Capstone Project Report

on

DevOps Lifecycle Implementation for Book-My-Show Application

Submitted To

Mentor: Ms. Rashi Rana

Company: Wipro

Domain: DevOps
Date: 15/09/2025

Submitted By

Shaik Mohammad Waseem Akram

DevOps Batch_4

1. **Introduction:** This Capstone Project simulates a real-world **DevOps workflow** for the **Book-My-Show (BMS) application**. The goal is to implement a complete **CI/CD lifecycle** – from task management and code collaboration to building, deploying, and monitoring the application.

2. Tools & Technologies used:

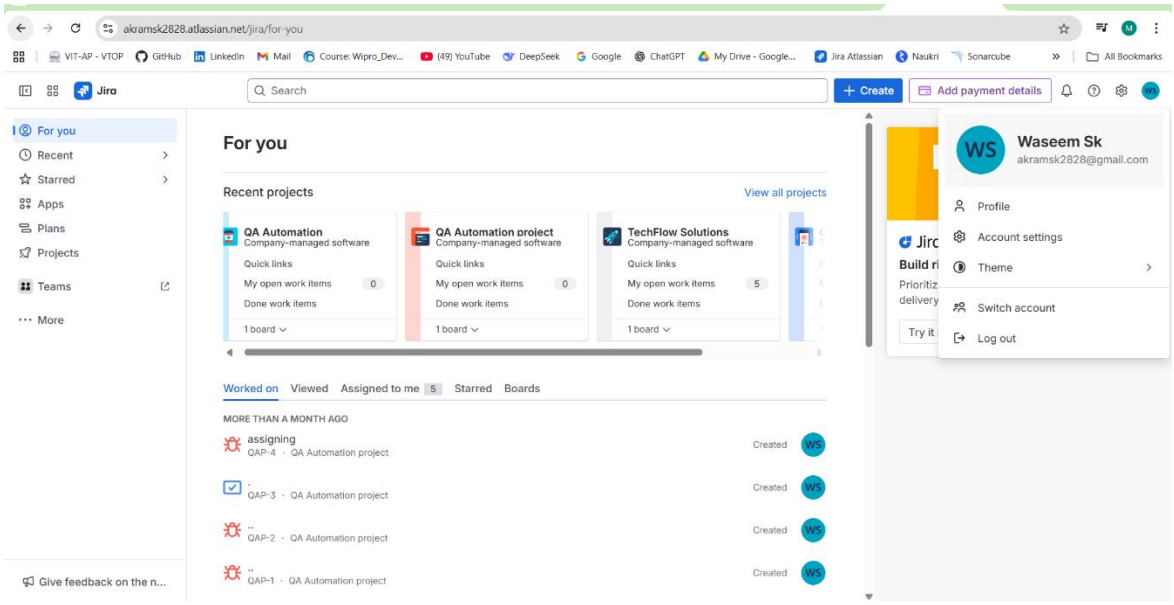
- **Project Management**: Jira
- **Version Control & Collaboration**: GitHub
- **CI/CD Automation**: Jenkins (Jenkinsfile)
- **Code Quality & Security**: SonarQube
- **Containerization**: Docker (Dockerfile)
- **Container Orchestration**: Kubernetes (EKS)
- **Monitoring & Observability**: Prometheus, Grafana
- **Notifications**: Jenkins Email Notifications
-

3. Table of Contents

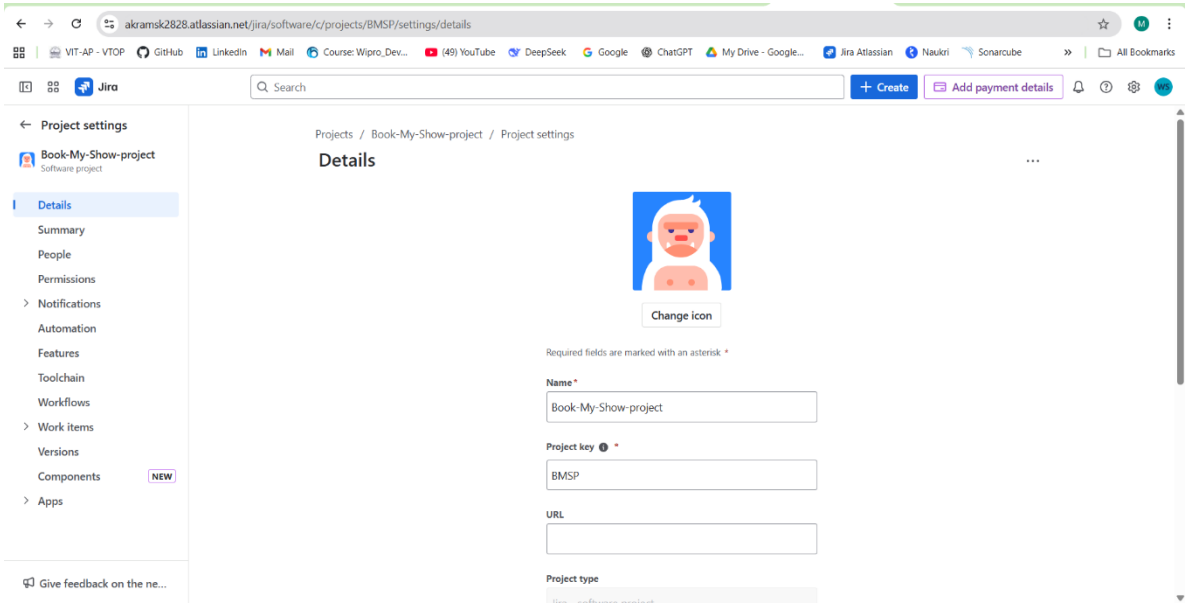
1. JIRA WORKFLOW

Jira: Jira is a project management tool used to track tasks across the DevOps lifecycle. It provides visibility into progress using a Kanban workflow (To Do → In Progress → Done). For this project, Jira helped in planning and tracking each phase — GitHub, Jenkins, Docker, Kubernetes, Monitoring, and Reporting.

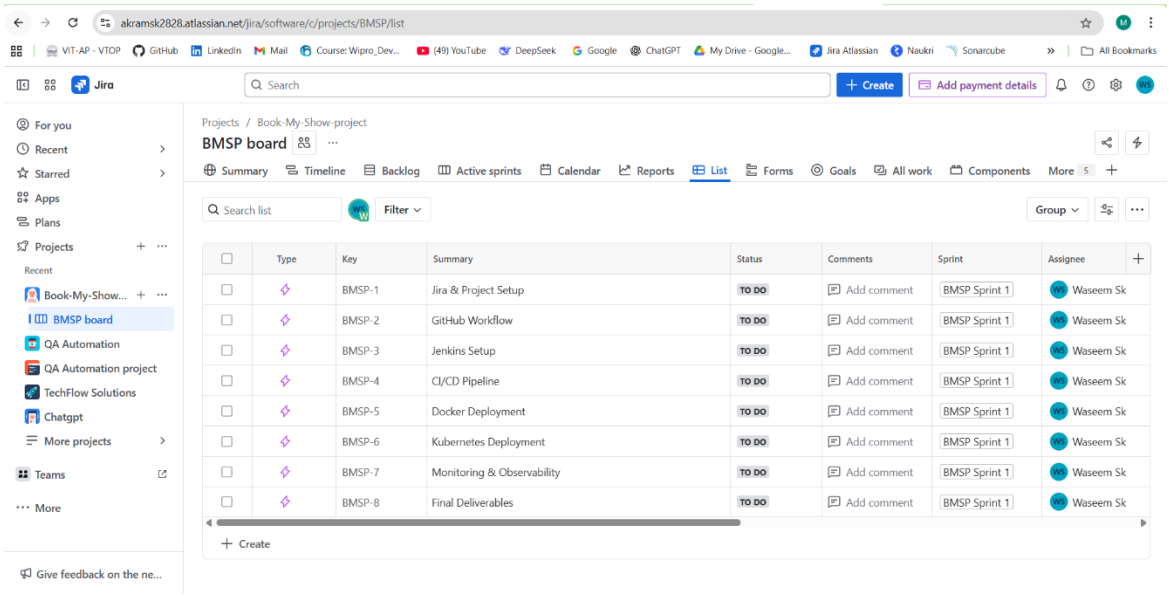
- **Login Page:** Logged into my JIRA Account



