**DOCKER PROJECT : Continuous Deployment using Git, Jenkins, SonarQube, and Docker**

**Overview:**

This project aimed to design and implement a Continuous Integration/Continuous Deployment (CI/CD) pipeline using Git, Jenkins, SonarQube, and Docker. The goal was to improve code quality, deployment efficiency, and security.

**CI/CD Pipeline Architecture:**

1. Source Code Management: Git

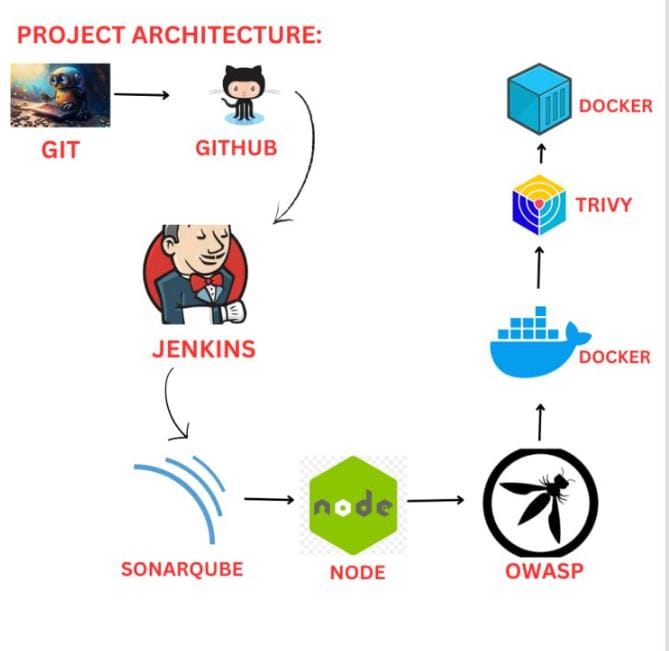
2. Continuous Integration Server: Jenkins

3. Code Quality Analysis: SonarQube

4. Security Scanning: OWASP (dependency check) and Trivy (image scanning)

5. Containerization: Docker

6. Image Repository: Docker Hub



**Implementation Steps:**

Step 1: Launch EC2 Instance

- Instance type: t2.large

- Storage: 25 GB

Step 2: Install Java, Git, and Docker -

sudo yum install java-17 git docker -y

- Started and enabled Docker service:

systemctl start docker

systemctl enable docker

- Verified versions:

java -version

git --version

docker --version

Step 3: Configure Security Group

- Added rule to allow all traffic in security group of instance.

Step 4: Install Jenkins and sign in

Step 5: Install Trivy

- Install Trivy by the following command :

wget https://github.com/aquasecurity/trivy/releases/download/v0.18.3/trivy\_0.18.3\_Linux- 64bit.tar.gz tar zxvf trivy\_0.18.3\_Linux 64

-extracted by the command: tar -zxvf filename

-moved to local bin – mv trivy /usr/local/bin

- Exported path [export PATH=$PATH:/usr/local/bin/] using vi .bashrc and source .bashrc.

- Verify Trivy version.

