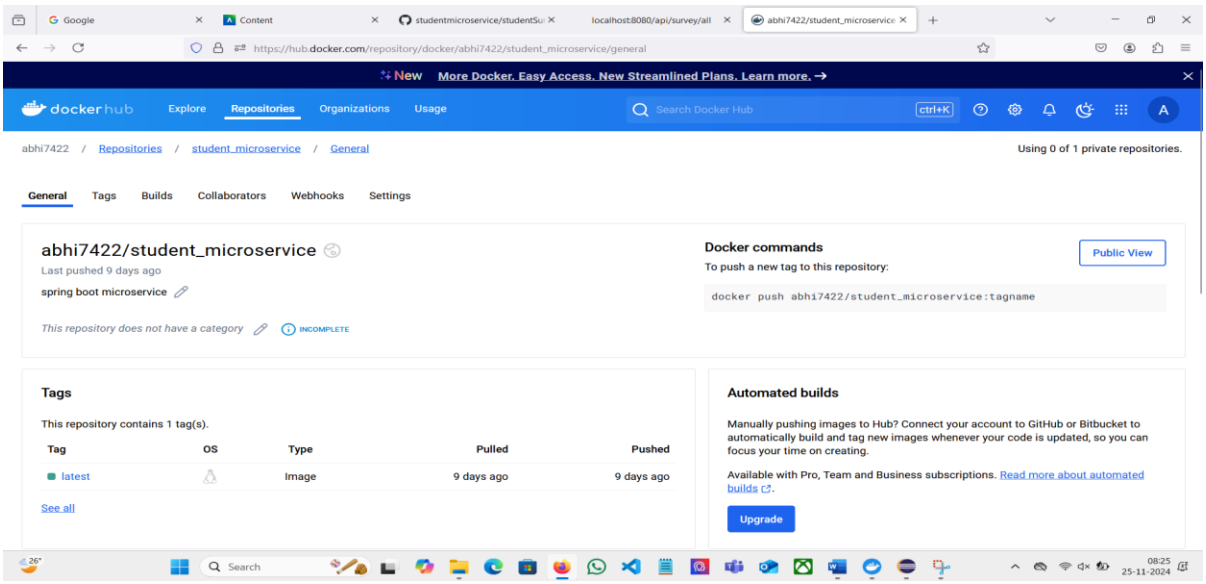


- Next, launch the command prompt from the directory containing the dockerfile and war file.
- To create a docker image, execute the following commands: "docker build -t studentsurveymicroservice."
- At this point, Docker Desktop creates an image of the program on the local machine. The command "docker run -it -p 8080:8080 studentsurveymicroservice" must be typed into the command prompt in order to launch this application on localhost.

- The server needs to be operational and the application needs to be installed on localhost:8080.
- Next, launch a browser and navigate to localhost:8081. Then, add the URL "/api/survey/all" to localhost. The application webpage ought to appear in the browser.
- URL : http://localhost:8080/api/survey/all

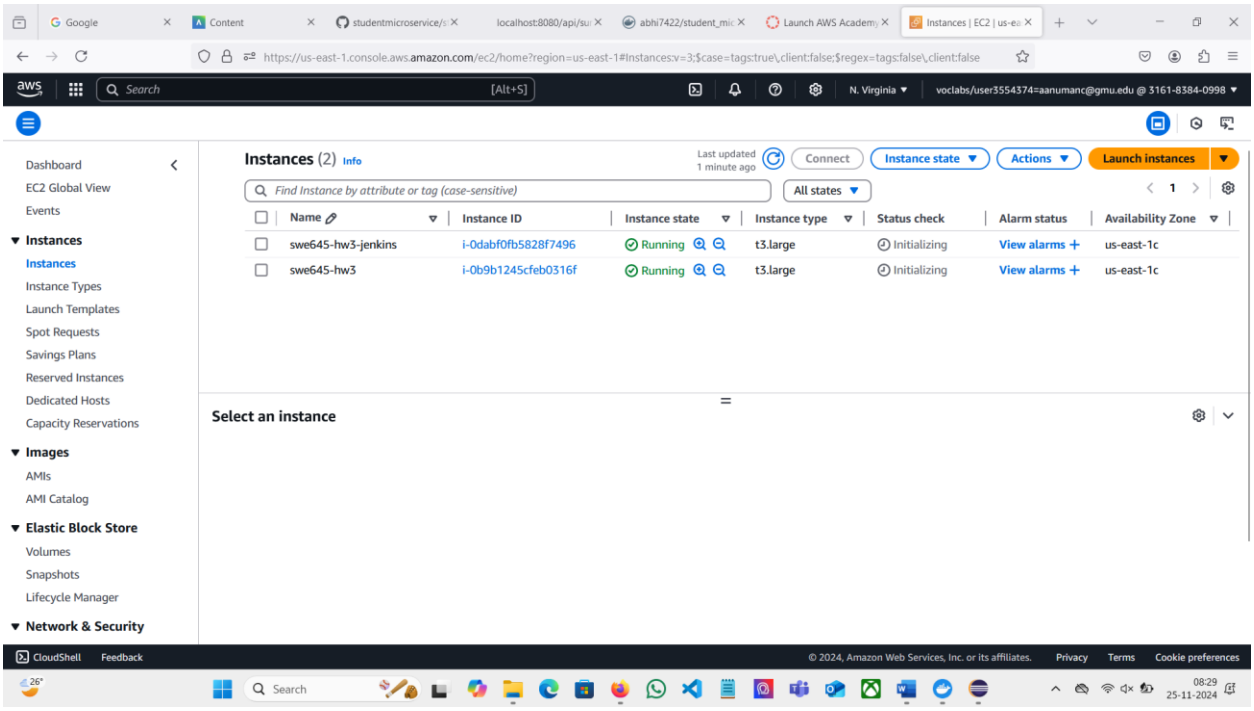


- The application image is now locally installed, and it has to be pushed from the Docker desktop onto the Docker Hub. To do this, we execute the subsequent commands from the same directory as the previous commands on the command prompt.
- To begin, connect in to Docker Hub with the previously generated account's credentials: "docker login -u abhi7422." When asked, enter the password.
- Next, use the commands "docker tag studentsurveymicroservice abhi7422/student_microservice" and "docker push abhi7422/student_microservice" to tag the image to be pushed and push it into the hub.
- As seen in the image below, the image is currently accessible in Docker Hub.

