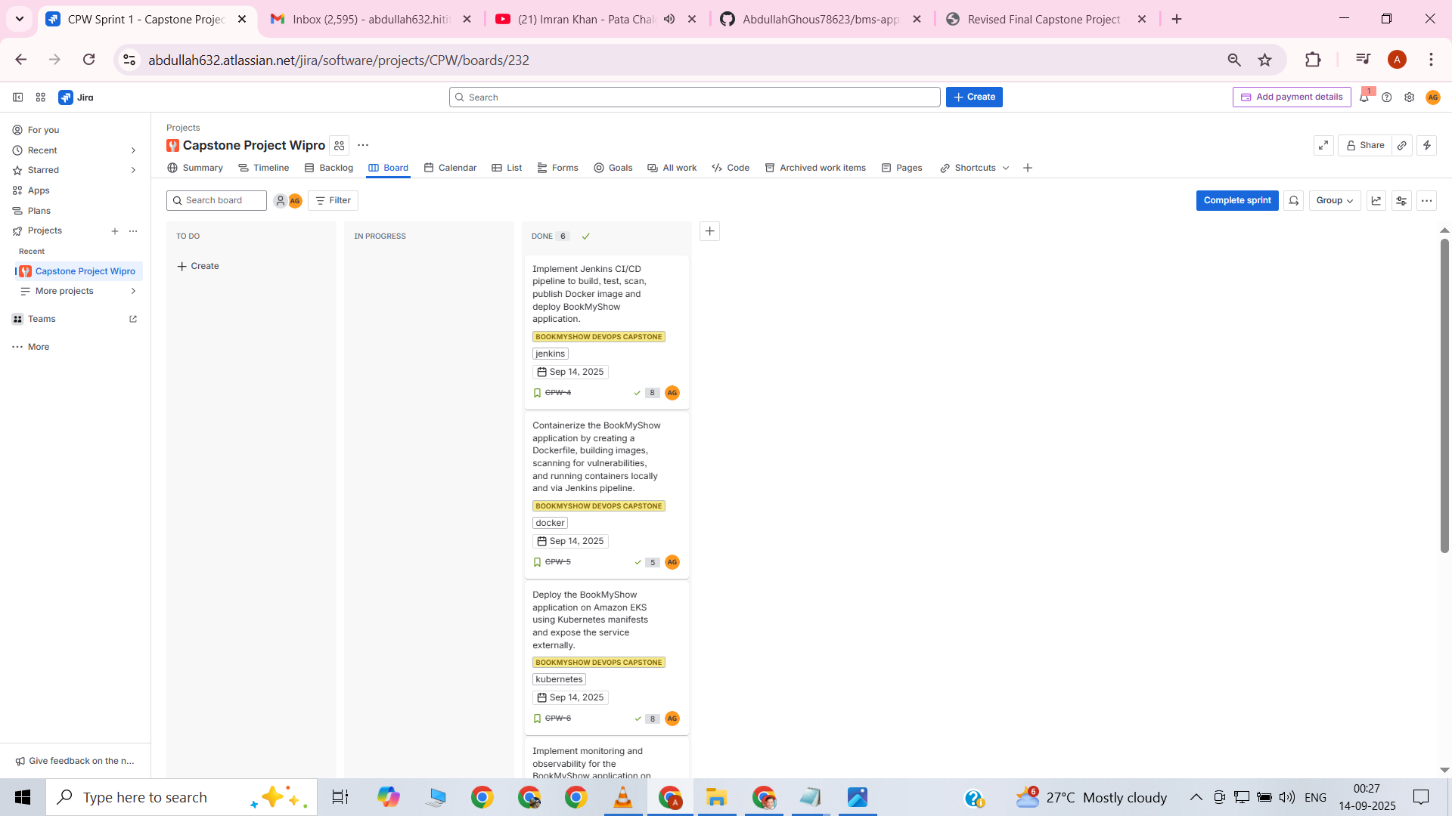
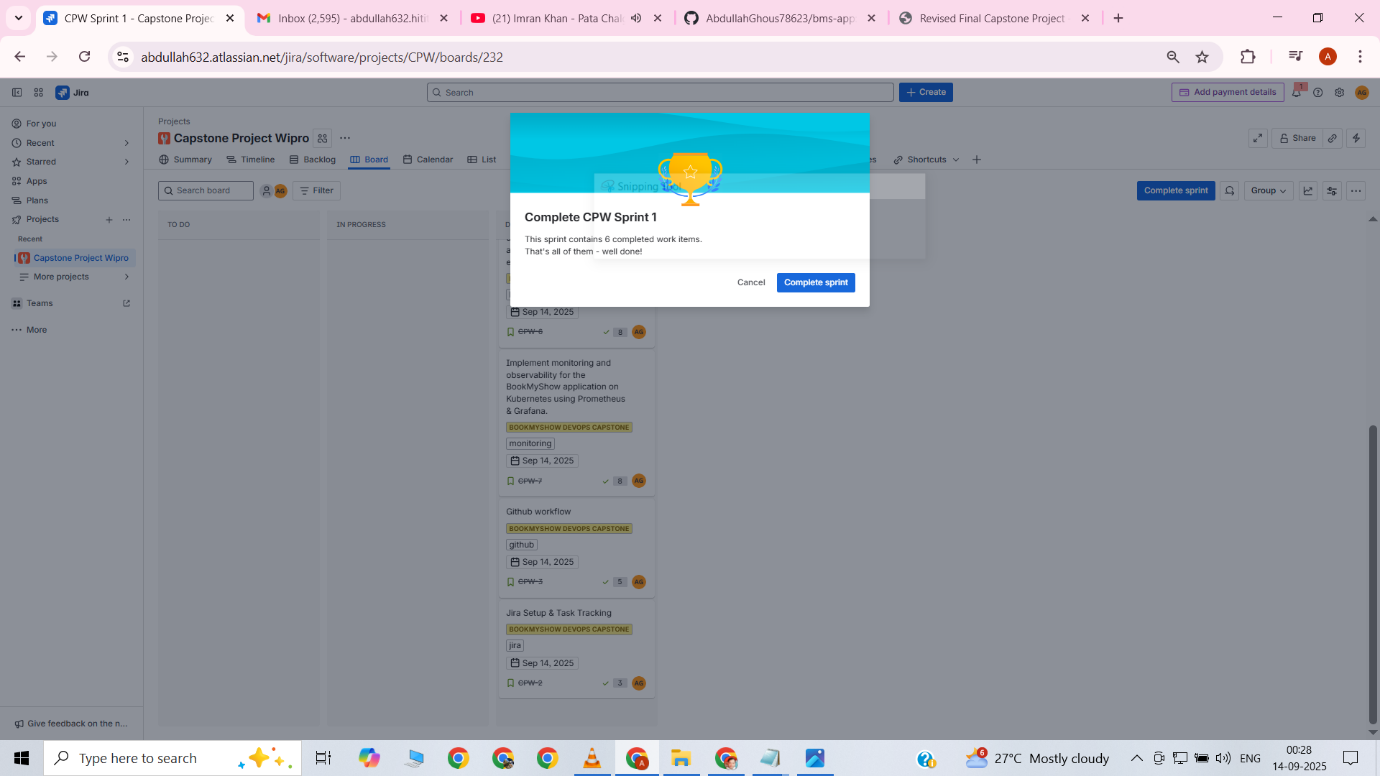
Final Deliverables