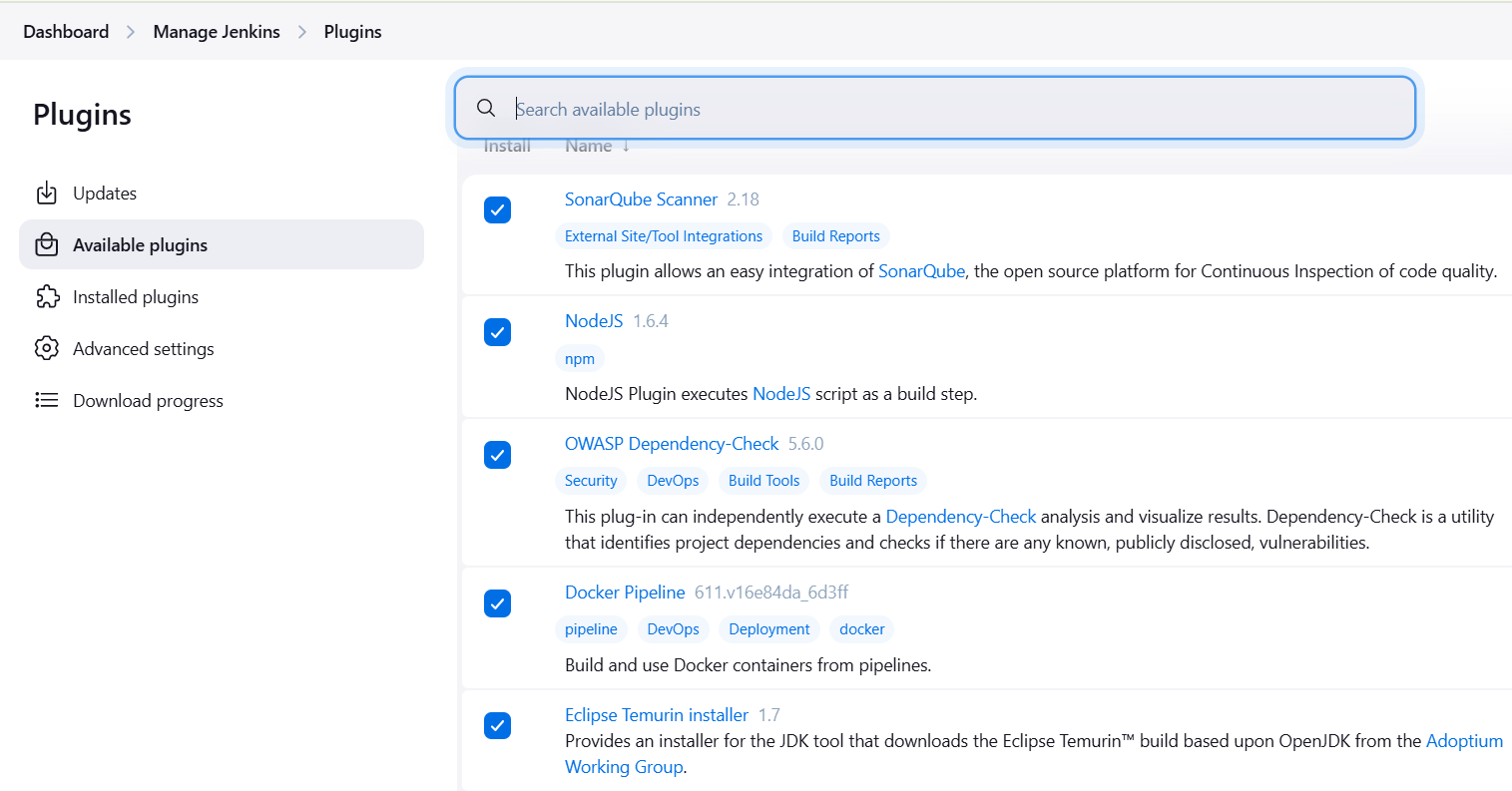
Step 6: Configure Docker Permissions

- Run the command to give permission for Docker:

sudo chmod 777 /var/run/docker.sock

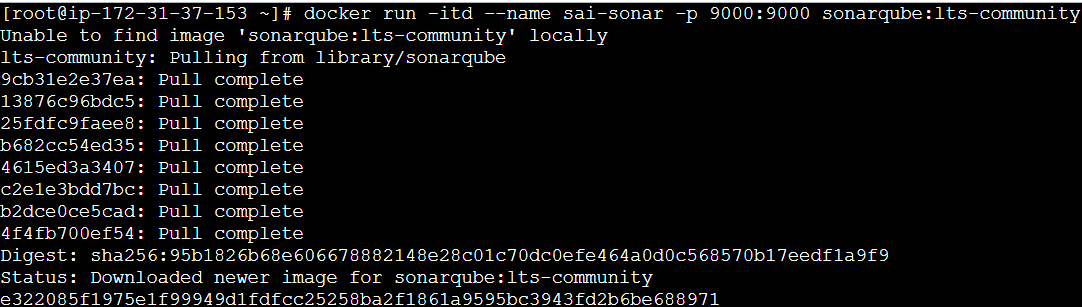
Step 7: Install Jenkins Plugins

- Installed SonarQube Scanner, NodeJS, OWASP Dependency Check, Docker Pipeline, and Eclipse Temurin Installer plugins.