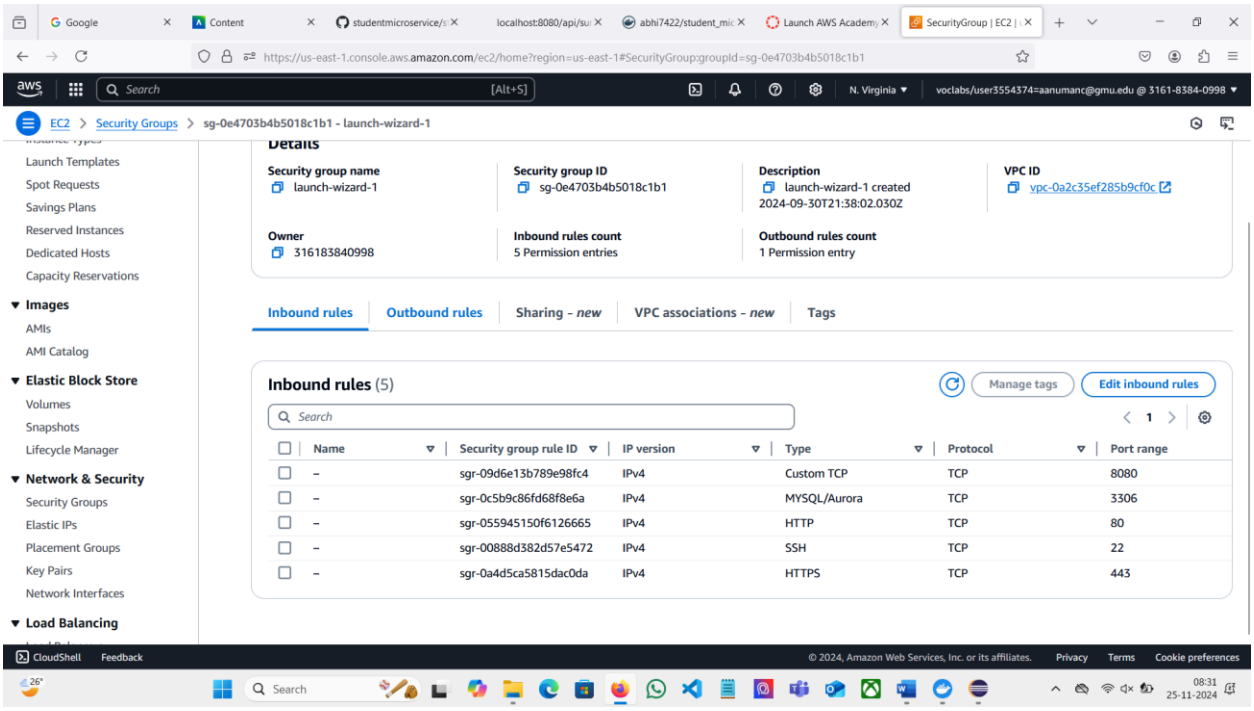


2. Configuring AWS EC2 instances to deploy the application on a Kubernetes cluster using Rancher.

- An AWS EC2 instance is required for the remainder of the assignment.
- After logging into AWS Academy, launch AWS, select the EC2 service, and then launch the dashboard.
- Here, click on “Launch Instance” to create the instance.
- Enter the instance name as "swe645-hw3," the instance type as "t3.large," and the AMI as "Ubuntu Server SSD volume Type." Choose a key pair that you already own, such as "HW2." To enable HTTP and HTTPS traffic from the internet, click the boxes. Lastly, expand storage from the default 8GB to 30GB. Lastly, select "Launch instance."
- After the instance has been established and is operational, assign it an elastic IP address. Go to the EC2 menu on the left pane of the screen, select "Elastic Ips," create an elastic IP using the default configuration, and link it to this instance. We take this action because every time the AWS lab is rebooted, the EC2 instances are restarted, and the IP addresses are changed, which causes problems with our Jenkins and Rancher configuration.



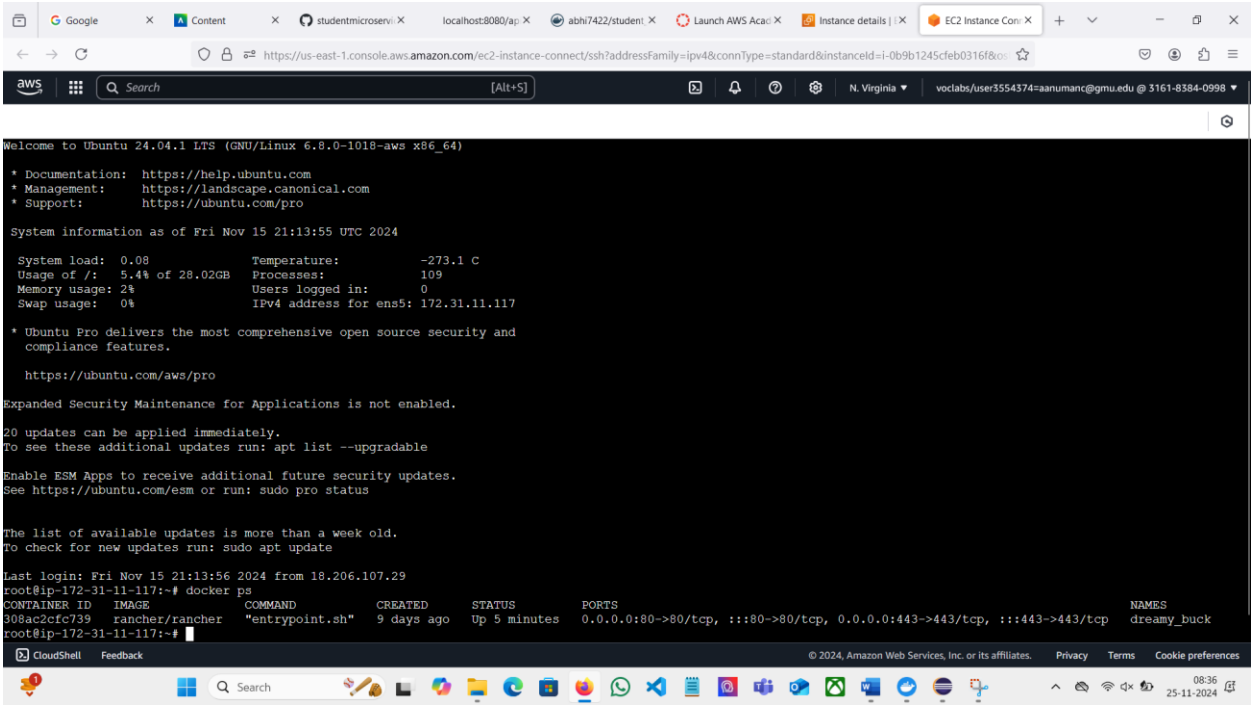
- Next, select the security tab by clicking on the instance. Click on the security group wizard in the "inbound rules"/"outbound rules" section.
- From the "inbound" or "outbound" tabs, scroll down and select the "edit rule" option.
- Click "Save rules" after adding a rule with the following settings: Type: "Custom TCP," Port range: "8080," Source: "custom" and "0.0.0.0/0."
- After selecting the EC2 instance, select "connect." Select the "EC2 instance connect" option, enter "root" as the username, and then click "connect." A new tab opens with a shell.
- Here, we'll provide instructions for installing Docker on the instances.
- Commands to install docker : “ sudo apt update” followed by “sudo apt install docker.io -y”.
- Docker should be successfully installed following the successful execution of the tasks.



3. Setting up Rancher.

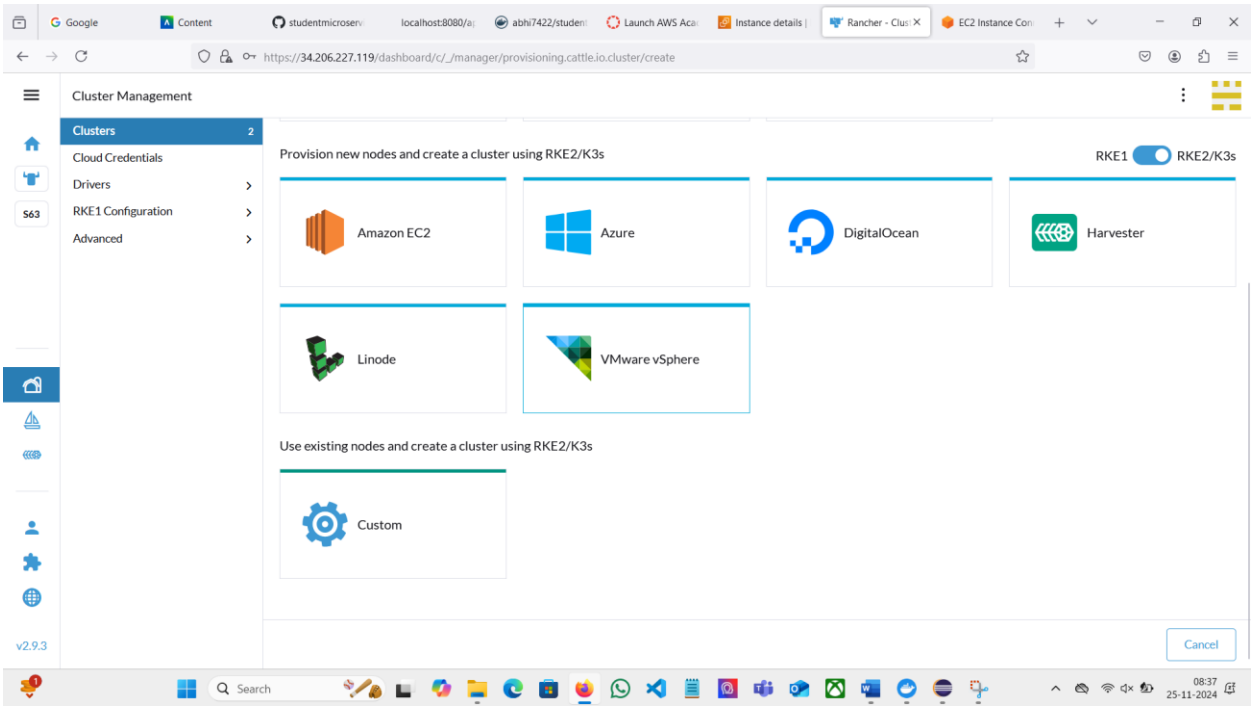
- Launch a web browser and navigate to https://www.rancher.com/quick-start. Here, under "Deploy Rancher," scroll down to the "Start the Server" section.
- Copy the command present there, which is : " $ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher ".
- Now, connect to the "swe645-hw3" instance in the same way as described in the earlier section.
- Run the copied command here. Rancher will have been installed on this instance and accessible via the instance's public IPv4 DNS address once it has completed its successful execution.
- To obtain information about the container that is operating on the instance, execute the following command on the instance: "sudo docker ps." We will need this information later, so save the result somewhere (ideally in a notebook).
- Next, open the "Public IPv4 DNS" URL in a new tab on the EC2 instance page. Permit access to the URL despite the fact that it is dangerous. The rancher login page ought should appear after loading.
- 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 to have the container id from the earlier stored output and run it on the instance console. The command should look like this : "sudo docker logs 308ac 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.