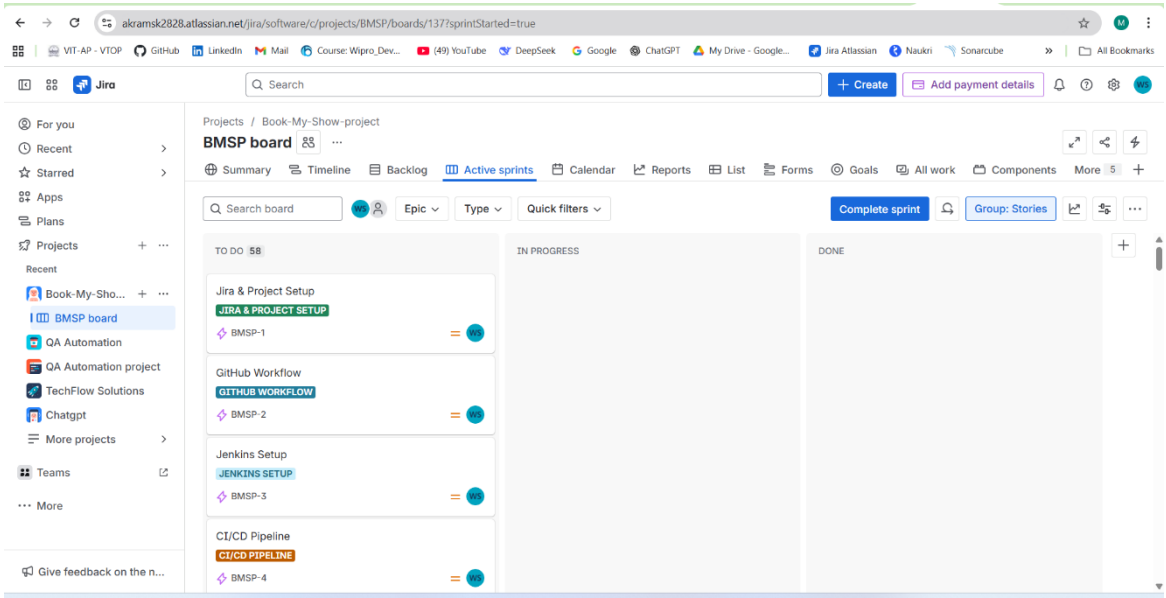
- **Project Settings:** Project settings page displaying the Book-My-Show project details with project key BMSP



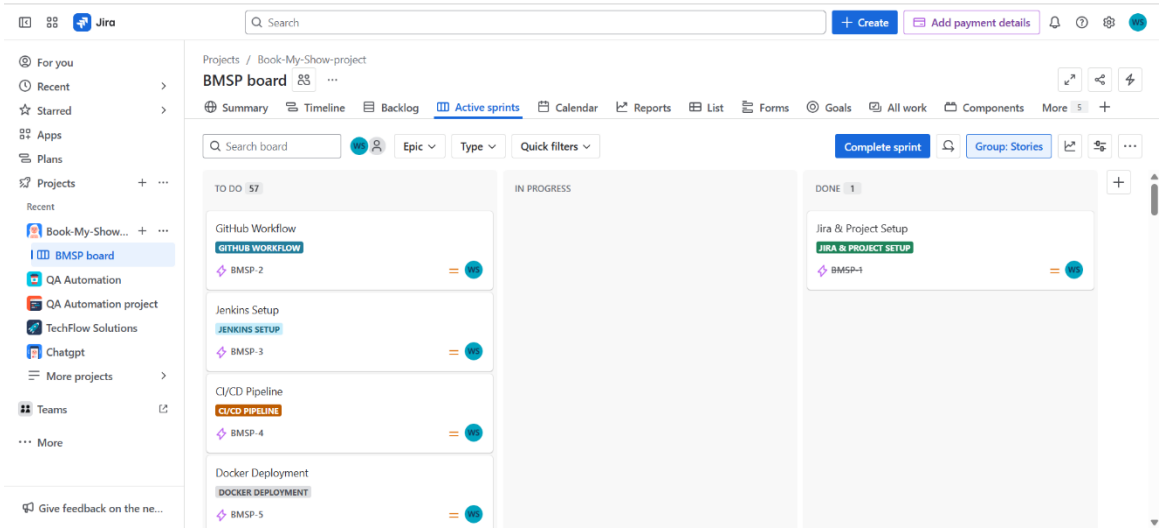
- **Task View List:** The Jira board in list view showing all major tasks such as Jira Setup, GitHub Workflow, Jenkins, CI/CD, Docker, Kubernetes, Monitoring, and Deliverables.



- **Active Sprint Board:** The Jira sprint board visualizing tasks under To Do, In Progress, and Done categories for workflow tracking.



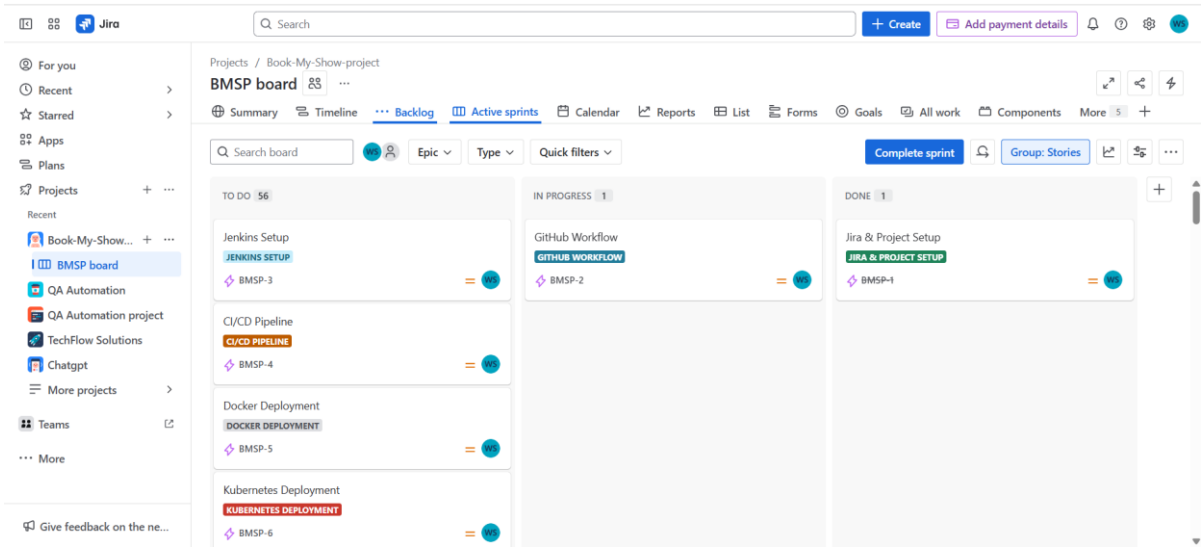
- **Task Completion:** I have completed the task of **Jira & Project Setup (BMSP-1)**, and it is now moved to the **Done** column as part of Step-1 in the capstone project.



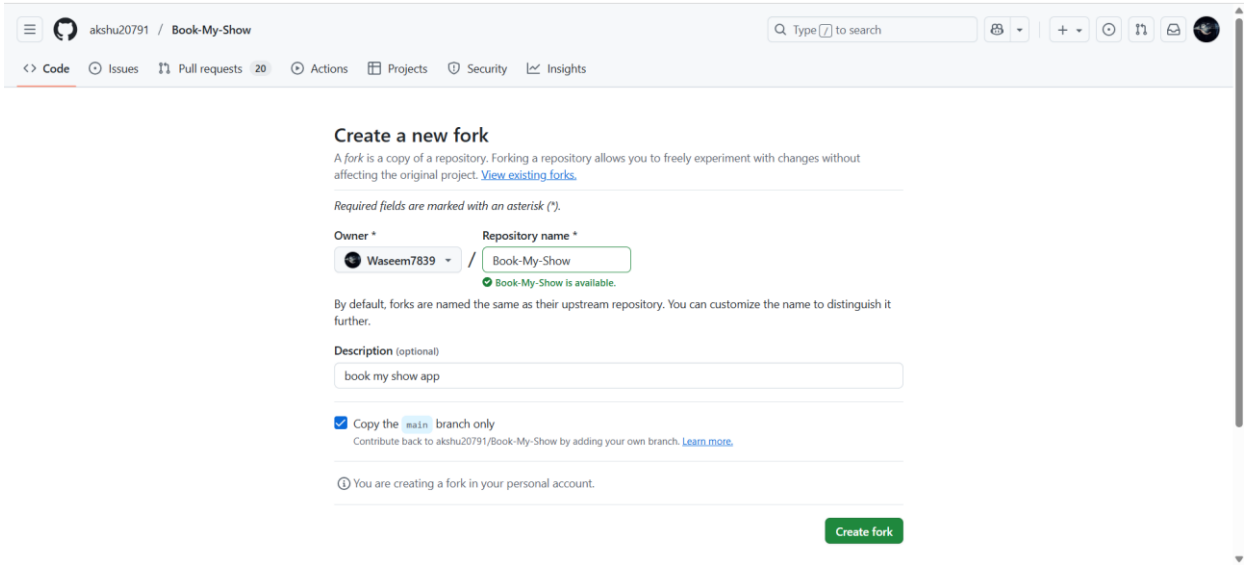
2. GITHUB WORKFLOW:

GitHub serves as the source code management system. A **branching strategy** was used:
main branch → stable production code
feature-* branches → new changes
Changes were merged using Pull Requests (PRs) after review.

