SWE 645 : HOME WORK 2

Kiran Muddana – G01413581

Venu Dammalapati – G01460891

Yashasree Marram – G01380842

This guide will walk through deploying an application using a CI/CD pipeline. Here's a breakdown of the steps:

1. Containerize your application: We'll build a Docker image and store it in Docker Hub.
2. Prepare the deployment environment: We'll set up Amazon EC2 instances to run a Kubernetes cluster using Rancher.
3. Deploy the application: We'll create a Kubernetes cluster on the EC2 instances and deploy your containerized application there.
4. Set up continuous integration and delivery (CI/CD): We'll configure a Jenkins server on another EC2 instance and create a pipeline to automate building, testing, and deploying your application.
5. Version control: We'll assume you have a Git repository set up to store your application code.

1. **Containerize your application:**

1. Setting Up Docker:

- Create a Docker Hub account at https://hub.docker.com/.
- Download Docker Desktop for a graphical interface (optional).
- Login to Docker Desktop or Docker Hub using your account credentials.

2**. Building the Docker Image:**

Create a file named Dockerfile in the same directory as your ".war" file (e.g.SurveyForm.war).

Edit Dockerfile to include the specific instructions for your application.

```
FROM tomcat:10.1-jdk21
COPY SurveyForm.war /usr/local/tomcat/webapps/
EXPOSE 8080
```

Open a command prompt or terminal in the directory containing Dockerfile and the ".war" file.

Run the command "**docker build -t studentsurveyform .**" to build a Docker image named studentsurvey. This uses the current directory (.) as the context for building the image.

```
C:\Windows\System32\cmd.e
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\skmud\OneDrive\Desktop\MS\645\Assignment2>docker build -t studentsureyform .
[+] Building 3.7s (5/7)                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                0.1s
 => => transferring dockerfile: 120B                                                                0.0s
 => [internal] load metadata for docker.io/library/tomcat:10.1-jdk21                                1.4s
 => [auth] library/tomcat:pull token for registry-1.docker.io                                       0.0s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [1/2] FROM docker.io/library/tomcat:10.1-jdk21@sha256:c83cd6449a4d34d6c5a9fe11934c15a09c4762a2b0db46c916a4fba  2.0s
 => => resolve docker.io/library/tomcat:10.1-jdk21@sha256:c83cd6449a4d34d6c5a9fe11934c15a09c4762a2b0db46c916a4fba  0.0s
 => => sha256:56e22531d1d0ae3c4cc8c3288522e9f1df7b534c2f1988b48de8f2ab6276575d 2.00kB / 2.00kB      0.0s
 => => sha256:e2670537dcebc0b445ca47e68b31667f3572b8758b8388e24d8a6770e860ecc8 14.68MB / 17.46MB    2.0s
 => => sha256:c44aaba36909c15b3b9d03a8a296637d8661ba2986ba8ff6927f631fc988cdc7 16.78MB / 159.59MB   2.0s
 => => sha256:c83cd6449a4d34d6c5a9fe11934c15a09c4762a2b0db46c916a4fba032e6b117 979B / 979B          0.0s
 => => sha256:405afe63d576023f51af416d0ac163f73268104891053b0a1d557a5435137bc6 13.02kB / 13.02kB    0.0s
 => => sha256:23828d760c7b04df02891af556c40ca44c2dd79d6837ea6f18fac24f4108448c 18.87MB / 30.45MB    2.0s
 => [internal] load build context                                                                   0.0s
 => => transferring context: 36B                                                                    0.0s
```

3. Running the Application Locally:

Run the command "**docker run -it -p 8184:8080 studentsurveyform**" to start a container from the studentsurveyform image.

-it: Runs the container in interactive mode.

-p 8184:8080: Maps port 8183 on your host machine to port 8080 inside the container (standard web server port).

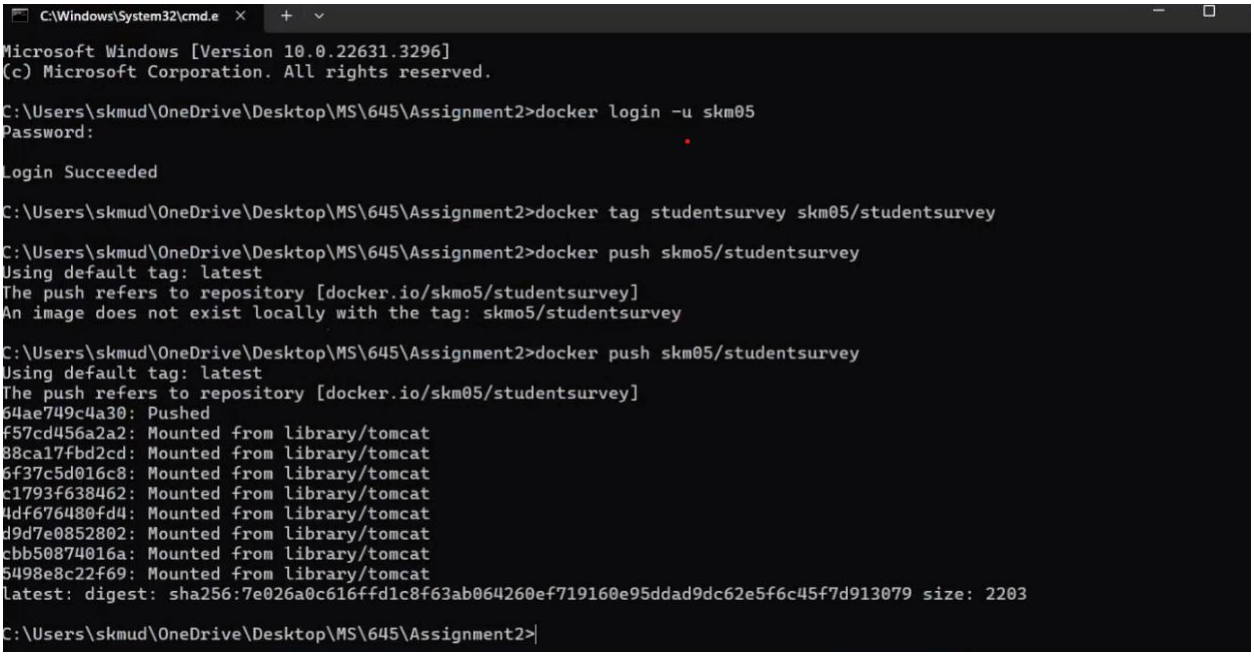
Access the application in your browser at "http://localhost:8184/StudentSurvey"

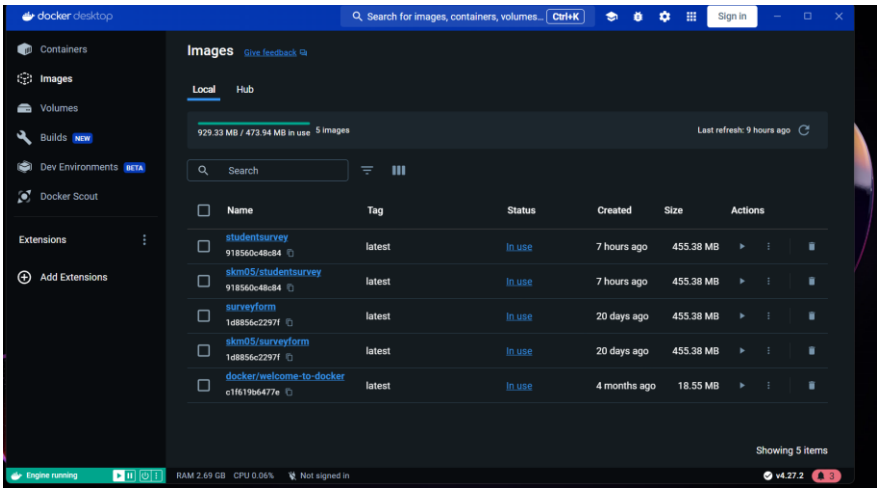
```
C:\Users\skmud\OneDrive\Desktop\MS\645\Assignment2>docker run -it -p 8184:8080 studentsurvey
Using CATALINA_BASE:   /usr/local/tomcat
Using CATALINA_HOME:   /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME:        /opt/java/openjdk
Using CLASSPATH:       /usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Using CATALINA_OPTS:
18-Mar-2024 18:06:10.566 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version name:   Apache
 Tomcat/10.1.19
18-Mar-2024 18:06:10.573 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:          Feb 14
 2024 19:36:50 UTC
18-Mar-2024 18:06:10.573 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version number: 10.1.1
9.0
18-Mar-2024 18:06:10.573 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name:               Linux
18-Mar-2024 18:06:10.573 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version:            5.15.1
```

4. Pushing the Image to Docker Hub:

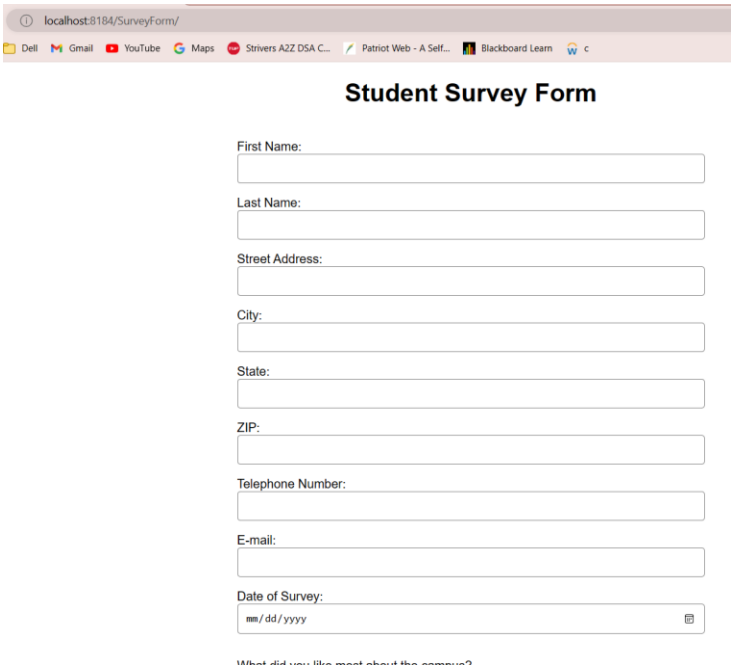
Login to Docker Hub using your credentials: docker login -u <username> (skm05 in my case) and enter the password
Tag the image for pushing: docker tag studentsurveyform <username>/studentsurveyform (replace <username> with your actual username).

Push the image to Docker Hub: docker push <username>/studentsurveyform.





