**Phase End Project**

**eCommerce Platform Testing**

**Objective:** To deploy applications using Docker containers and later automate the deployment using Jenkinsfile on EC2 instance

**Tools Required:** Visual Studio Code, Eclipse IDE, MySQL, Angular CLI, Docker, Jenkins, AWS

**Prerequisites:** None

**Steps to be followed:**

1. Downloading and extracting the code files
2. Creating a network and running a MySQL container
3. Creating Dockerfiles and running containers for applications
4. Verifying the applications function
5. Configuring the Git repository for both the applications
6. Setting up the Jenkins
7. Creating a Jenkinsfile and CI/CD pipeline in local machine
8. Setting up the EC2 instance
9. Deploying the applications on EC2 using CI/CD Jenkins pipeline

**Step 1: Downloading and extracting the code files**

1. Download the **eStore** code files that encompass the code for both the frontend Angular application and the Java backend application

A computer screen shot of a computer

Description automatically generated with low confidence

1. Open the terminal inside the **Downloads** directory and unzip the downloaded zip folder

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer

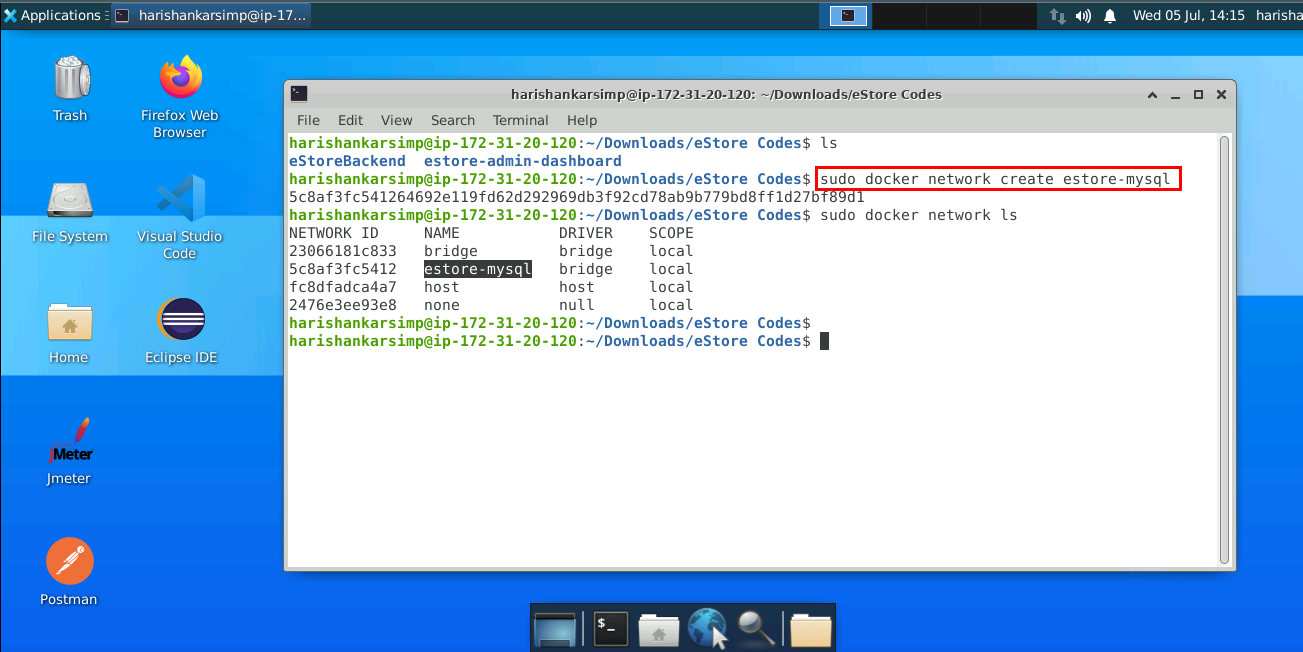
Description automatically generated with low confidence

You can see both frontend and backend application folders. To create a Docker container for both application. You need to setup MySQL and Docker initially.

**Step 2: Creating a network and running a MySQL container**

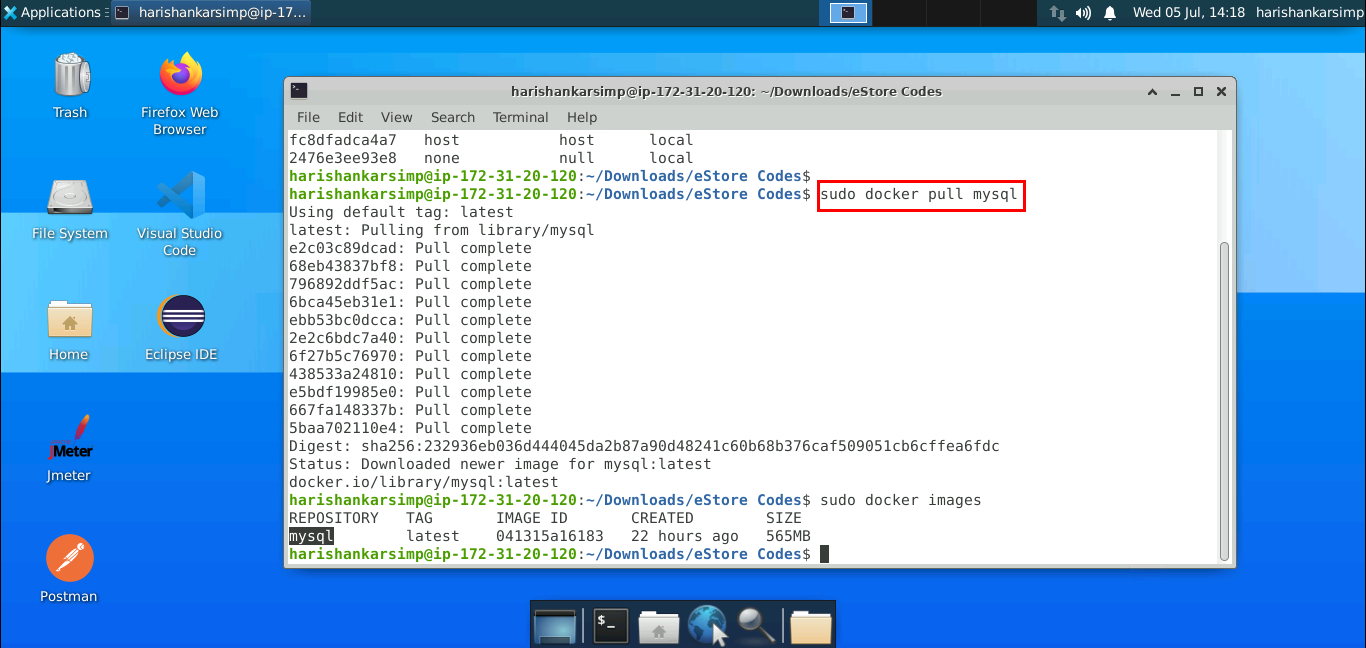
1. Create a new network in Docker so that we can make a connection between MySQL and Java backend and Angular frontend in Docker. Execute the command:

**sudo docker network create estore-mysql**



1. Now, pull latest image of MySQL database in Docker:

**sudo docker pull mysql**



1. Now, run container from MySQL on the created network:

**sudo docker run --name mydb --network estore-mysql -e MYSQL\_ROOT\_PASSWORD=password -e MYSQL\_DATABASE=estoredb -d mysql**

A screenshot of a computer

Description automatically generated

1. Check if the database has been created in MySQL container. Execute the command to access it:

**sudo docker exec -it mydb bash -l**

**A computer screen shot of a computer screen

Description automatically generated**

We are inside the MySQL container and able to access the shell of container. Now we can create tables and add admin record which is later required to login for frontend.

1. Run the below command to access the MySQL server:

**mysql -u root -p**

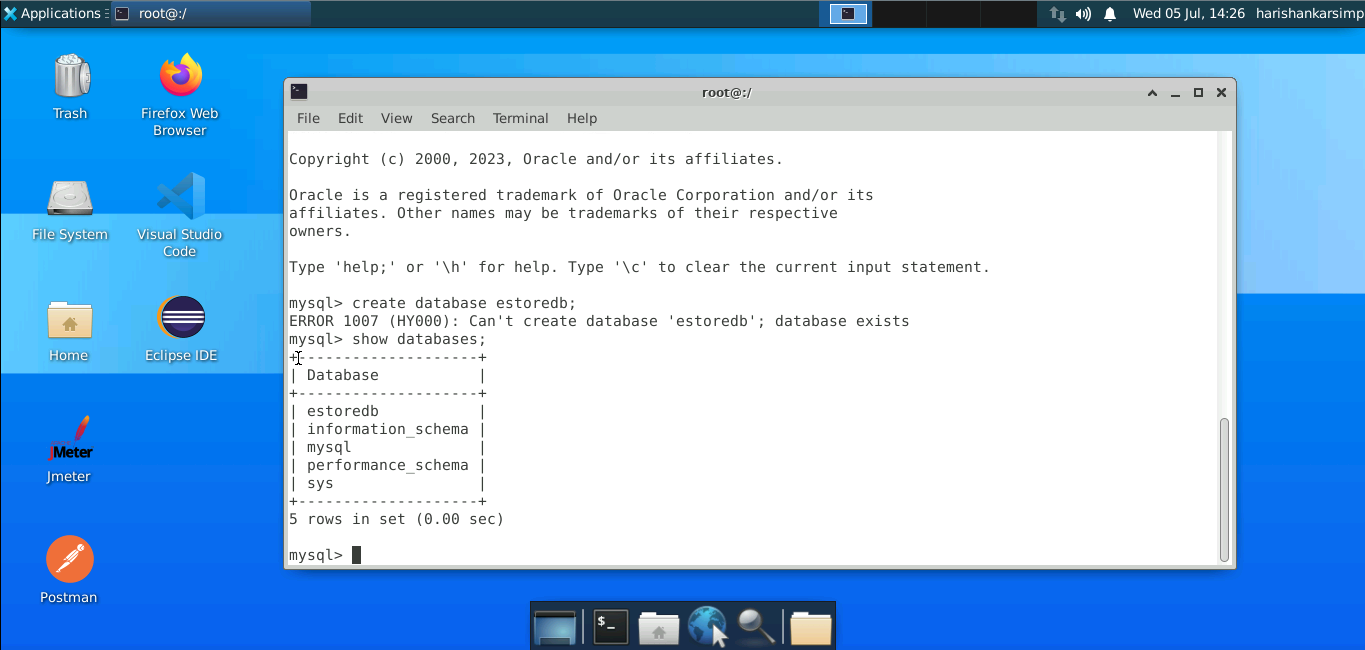
A screenshot of a computer

Description automatically generated

**Note:** Enter **password** if prompt for password.

1. Now you can setup the MySQL database which is required for the application to work. Firstly, create a database in MySQL using the command:

**create database estoredb;**



1. Now, run **use estoredb;** command and create admins table:

CREATE TABLE `admins` (

 `admin\_id` int NOT NULL,

 `added\_on` datetime(6) DEFAULT NULL,

 `email` varchar(255) DEFAULT NULL,

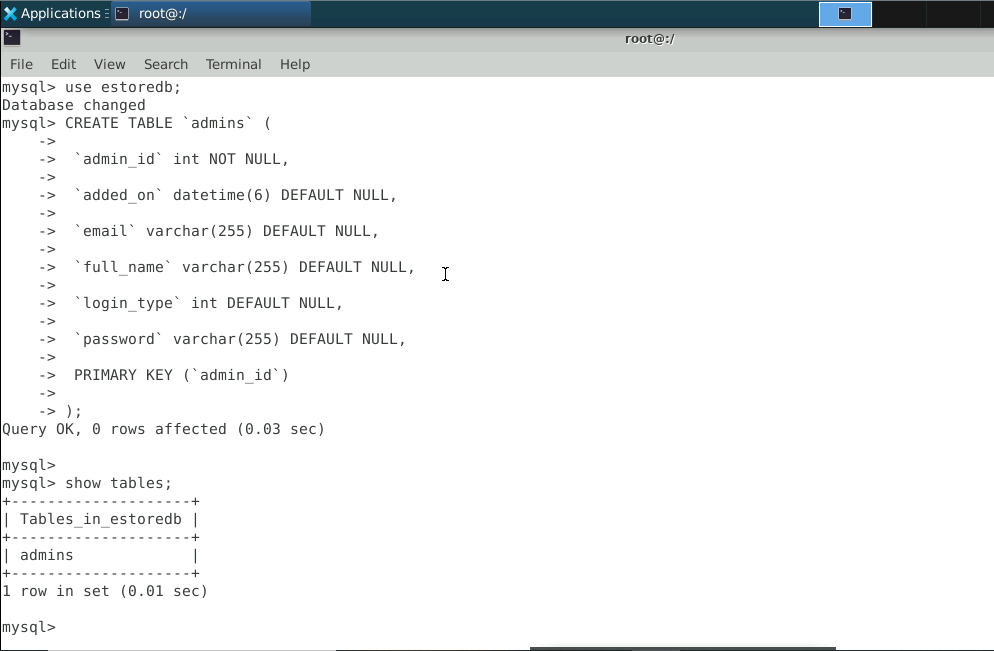
 `full\_name` varchar(255) DEFAULT NULL,

 `login\_type` int DEFAULT NULL,

 `password` varchar(255) DEFAULT NULL,

 PRIMARY KEY (`admin\_id`)

);



1. Insert a record to the admins table:

**INSERT INTO `admins` VALUES (1,'2023-02-10 00:00:00.000000','admin@example.com','John Watson',1,'admin123');**

**A screenshot of a computer

Description automatically generated**

Our Docker network and MySQL container are ready now.