1. . Jira Board





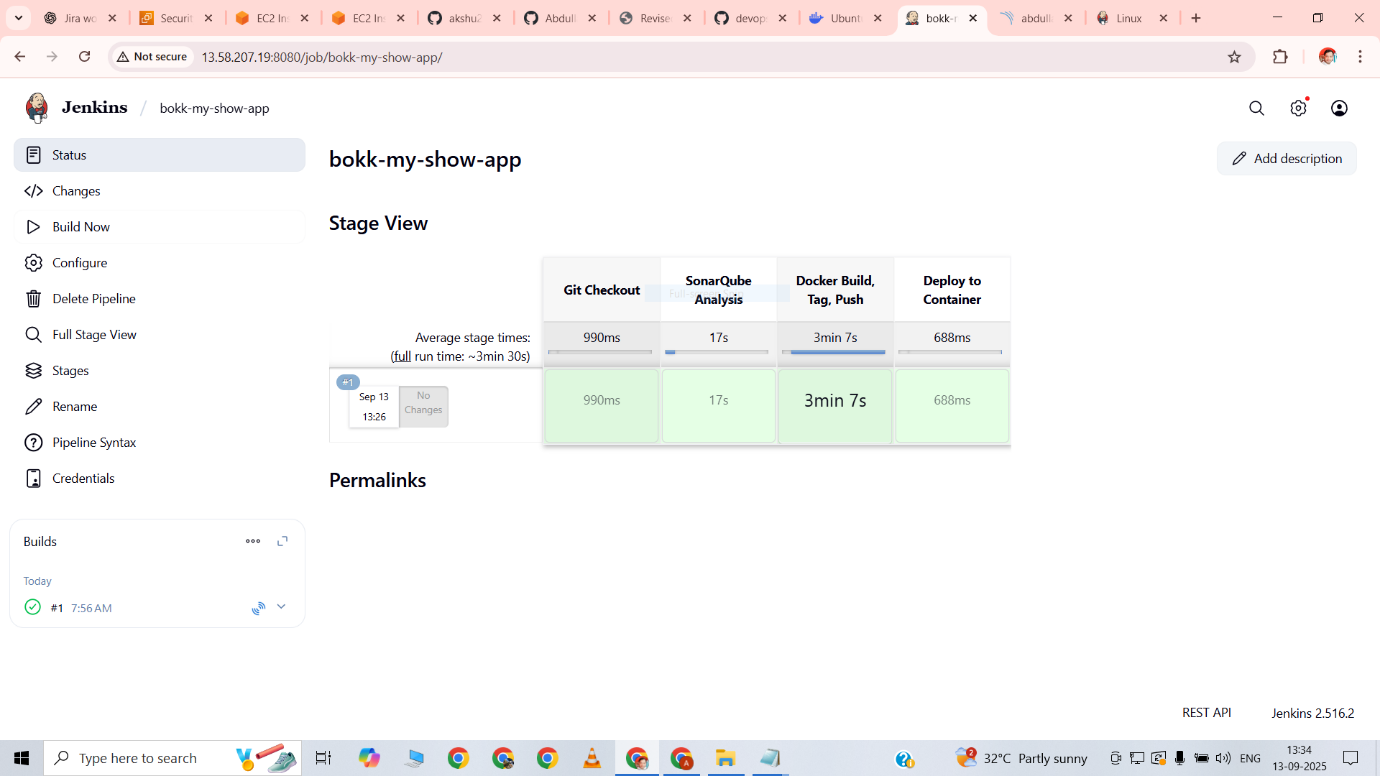
Created a dedicated Jira project board named “Automated CI/CD Pipeline with Jenkins, Terraform, and Monitoring Integration”  
Defined Epics, Stories, and Tasks to structure the project workflow  
Configured the board to follow To Do → In Progress → Done lifecycle  
This board served as the central place to track all project activities from start to finish

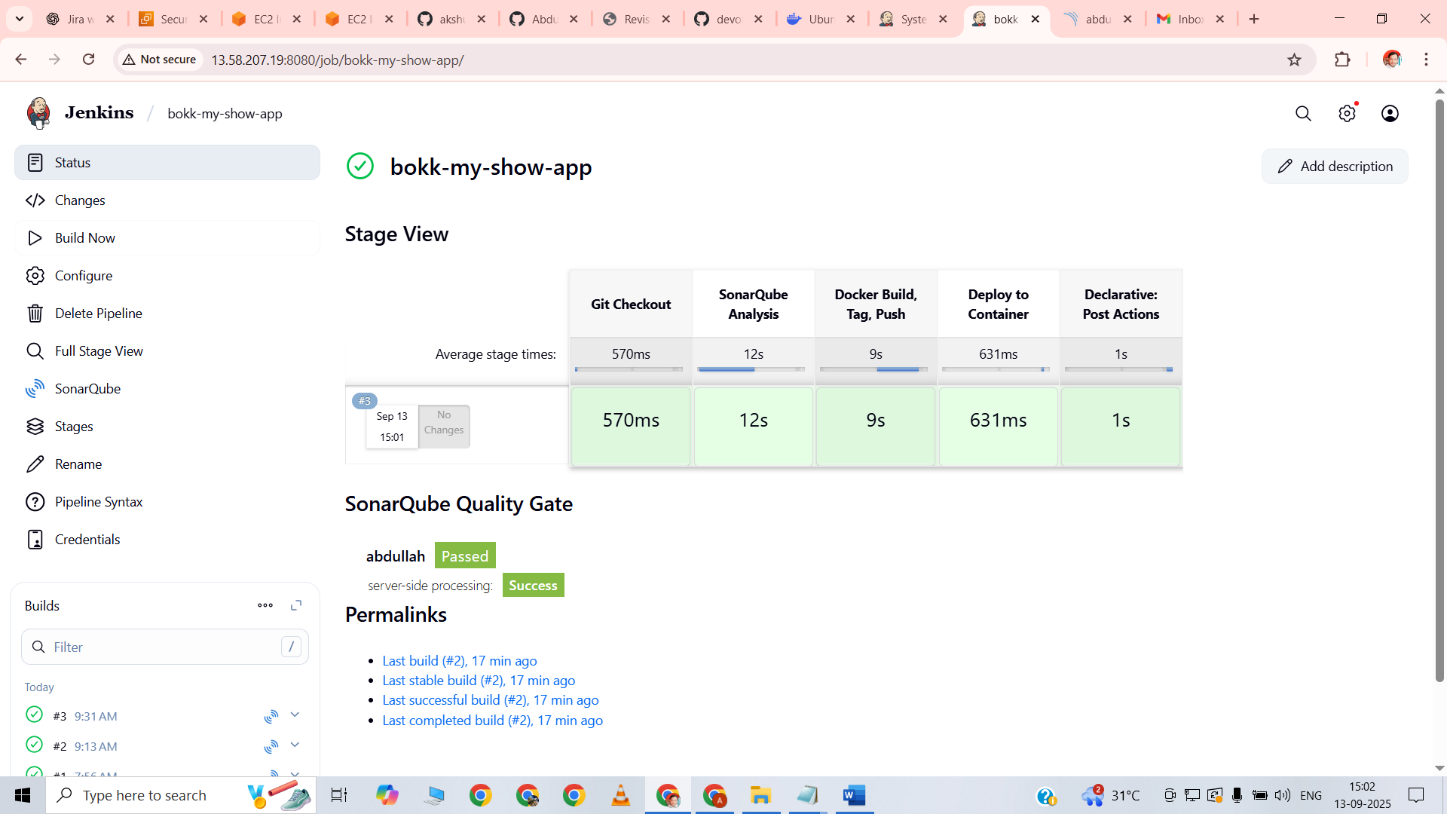
2.. GitHub PR link.