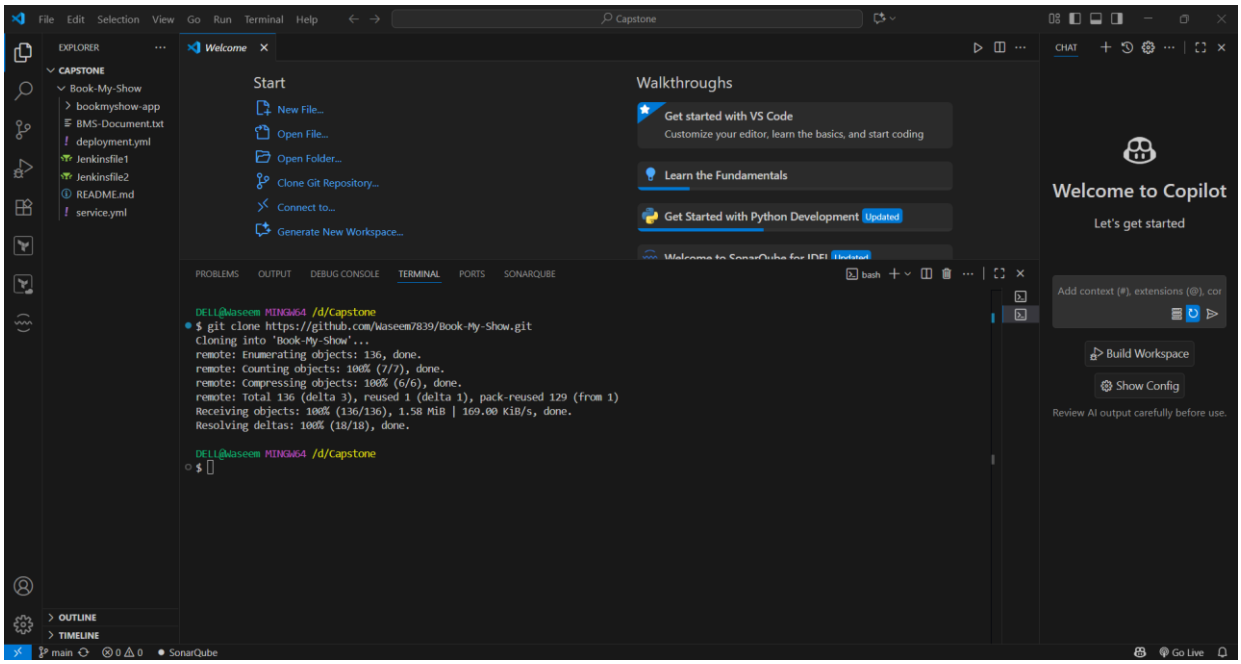
- **TO DO -→**



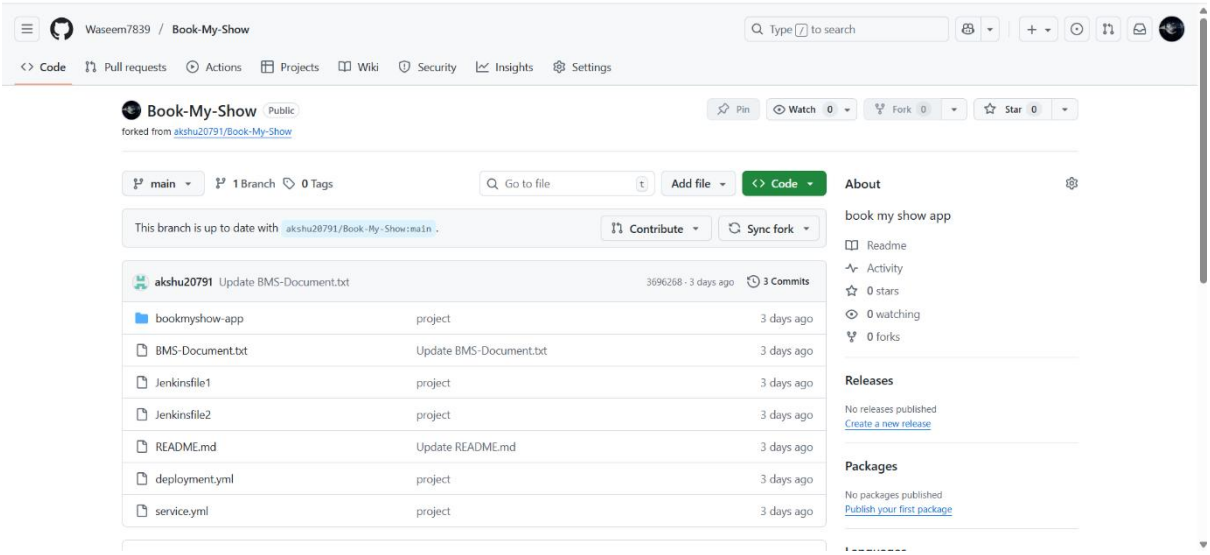
- **Forked Repository:** The original **Book-My-Show** repository was forked into the personal GitHub account.



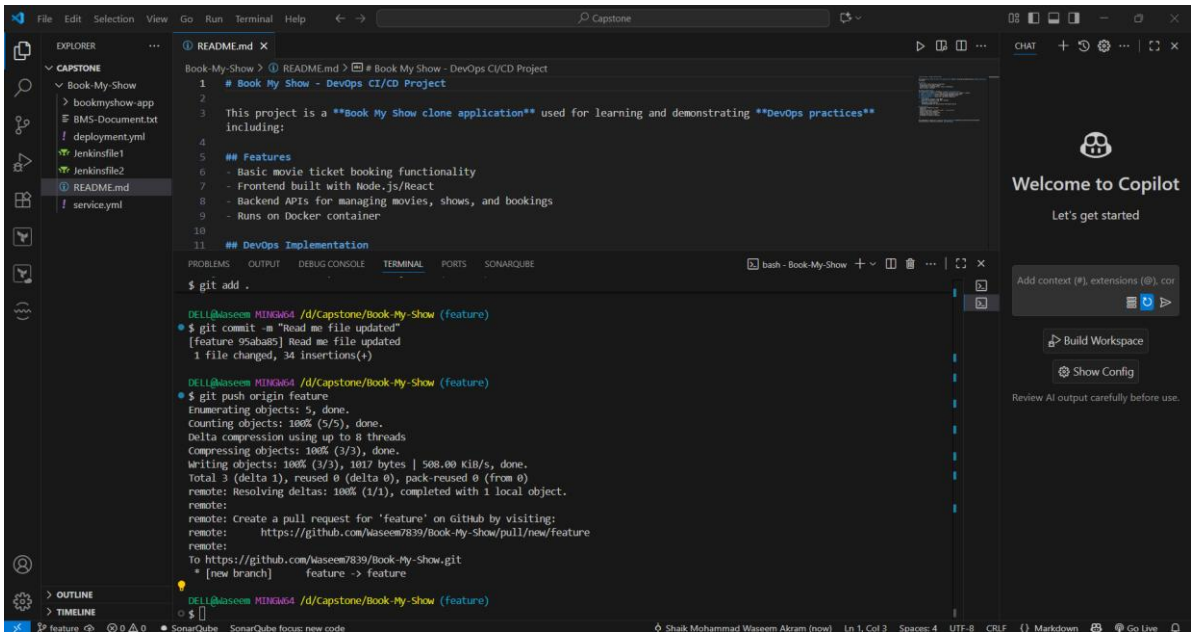
- **Cloned Repository:** The forked repository was cloned into the local system using VS Code.



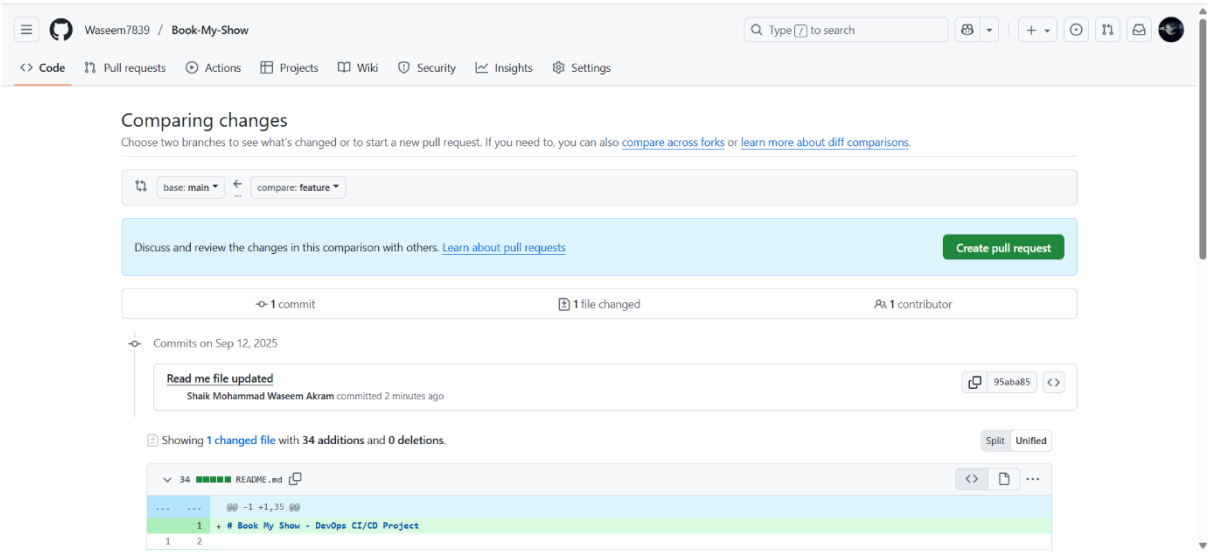
- **Repository Setup:** The forked repository contains files such as Jenkinsfile, deployment.yml, and service.yml