Step 8: Run SonarQube Docker Container

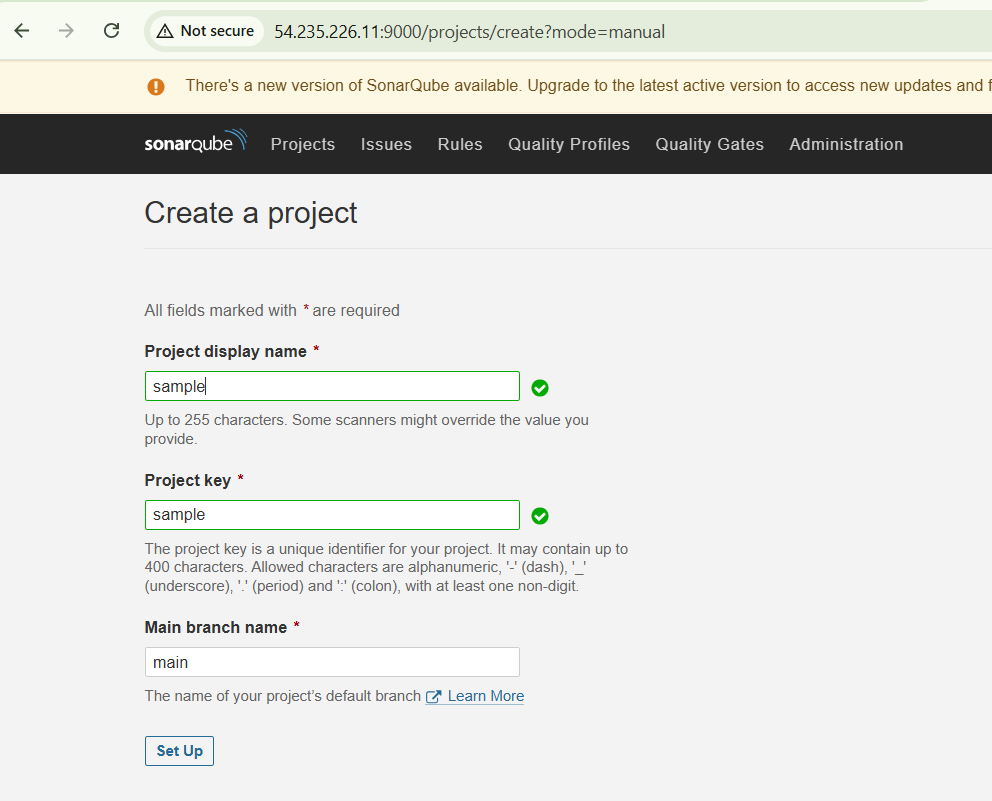
docker run -itd --name sonar -p 9000:9000 sonarqube:lts-community.



Step 9: Configure SonarQube

- Accessed SonarQube login page using instance public IP and port 9000.

- Logged in with default credentials (admin/admin) and changed password.

- Created sample project and copy the token. 

Step 10: Configure Jenkins

- Added JDK installations with version 17.0.8.1+1.

- Added SonarQube Scanner with version 6.1.0.4777.

- Added NodeJS with version 16.2.0.

- Added Dependency-Check with version 6.5.1.

Step11: Configured SonarQube server with URL and credentials:

Go to Jenkins > Manage Jenkins > Configure System.