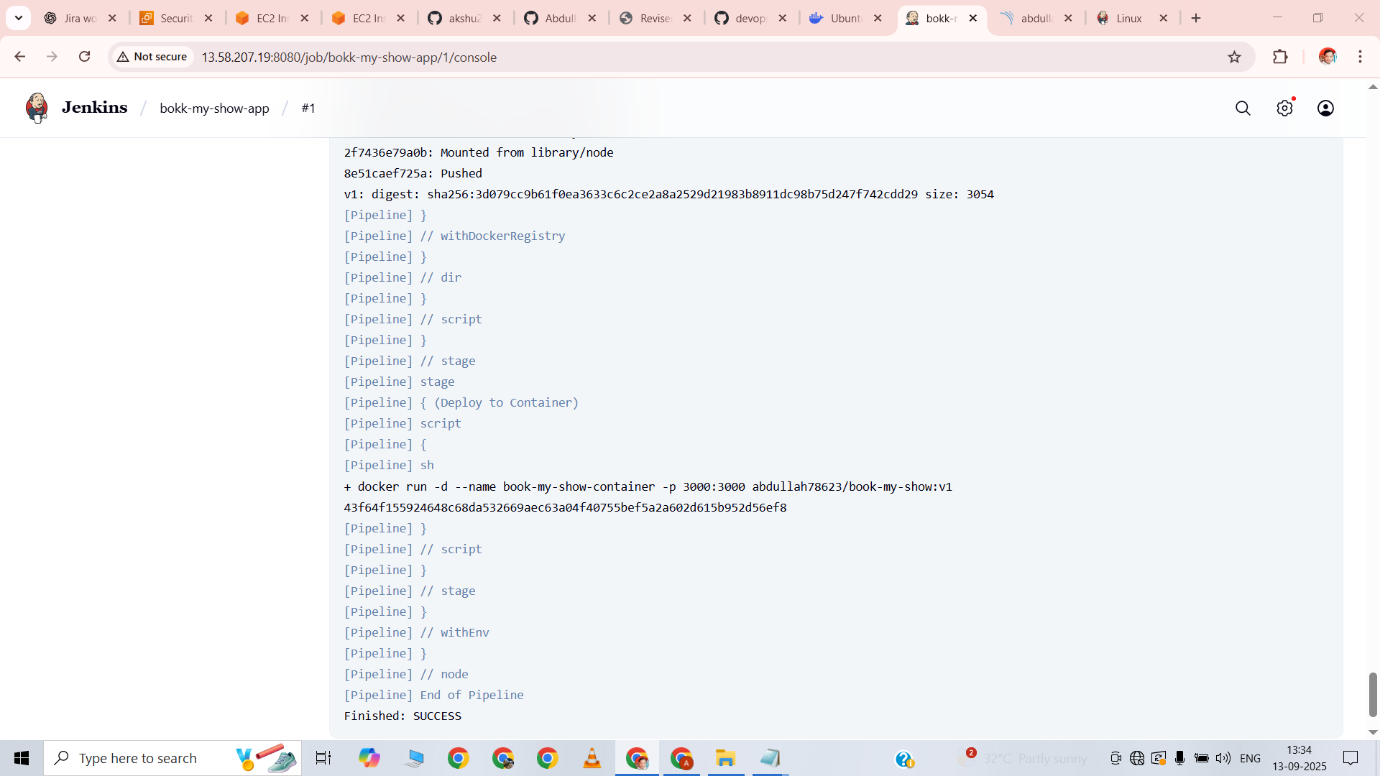
<https://github.com/akshu20791/Book-My-Show/pull/72>

Changed the Dockerfile to refine container configuration  
Created a docker-compose.yml file to run multiple services together  
Verified that the application and supporting services worked correctly with docker-compose

3.. Jenkins pipeline execution logs.







The Jenkins pipeline was executed successfully. All stages including Git Checkout, SonarQube Analysis, Docker Build & Push, and Deployment passed without errors. The console output displayed the final message “Finished: SUCCESS”, confirming the end-to-end pipeline execution.

4.1--Jenkinsfile-file-1-Deployment successfully done

pipeline {

agent any

environment {

SCANNER\_HOME = tool 'sonar-scanner'

}

stages {

stage('Git Checkout') {

steps {

git branch: 'main', url: 'https://github.com/AbdullahGhous78623/bms-app.git'

}

}

stage('SonarQube Analysis') {

steps {

withSonarQubeEnv('sonar') {

sh """

$SCANNER\_HOME/bin/sonar-scanner \

-Dsonar.projectKey=abdullah \

-Dsonar.projectName=abdullah \

-Dsonar.sources=.

"""

}

}

}

stage('Docker Build, Tag, Push') {

steps {

script {

dir('bookmyshow-app') {

withDockerRegistry(credentialsId: 'docker-cred', toolName: 'docker') {

sh 'docker build -t book-my-show:v1 .'

sh 'docker tag book-my-show:v1 abdullah78623/book-my-show:v1'

sh 'docker push abdullah78623/book-my-show:v1'

}

}

}

}

}

stage('Deploy to Container') {

steps {

script {

sh 'docker run -d --name book-my-show-container -p 3000:3000 abdullah78623/book-my-show:v1'

}

}

}

}

}

4.2-Jenkins—file-2—with email message of success message

pipeline {

agent any

environment {

SCANNER\_HOME = tool 'sonar-scanner'

}

stages {

stage('Git Checkout') {

steps {

git branch: 'main', url: 'https://github.com/AbdullahGhous78623/bms-app.git'

}

}

stage('SonarQube Analysis') {

steps {

withSonarQubeEnv('sonar') {

sh """

${SCANNER\_HOME}/bin/sonar-scanner \\

-Dsonar.projectKey=abdullah \\

-Dsonar.projectName=abdullah \\

-Dsonar.sources=.

"""

}

}

}

stage('Docker Build, Tag, Push') {

steps {

script {

dir('bookmyshow-app') {

withDockerRegistry(credentialsId: 'docker-cred', toolName: 'docker') {

sh 'docker build -t book-my-show1:v1 .'

sh 'docker tag book-my-show1:v1 abdullah78623/book-my-show1:v1'

sh 'docker push abdullah78623/book-my-show1:v1'

}

}

}

}

}

stage('Deploy to Container') {

steps {

script {

sh 'docker run -d --name book-my-show-container -p 3000:3000 abdullah78623/book-my-show1:v1"'

}

}

}

}

post {

success {

emailext (

subject: "SUCCESS: CI/CD Pipeline Passed",

body: """

Hi Abdullah,

✅ The Jenkins pipeline executed successfully.

- Code checkout ✅

- SonarQube Quality Gate ✅

- Docker Build & Push ✅

- Deployment ✅

You can check SonarQube report here:

http://13.58.207.19:9000/dashboard?id=abdullah&selectedTutorial=local

username: admin, password: 1234

Regards,

Jenkins

""",

to: 'abdullah632.hitit@gmail.com'

)

}

failure {

emailext (

subject: "FAILED: CI/CD Pipeline",

body: """

Hi Abdullah,

❌ The Jenkins pipeline has failed.

Please review the Jenkins console logs and fix the issue.