**Step 3: Creating Dockerfiles and running containers for applications**

1. Launch Visual Studio Code and open the **eStore Codes** folder

A screenshot of a computer

Description automatically generated

1. Create a **Dockerfile** for the angular frontend application in the root directory

A screenshot of a computer program

Description automatically generated with medium confidence

1. Similarly create a **Dockerfile** for Java backend application

A screenshot of a computer program

Description automatically generated with medium confidence

1. Open a terminal, navigate to the frontend application directory, and build the Docker image:

**sudo docker build -t estore-admin .**

A screenshot of a computer program

Description automatically generated with medium confidence

1. Run the below command to check if the image is generated successfully:

**sudo docker images**

****

1. Since, the image is generated now we can create container from the image and run it:

**sudo docker run --name=estore-admin-container -d -p 4200:80 --network=estore-mysql estore-admin**

**A screenshot of a computer

Description automatically generated**

Here we are using port **4200** to run the admin application.

1. Open the browser window and type **localhost:4200** to validate if Angular web admin project is running inside the docker container

A screenshot of a computer

Description automatically generated

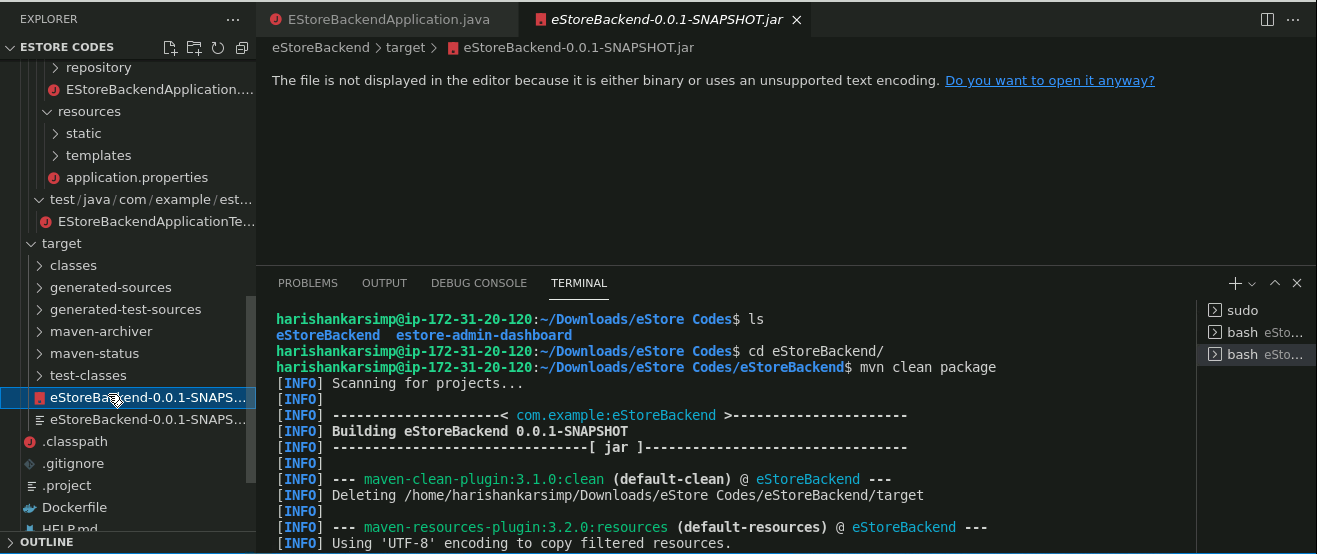
1. Moving ahead for Java backend, we need to externalize **application.properties** file in the main function so that we can give the configuration for MySQL database in Docker Container

A screen shot of a computer

Description automatically generated with medium confidence

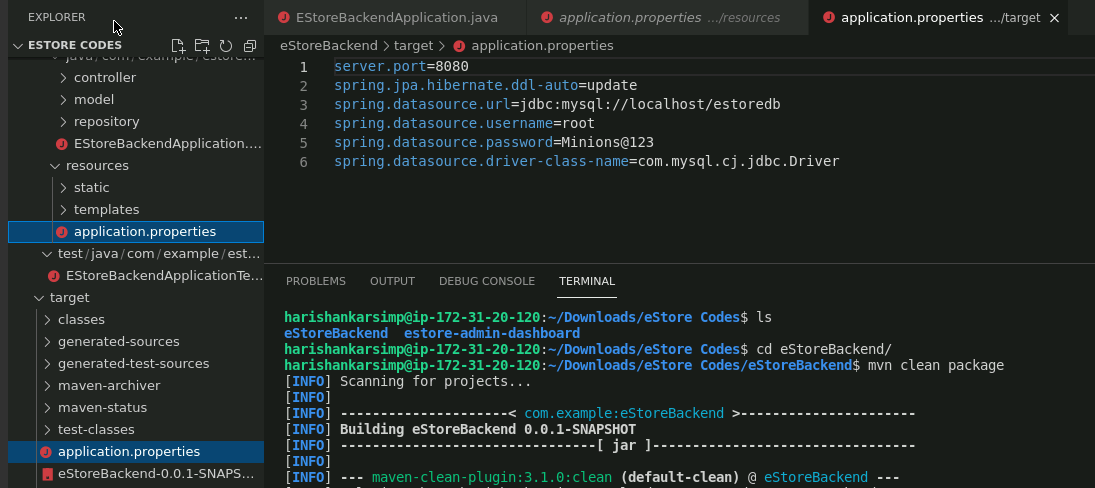
1. Open the terminal, navigate to the Java backend root directory, and run the command:

**mvn clean package**

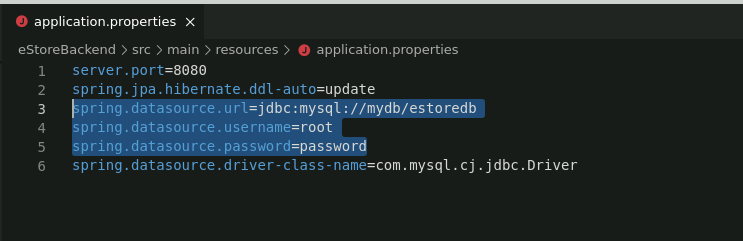


This command will generate a JAR compressed file inside the target directory.