- At this point, the rancher dashboard will appear, allowing us to move forward with cluster creation and application deployment.

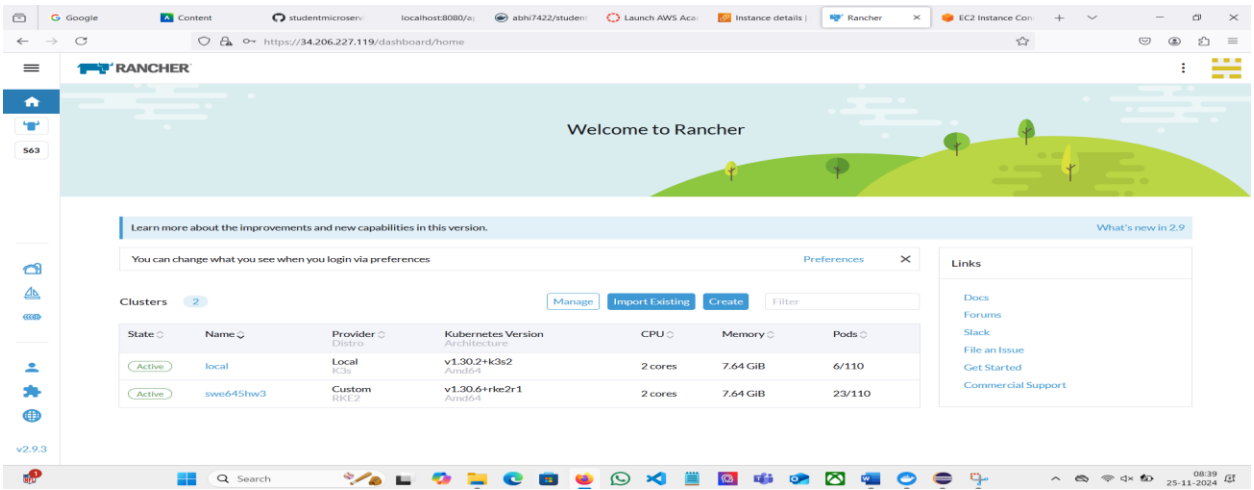


4. Using the Rancher UI to deploy the application and set up a Kubernetes cluster.

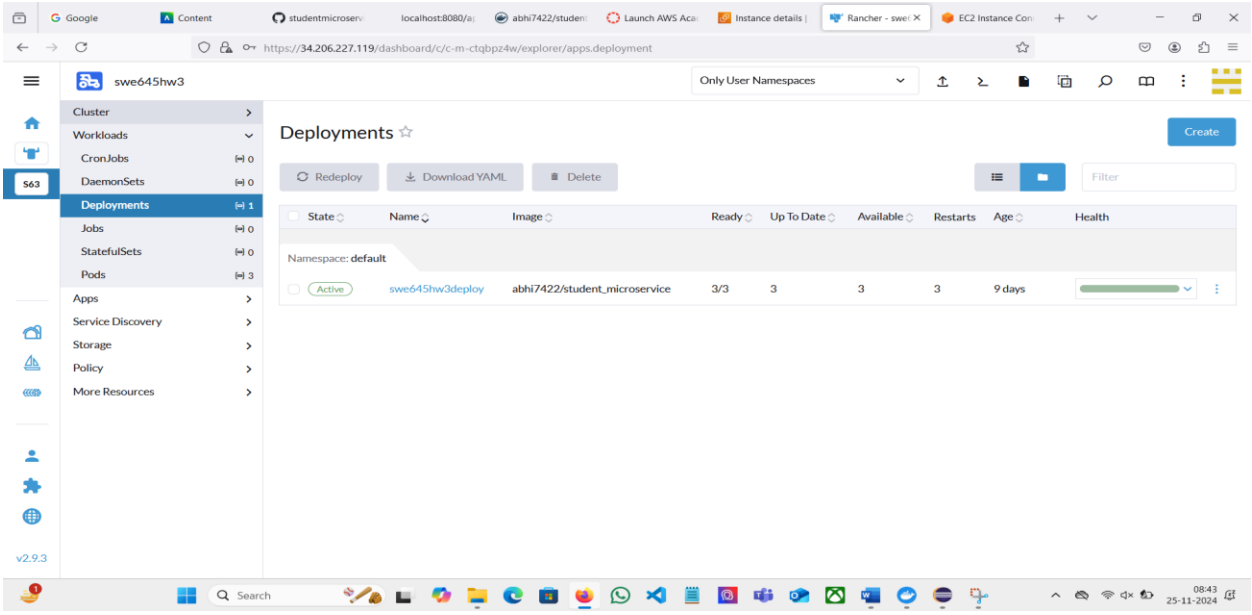
- Now that Rancher is operational, we will use the Rancher UI to create a cluster and deploy our application on it.
- Click the "Create Cluster" button on the dashboard. After choosing the "Custom" option from the list of cluster options, a form to establish a cluster should appear.