SonarQube Dashboard:

http://13.58.207.19:9000/dashboard?id=abdullah&selectedTutorial=local

username: admin, password: 1234

Regards,

Jenkins

""",

to: 'abdullah632.hitit@gmail.com'

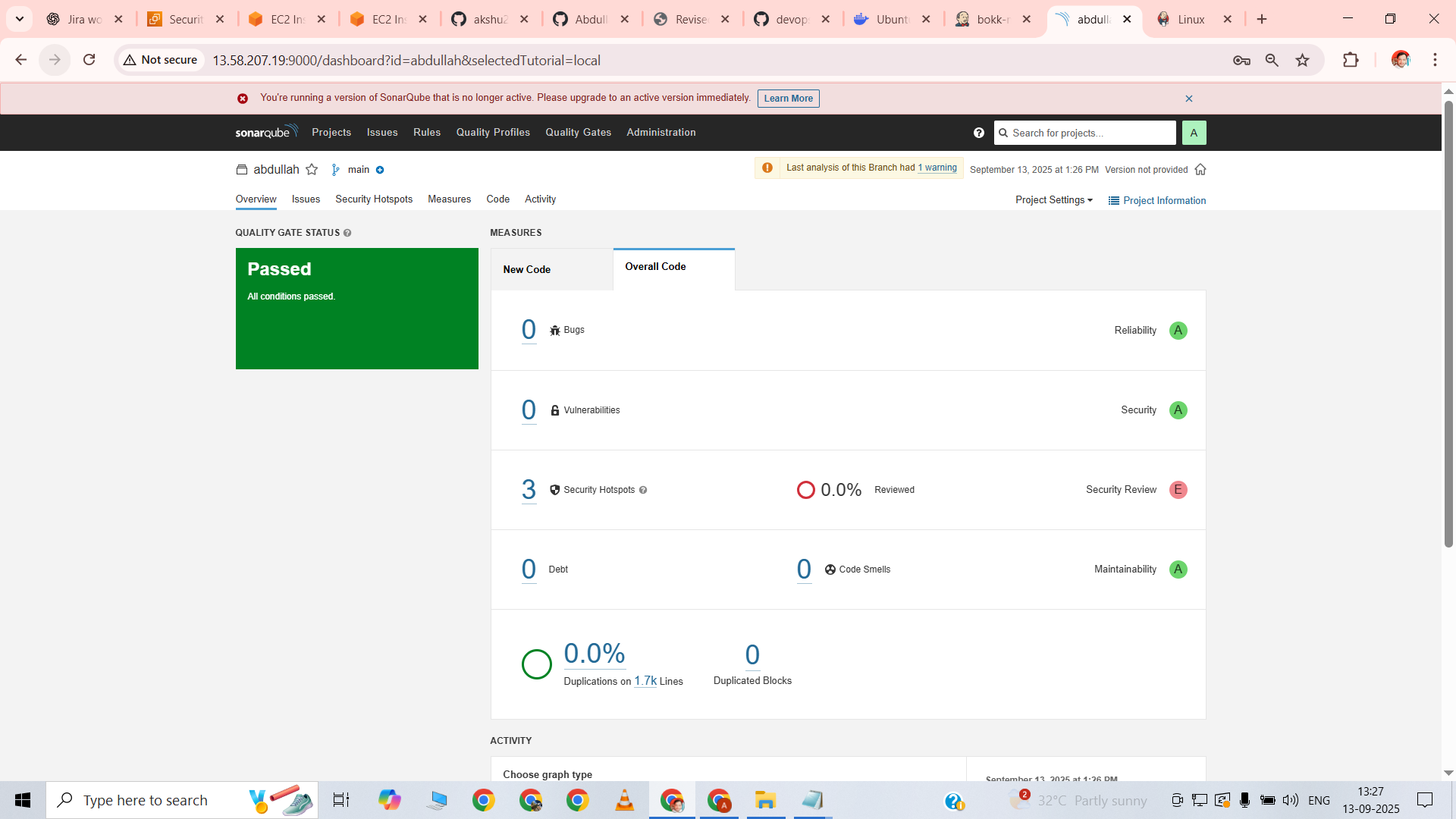
)

}

}

}

5. . SonarQube Quality Gate report.



SonarQube analysis was successfully completed, and the project passed the Quality Gate with no major code issues. This ensured code quality and maintainability before deployment.

6. DockerHub repository link.

<https://hub.docker.com/repository/docker/abdullah78623/book-my-show/tags>

docker pull abdullah78623/book-my-show:v1

The Docker image book-my-show:v1 was successfully built and pushed to Docker Hub, making it available for deployment in containers or Kubernetes.

7. Dockerfile

FROM node:18

WORKDIR /app

COPY package.json package-lock.json ./

RUN npm install chokidar@3 postcss@8.4.21 postcss-safe-parser@6.0.0 --legacy-peer-deps

RUN npm install --legacy-peer-deps

COPY . .

EXPOSE 3000

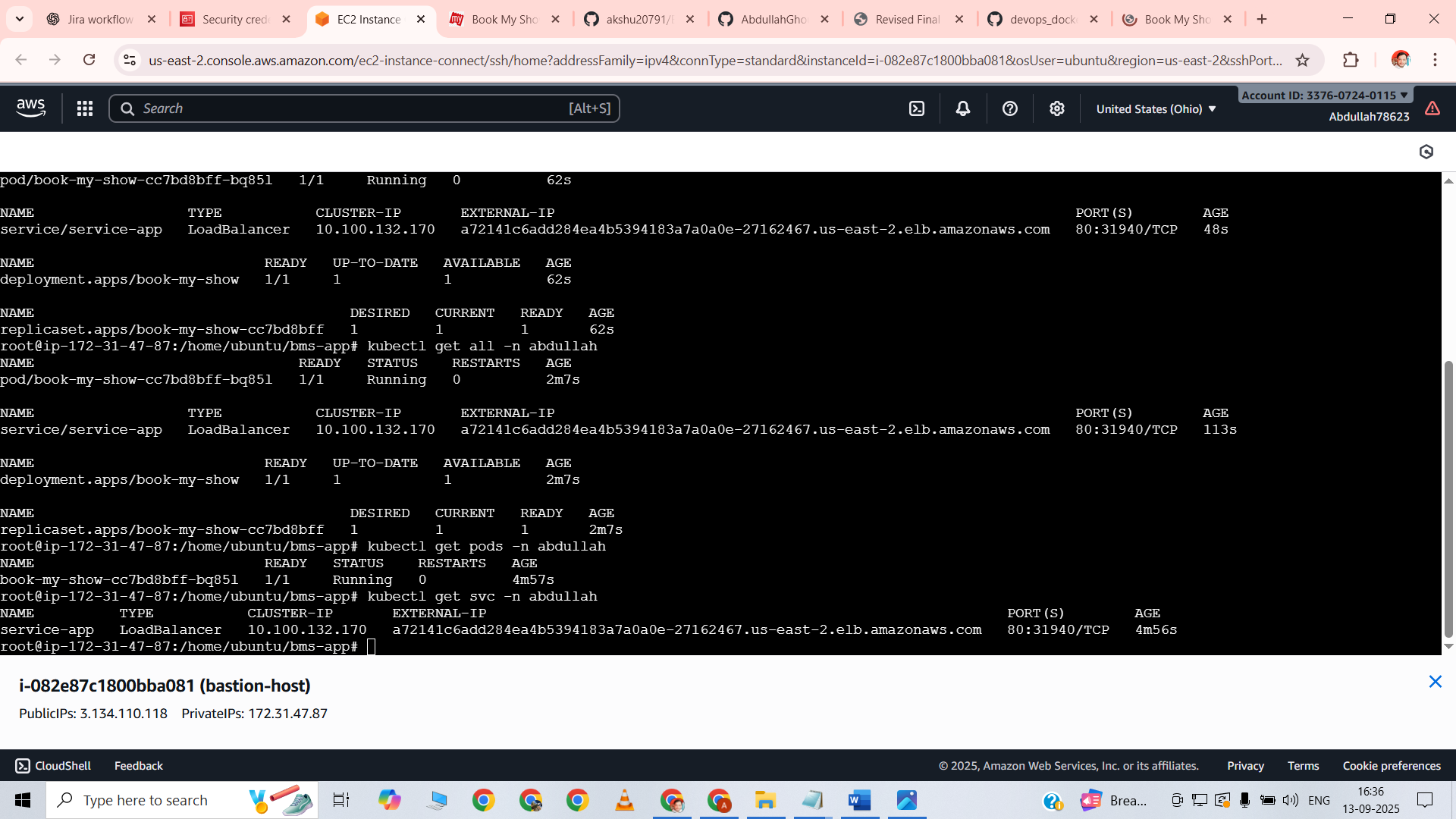
ENV NODE\_OPTIONS=--openssl-legacy-provider