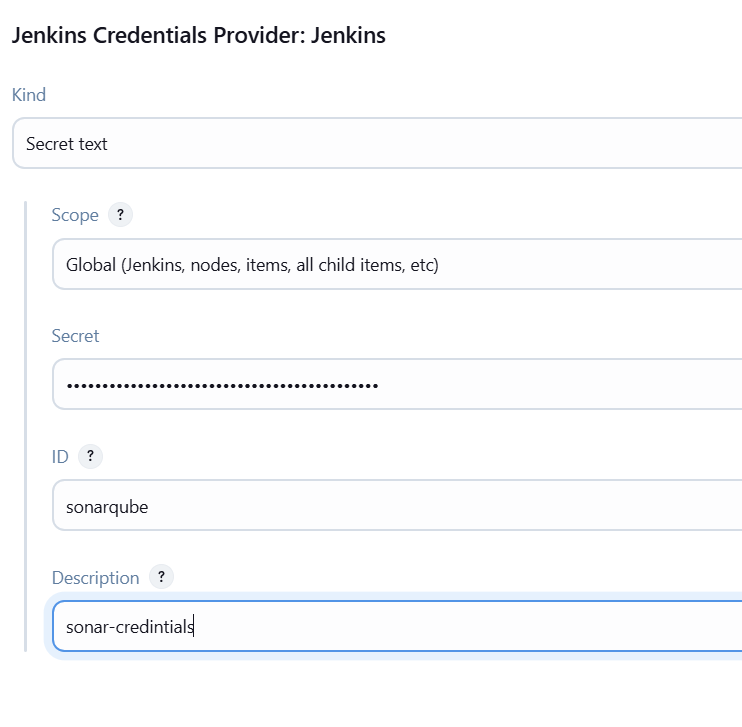
🡪 Scroll down to the SonarQube section and Click on the "Add SonarQube" button.

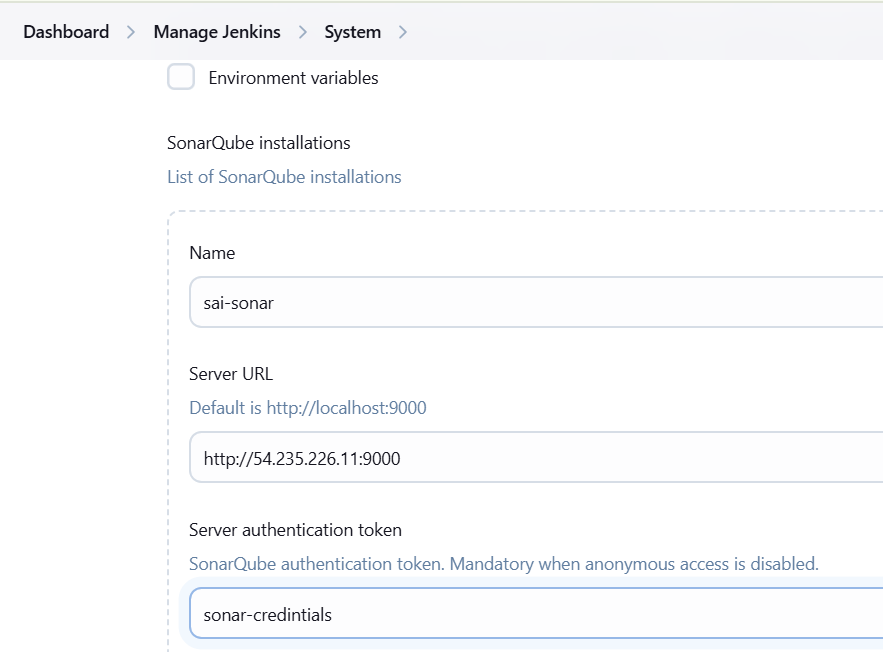
🡪Enter a name for the SonarQube instance.

🡪Enter the URL of the SonarQube instance.

🡪Add credentials by selecting "Secret text" and entering the token ID copied from SonarQube.