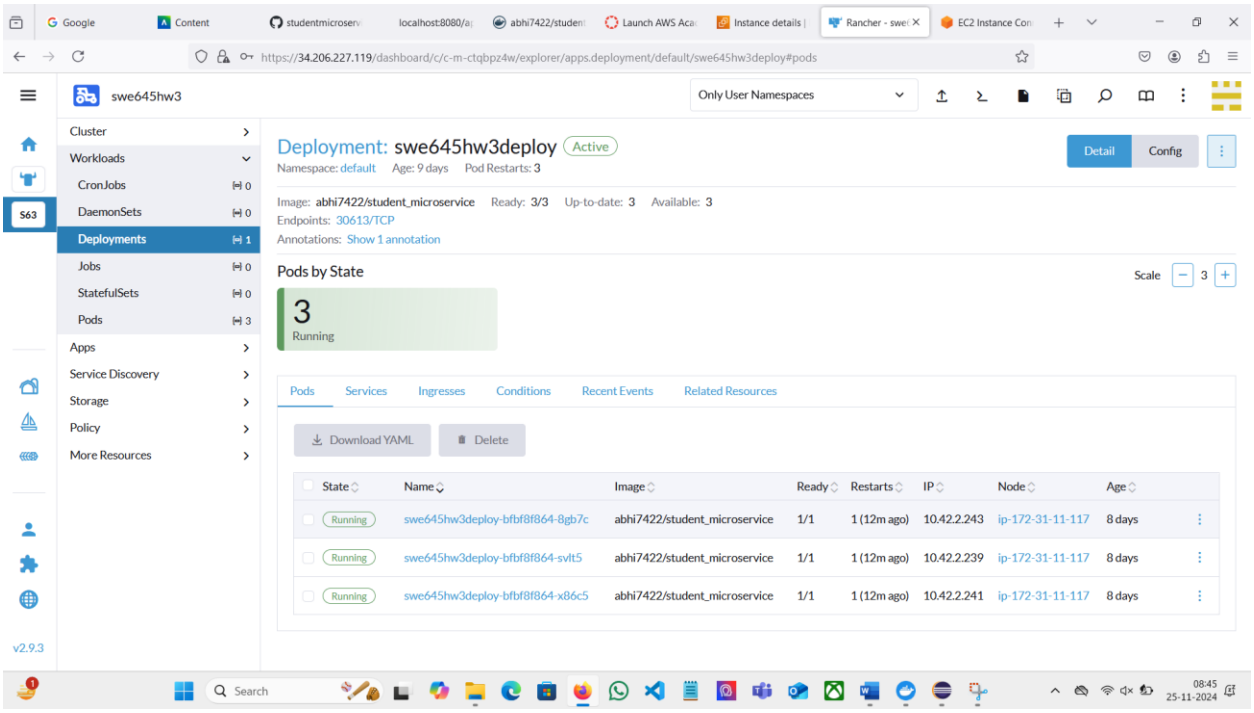


- In this case, fill out the form with the cluster name "swe645hw3", leaving all other fields empty, and press the "Create" button.
- Select the "Registration" option on the following page, and under "step1," confirm that the three checkboxes labelled "etcd," "control pane," and "worker" are selected.
- Now copy the command present below in “step2”.
- Here run the copied command but append the “—insecure" after the “curl” word in the command then run it.
- The cluster should be in the "Updating" state after the command has been successfully run; it will transition to the "Active" state a few minutes later, at which point it will be prepared for deployment. The Rancher UI's "Clusters" option under "cluster management" allows you to view this.
- Once the cluster is in “Active” state, click on the “Explore” button next to the cluster.

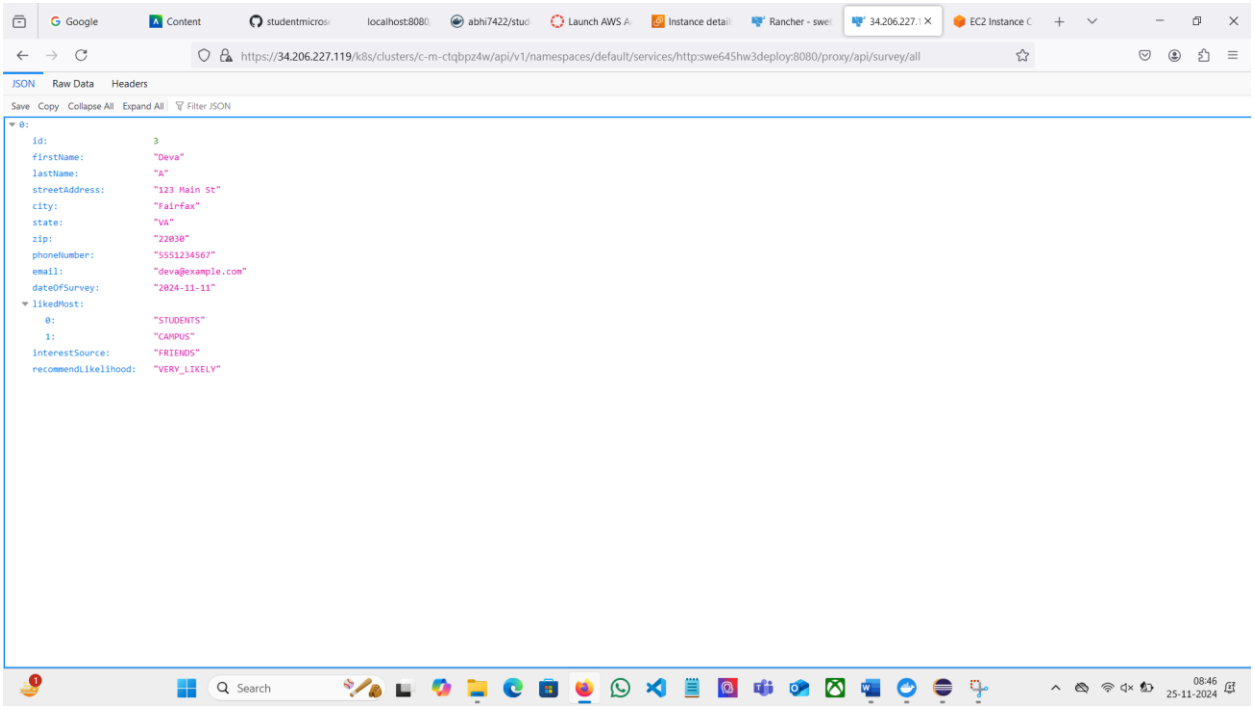


- On the left pane, under “workloads” select “Deployments”.
- When you click the "create" button, a form ought to appear on the screen.
- Fill in the form using the following details. Namespace : “default”, Name : “swe645hw3deploy”, Replicas : “3” (no. of pods), Container image : “abhi7422/student_microservice:latest” (same as the name of the image on the docker hub).
- Select the "Add port or service" button under Networking.
- Provide the following information: protocol: TCP, private container port: 8080, service type: "node Port," and name: "nodeport."
- Leave everything else as default and click on create.
- The deployment is now finished, and it ought to be in the "Updating" state. After a few minutes, it ought to be in the "Active" state.
- Once it is in the “Active" state, click on the deployment, navigate to the “Services” tab.
- Click the "nodeport" button to open the deployment in a new tab under the target option.

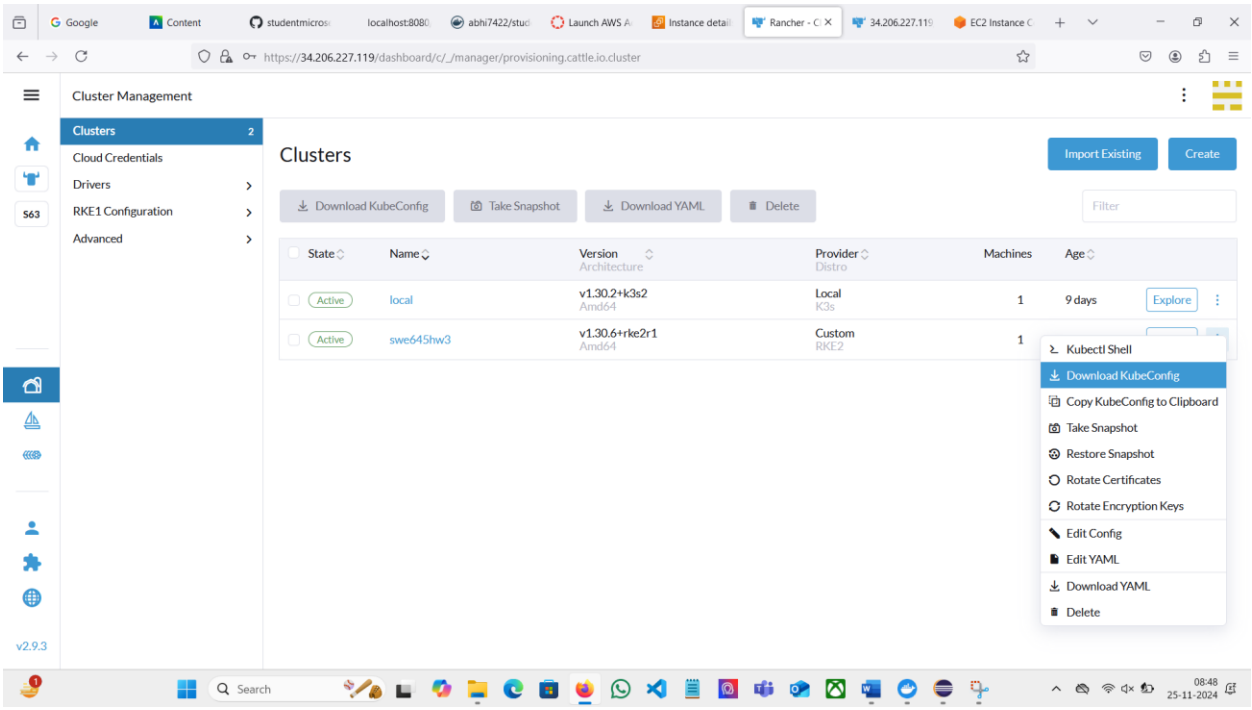




- Now add "/api/survey/all" to the URL and click "Open." The application ought to be visible to you as it runs.
- URL: https://34.206.227.119/k8s/clusters/c-m-ctqbpz4w/api/v1/namespaces/default/services/http:swe645hw3deploy:8080/proxy/api/survey/all