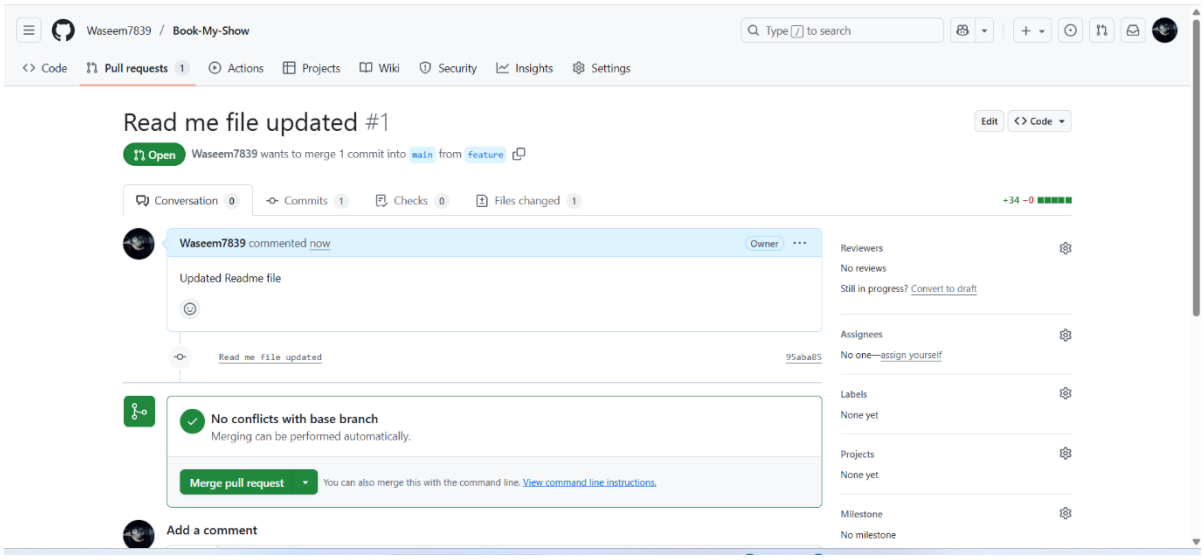
- **Feature Branch Update:** A new feature branch was created, and the README.md file was updated, committed, and pushed to GitHub.



- **Pull Request Creation:** A Pull Request (PR) was created from the feature branch to the main branch.

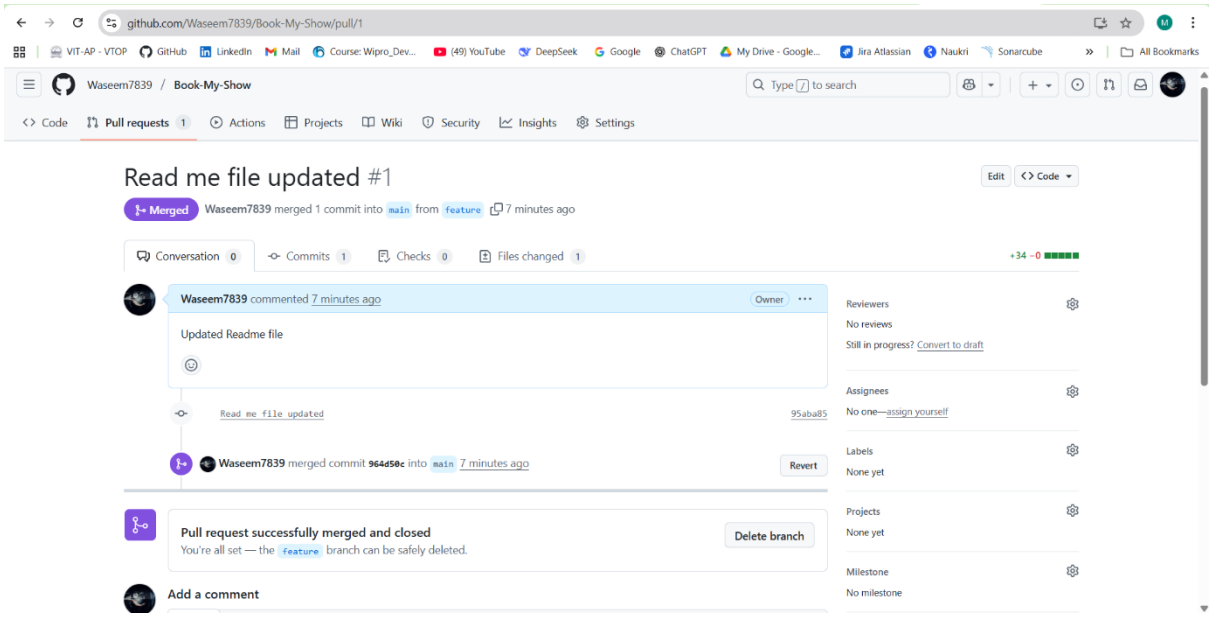


- **PR Merge:** The Pull Request was reviewed and successfully merged into the main branch.

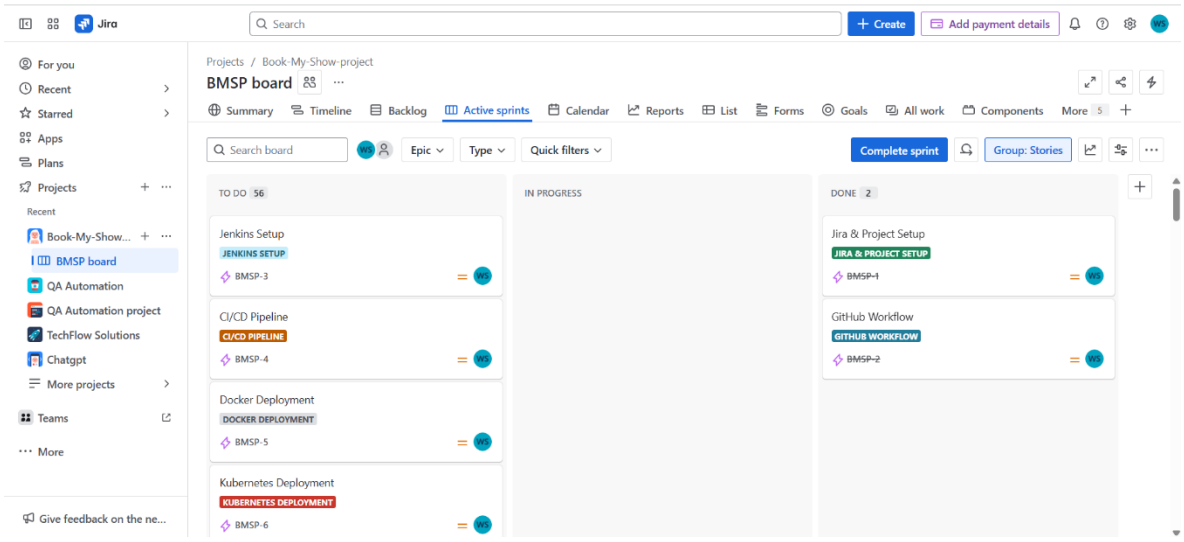


- **Merged PR confirmation.**

PR link: https://github.com/Waseem7839/Book-My-Show/pull/1



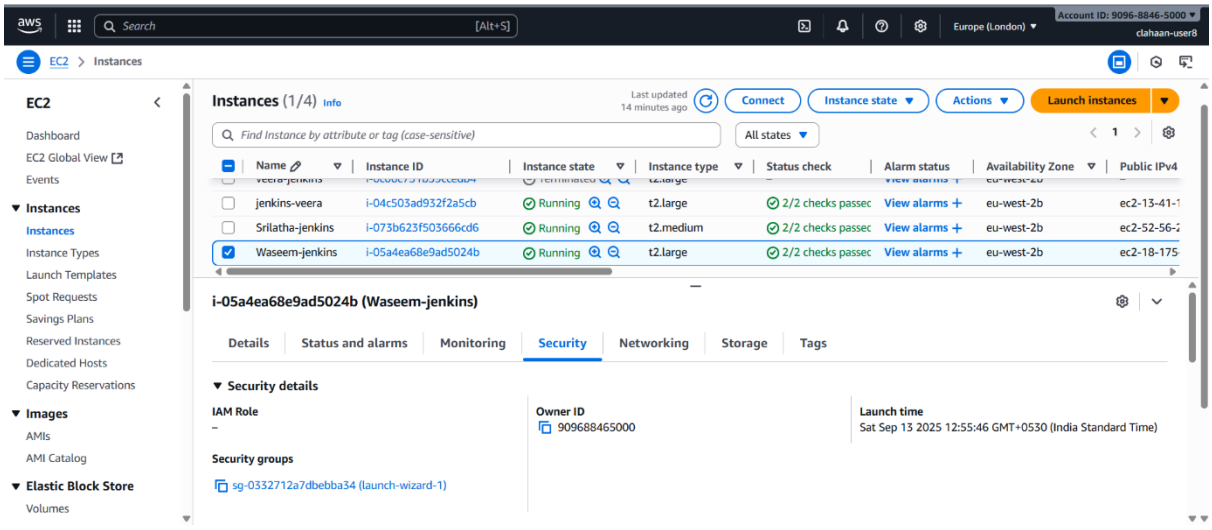
- **Task Completion:** The **GitHub Workflow (BMSP-2)** task has been successfully completed and is moved to the **Done** column in Jira.