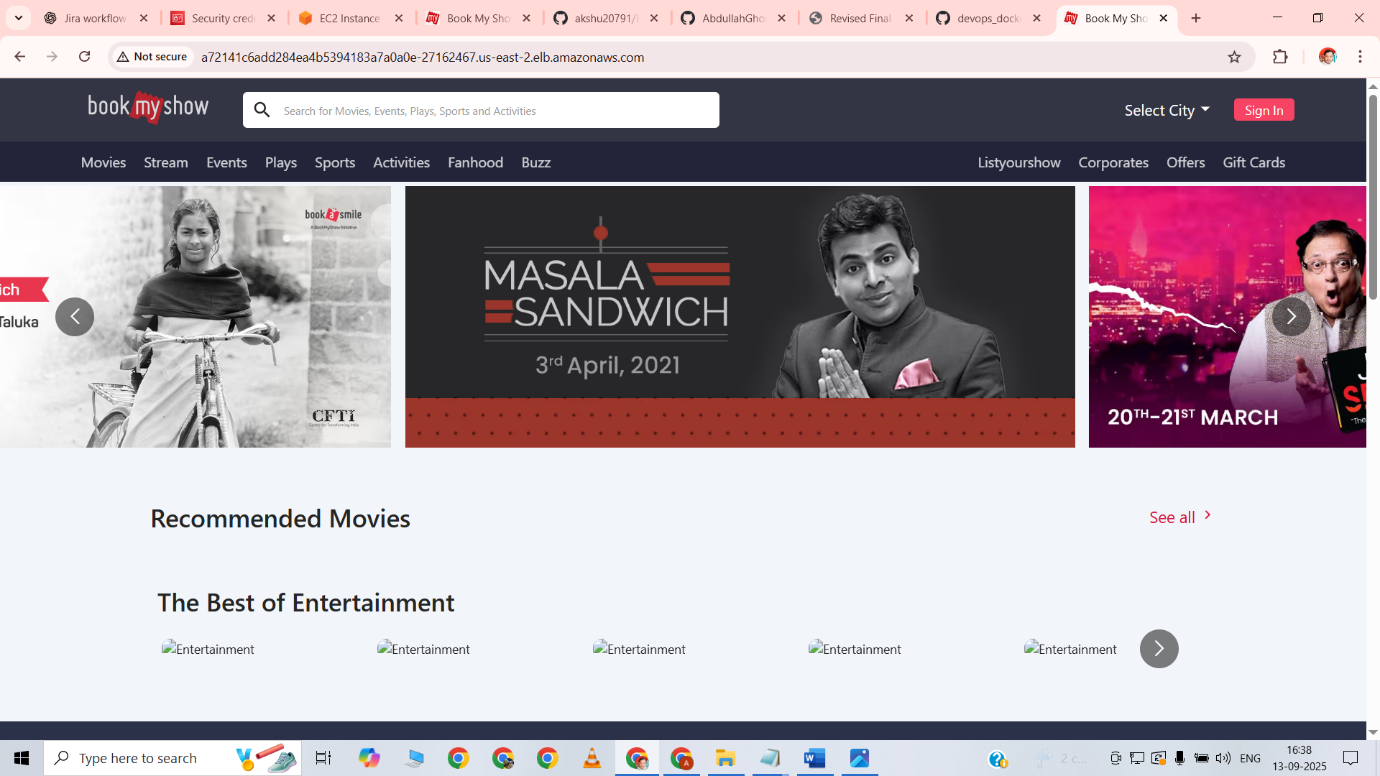
ENV PORT=3000

CMD ["npm", "start"]

The Dockerfile was created to containerize the Book-My-Show React application. It installs dependencies, sets the working directory, exposes port 3000, and defines the start command for the container.

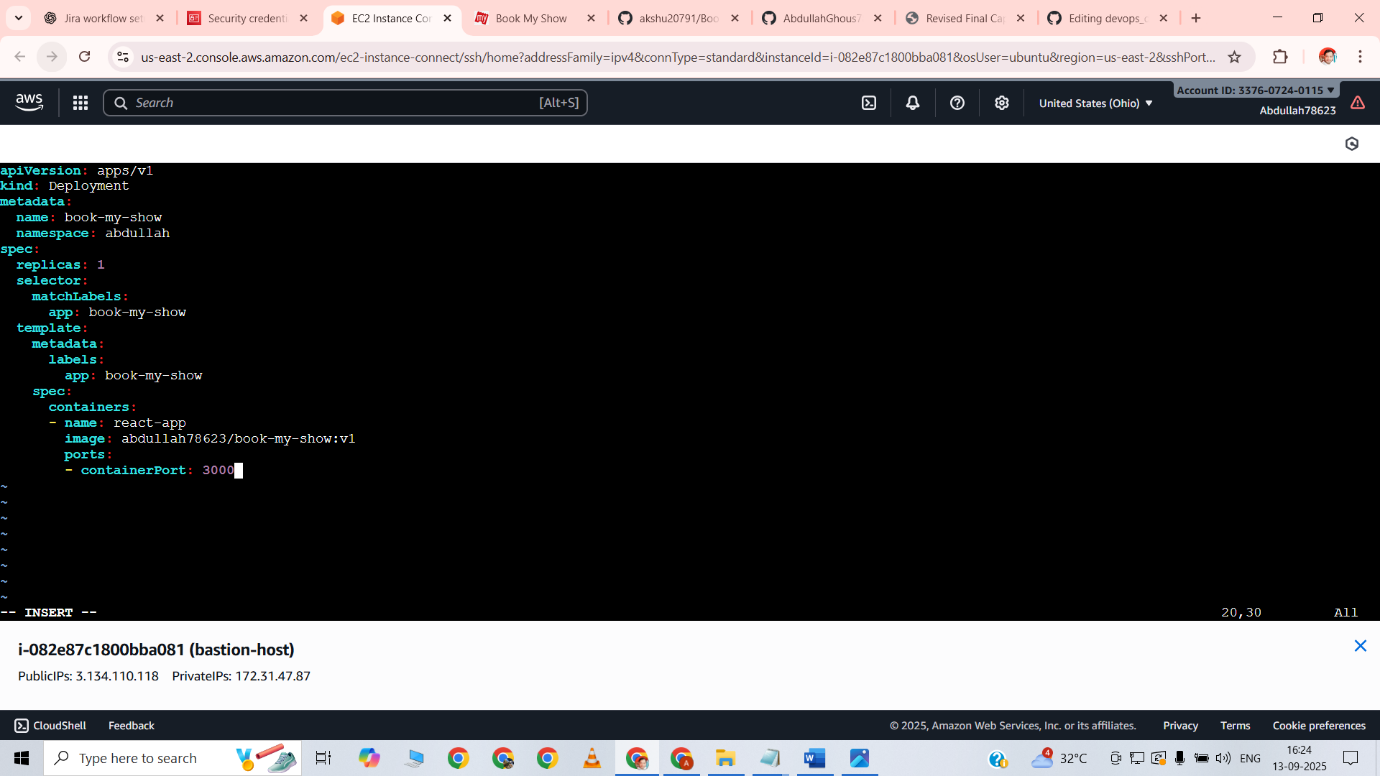
8. . EKS deployment proof





Commands executed:  
kubectl get all -n abdullah  
kubectl get pods -n abdullah  
kubectl get svc -n abdullah

**Observation:**  
All pods and services are running successfully. The Book-My-Show application is accessible through the LoadBalancer URL in a web browser.