This process creates a Docker image for your application, allowing you to run it consistently on any machine with Docker installed. You can then push the image to Docker Hub for sharing or deployment in a containerized environment.



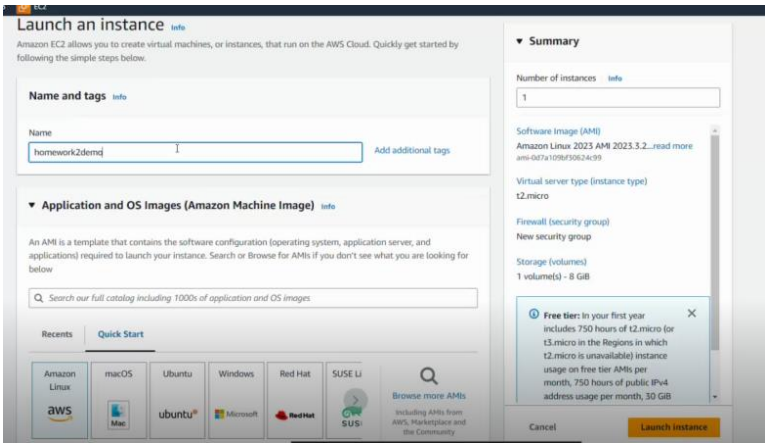
2. Prepare the deployment environment :

1. Launch and Configure the Instance:

Log in to your AWS account and navigate to the Amazon Elastic Compute Cloud (EC2) service.

Click on "Launch Instance" to create a new instance.

Name: Enter "homework2demo" for the instance name.



Machine Image (AMI): Select "Ubuntu Server 22.04 LTS (HVM) SSD volume Type".

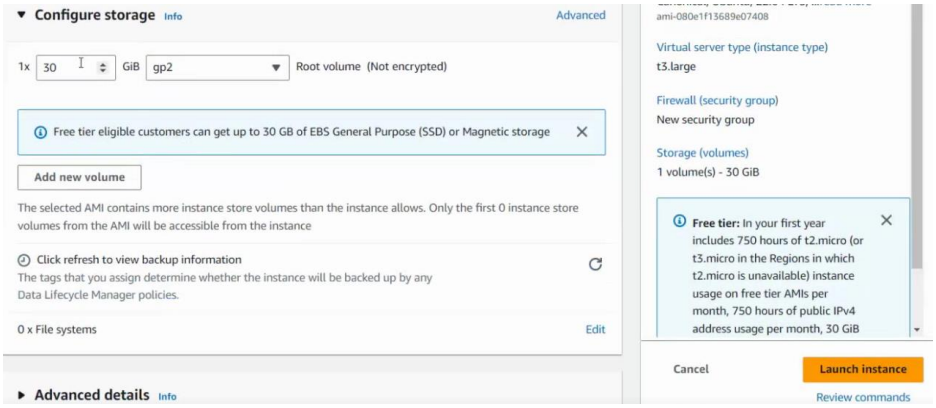
Instance Type: Choose "t3.large" for the instance type.

Key Pair: Select your existing key pair named "homework2". This allows you to connect to the instance securely.

Security Group: Check the boxes to allow both HTTP and HTTPS traffic from anywhere on the internet. This is necessary for certain functionalities.

Storage: Increase the storage size from the default 8 GB to 30 GB for better performance.

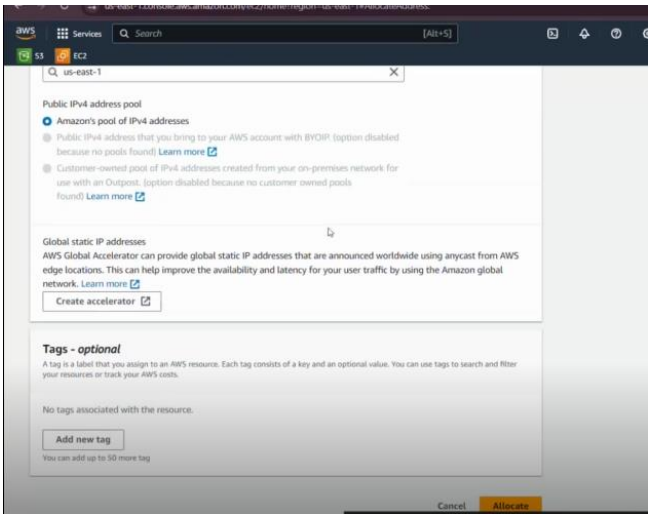
Click "Launch Instances" to create the instance.



2. Assign an Elastic IP Address:

Once the instance is running, navigate to "Elastic IPs" in the EC2 menu on the left side of the screen.

Create a new Elastic IP address using the default settings.



Associate this newly created Elastic IP address with the "homework2demo" instance you just created.

3. Configure Security Group for Docker Access:

Go to the "homework2demo" instance and navigate to the "Security Groups" tab.

Click on "Security group wizard" within the "Inbound rules" section.

Scroll down and click "Edit rule" in the "Inbound" tab.

Create a new rule with these settings:

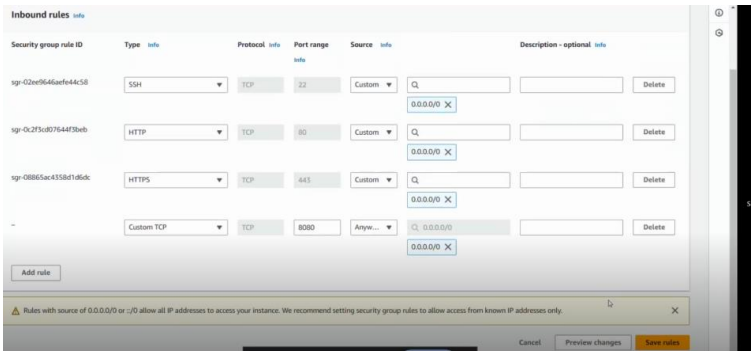
Type: "Custom TCP"

Port Range: "8080"

Source: "Custom" (allows access from anywhere)

Source Address: "0.0.0.0/0" (allows access from any IP address)

Click "Save rules" to apply the changes.



4. Connect and Install Docker:

Click on the " homework2demo" instance and then "Connect."

Go to the "EC2 Instance Connect" tab, enter username "ubuntu," and click "Connect." This opens a terminal window.

Update package lists: “sudo apt-get update”.

Install Docker: “sudo apt install docker.io”

When prompted, type "Y" to grant permission for the installation.

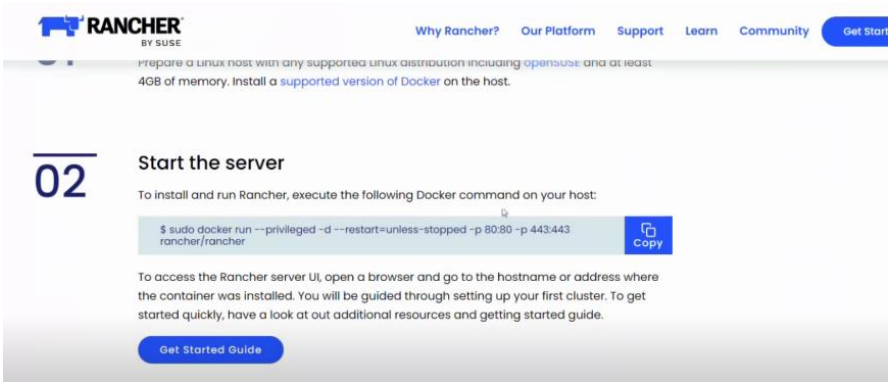
Verify successful installation.

5. Get the Rancher Server Command:

Visit the Rancher quick start guide: https://www.rancher.com/quick-start

Scroll down to "Start the Server" under "Deploy Rancher."

Copy the provided command, which will look something like: $ sudo docker run --privileged -d --restart=unless-stopped -p 80:80 -p 443:443 rancher/rancher

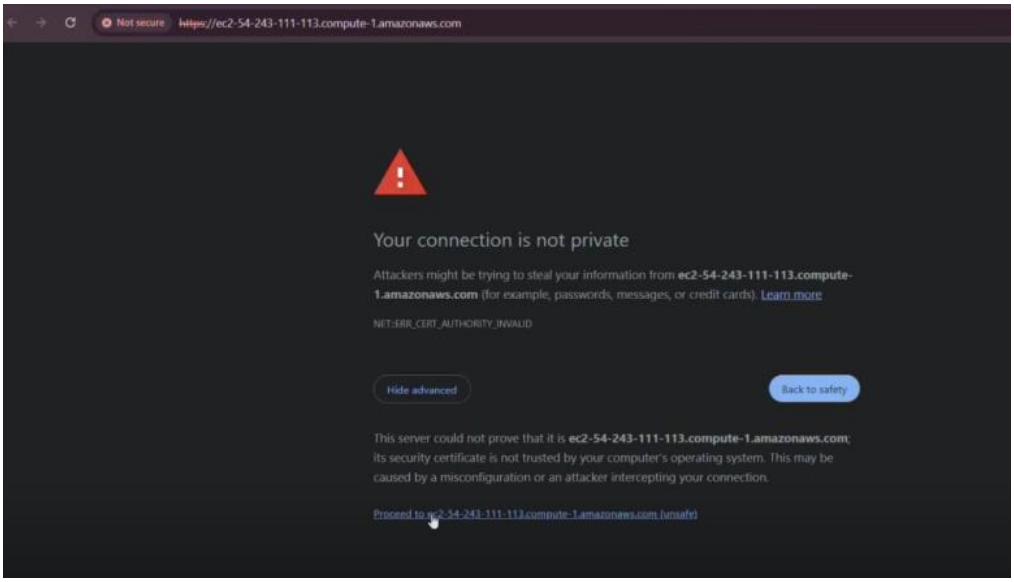


6. Install Rancher on the Node:

Connect to your " homework2demo " instance using EC2 Instance Connect (same method as before).

Paste the copied Rancher server command and run it.

Once successful, Rancher is installed and accessible through your master node's public IP address.



7. Get the Container ID and Password:

Run sudo docker ps on the master node to list running containers.

Note the container ID for the Rancher container.