1. Now, copy the **application.properties** file in the **target** directory

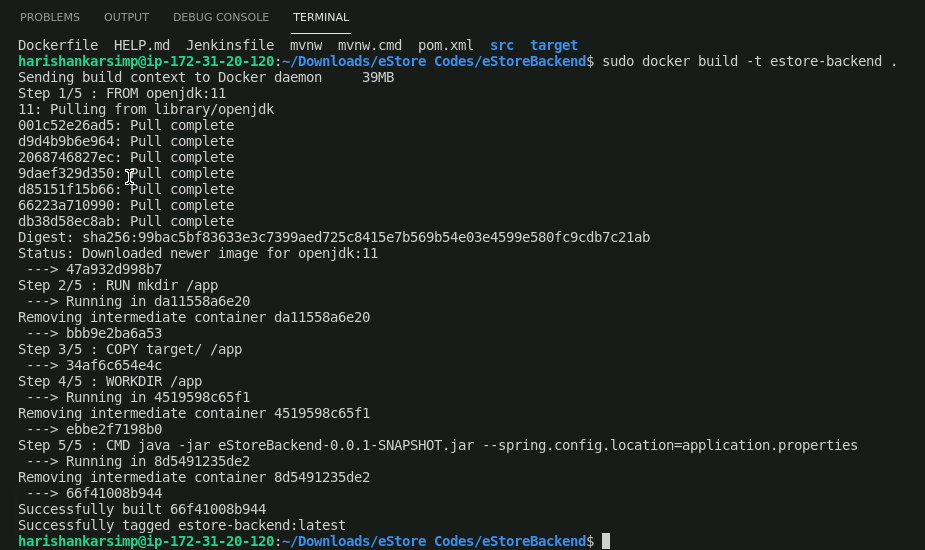


1. Update the **application.properties** file with updated details



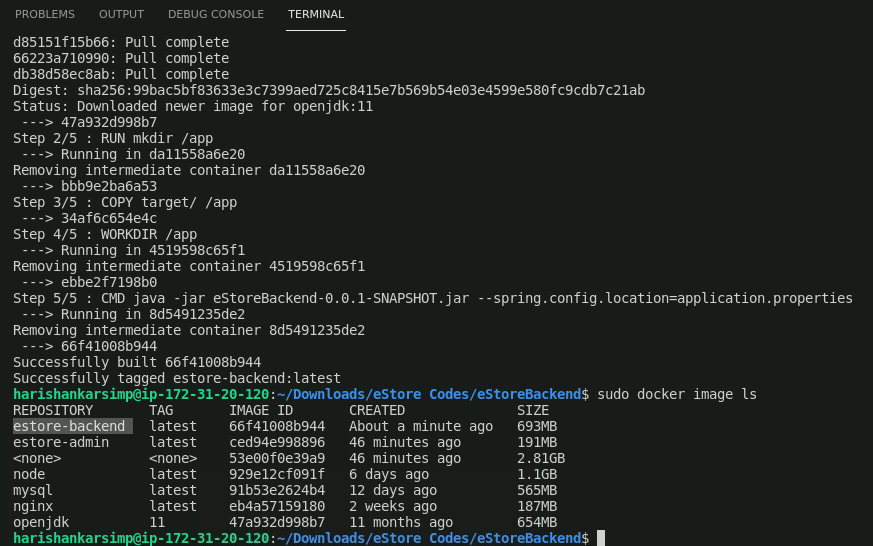
1. Now, create an image for the project. Open the terminal window and execute the command:

**sudo docker build -t estore-backend .**

****

1. Run the command to check if the image is generated successfully:

**sudo docker image ls**

****

1. Image is generated now we can create container from the image and run it. Use the command:

**sudo docker run --name estore-backend-container -d -p 8080:8080 --network=estore-mysql estore-backend**

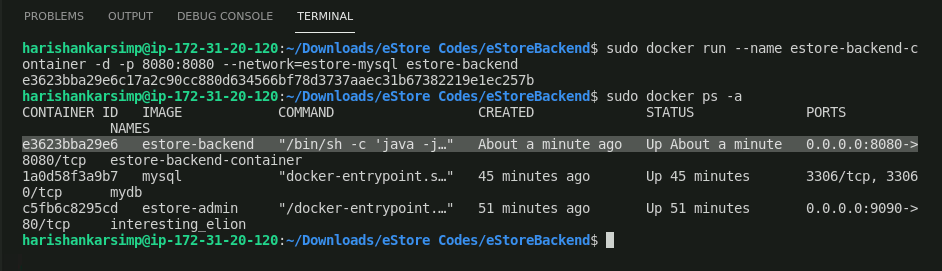
A screen shot of a computer

Description automatically generated with medium confidence

Here we are using port **8080** to run the java backend application, which can be modified in **application.properties** file.