🡪Save the changes

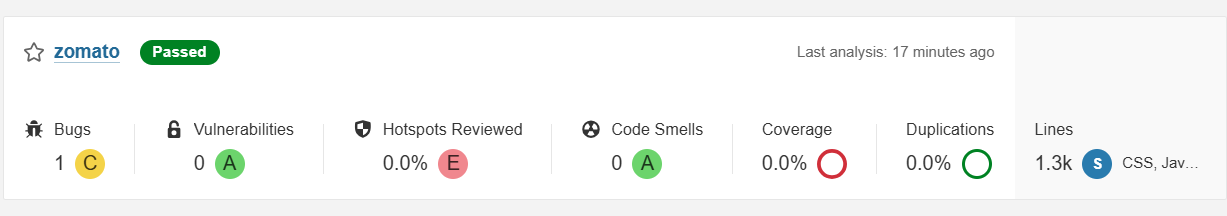




Step12: Create a New SonarQube Project

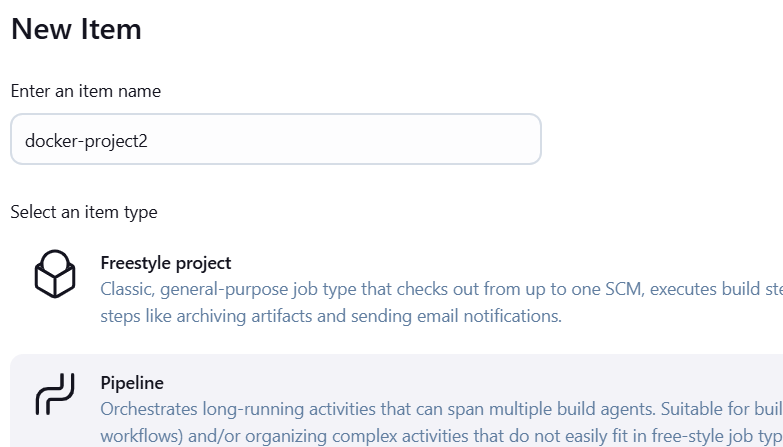
- Log in to your SonarQube instance and create a new project.

- Enter the project name as "Zomato" and project key as "Zomato" and create .



Step13: Create a New Jenkins Pipeline by the following steps:

Click on "New Item" and select "Pipeline" as the item type.



Step14: Configure the pipeline script with the necessary credentials and settings.