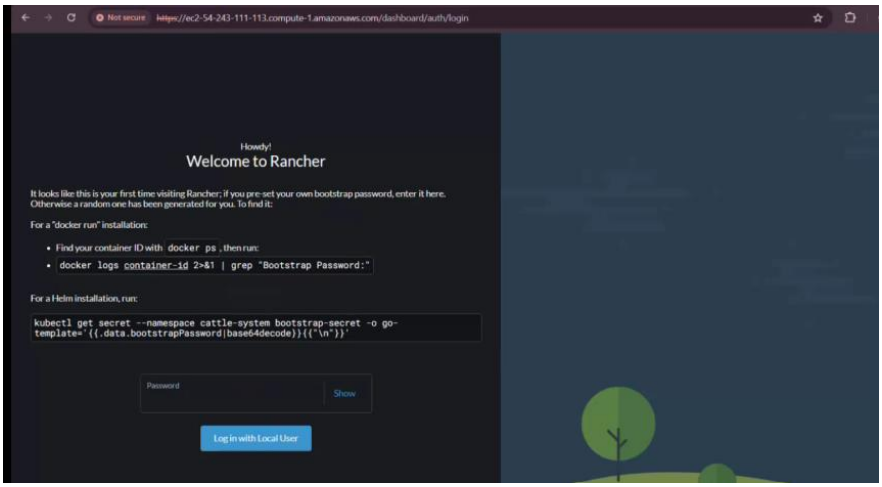
In your browser, open the master node's public IP address (even if the security warning appears). The Rancher login page should load.

On the login page, you'll need a password.

Go back to the instance console and run a modified version of the docker logs command, replacing the container ID with the one you noted:

sudo docker logs <container_id> 2>&1 | grep "Bootstrap Password:"

This will display the Rancher password in the console.



8. Login to Rancher:

Copy the password from the console and paste it into the Rancher login screen.

Login and set up a new password following the on-screen prompts.

9. Rancher Dashboard:

You'll now see the Rancher dashboard, where you can create a Kubernetes cluster and deploy your application.

3. Deploy the application

Create a Custom Cluster:

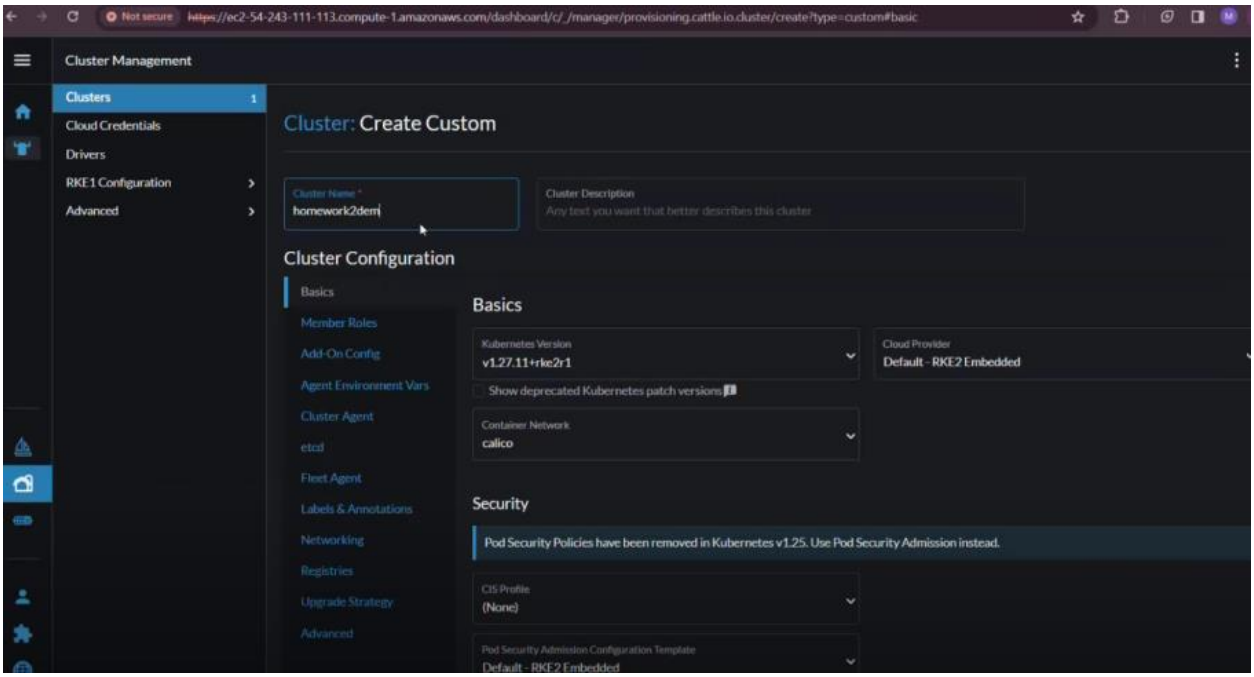
In the Rancher dashboard, click "Create Cluster."

Choose "Custom" from the options and fill out the form:

Cluster Name: "homework2demo"

Leave all other settings as default.

Click "Create" to initiate cluster creation.



2. Register the Node :

Navigate to the "Registration" tab on the cluster creation page.

Under "Step 1," ensure all checkboxes for "etcd," "Control Plane," and "Worker" are selected.

Copy the registration command displayed under "Step 2."

Connect to your " homework2demo " node via EC2 Instance Connect

Run the copied command on the worker node, appending --insecure after curl.

3.Verify Cluster Status:

The cluster status should change from "Updating" to "Active" after a few minutes.

You can monitor the progress in the "Clusters" section under "Cluster Management" on the Rancher UI.

4. Deploy the Application:

Once the cluster is active, click "Explore" next to the cluster name.

In the left pane, navigate to "Workloads" and select "Deployments."

Click "Create" to start defining your deployment.

Fill in the deployment details:

Namespace: "default"

Name: "assignment2-deployment"

Replicas: "3" (number of pods running the application)

Container Image: "skm05/studentsurveyform:latest" (your Docker Hub image name)

5. Configure Service for NodePort Access:

Under the "Networking" section, click "Add port or service."

Configure the service with:

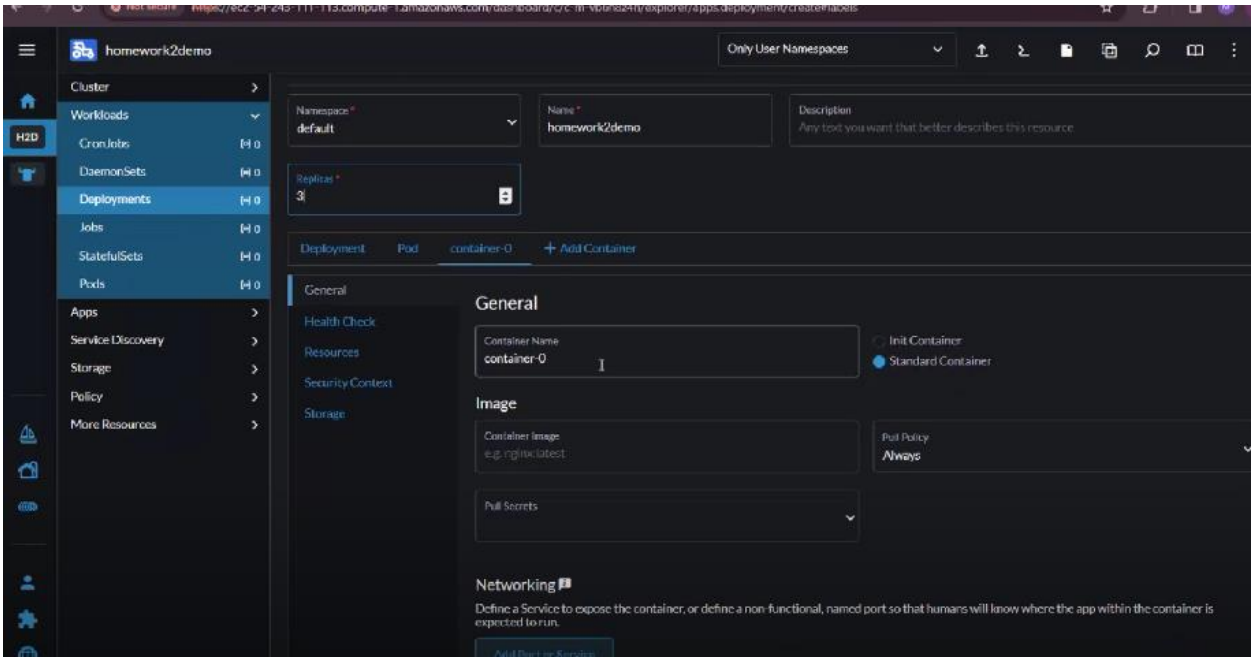
Service Type: "NodePort" (allows access from outside the cluster)

Name: "nodeport"

Private Container Port: "8080" (port your application listens on)

Protocol: "TCP" (communication protocol)

Leave other settings as default and click "Create."



6. Access the Application:

After the deployment finishes (may take a few minutes), navigate to the "Services" tab in the deployment details.

Find the "nodeport" service and click on it.

In the target URL, append "/SurveyForm" (your application endpoint) and open it in a new browser tab.

You should now see your application running at the provided URL (similar to this format):

7. Download Kubeconfig File:

Go back to the cluster page in Rancher.

From the top right menu, click "Download KubeConfig file." Save this file locally for future interaction with your Kubernetes cluster.

4. Set up continuous integration and delivery (CI/CD):

Begin by creating a new AWS EC2 instance in the AWS lab environment.

Follow the same process as before for creating EC2 instances to deploy the application on the Kubernetes cluster.

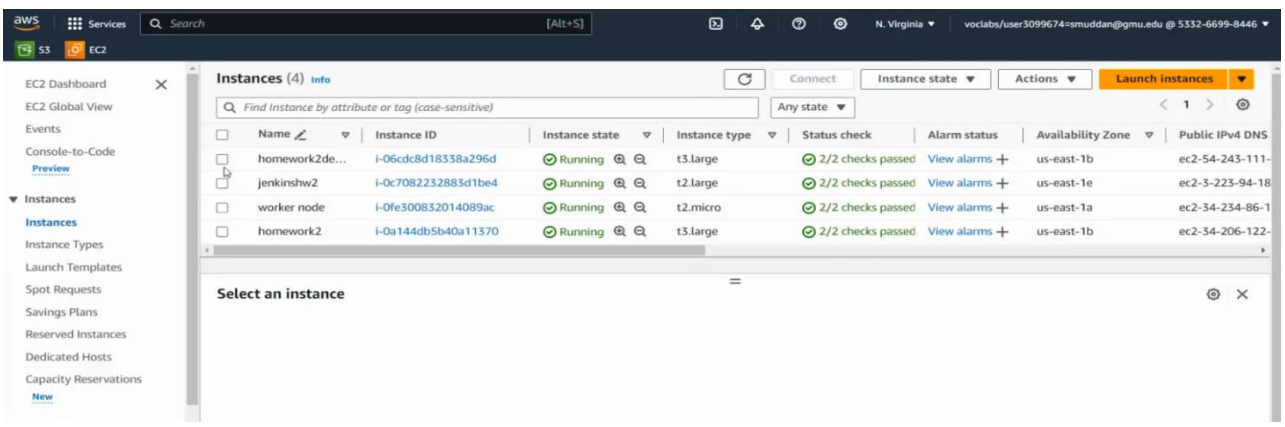
Name the EC2 instance "Jenkins".