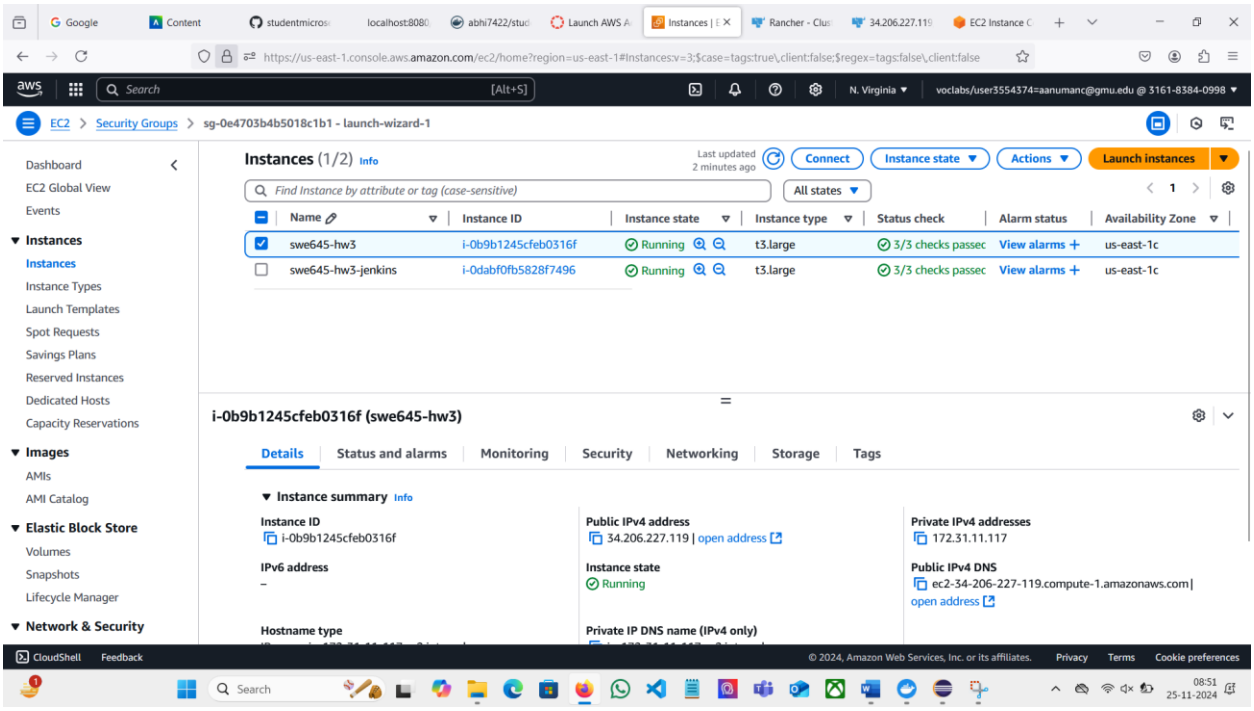


- The "KubeConfig" file is required for the following step. Visit our cluster page to get it. Click "Download KubeConfig file" from the option on the top right and save it to your computer.



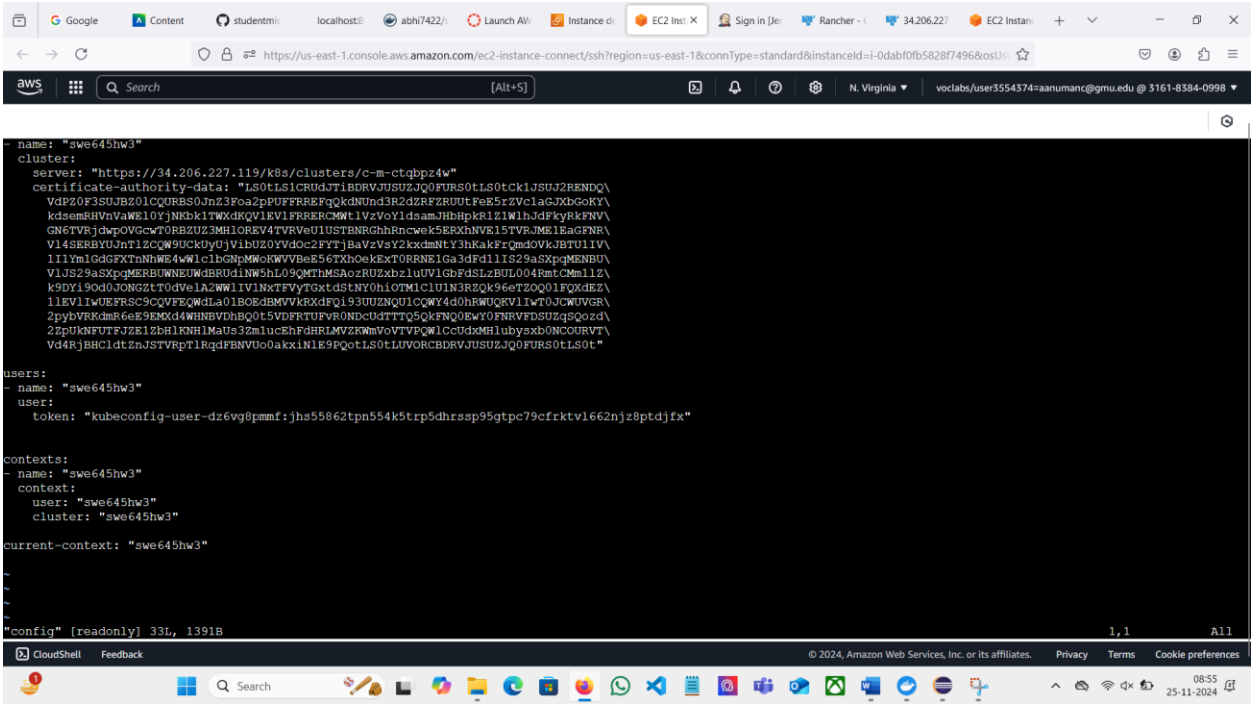
5. Setting up an AWS EC2 instance for Jenkins installation and operation.

- Using Jenkins, we will now concentrate on developing a CI/CD pipeline that will automatically build and update our source code as needed.
- To do this, we must first install Jenkins on our computer, thus we will begin by setting up an AWS EC2 instance in our AWS lab.
- To deploy our application on the Kubernetes cluster, we will construct the instance in the same way that we created our two EC2 instances.
- Give the EC2 instance the name "swe645-hw3-jenkins," follow the same setup, set an elastic IP, and modify the security group.



- Use "EC2 instance connect" to establish a connection to the instance. Verify that the instance is in the state of "running."
- Before using Jenkins, Java must be installed.
- The commands "sudo apt update" are used for that.
- Now run “sudo apt install openjdk-21-jdk -y”. This should install java jdk-17.
- Now Jenkins can be installed by running the following commands “sudo wget -O /usr/share/keyrings/jenkins-keyring.asc\https://pkg.jenkins.io/debian-stable/jenkins.io2023.key
  echo"deb[signedby=/usr/share/keyrings/jenkins-keyring.asc]"
  \https://pkg.jenkins.io/debian-stable binary/ | sudo tee \ /etc/apt/sources.list.d/jenkins.list > /dev/null
- “sudo apt-get update” followed by next step,
- “sudo apt-get install Jenkins -y” after that Jenkins should be successfully installed.
- 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”.
- Now go to AWS EC2 instance “Jenkins” and open the “Public IPv4 address” in a new tab.
- To the url append the port 8080 and change from “https” to “http”. It should look like the following: http://3.220.79.129:8080.
- Enter the admin password obtained earlier to unlock Jenkins.
- Next, install the recommended plugins.