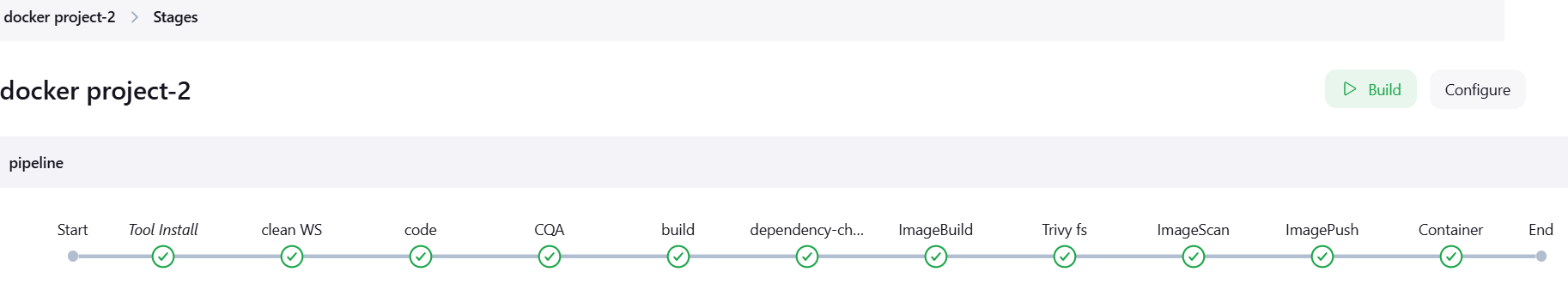
🡪 pipeline script

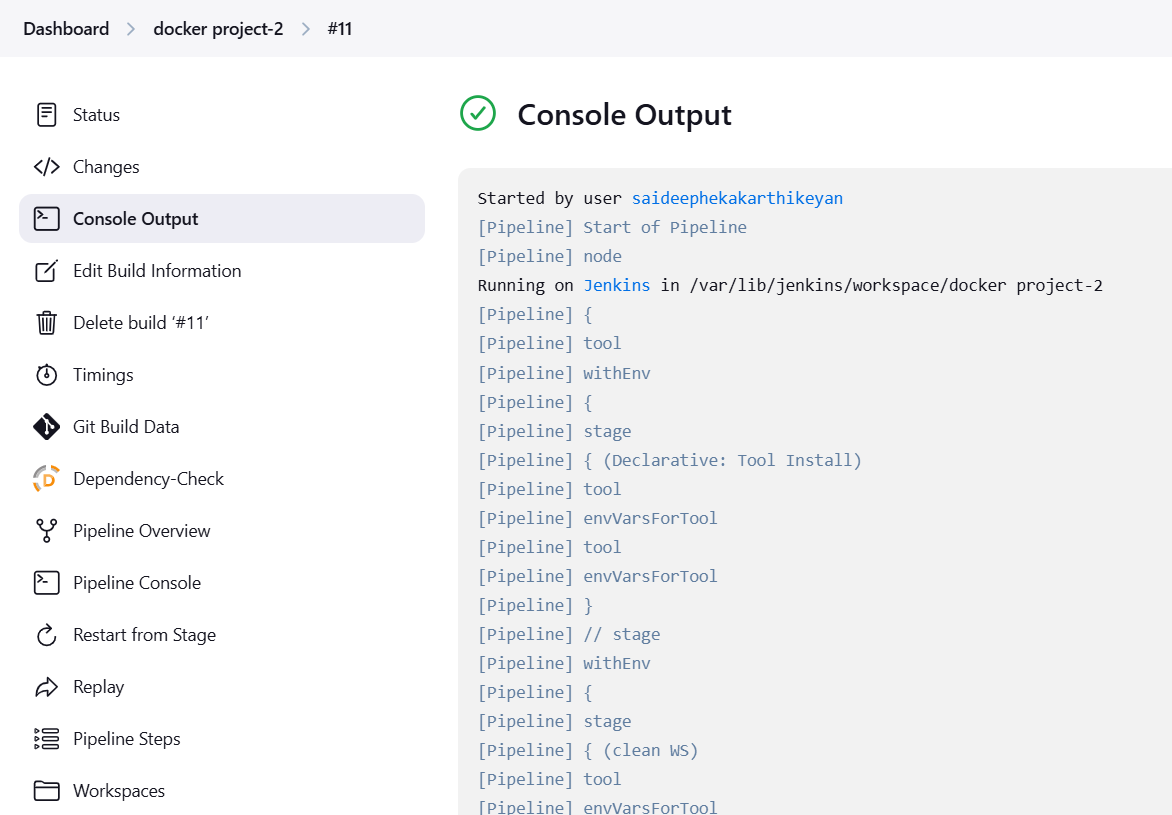
Step15: Save the pipeline configuration and Click on "Build Now" to start the pipeline.

Step16: Verify the Pipeline Build Result

- Monitor the pipeline build result in the Jenkins console output.