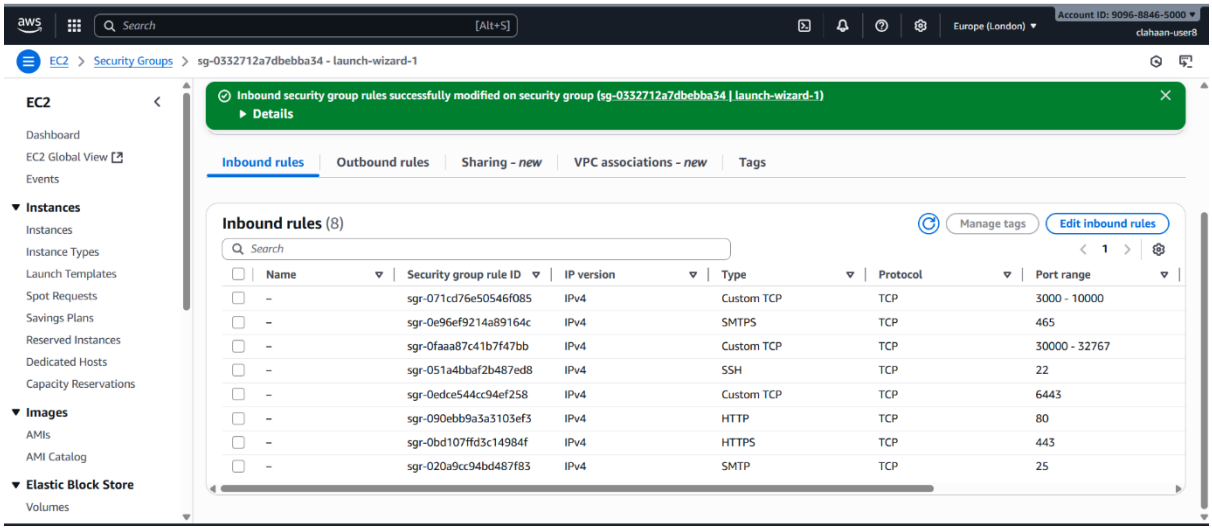
3. JENKINS SETUP

Jenkins is used to automate the CI/CD pipeline, integrating code builds, tests, SonarQube scans, Docker builds, and Kubernetes deployments.

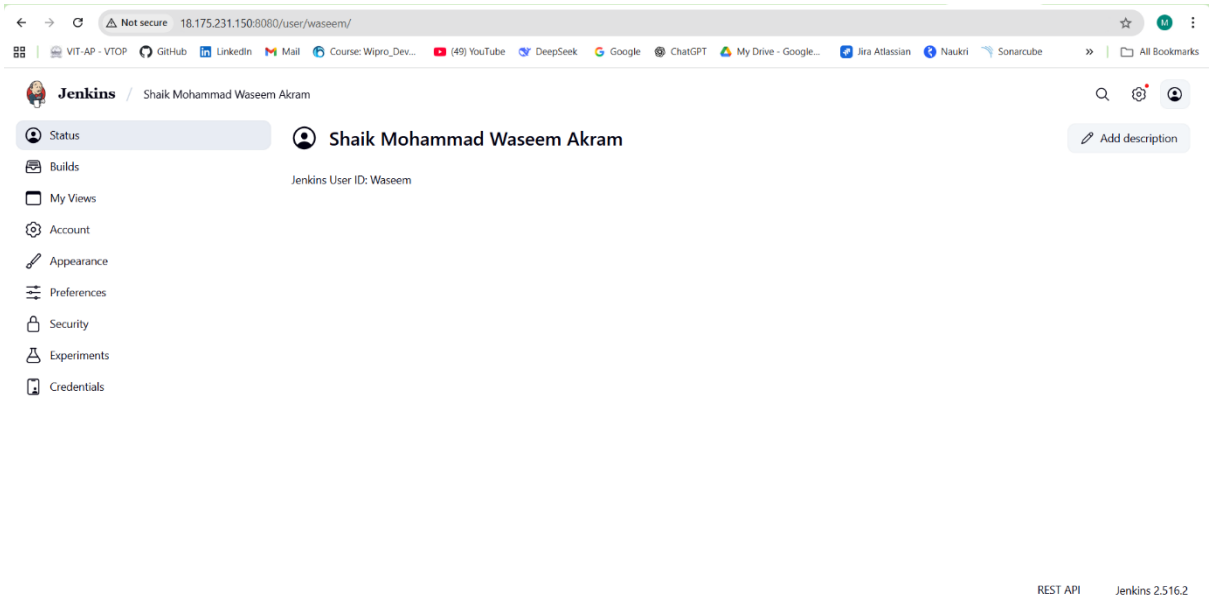
- Installed Jenkins on EC2
  - Launched an EC2 instance (Ubuntu).
  - Installed Jenkins, Java (JDK 17), and required dependencies.



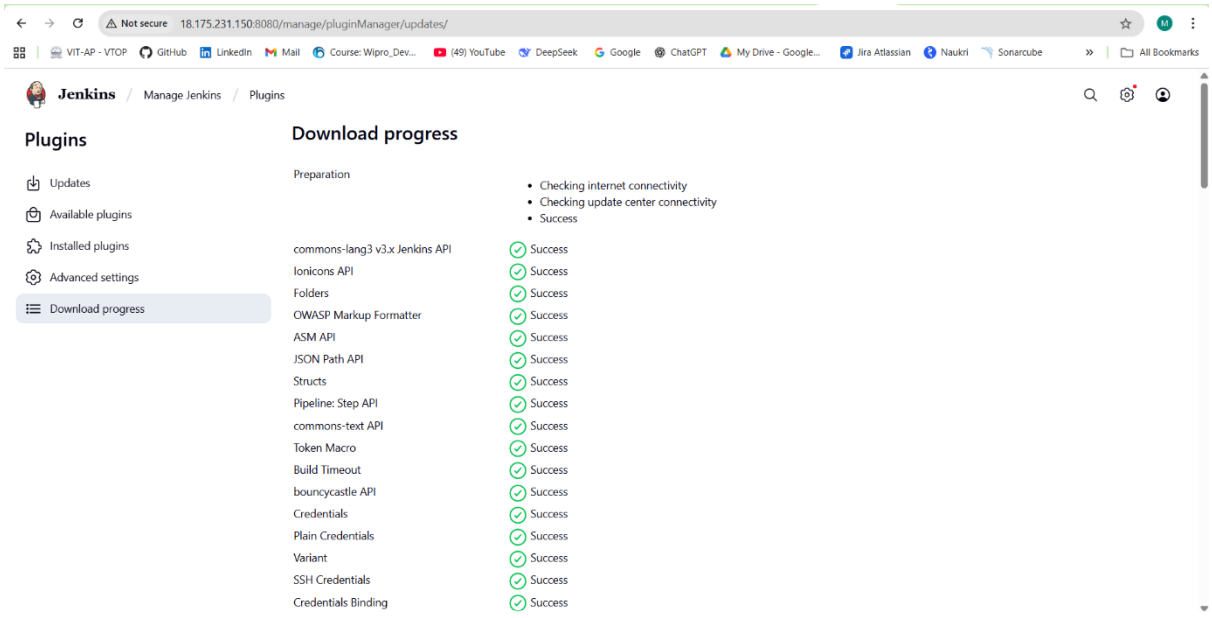
Ports



- **Jenkins Installation & Access:** Jenkins was installed on Ubuntu EC2 and accessed at http://18.175.231.150:8080



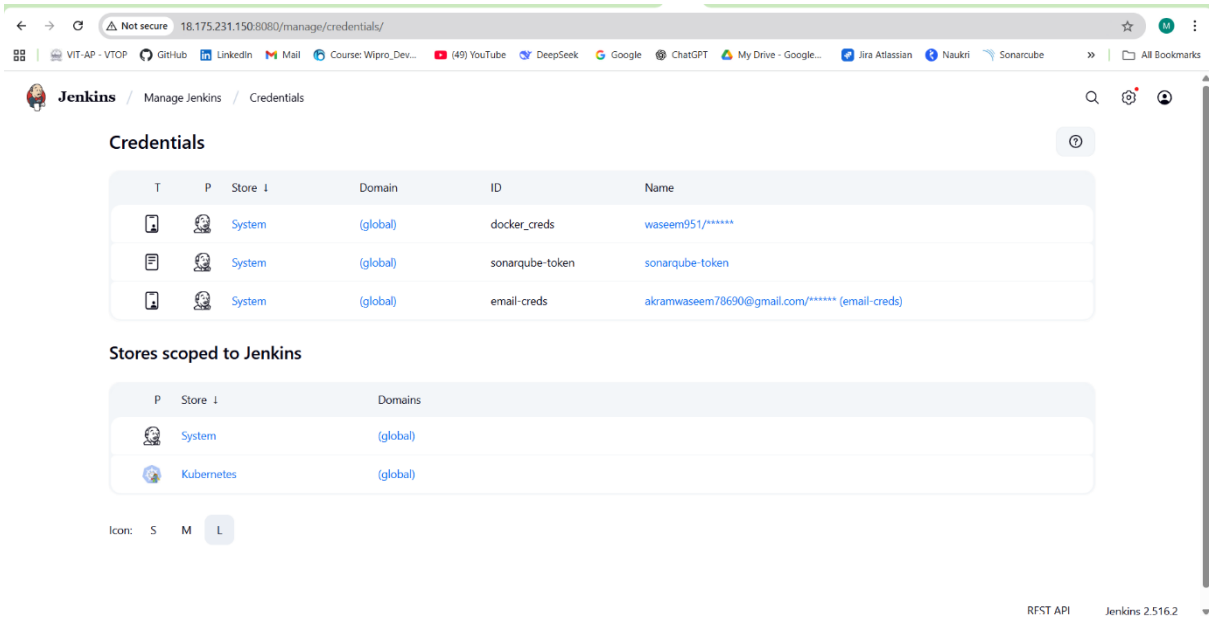
- **Plugin Installation:** Essential plugins such as Git, Pipeline, Docker, SonarQube Scanner, and Email Extension were installed.