- Next, make an administrator user. Select "Save and create user," "Save and finish," and "Start using Jenkins" in that order.
- The Jenkins dashboard ought to be visible.
- Return to the EC2 console now. We must now build a configuration 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 it : “mkdir .kube” followed by “cd .kube”.
- Open the file by creating it with "vi config."
- Now copy the contents from the “KubeConfig” file downloaded earlier and paste it here.
- Save and quit the file using “:wq”.

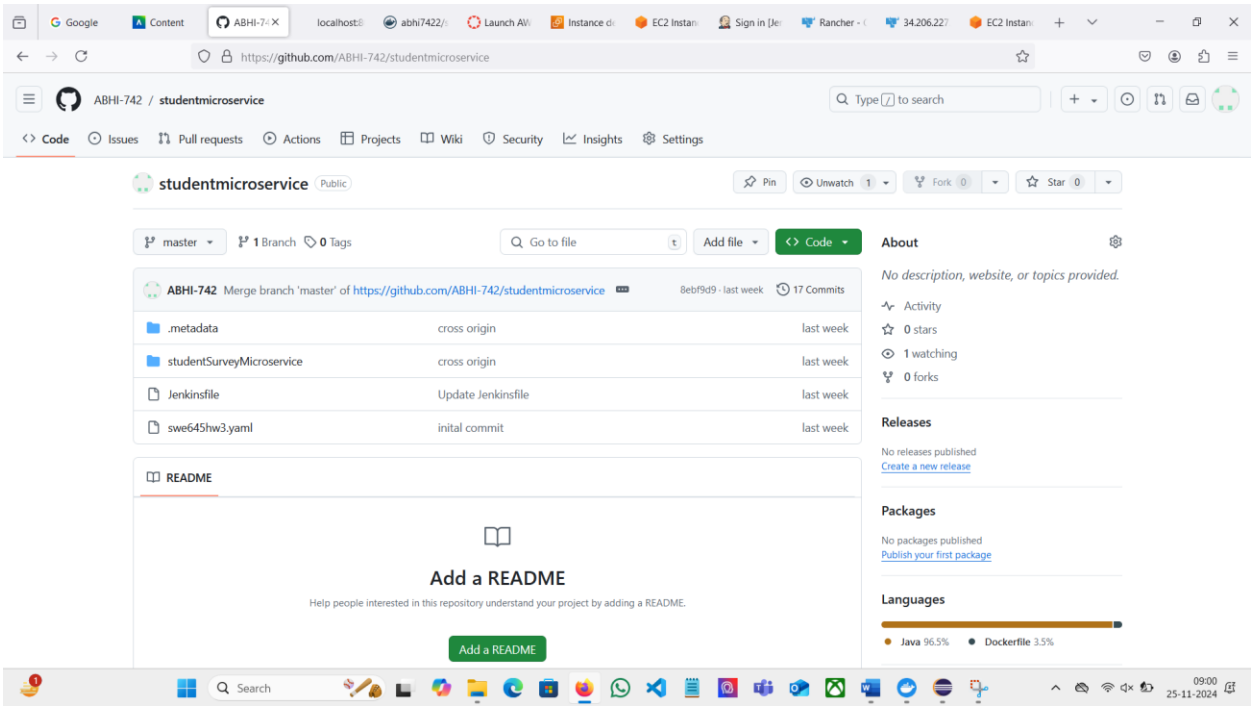


- Use the command "kubectl config current-context" to confirm. The cluster name ought to appear as an output.
- Next, use the "exit" command to end Jenkins mode.
- Now install docker with the given commands : “sudo apt update” followed by “sudo apt install docker.io -y”.
- Using the commands "sudo apt-get update" and "sudo apt update," followed by "sudo apt install docker.io," install Docker now. Permit as "Y."

6. Setting up the project's GitHub repository.

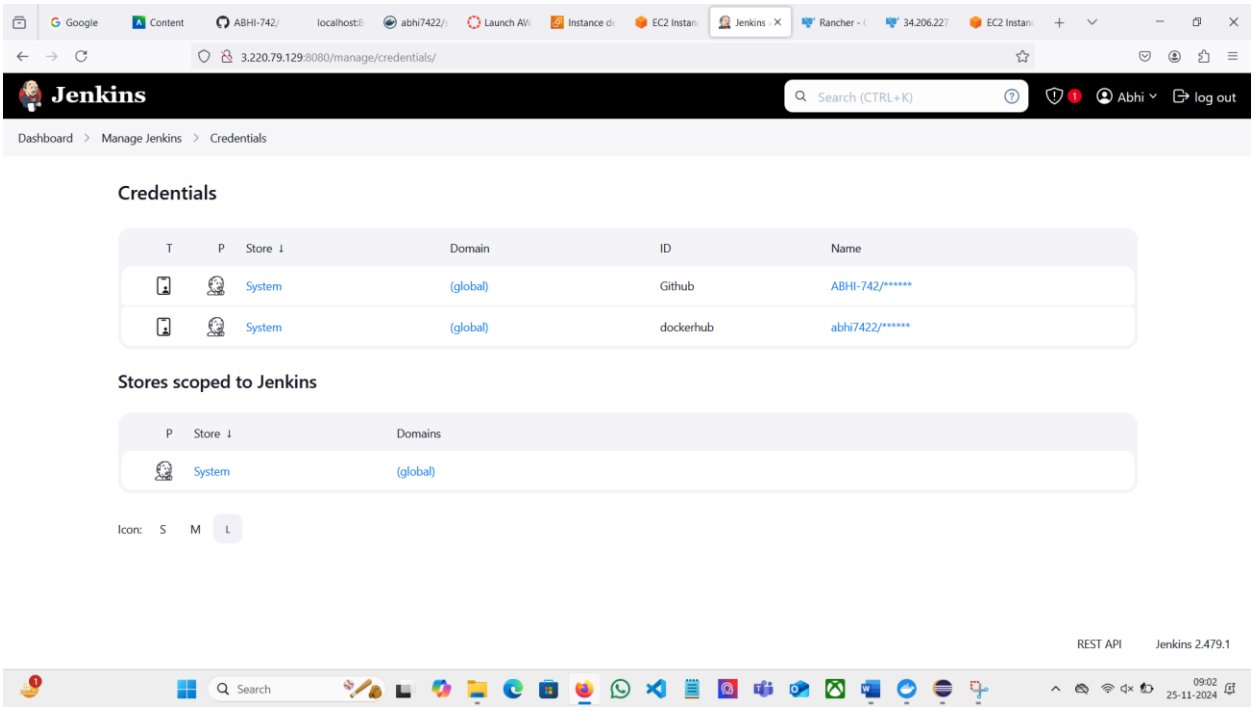
- The source code must be saved somewhere so that it can be attached to the pipeline and used to detect changes before we can generate it. We're going to use GitHub for this.
- Create an account and log in at https://github.com/.