8.1 . Kubernetes manifest files – deployment.yaml 

apiVersion: apps/v1

kind: Deployment

metadata:

name: book-my-show

namespace: abdullah

spec:

replicas: 1

selector:

matchLabels:

app: book-my-show

template:

metadata:

labels:

app: book-my-show

spec:

containers:

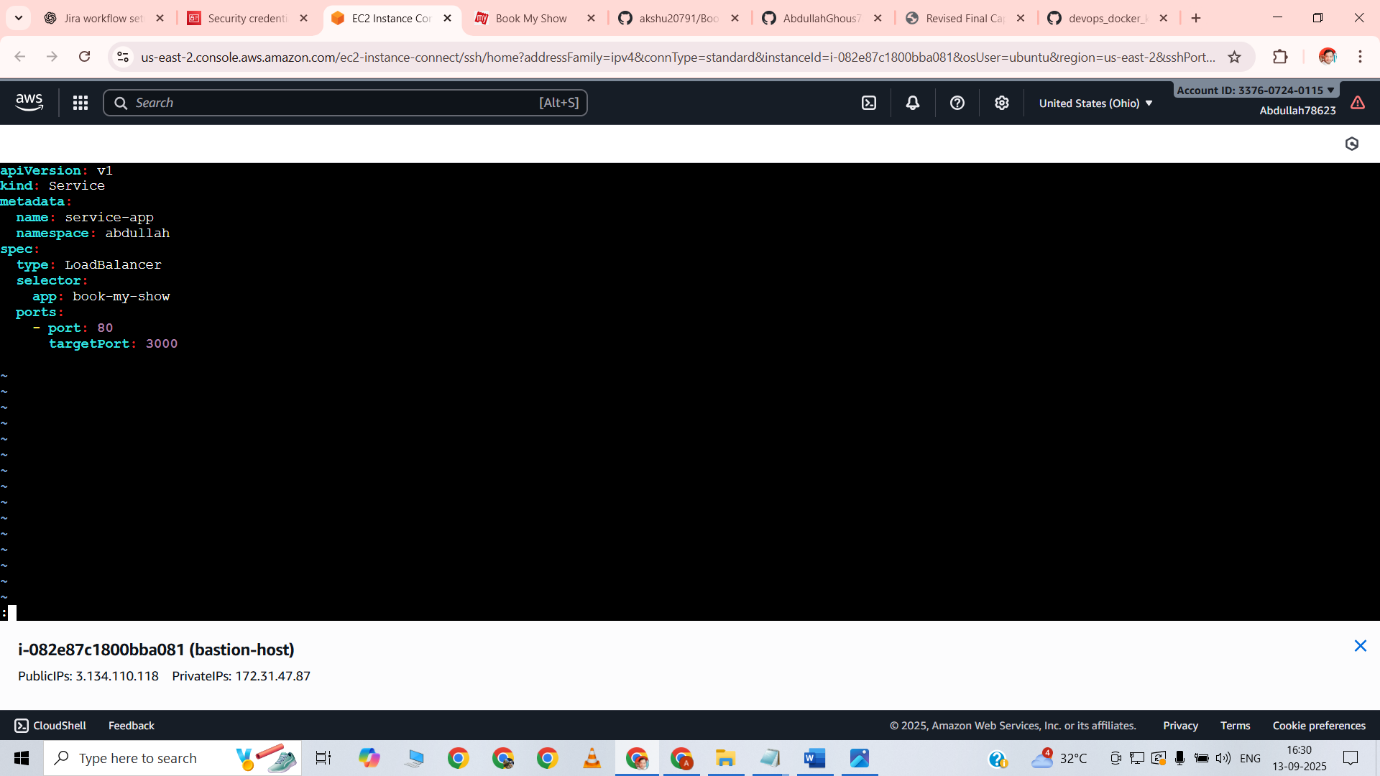
- name: react-app

image: abdullah78623/book-my-show:v1

ports:

- containerPort: 3000

8.1 . Kubernetes manifest files –service.yaml



apiVersion: v1

kind: Service

metadata:

name: service-app

namespace: abdullah

spec:

type: LoadBalancer

selector:

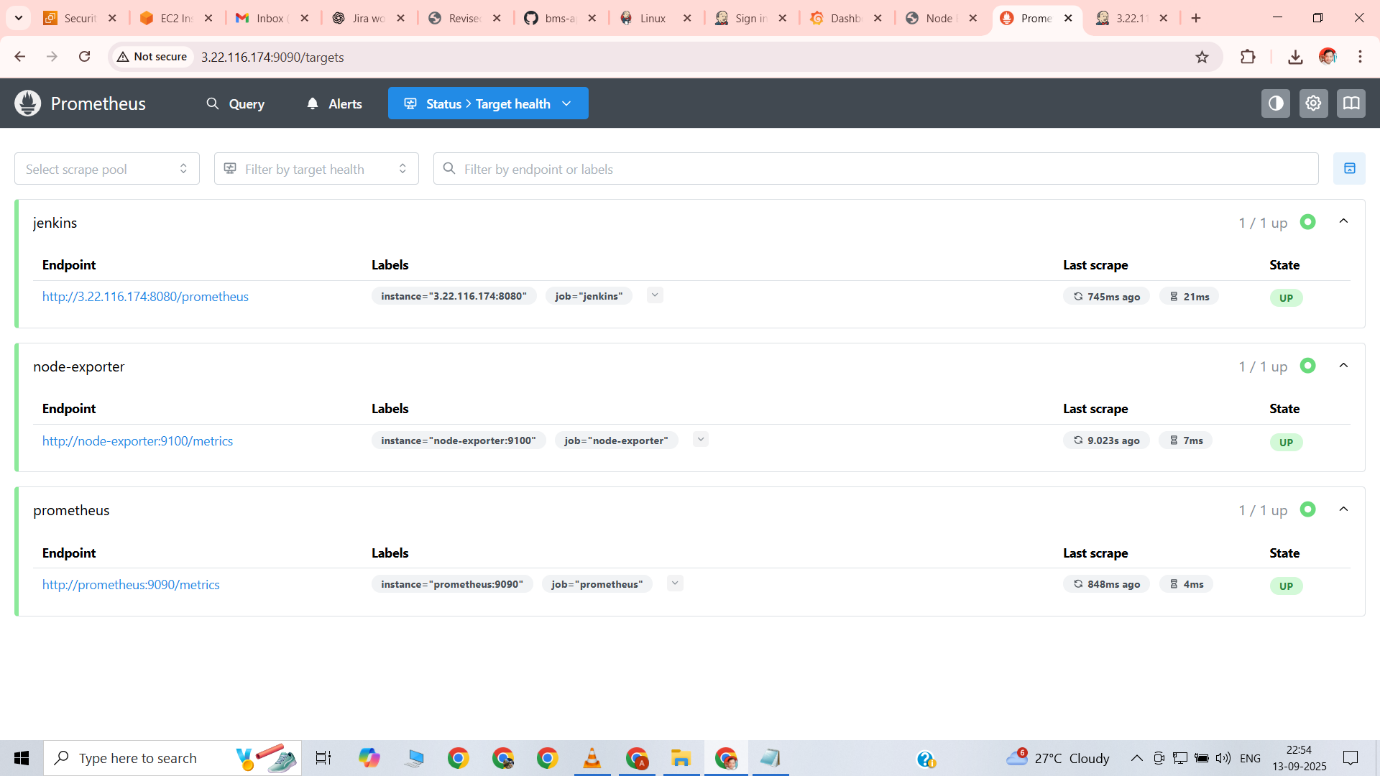
app: book-my-show

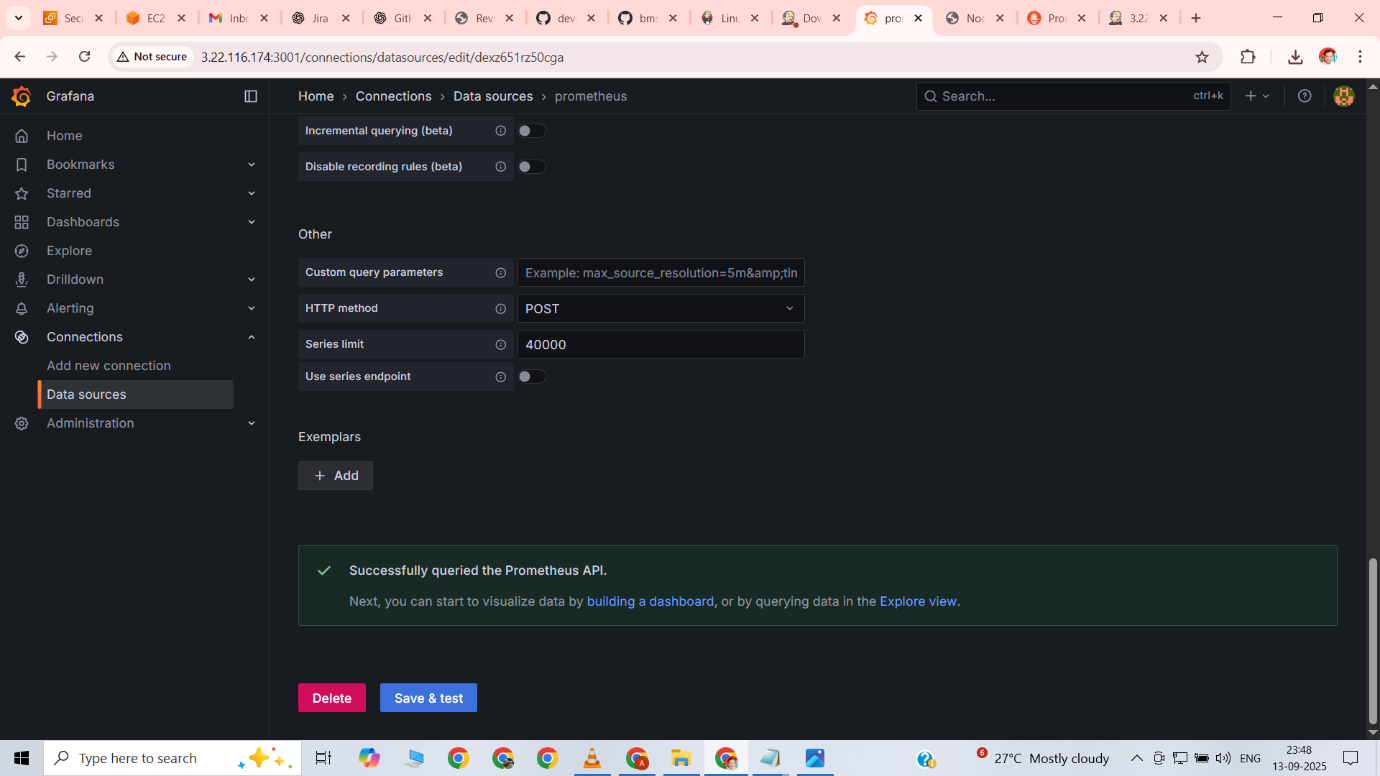
ports:

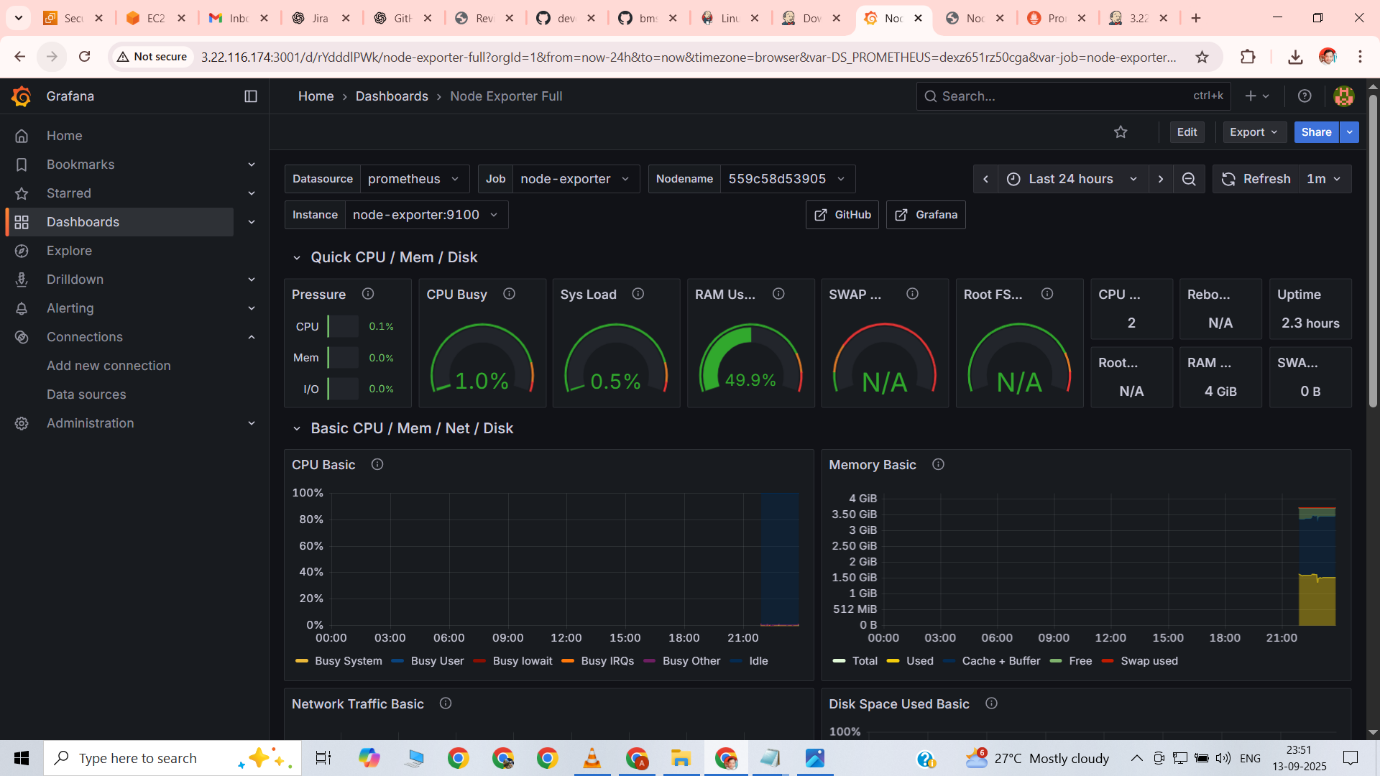
- port: 80

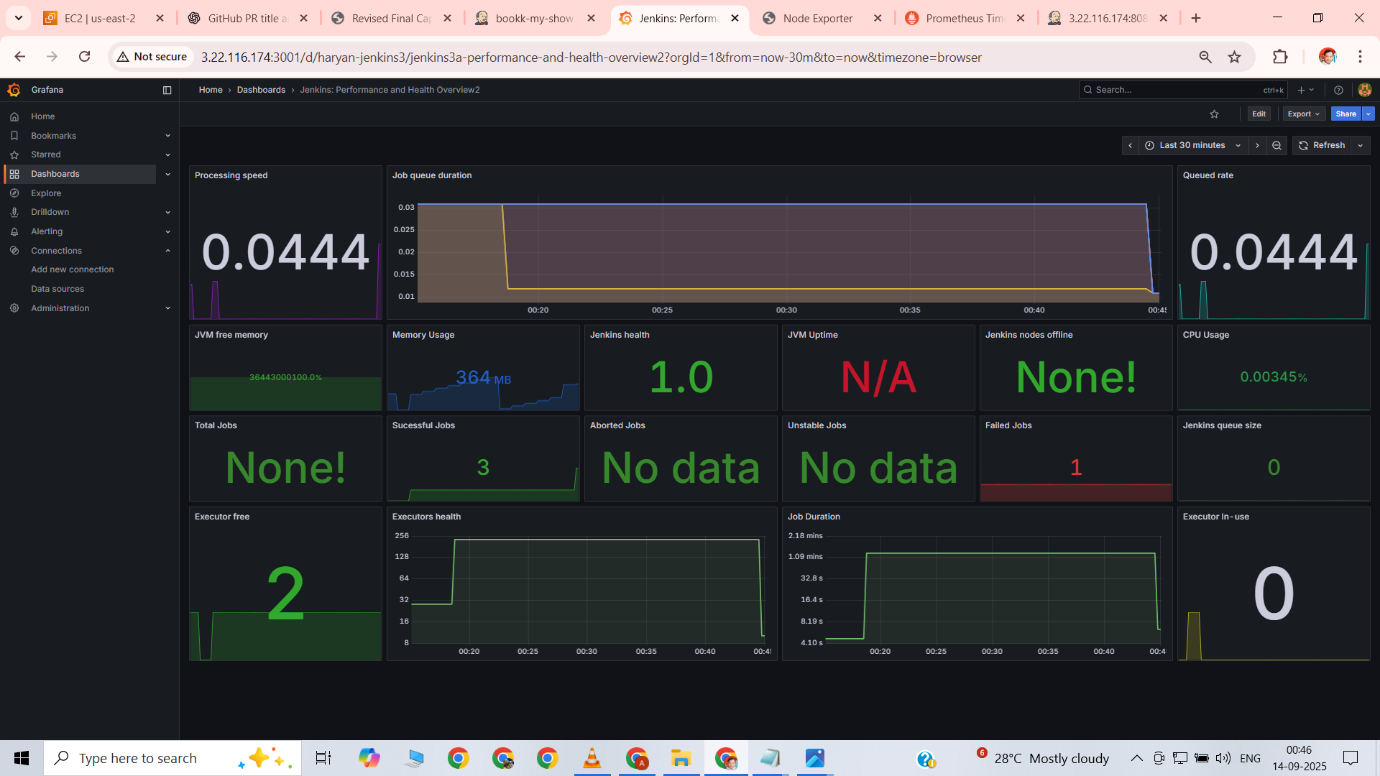
targetPort: 3000

9. Prometheus & Grafana screenshots



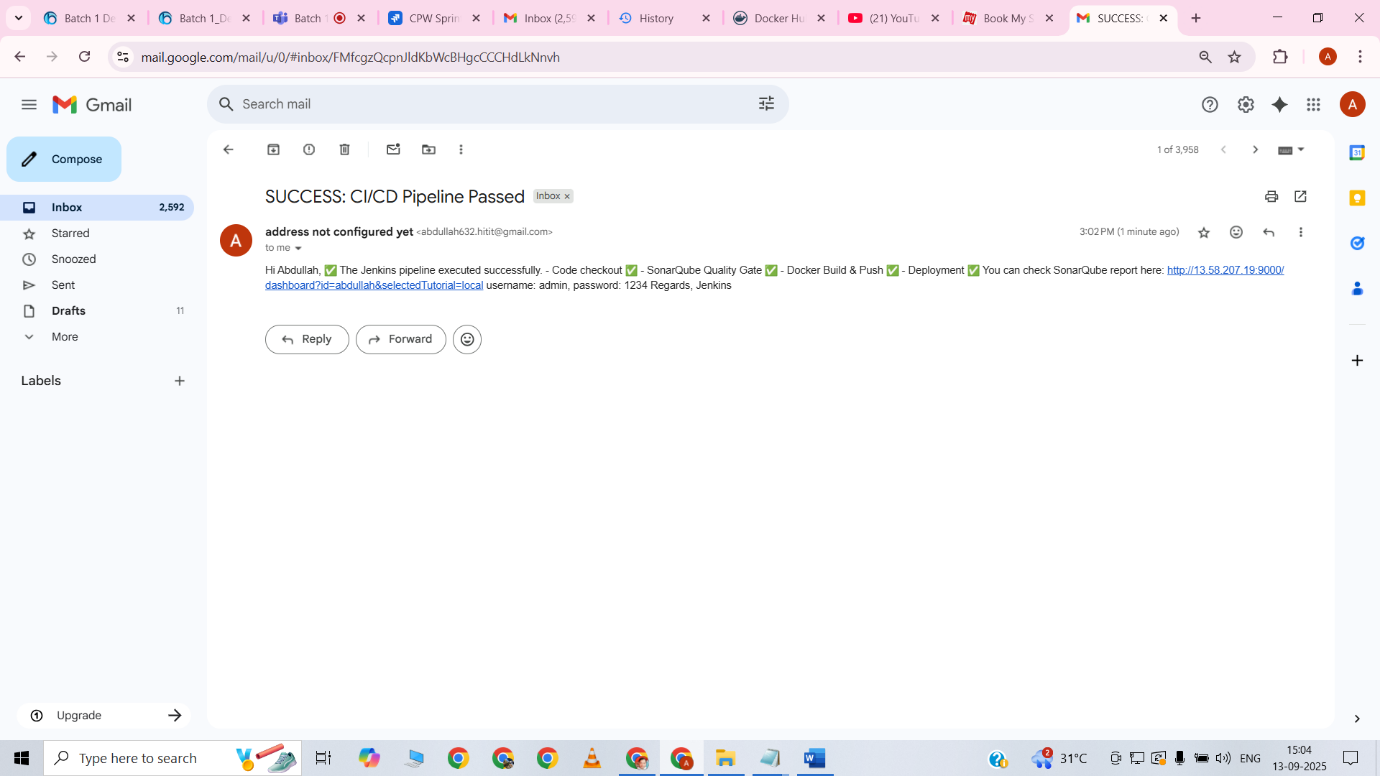






Prometheus and Node Exporter were successfully set up to collect system metrics. Jenkins metrics were integrated into Prometheus, and Grafana dashboards were configured to visualize Node health and CI/CD performance in real time.

10. Email notification screenshot.



Jenkins was configured to send email notifications upon pipeline completion. A success notification was received in the registered email, confirming that all CI/CD stages (Git checkout, SonarQube analysis, Docker build & push, and deployment) executed successfully.