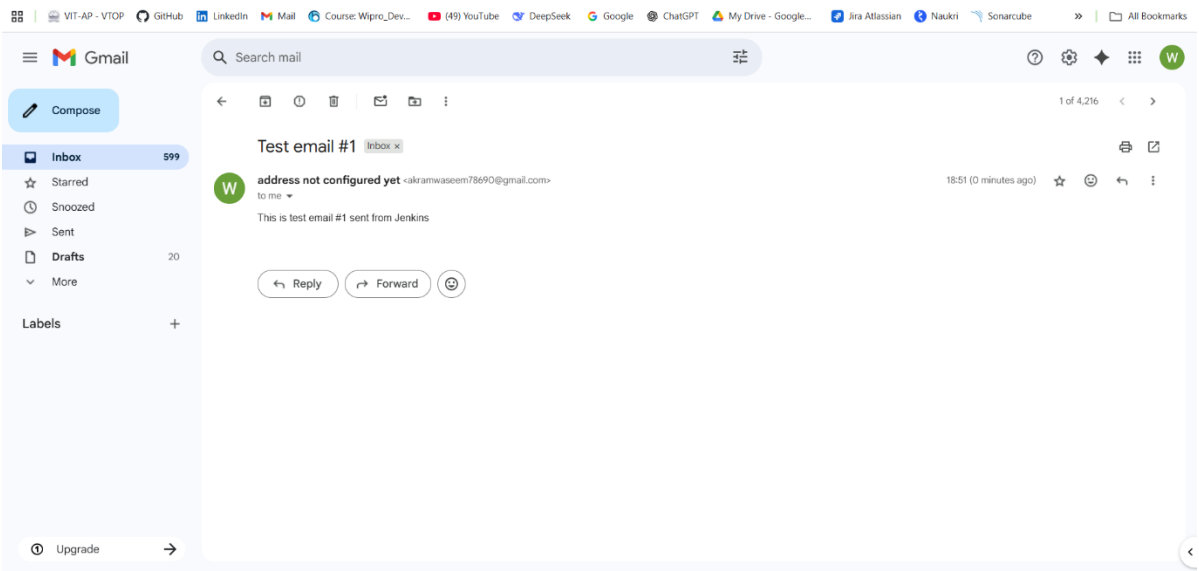
➢ **Credential Configuration:**

Configured credentials for:

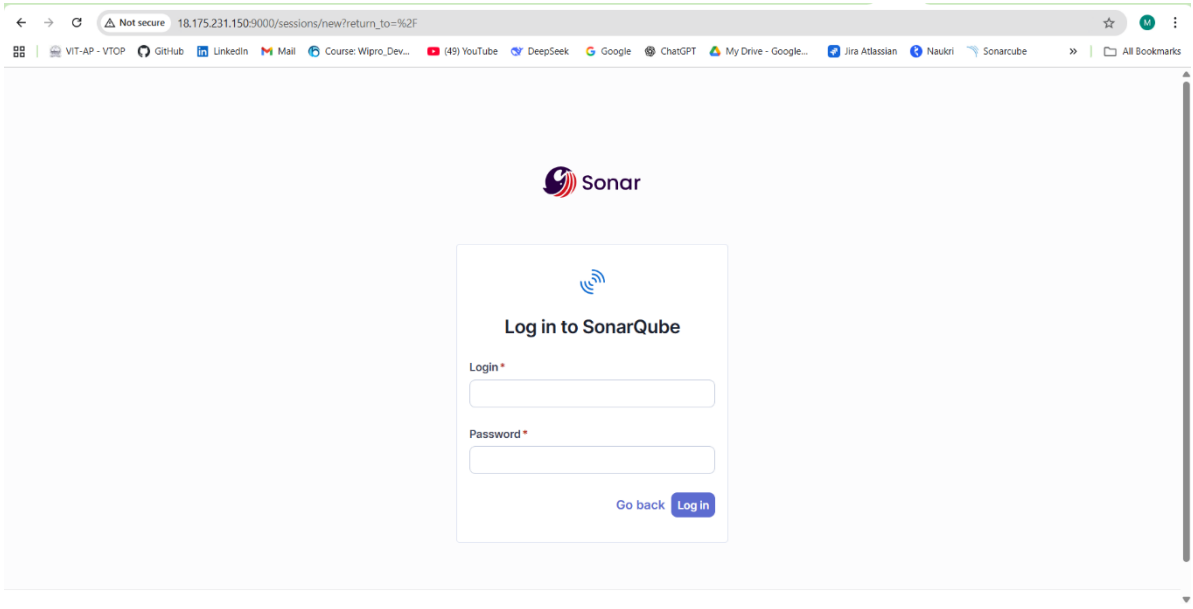
- DockerHub (for pushing images)
- SonarQube (for code analysis)
- Gmail (for email notifications)



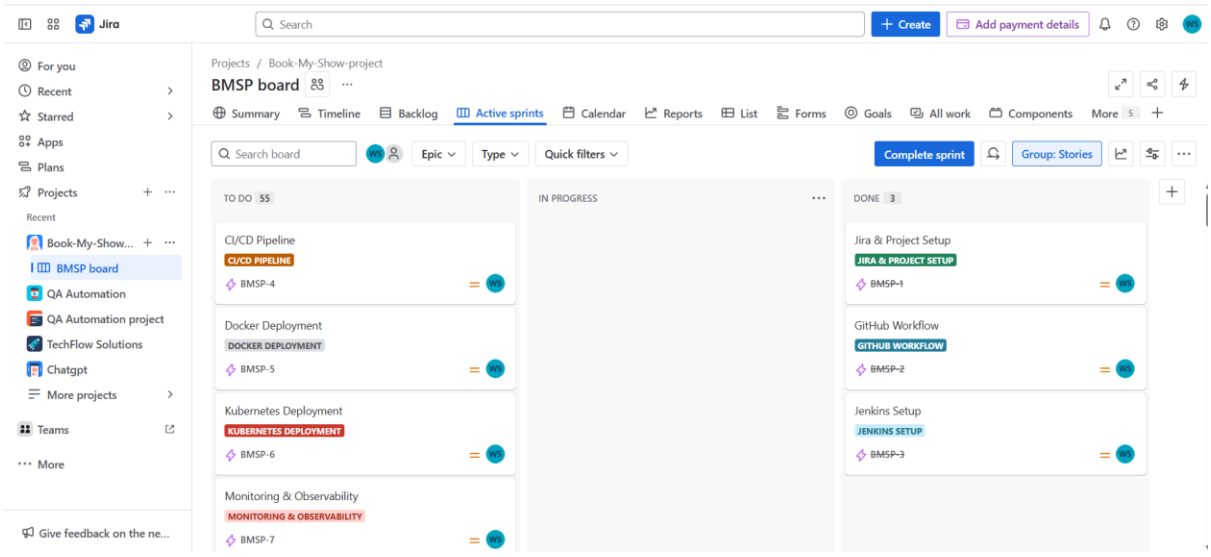
➢ **Test Email Received in Gmail**: A sample test mail from Jenkins confirming that email notifications are configured.



- **SonarQube Login Page:** Jenkins integrates with SonarQube to perform static code analysis and maintain code quality.
  url: http://18.175.231.150:9000



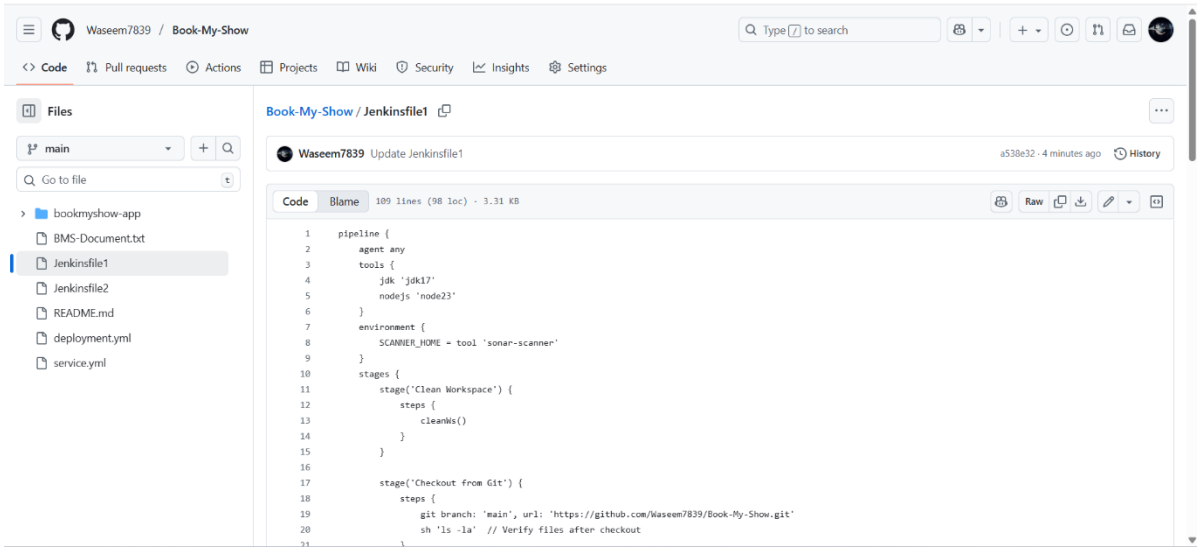
- **Task completion:** The *Jenkins Setup* task (BMSP-3) is now marked as **Done** on the Jira



4. CI/CD PIPELINE:

Jenkins is used for automating the build, test, and deployment process. A pipeline consists of multiple stages such as Checkout, SonarQube Analysis, Build, Push, Deploy, and Email Notification. This ensures continuous integration and continuous delivery (CI/CD).