Utilize the same configuration parameters as used previously, including choosing the appropriate AMI, instance type, and key pair.

Assign an elastic IP to the Jenkins instance to maintain a static IP address, ensuring consistency even during AWS lab restarts.

Adjust the security group settings for the Jenkins instance as needed, ensuring necessary inbound and outbound rules are configured.

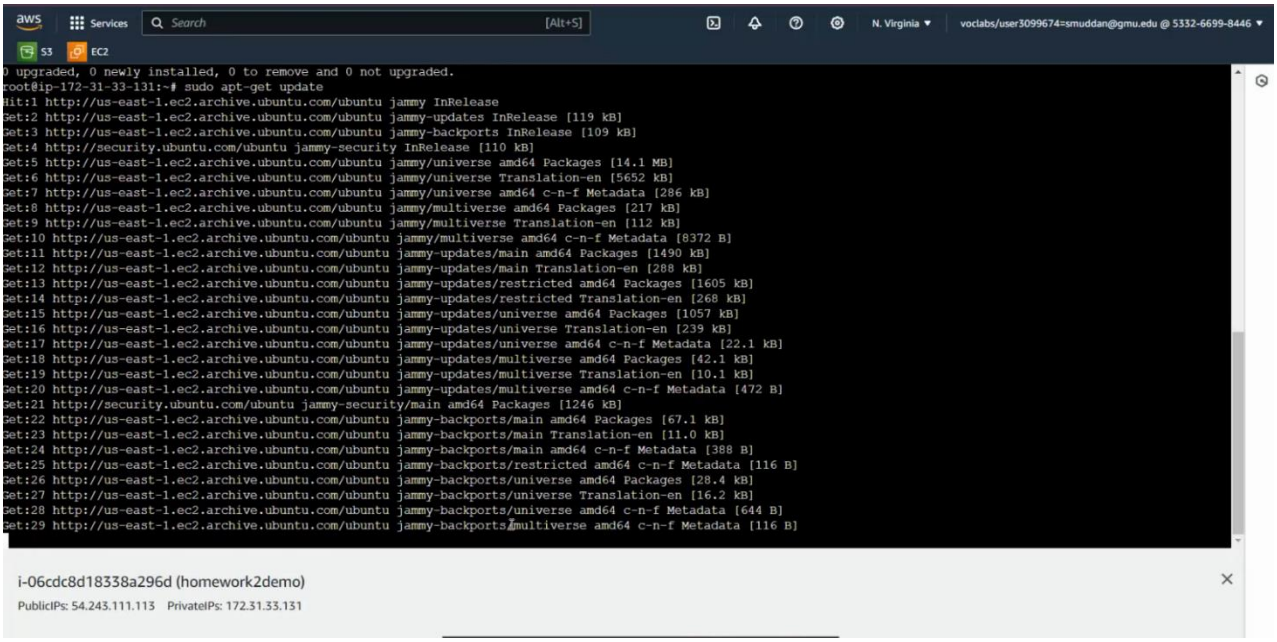
Once the instance is created and running, proceed to set up Jenkins on this instance to establish a CI/CD pipeline for automated build and deployment processes.



To connect to the EC2 instance using "EC2 instance connect," ensure that the instance is in a "running" state. Then follow these steps:

Update the package index on the instance by running: sudo apt-get update

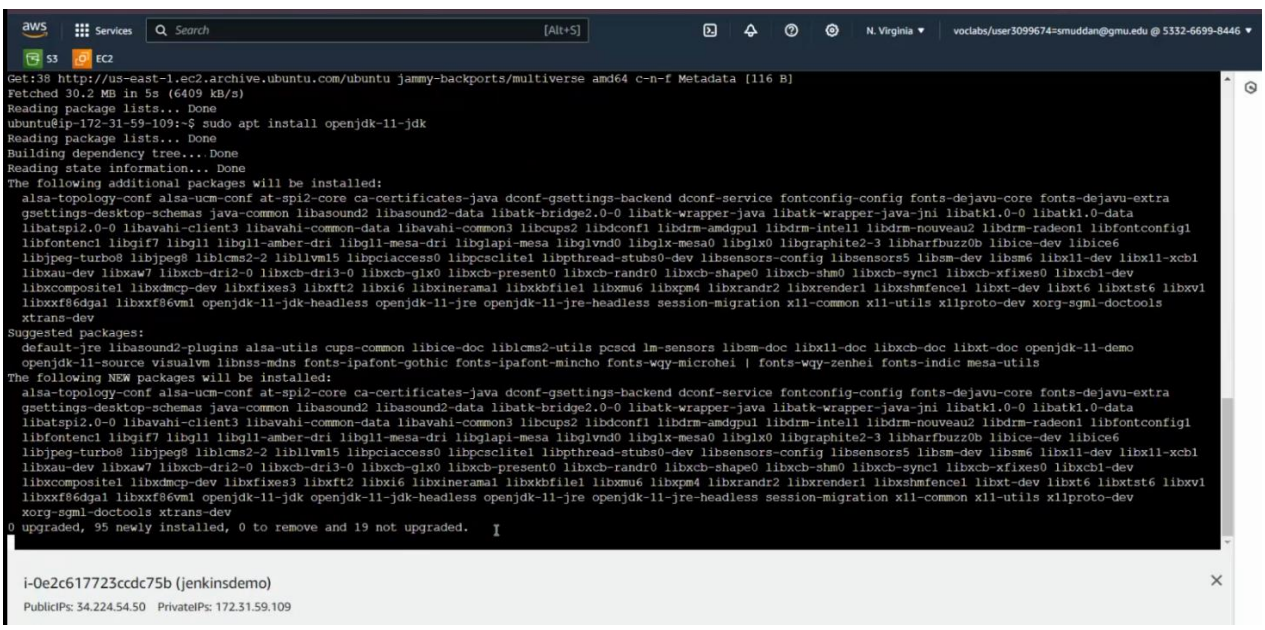
Update the package index again using: sudo apt update



Install OpenJDK 11 by running the following command: sudo apt install openjdk-11-jdk

When prompted for permission, type "Y" and press Enter to proceed with the installation.

Following these steps will install Java JDK 11 on the EC2 instance, preparing it for Jenkins installation.



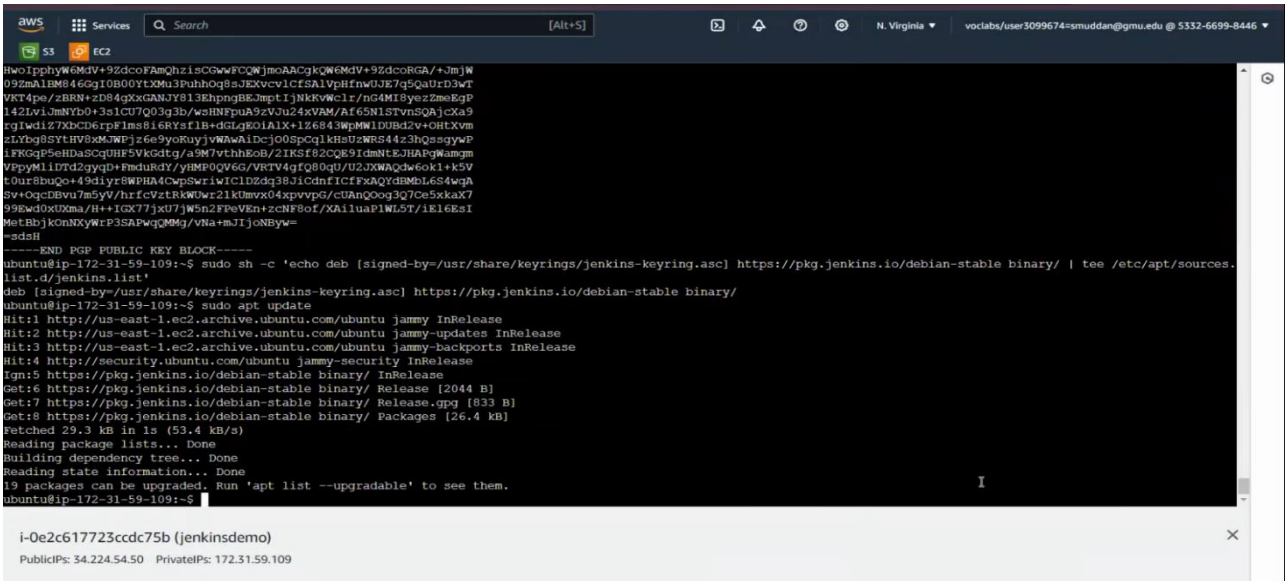
To install Jenkins on the EC2 instance, follow these steps:

Obtain the Jenkins package key by running the following command:

sudo curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

Add the Jenkins package repository by running:

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



These commands will download the Jenkins package key and add the Jenkins package repository to the list of package sources on the EC2 instance. This prepares the instance for installing Jenkins.