1. Check the container is live or not by executing the command:

**sudo docker ps -a**

****

1. Now, open the browser and type **localhost:8080** to check if Java backend is up

A screenshot of a computer

Description automatically generated

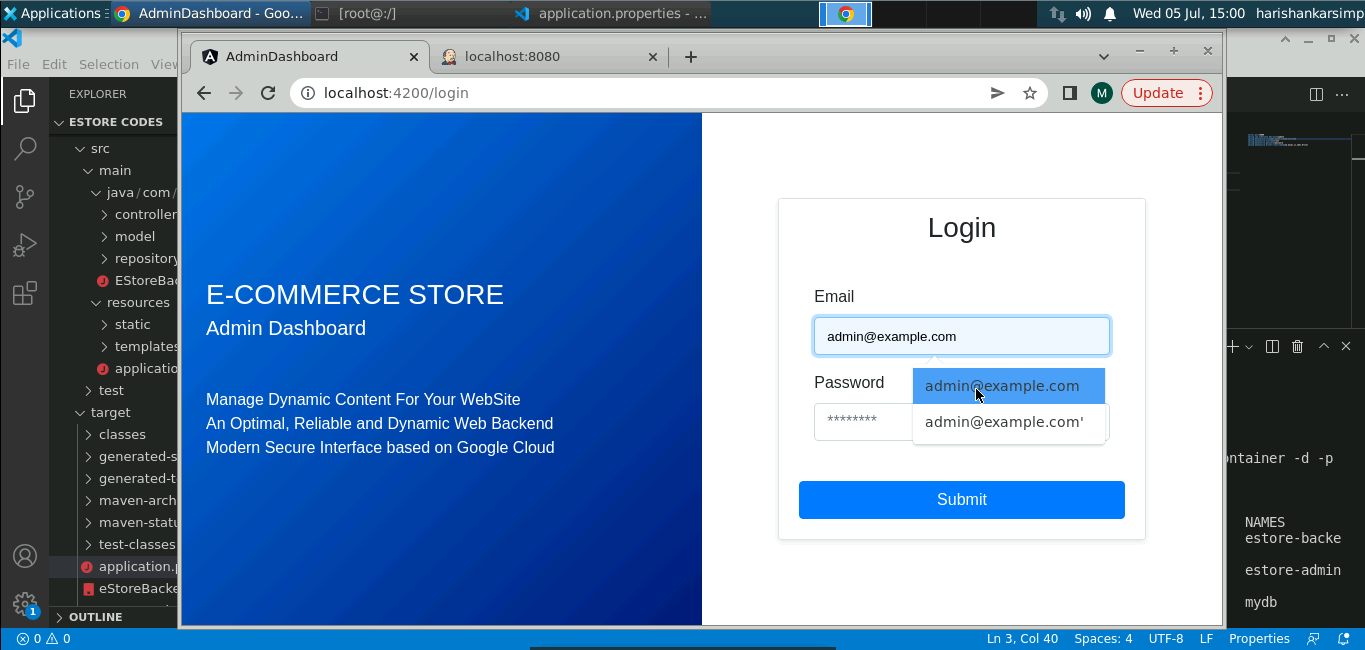
Our frontend and backend applications are up and running using the Docker containers environment and within a same network **(estore-mysql)**.

**Step 4: Verifying the applications function**

1. Navigate to the Angular application and login using the admin credentials we added in the MySQL container:

**Email: admin@example.com**

**Password: admin123**

****

1. Click **Submit**

A screenshot of a computer

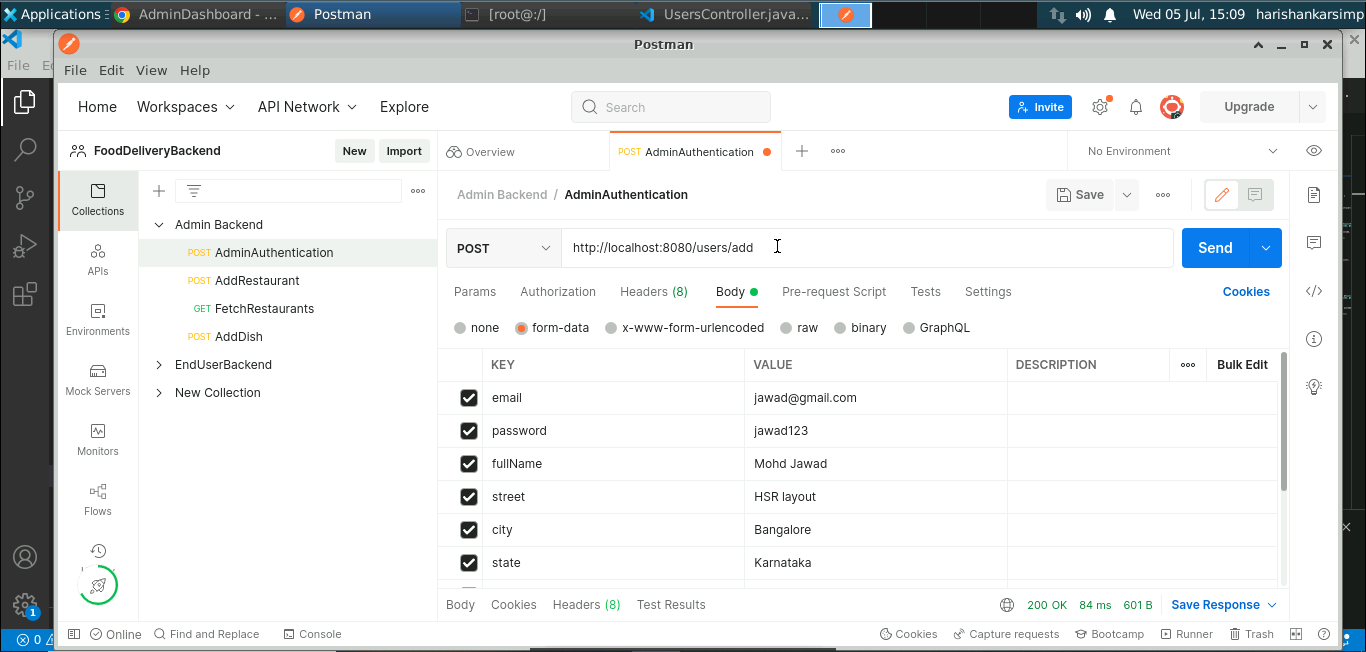
Description automatically generated

A screenshot of a computer

Description automatically generated

You can see the login is successful. Right now, there are no users available in the database thus the table is empty.

1. You can use postman to add users, products, orders etc



You should receive a **200** status code and successful message on the console for the right insertion.

A screenshot of a computer

Description automatically generated

1. Once users are added you can refresh the admin home dashboard page

A screenshot of a computer

Description automatically generated

You can see the user details been display that currently added through Postman.