- **Jenkinsfile in GitHub Repo:** The pipeline script defines stages: Clean Workspace → Git Checkout → SonarQube Analysis → Build → Push to DockerHub → Deploy → Email Notification.



- **CI/CD Pipeline Execution:**

  Jenkins pipeline executed successfully with all stages:

  - Clean Workspace
  - Checkout from Git
  - SonarQube Analysis + Quality Gate
  - Build & Push Docker image
  - Deploy to Container

Repository Link: https://github.com/Waseem7839/Book-My-Show

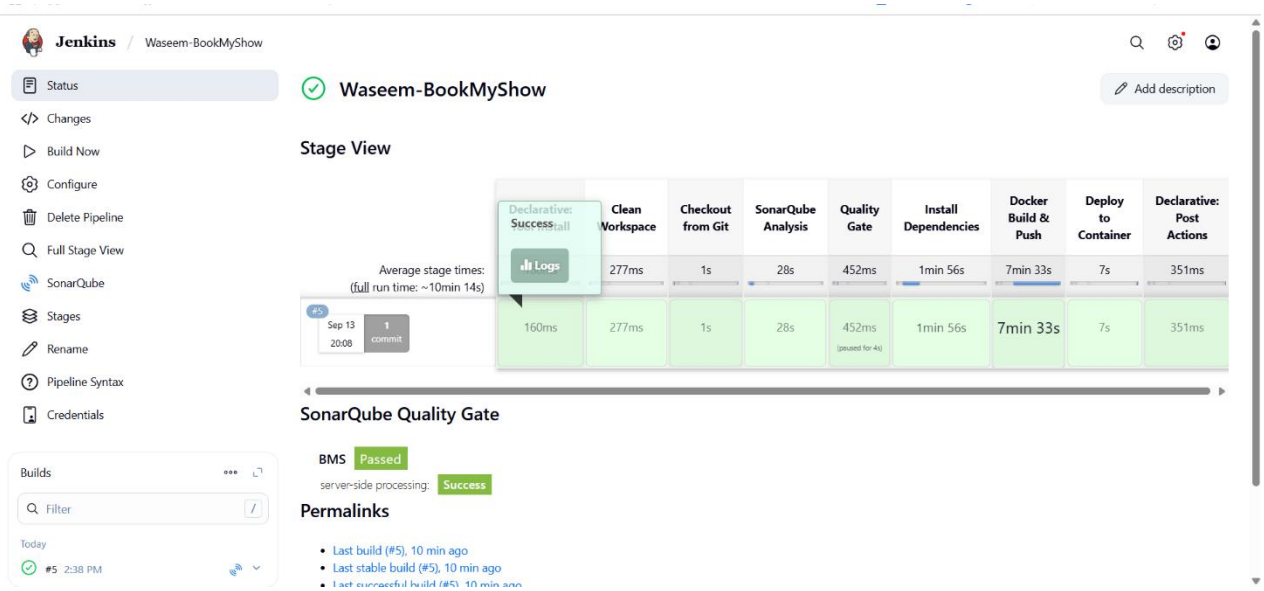
Code: Jenkins Pipeline Code

```
pipeline {
    agent any
    tools {
        jdk 'jdk17'
        nodejs 'node23'
    }
    environment {
        SCANNER_HOME = tool 'sonar-scanner'
    }
    stages {
        stage('Clean Workspace') {
            steps {
                cleanWs()
            }
        }

        stage('Checkout from Git') {
            steps {
                git branch: 'main', url: 'https://github.com/Waseem7839/Book-My-Show.git'
                sh 'ls -la'  // Verify files after checkout
            }
        }

        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('sonar-server') {
                    sh '''
                        $SCANNER_HOME/bin/sonar-scanner \
```

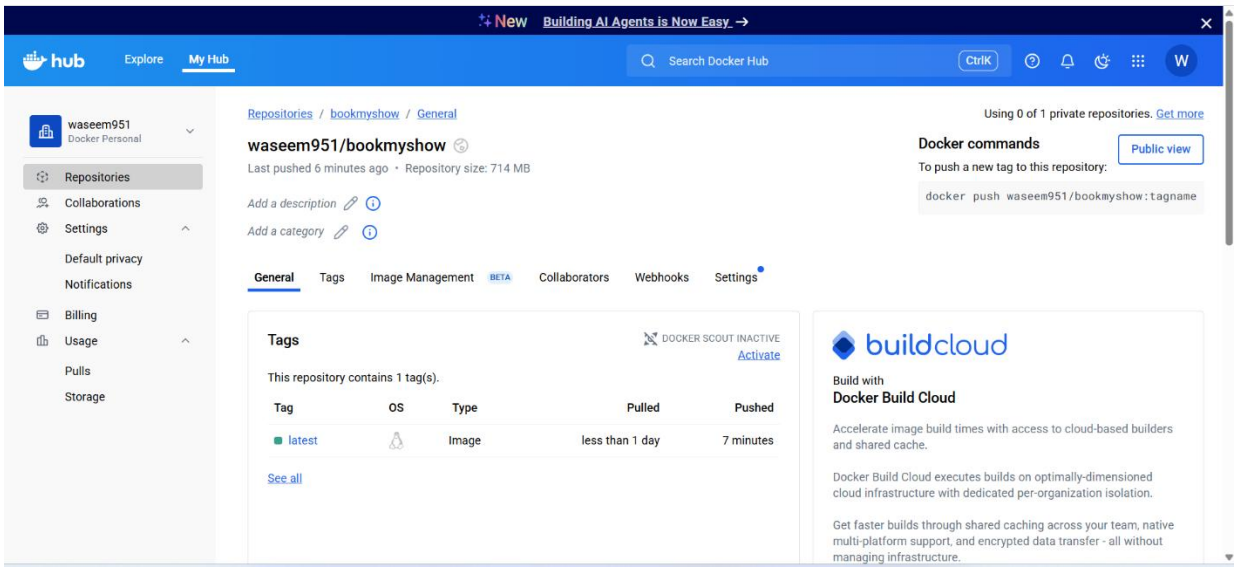
```
            -Dsonar.projectName=BMS \
            -Dsonar.projectKey=BMS
        '''
      }
    }
}

stage('Quality Gate') {
    steps {
      script {
        waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-token'
      }
    }
}

stage('Install Dependencies') {
    steps {
      sh '''
        cd bookmyshow-app
        ls -la  # Verify package.json exists
        if [ -f package.json ]; then
          rm -rf node_modules package-lock.json  # Remove old dependencies
          npm install  # Install fresh dependencies
        else
          echo "Error: package.json not found in bookmyshow-app!"
          exit 1
        fi
      '''
    }
```

```
        }

        stage('Docker Build & Push') {
            steps {
                script {
                    withDockerRegistry(credentialsId: 'docker_creds', toolName: 'docker') {
                        sh '''
                            echo "Building Docker image..."
                            docker build --no-cache -t waseem951/bookmyshow:latest -f bookmyshow-app/Dockerfile bookmyshow-app

                            echo "Pushing Docker image to registry..."
                            docker push waseem951/bookmyshow:latest
                        '''
                    }
                }
            }
        }

        stage('Deploy to Container') {
            steps {
                sh '''
                    echo "Stopping and removing old container..."
                    docker stop bookmyshow || true
                    docker rm bookmyshow || true

                    echo "Running new container on port 3000..."
                    docker run -d --restart=always --name bookmyshow -p 3000:3000 waseem951/bookmyshow:latest
```