To start Jenkins and expose port 8080 on the EC2 instance, follow these steps:

Start Jenkins service by running the command: sudo systemctl start jenkins.service

Allow incoming traffic on port 8080 using the Uncomplicated Firewall (UFW) by running:

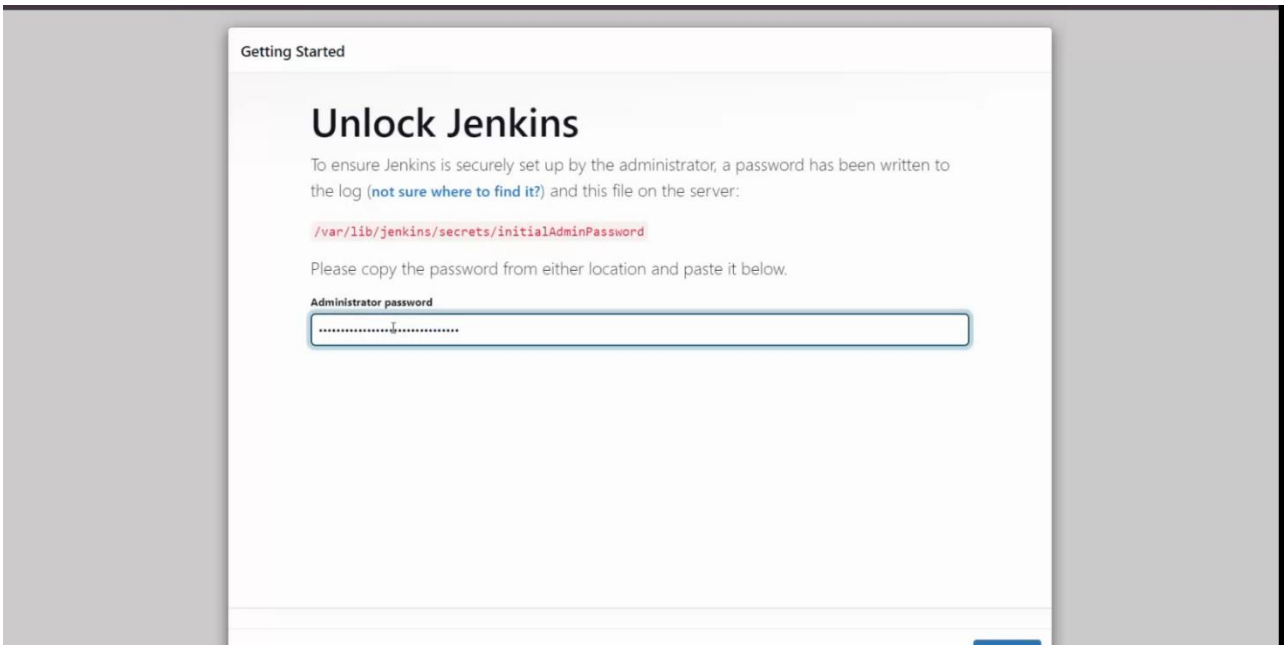
sudo ufw allow 8080

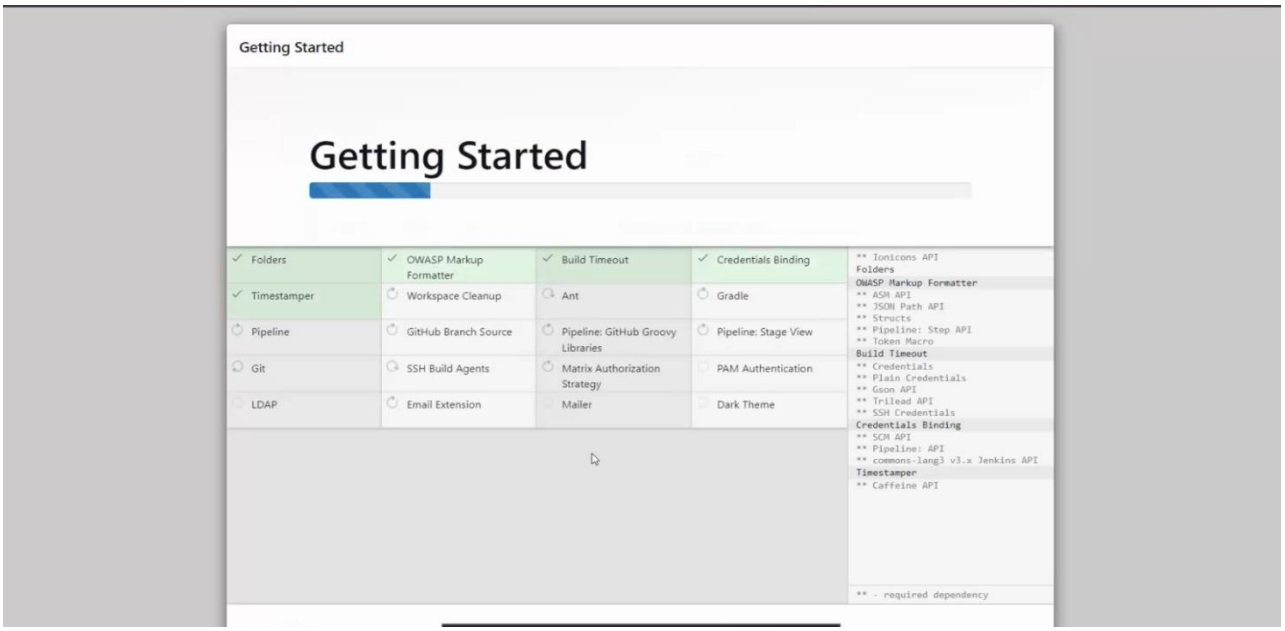
Retrieve the initial admin password generated by Jenkins using the following command:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

This command will display the generated password. Make sure to store it securely as it is required during the Jenkins setup process.

These steps will start Jenkins, open port 8080 for incoming traffic, and provide you with the initial admin password needed to set up Jenkins.





To install snapd and then kubectl, follow these steps:

Install snapd by running the command: sudo apt install snapd

Once snapd is installed, use snap to install kubectl by running: sudo snap install kubectl –classic

To access Jenkins on the AWS EC2 instance, follow these steps:

Open a web browser and enter the public IPv4 address of the Jenkins instance in the address bar. For example: http://34.224.54.50

Append the port 8080 to the URL, so it looks like this: http://34.224.54.50:8080

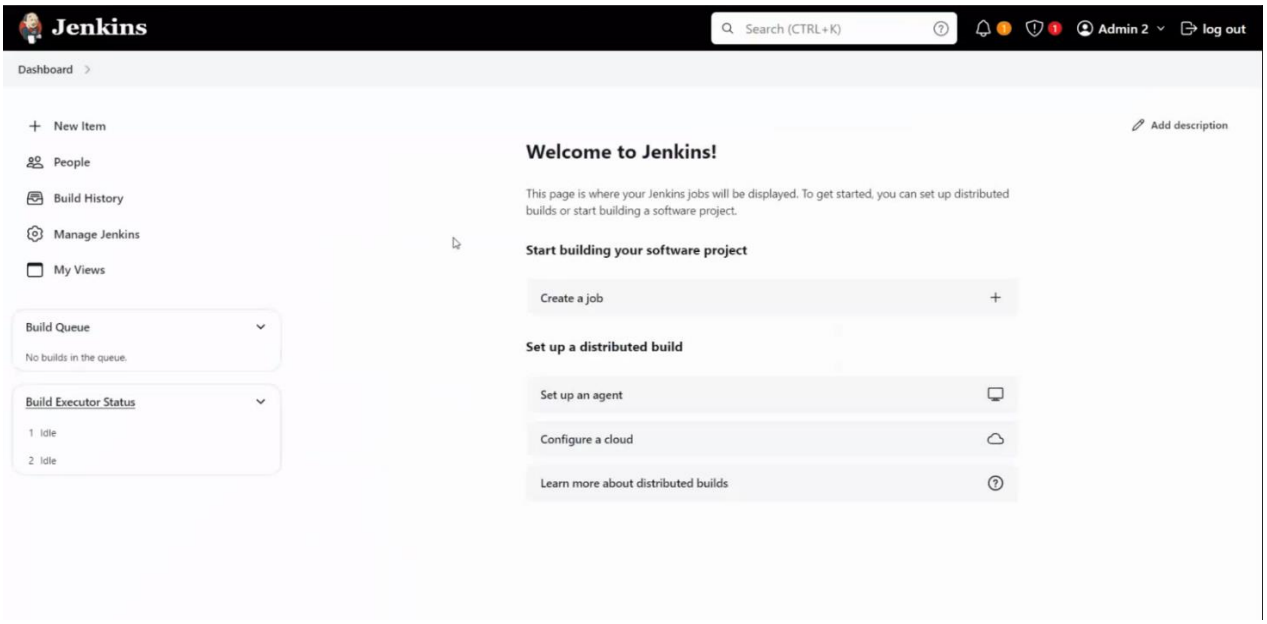
Press Enter to navigate to the Jenkins login page.

If prompted, enter the initial admin password obtained earlier when prompted to unlock Jenkins.

Following these steps will allow you to access Jenkins via its public IPv4 address on port 8080 and unlock it using the admin password.

Now 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”.



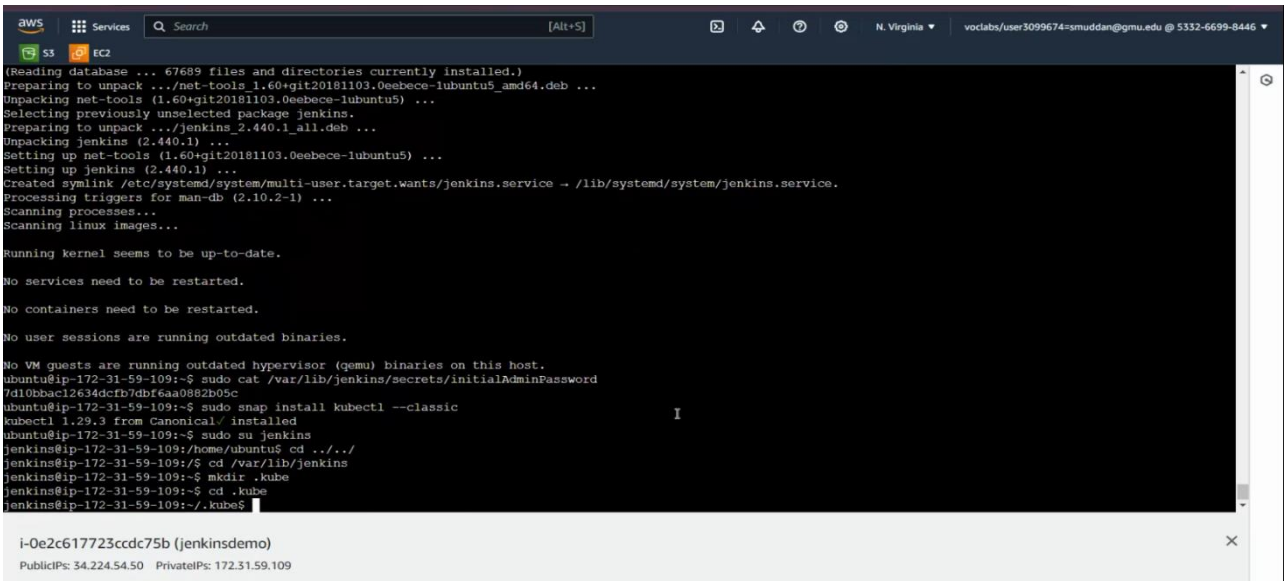
To create the Kubernetes config file for Jenkins, follow these steps:

Go back to the EC2 console.