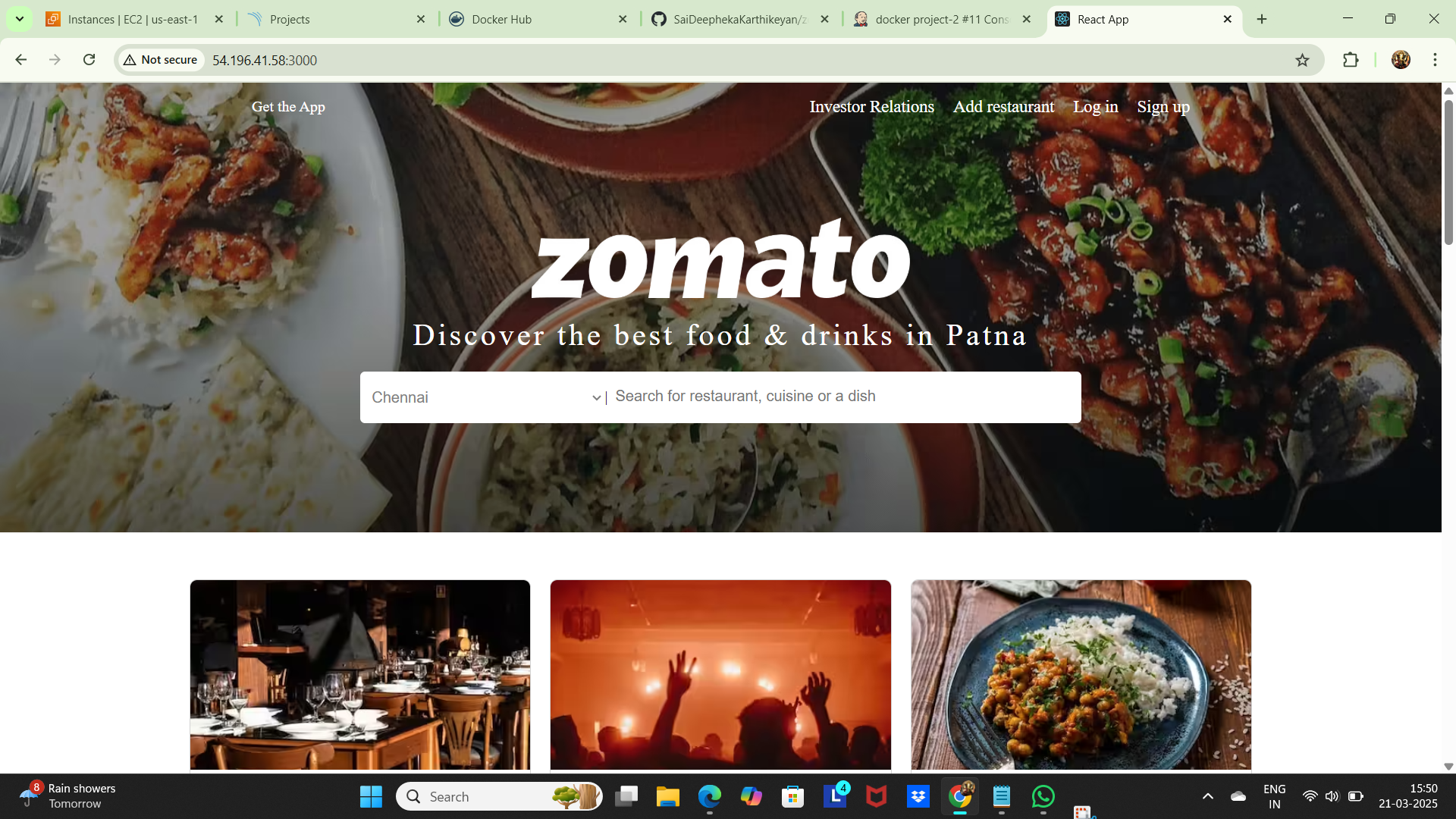
- Verify that all stages have completed successfully.





Step17: Access the deployed application using http://<IP\_ADDRESS>:3000.

Step18: Verify that the Zomato application is up and running.



**Benefits:**

1. Improved code quality through SonarQube analysis.

2. Enhanced security through OWASP dependency check and Trivy image scanning.

3. Faster deployment through automated CI/CD pipeline.

4. Simplified containerization using Docker.

**Conclusion:**

This project demonstrated the successful implementation of a CI/CD pipeline using Git, Jenkins, SonarQube, and Docker. The pipeline improved code quality, deployment efficiency, and security, making it an essential tool for modern software development.