- Now create a new repository : “ **studentmicroservice**”.
- To that repository add your source files, Dockerfile and the war file.
- After then commit them.
- This repository will be used by our pipeline.
- Github repository URL : https://github.com/ABHI-742/studentmicroservice.git

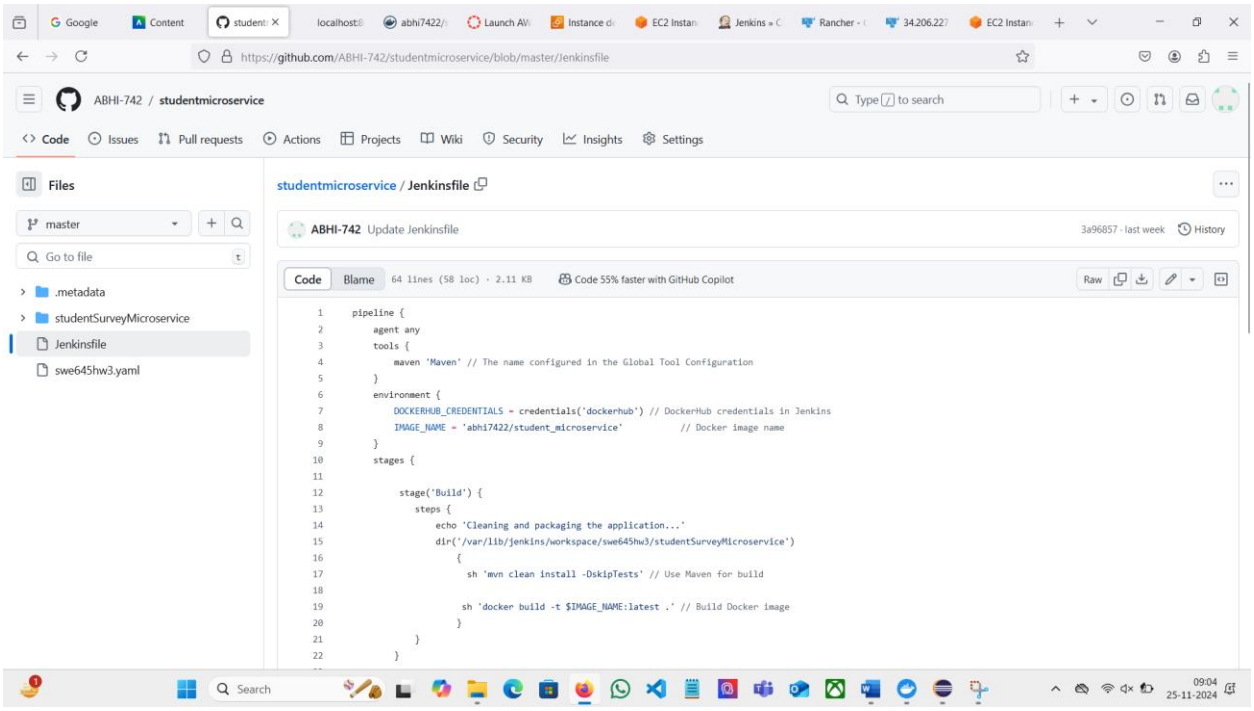


7. Constructing a CI/CD pipeline and running it fully configured on Jenkins.

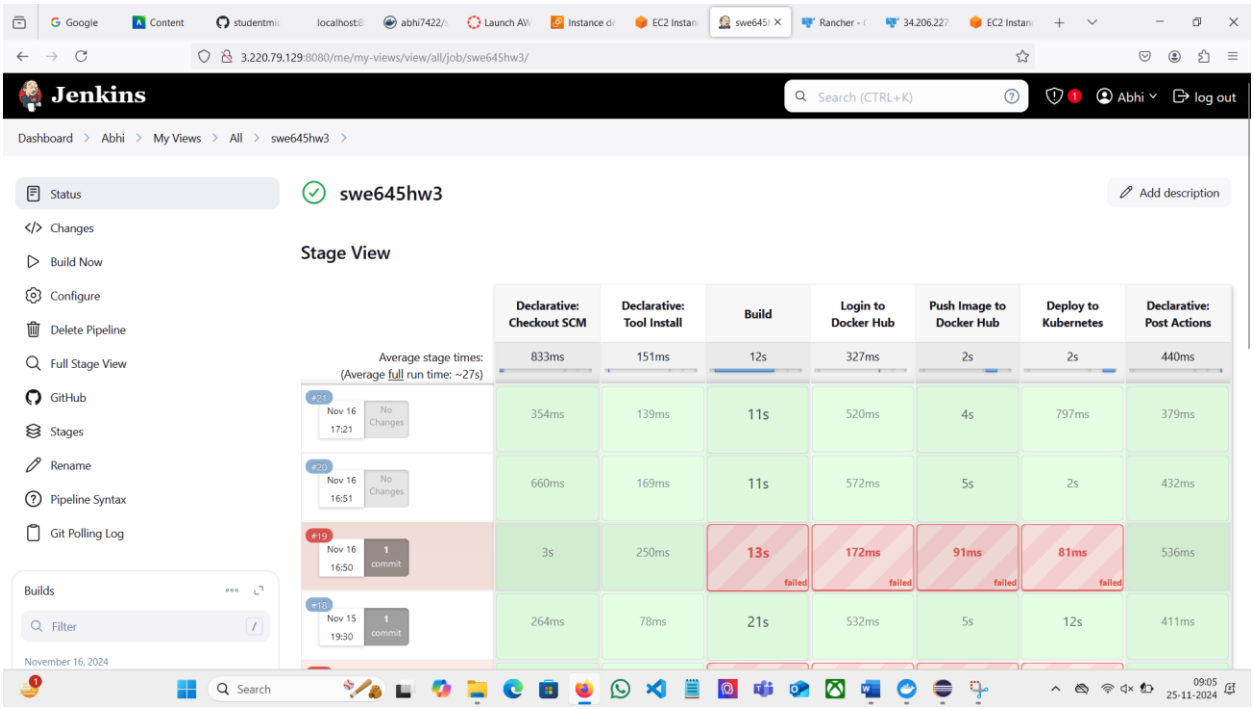
- Everything is now prepared for the creation of our pipeline.
- Navigate to the Jenkins dashboard, pick "Credentials," scroll, and click "Mange Jenkins."
- Click "Add credentials" after choosing the "global" option.
- Click on Create after choosing Kind: "Username and password," entering the dockerhub username and password, and providing the ID "dockerhub."
- Repeat the above step to save GitHub credentials as well, give id as “github”.



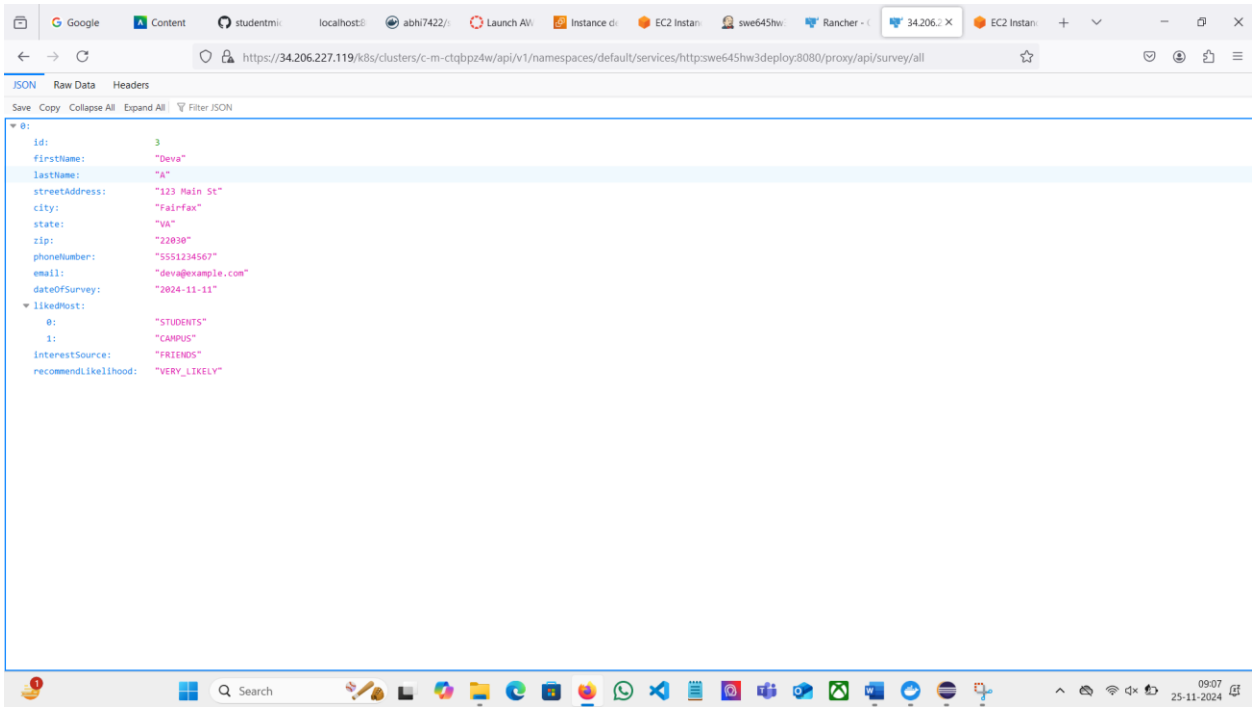
- Go to the dashboard now and select "New Item."
- Give name as “swe645hw3” and click on “Pipeline”.
- Now on the next page, check the “GitHub Project” checkbox, and in the project URL paste the URL from GitHub repository: “https://github.com/ABHI-742/studentmicroservice.git”.
- Scroll down and check 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/ABHI-742/studentmicroservice.git”.
- Under Credentials select the GitHub credentials.
- Then change the branch specifier: “*/master”.
- Go to 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 edit.
- Commit the changes after entering the commands shown in the screenshot below. These commands, which include instructions on how to create a new war file, log in to Docker and push the image, and deploy the image on a Kubernetes cluster, should be run whenever there is a commit or modification to the files in the repository.