**Step 5: Configuring the Git repository for both the applications**

1. Go to the GitHub and create two repositories for Angular frontend and Java backend applications

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

1. Push the files to the created repositories follow git commands

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

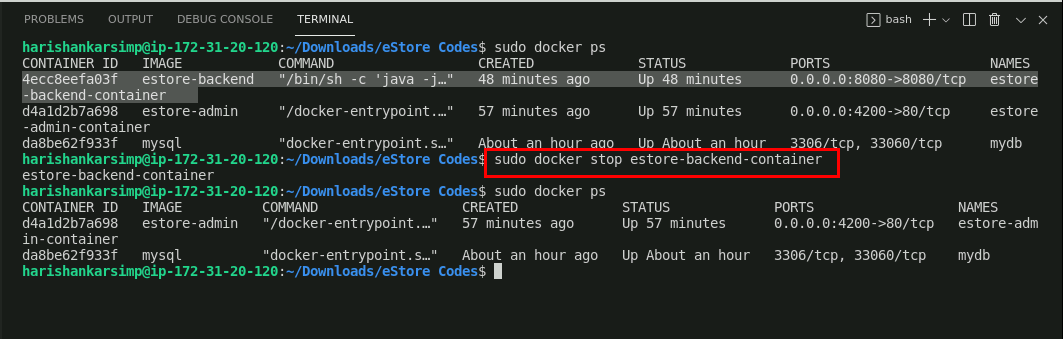
**A screenshot of a computer

Description automatically generated**

**Step 6: Setting up the Jenkins**

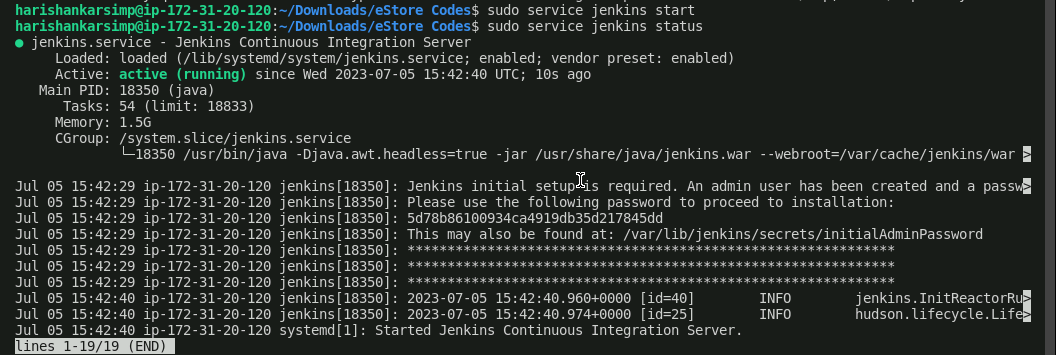
1. Since Jenkins by default uses port **8080** which is already occupied by Java backend, let’s stop it:

**sudo docker stop estore-backend-container**

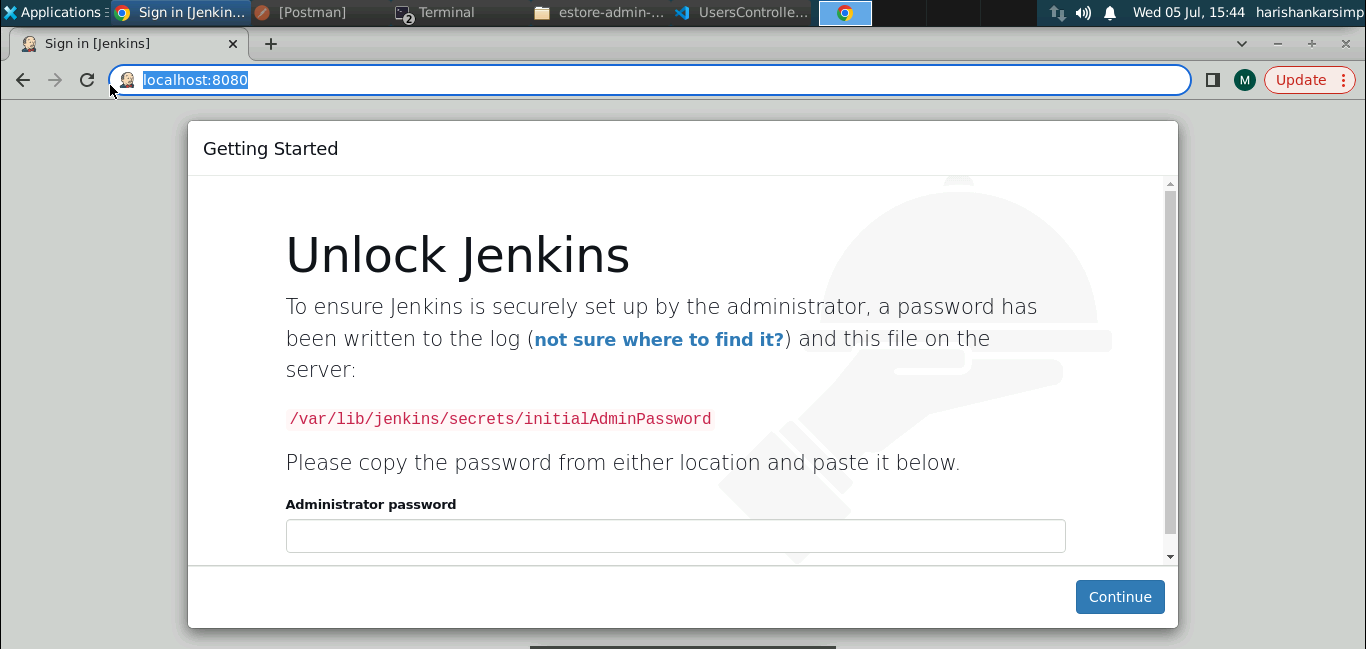
****

1. Start the Jenkins service using the command:

**sudo service jenkins start**



1. Navigate to the **localhost:8080** in the browser



You should see the Jenkins initial page.

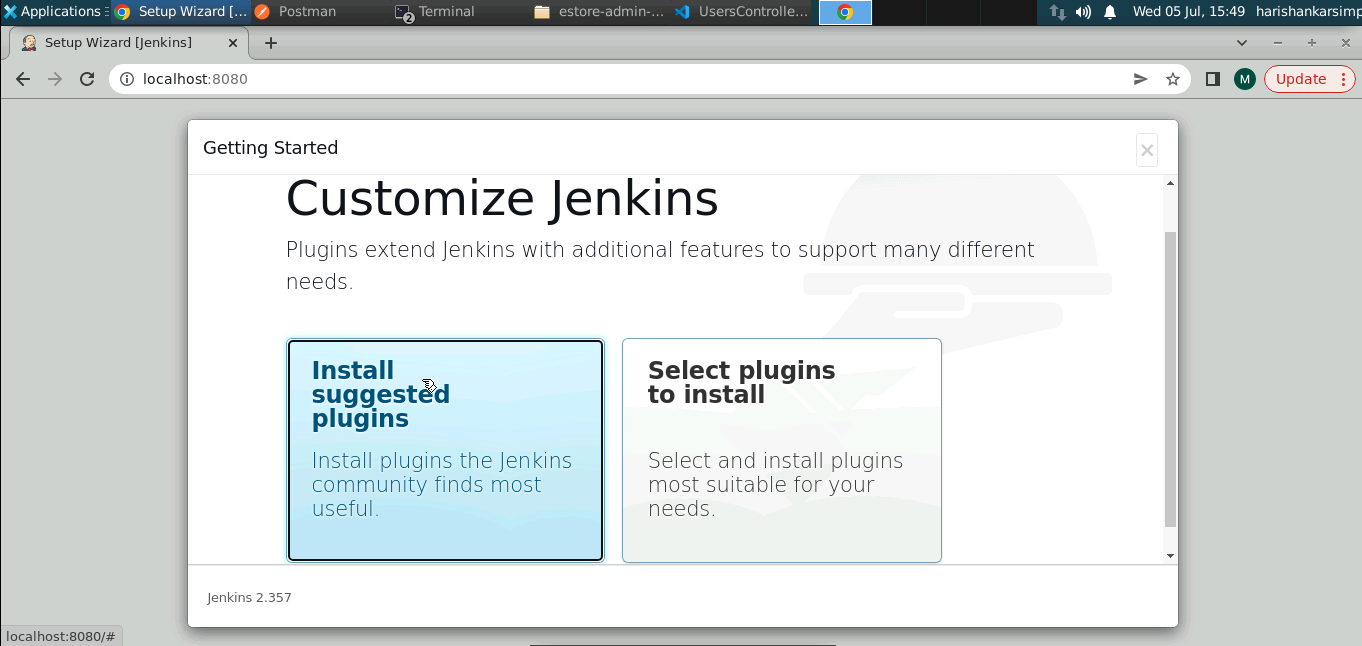
1. To unlock the Jenkins, copy the given path and print the hash password using command:

**sudo cat [path]**

**A screenshot of a computer

Description automatically generated**

1. Copy paste the hashcode and click **Continue**. Next click on **Install suggested plugins**.



1. Then, provide admin credentials that will be required to login with Jenkins and click on **Save and Continue**

A screenshot of a computer

Description automatically generated

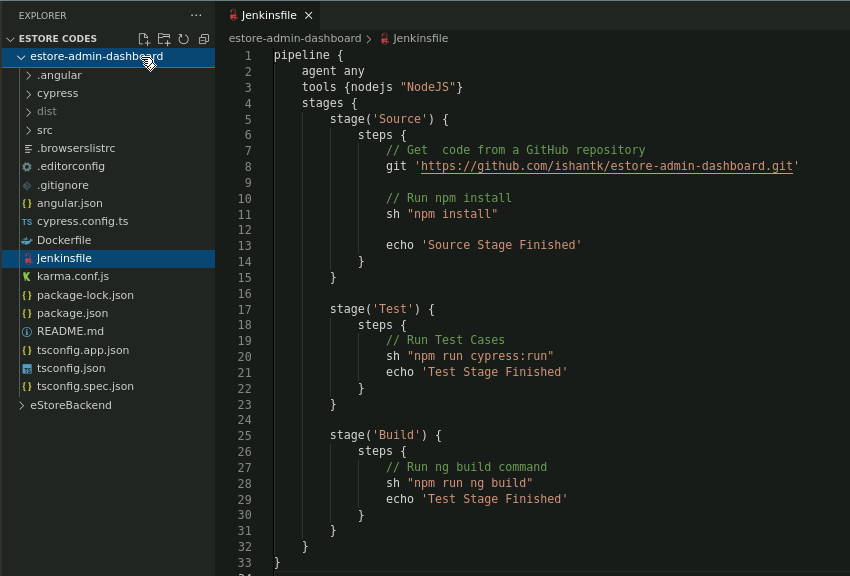
A screenshot of a computer

Description automatically generated

Jenkins is ready to use now.