Switch to the Jenkins user by running: sudo su Jenkins

Navigate to the root directory: cd ../../

Go to the Jenkins home directory: cd /var/lib/Jenkins



Create a directory named ".kube" and enter it:

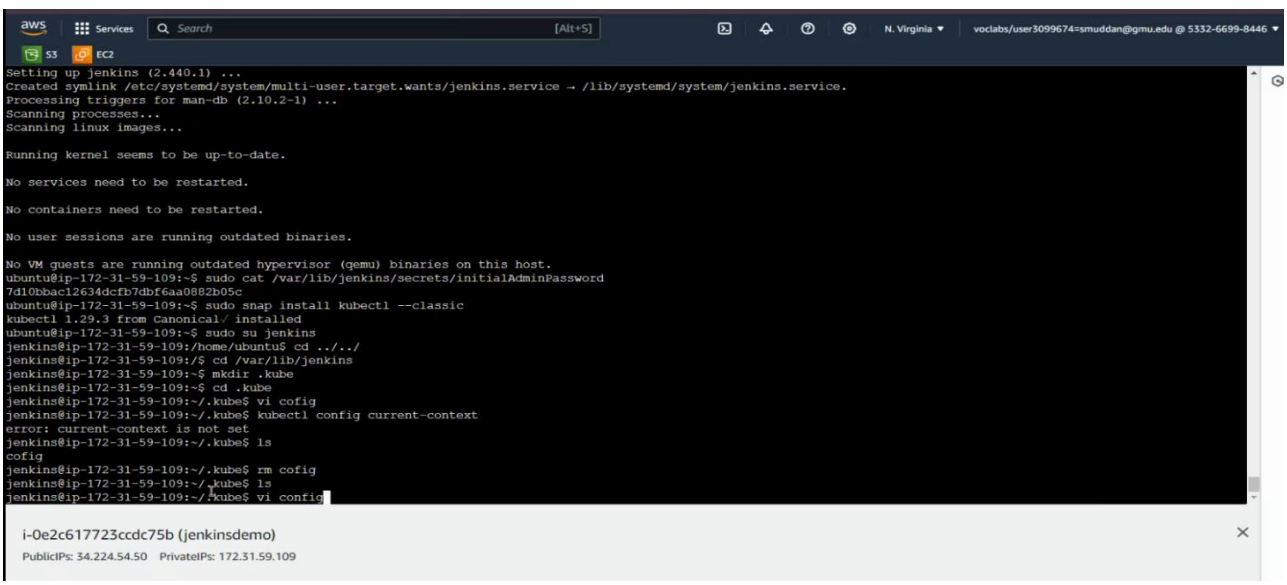
mkdir .kube

cd .kube

Create and open the config file using the vi editor: vi config

Copy the contents from the "KubeConfig" file downloaded earlier and paste it into the vi editor.

Save the file and exit vi by pressing Esc, then typing ":wq" and pressing Enter.



These steps will create the Kubernetes config file named "config" in the ".kube" directory within the Jenkins user's home directory, allowing Jenkins to interact with the Kubernetes cluster.

To verify the Kubernetes configuration and install Docker, follow these steps:

Run the following command to check the current Kubernetes context:

kubectl config current-context

This command should display the cluster name as expected.

Exit the Jenkins mode by running: exit

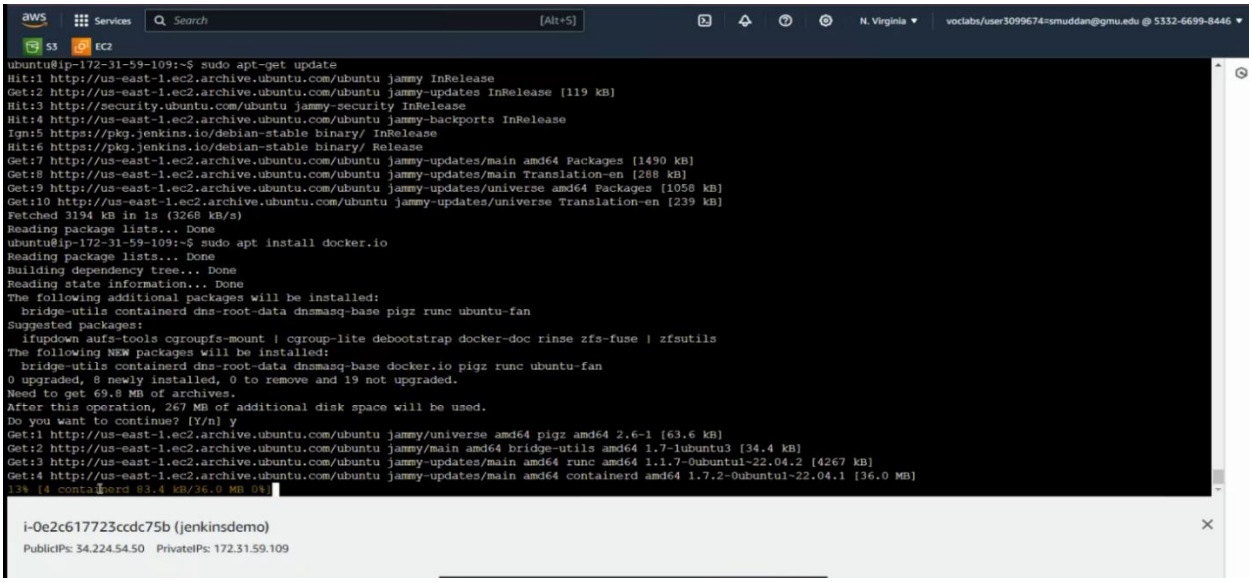
Update the package index and upgrade the packages by running:

sudo apt-get update

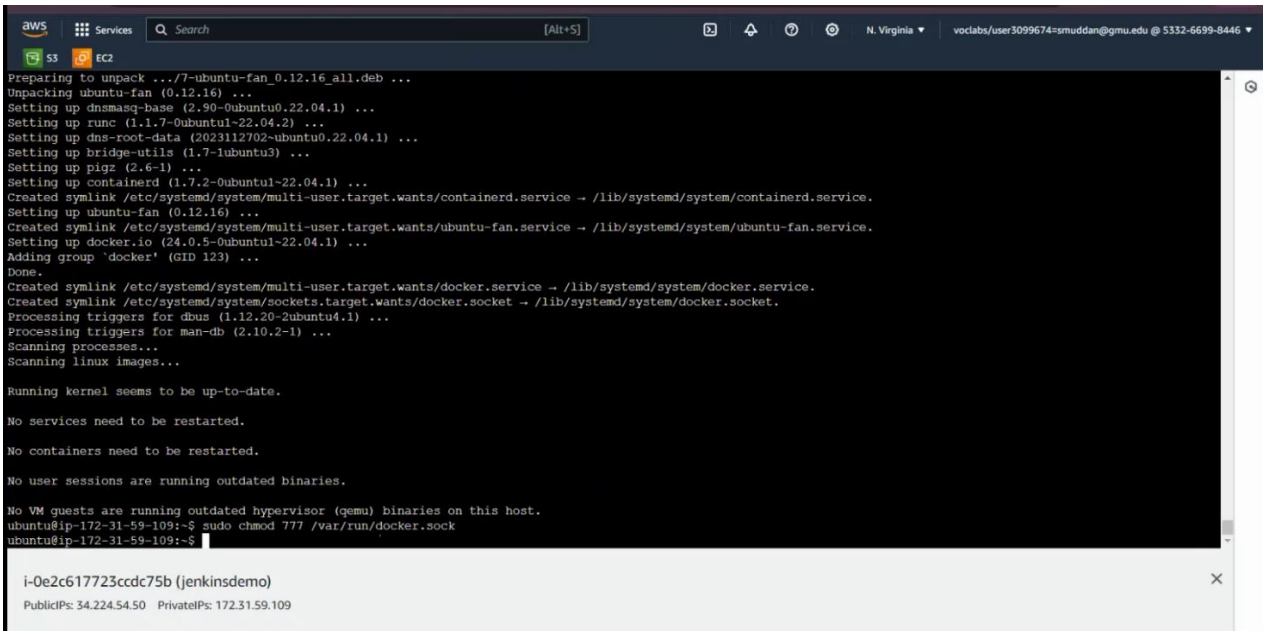
sudo apt update

Install Docker by running: sudo apt install docker.io

When prompted, type "Y" and press Enter to confirm the installation.



After installing Docker, change the file permissions for the Docker socket by running: sudo chmod 777 /var/run/docker.sock



These steps will verify the Kubernetes configuration, exit the Jenkins mode, and then install Docker while ensuring proper permissions for the Docker socket.

Setting up Github Repo:

To set up the GitHub repository for the pipeline, follow these steps:

Go to https://github.com/ and create an account or log in if you already have one.

Once logged in, click on the "+" icon in the top-right corner and select "New repository".

Name the repository "Swe645Homework2".

Choose whether the repository will be public or private (this depends on your preference and needs).

Click on the "Create repository" button.

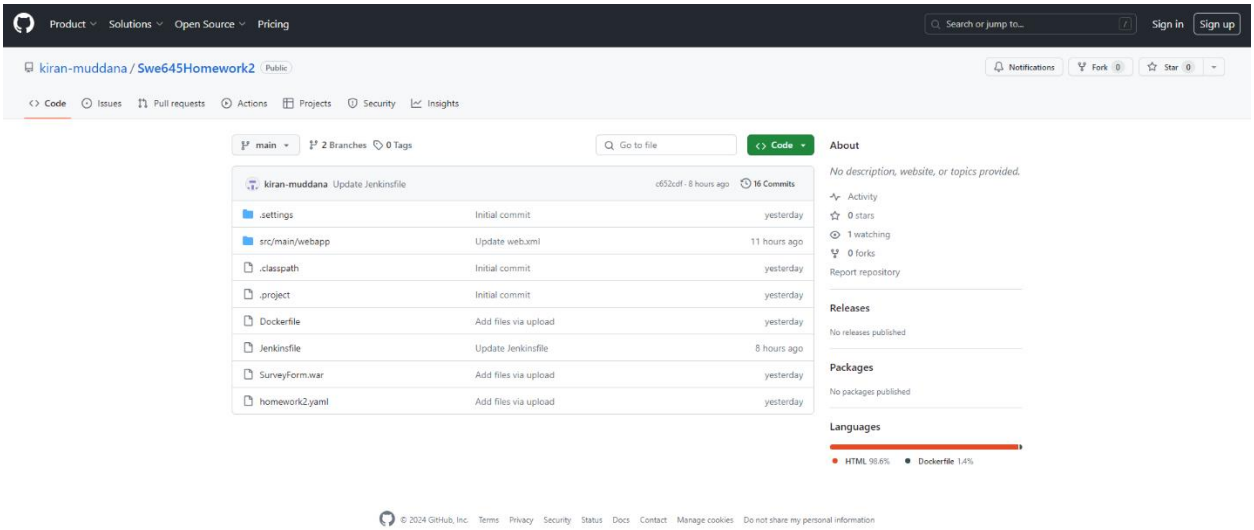
Once the repository is created, navigate to it and upload your source files, Dockerfile, and the WAR file by clicking on the "Add file" button and selecting "Upload files".