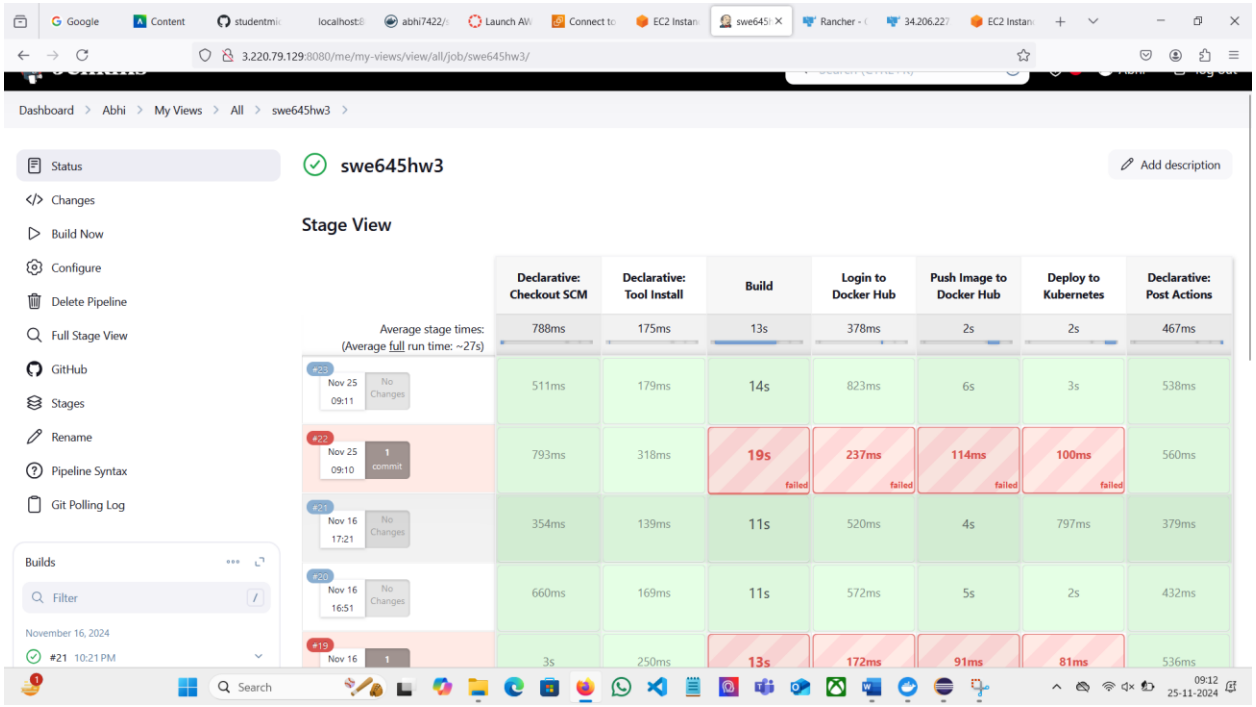
• During a build, there may be path and all issues; fix these by modifying the Jenkins file.



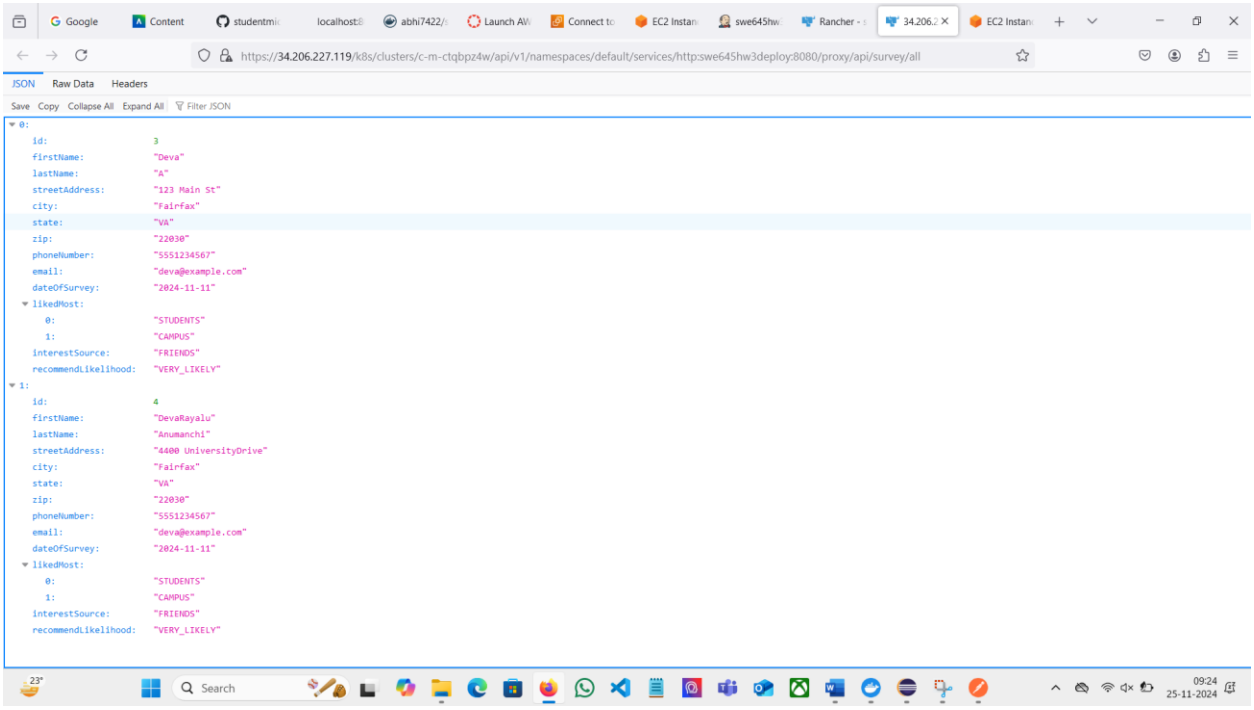
• Using the same URL, launch the application that was deployed on the Kubernetes cluster following a successful build. Additionally, the application webpage ought to be visible to you.



- Now to test the pipeline, open your source file in github and perform some changes.



- URL https://34.206.227.119/k8s/clusters/c-m-ctqbpz4w/api/v1/namespaces/default/services/http:swe645hw3deploy:8080/proxy/api/survey/all



URL/Links of the application deployed:

1. GitHub URL : https://github.com/ABHI-742/studentmicroservice.git
2. Docker hub URL : https://hub.docker.com/repository/docker/abhi7422/student_microservice/general
3. Application deployed on cluster URL : https://34.206.227.119/k8s/clusters/c-m-ctqbpz4w/api/v1/namespaces/default/services/http:swe645hw3deploy:8080/proxy/api/survey/all

References:

1. Assignments The assignment folder contains two linked papers.

File description:

1. Dockerfile – Docker file that is used to build an image.
2. Jenkinsfile – Jenkinsfile used to perform build after ever commit or change in the source code.
3. studentSurveyMicroservice – that contains jar file in the target foleder of the application
4. swe645hw3.yaml – Kubeconfig file of the cluster
5. SWE645_HW3_README_FILE – PDF version of the README FILE.

Group Members:

1. Abhi Venkata Sri Krishna Devarayalu Anumanchi (G01459507).