**Step 7: Creating a Jenkinsfile and CI/CD pipeline in local machine**

1. Create a Jenkinsfile for the Angular project in the root directory



1. Similarly create Jenkinsfile for Java backend project

A screenshot of a computer program

Description automatically generated

**Note:** Please commit and push the updates in the GitHub repositories.

1. Navigate to the Jenkins dashboard install the NodeJS plugin for the Angular project and then restart Jenkins

A screenshot of a computer

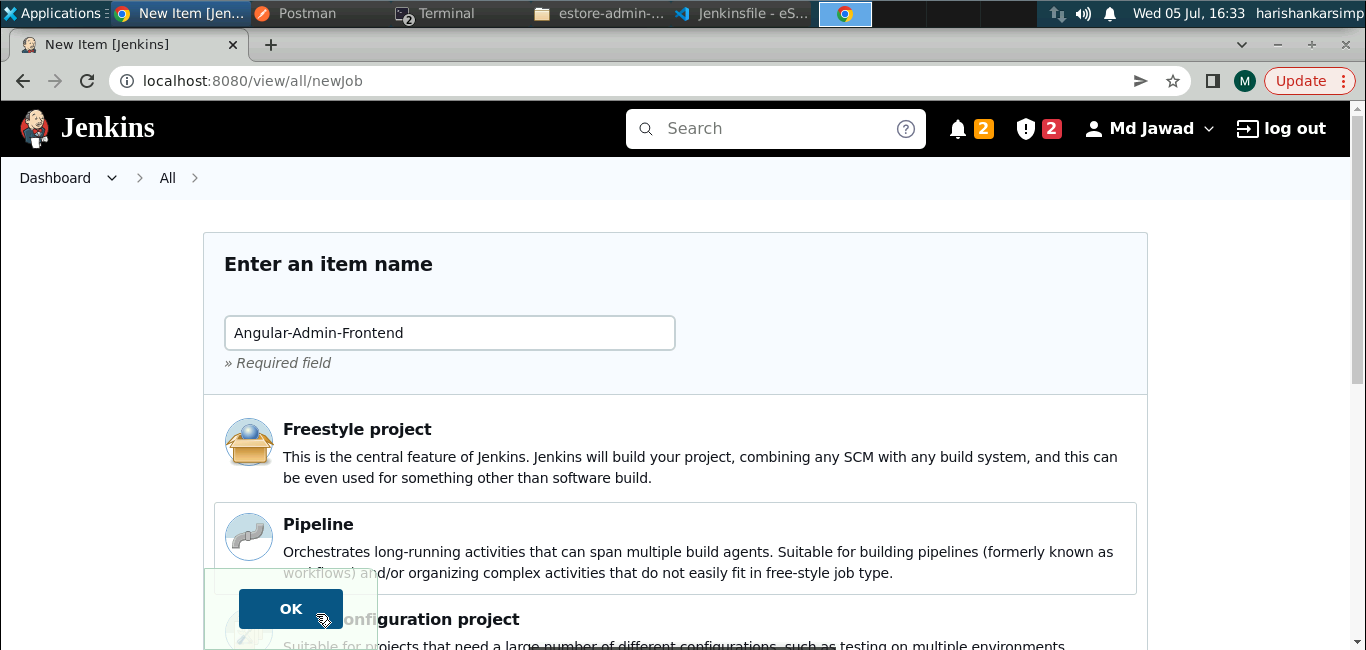
Description automatically generated

1. You need to make sure that the NodeJS tool is properly configured in your Jenkins environment

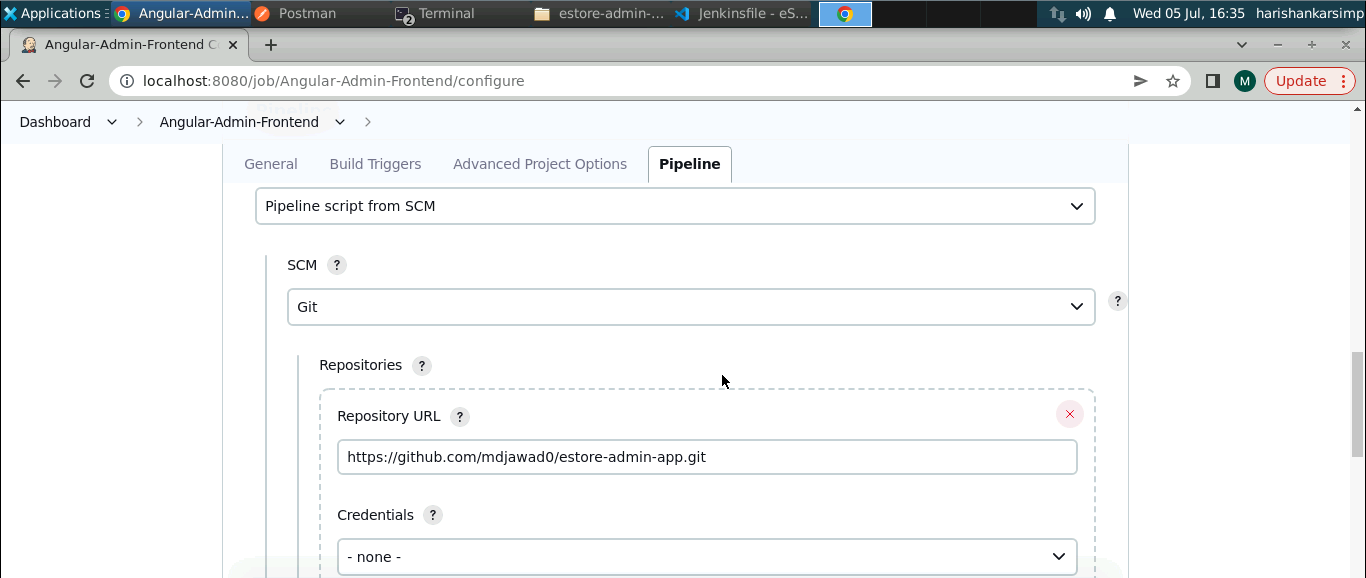
A screenshot of a computer

Description automatically generated

1. Let’s create a new pipeline for Angular project



1. Provide the GitHub repository URL under **Advanced Project Options** tab



1. Keep the Script Path as **Jenkinsfile** and click on **Save**

A screenshot of a computer

Description automatically generated

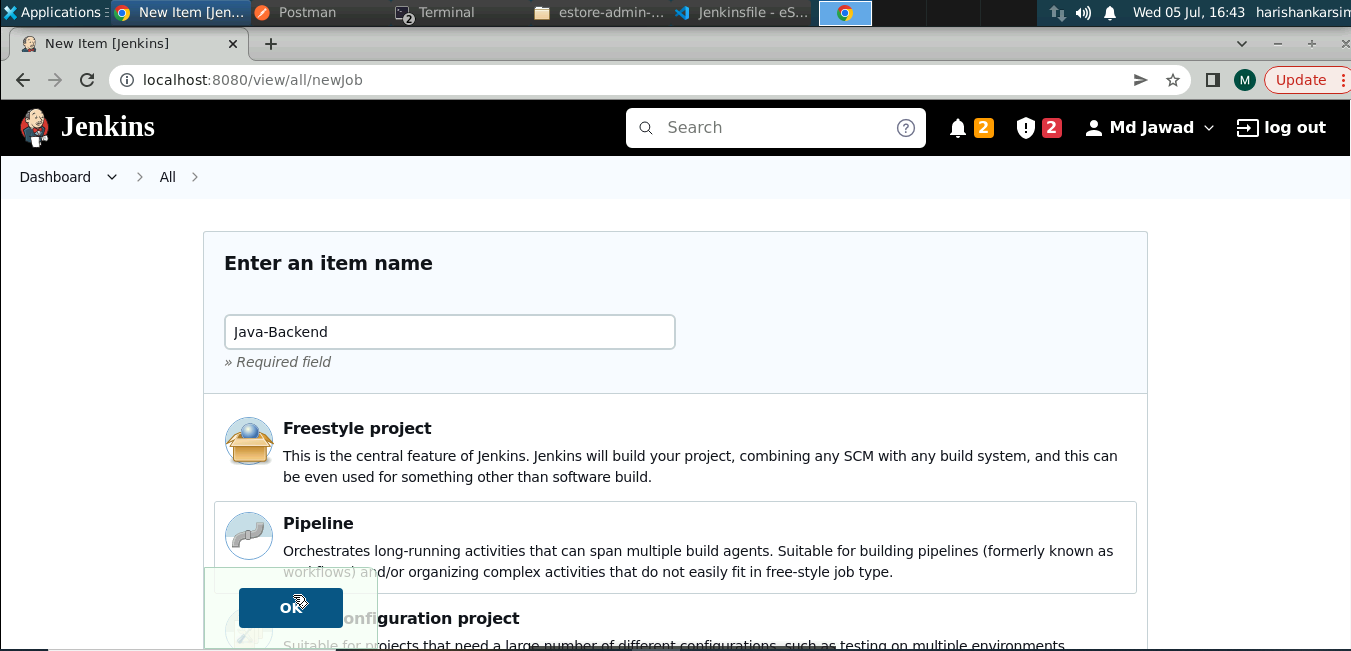
1. Click on **Build Now** to execute the pipeline

A screenshot of a computer

Description automatically generated

You can see all the defined stages in the Jenkinsfile are passed and successful. Please review your cypress testcases if pipeline fails.

1. Similarly create another pipeline for Java backend project following same steps and click on **Build Now**



A screenshot of a computer

Description automatically generated

**Note:** if you get any **./mvnw permission denied** error, update the script in Jenkins file to set permission on **mvnw** scripts.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

You can see all the defined stages in the Jenkinsfile are passed and successful. Please review your testcases if the pipeline fails.

**Step 7: Setting up the EC2 instance**