Drag and drop the files or use the file selector to add them to the repository.

Once the files are added, scroll down and click on the "Commit changes" button to commit them to the repository.

Now, you can use this GitHub repository URL for your pipeline:

URL: https://github.com/kiran-muddana/Swe645Homework2



To create credentials for DockerHub in Jenkins, follow these steps:

Go to the Jenkins dashboard and click on "Manage Jenkins".

Scroll down and select the "Manage Credentials" option.

Click on "Global credentials (unrestricted)".

Click on "Add credentials".

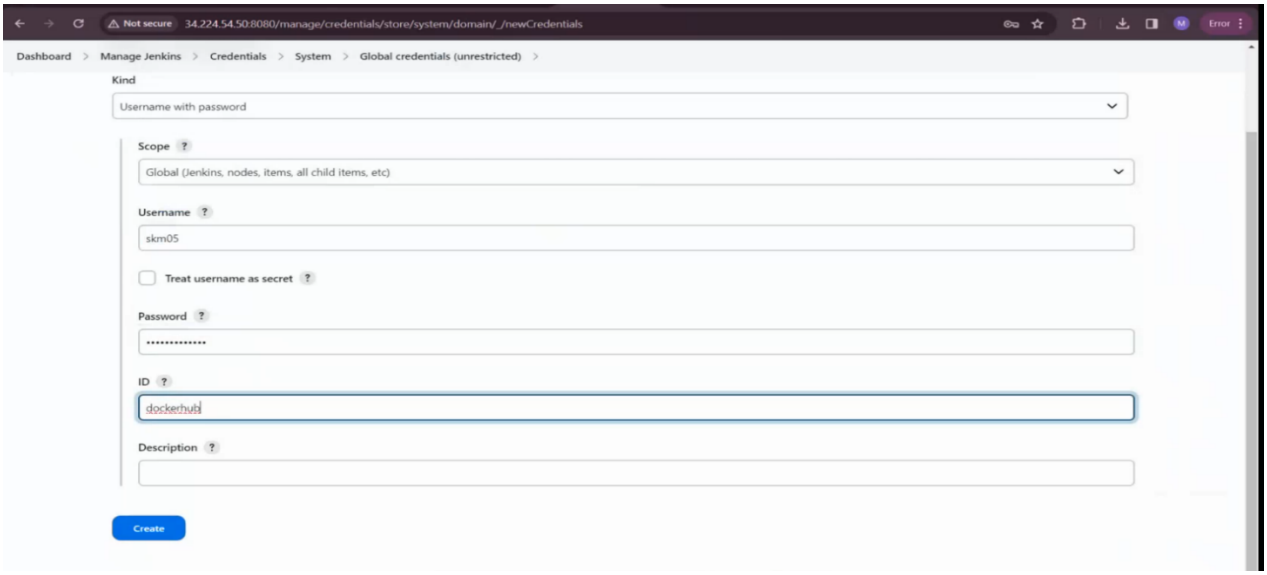
In the Kind dropdown menu, select "Username with password".

Enter your DockerHub username and password in the respective fields.

Set the ID field as "dockerhub".

Click on the "OK" or "Create" button to save the credentials.

These steps will add DockerHub credentials to Jenkins, allowing it to access DockerHub repositories during pipeline execution.



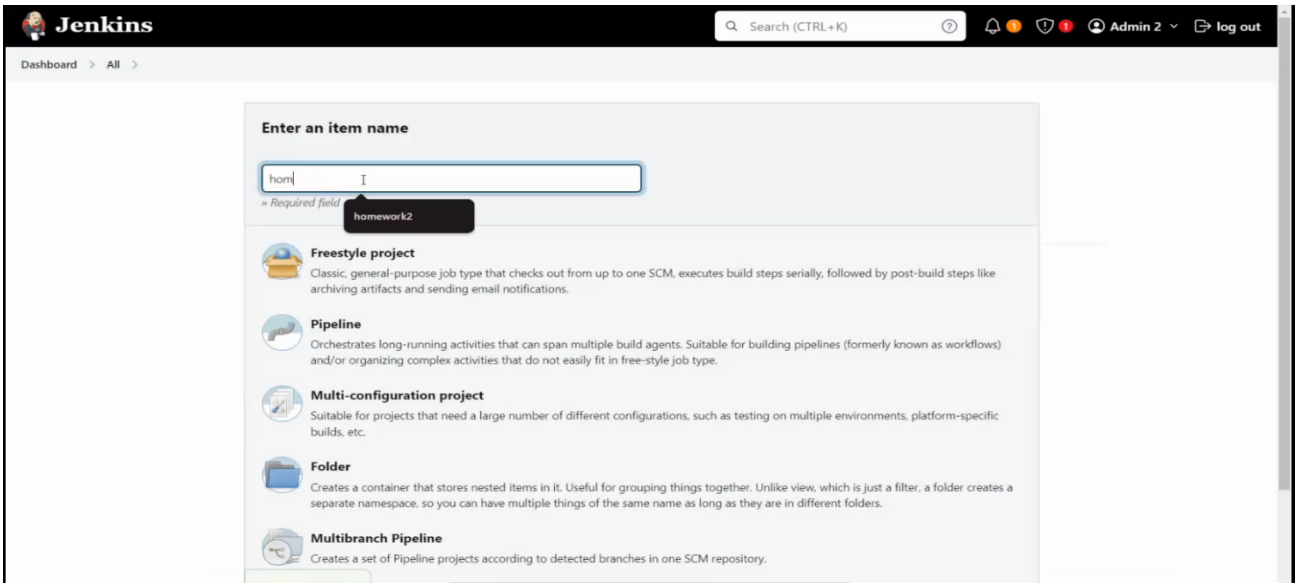
Repeat the above step to save GitHub credentials as well, give id as “github”.

Now go to dashboard and click on “New Item”.

Give name as “homework2demo” and click on “Pipeline”.

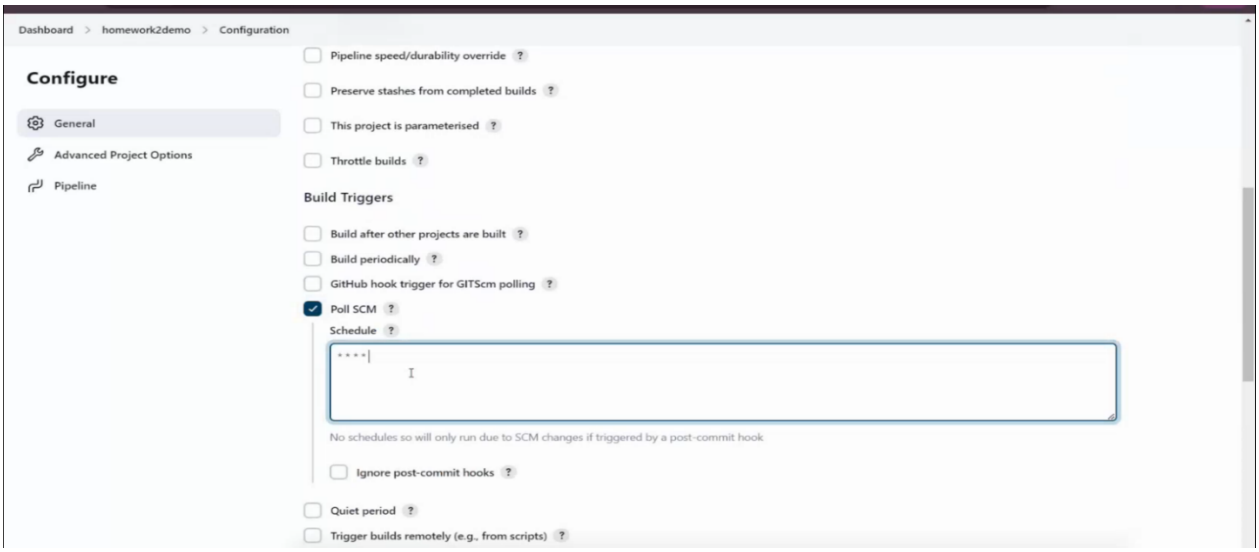
Now on the next page, check the “Github Project” checkbox, and in the project URL paste the URL from

Github repository: https://github.com/kiran-muddana/Swe645Homework2



Scroll down and check the “Poll SCM” checkbox.

Give the Schedule as : “* * * * *”.



To configure the Jenkins pipeline to fetch the Jenkinsfile from the GitHub repository, follow these steps:

Scroll down to the "Pipeline" section in Jenkins.

Select the "Pipeline script from SCM" option.

In the SCM dropdown menu, select "Git".

Enter the URL of your GitHub repository in the "Repository URL" field:

https://github.com/kiran-muddana/Swe645Homework2.git

Choose the appropriate credentials from the "Credentials" dropdown menu.

In the "Branches to build" or "Branch specifier" field, specify the branch where your Jenkinsfile is located. If it's the main branch, enter: */main

Save the configuration.

Go to github repository and then create a new file : “Jenkinsfile” and then commit changes.

Click on Save.

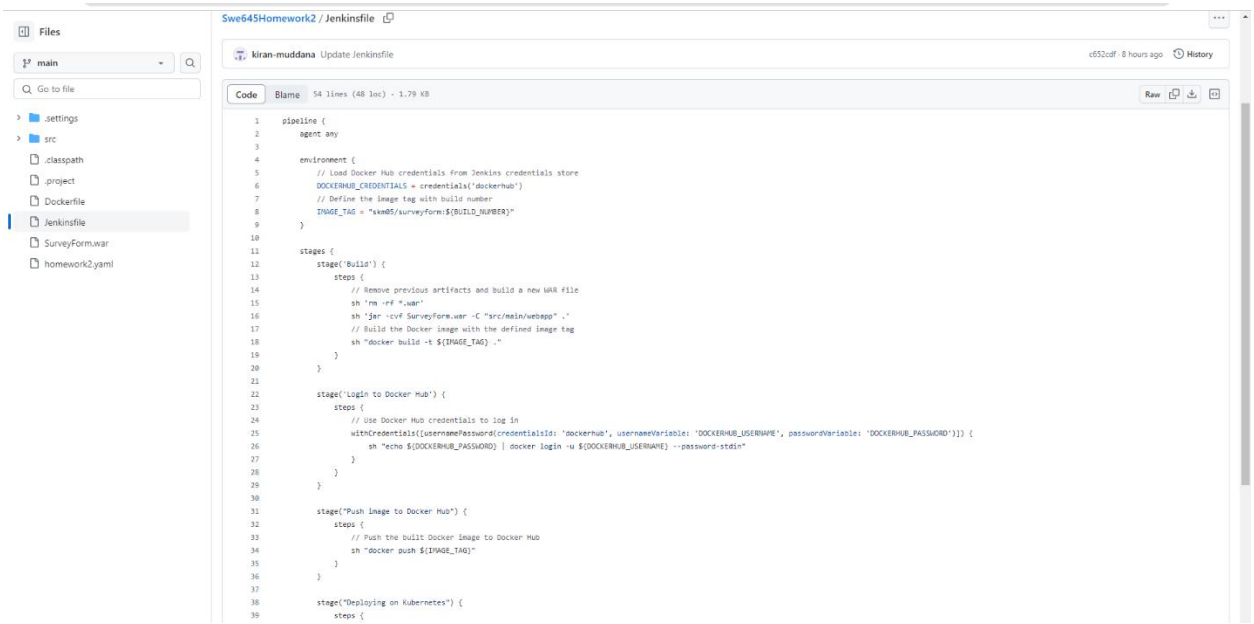
Now a build will automatically start on this pipeline.

Now go to the Github repository -> “Jenkinsfile” and click on edit.

Enter the commands given in the screenshot below and 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 commands on how to generate a new war file, create a docker image, login to docker and push image, deploy image on Kubernetes cluster.



To address any errors that may occur during the build process by adjusting the Jenkinsfile, follow these steps:

Navigate to your GitHub repository.

Locate the Jenkinsfile and click on "Edit".

Review the Jenkinsfile for any paths, commands, or configurations that may be causing errors.

Make necessary corrections or adjustments to resolve the errors. Here are some common adjustments you might need to make:

Check the paths to ensure they are correctly pointing to the necessary files or directories.

Verify that commands are written correctly and include any required arguments or options.

Double-check configurations to ensure they match the setup of your environment and project.

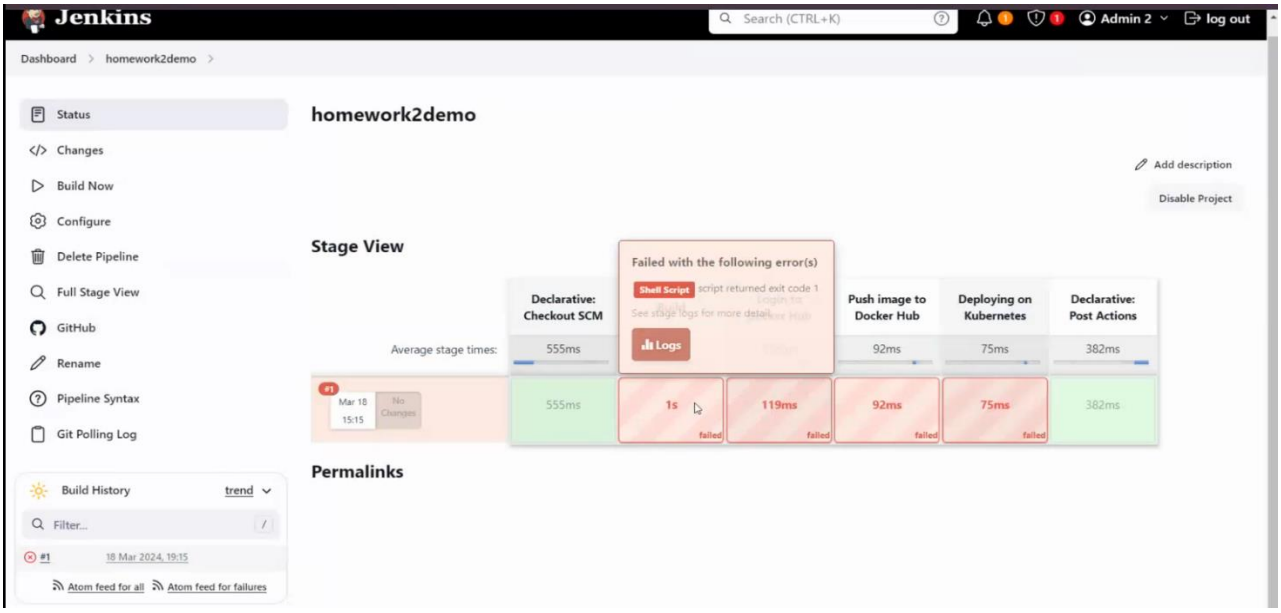
Save the changes and commit them to the repository.

Trigger a new build in Jenkins to apply the changes made to the Jenkinsfile.

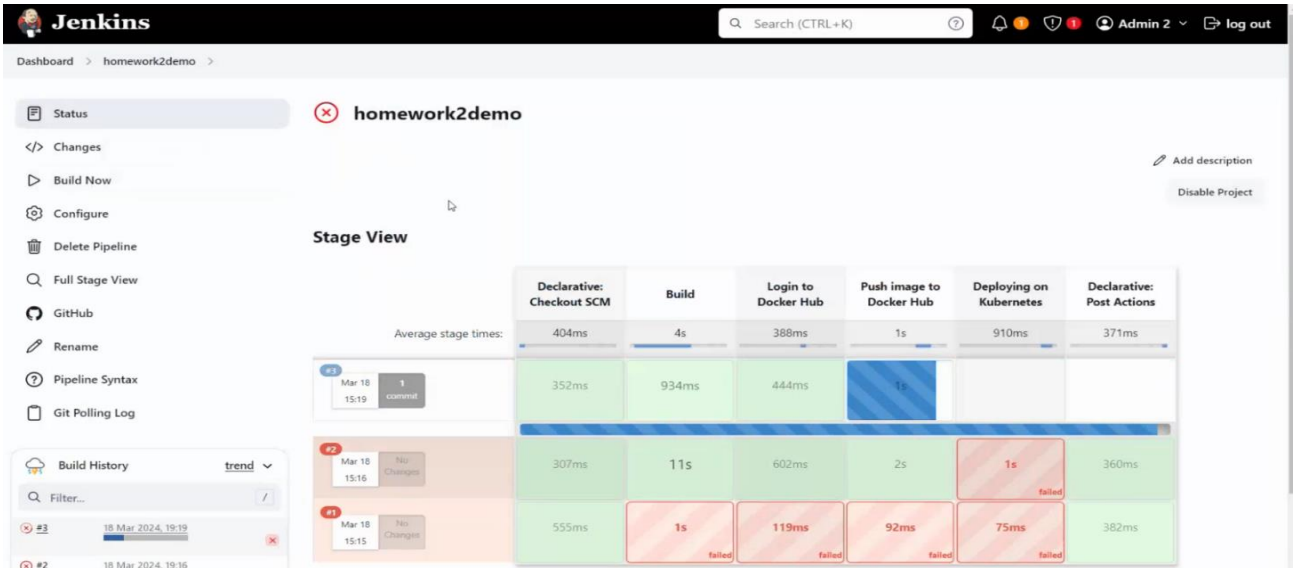
Monitor the build process for any errors or issues and adjust the Jenkinsfile further if necessary.

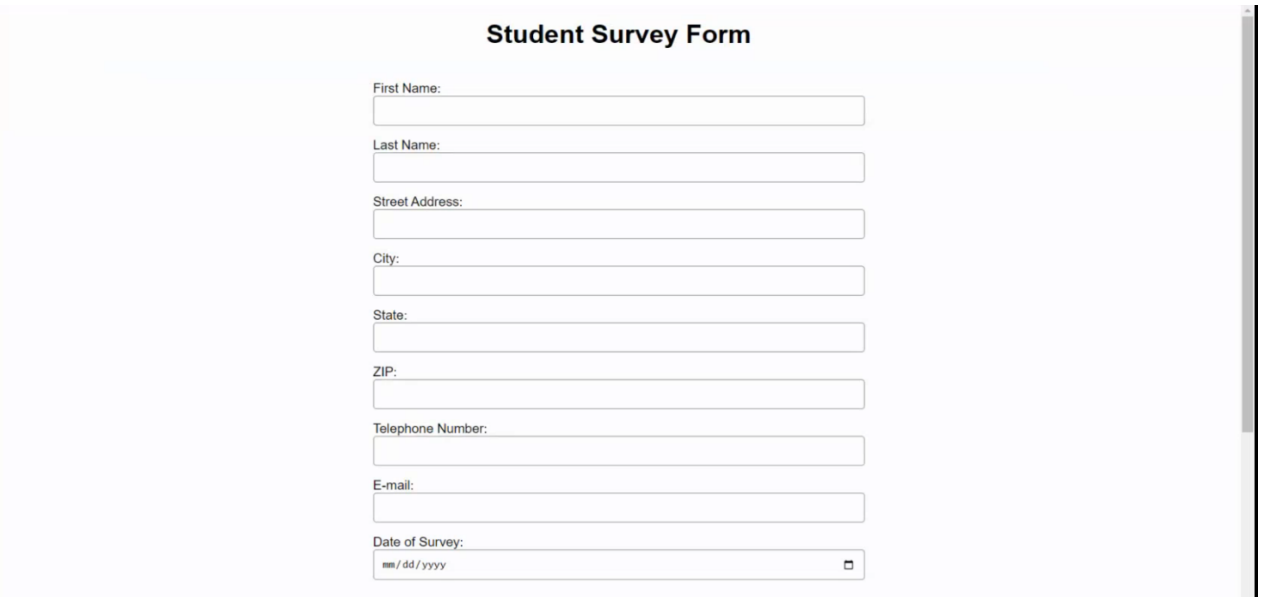
Repeat this process as needed until the build process completes successfully without errors.

By iteratively adjusting the Jenkinsfile based on error feedback and testing, you can gradually refine the build process and ensure its smooth execution.



After the build is successful, if we reload the webpage, we will be able to see the changes being reflected.





Links:

Github URL: https://github.com/kiran-muddana/Swe645Homework2

Docker URL: https://hub.docker.com/r/skm05/studentsurvey

Application URL: http://ec2-54-243-111-113.compute-1.amazonaws.com:30463/SurveyForm/

References:

1. https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-20-04

2. Lecture 6 class notes.

3. Lecture 7 class notes.

4. https://docs.docker.com/get-started/02_our_app/

5. https://docs.rke2.io/

Contributions:

Kiran: Developed the source file, worked on creating docker image and the Github repository, worked on solving issues and errors in setting up jenkins and CI/CD pipeline. (making of video)

Venu: Setting up CI/CID pipeline and documentation. (edits/corrections made for the video)

Yashasree: Solving errors and issues, helped in setting up Jenkins and CI/CD pipeline