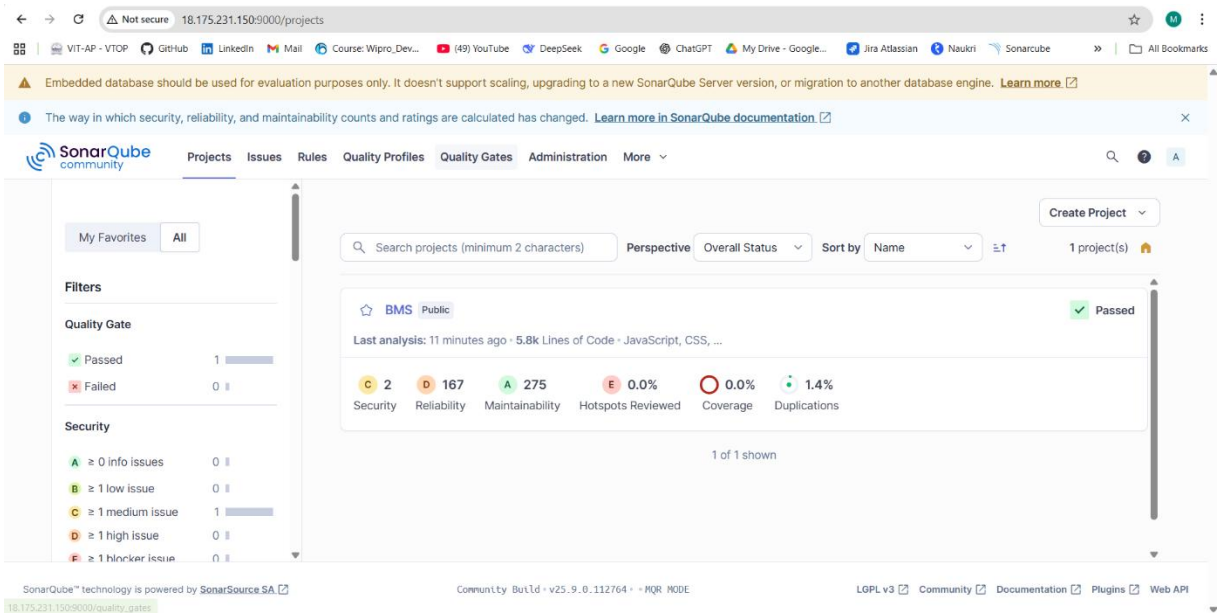
```
                echo "Checking running containers..."
                docker ps -a

                echo "Fetching logs..."
                sleep 5  # Give time for the app to start
                docker logs bookmyshow
            '''
        }
    }
}

post {
    always {
        emailext(
            attachLog: true,
            subject: "'${currentBuild.result}'",
            body: """Project: ${env.JOB_NAME}<br/>
                Build Number: ${env.BUILD_NUMBER}<br/>
                URL: ${env.BUILD_URL}<br/>""",
            to: 'akramwaseem78690@gmail.com'
        )
    }
}
}
```

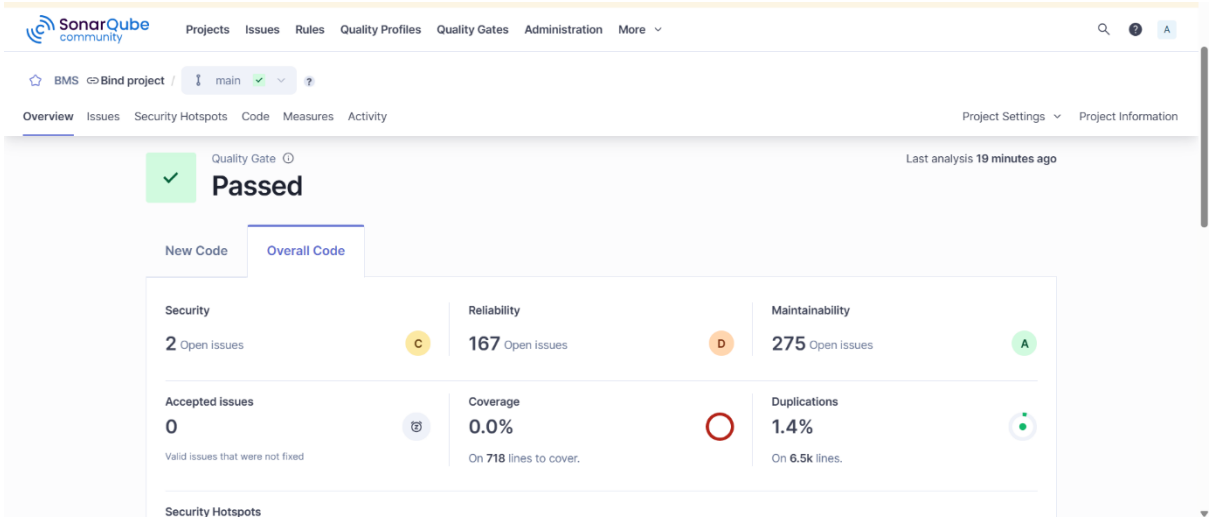


- **DockerHub Deployment:** Docker image waseem951/bookmyshow:latest pushed successfully to Docker Hub.

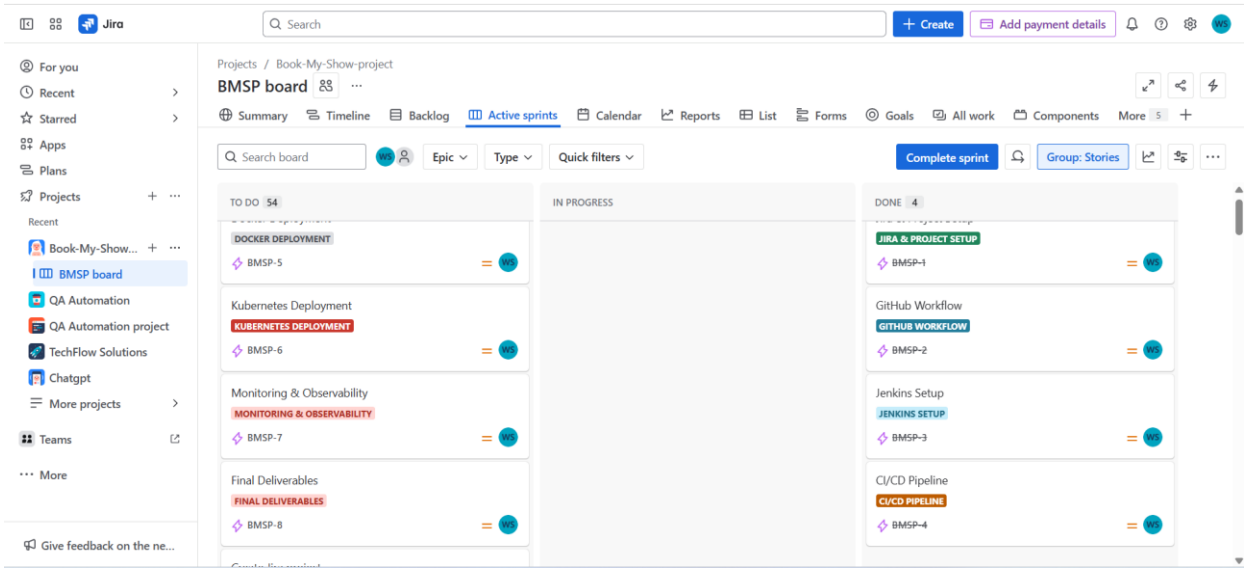


- **SonarQube Quality Gate Result:**
  Quality Gate **Passed**.





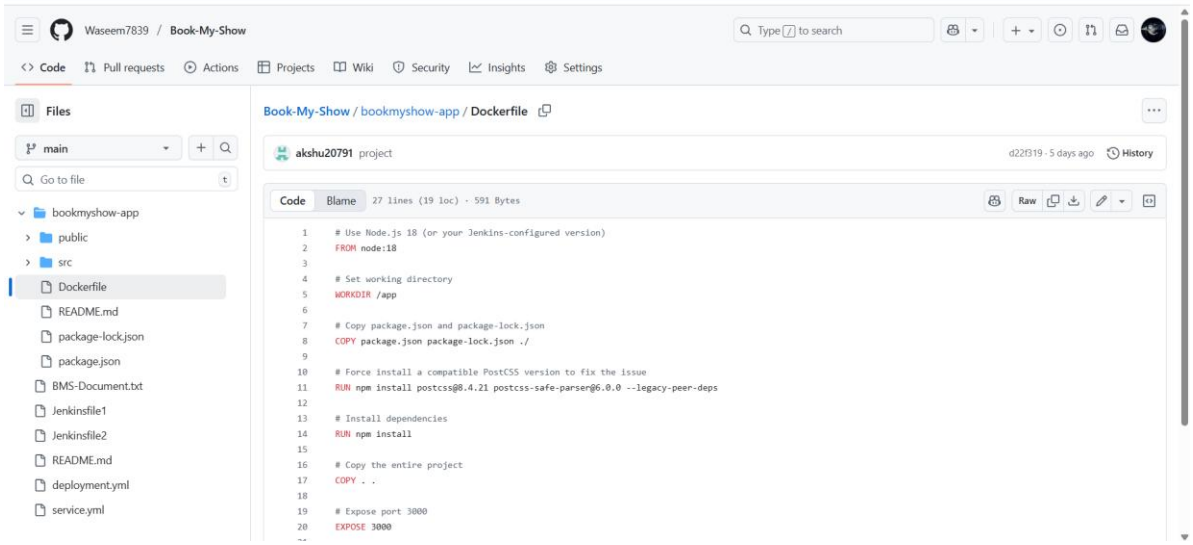
- **Task Completion:** The *CI/CD Pipeline* task (BMSP-4) has been marked as **Done** on the Jira board.



5. ## Docker Deployment:

The **Docker Deployment** step was implemented to containerize and run the Book-My-Show application.

- **Dockerfile Creation:** A Dockerfile was written to define how the application should be built and run inside a container.
  It included the base image, dependencies, and instructions to copy project files and run the app.



CODE:

```
# Use Node.js 18 (or your Jenkins-configured version)

FROM node:18


# Set working directory

WORKDIR /app


# Copy package.json and package-lock.json

COPY package.json package-lock.json ./


# Force install a compatible PostCSS version to fix the issue

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

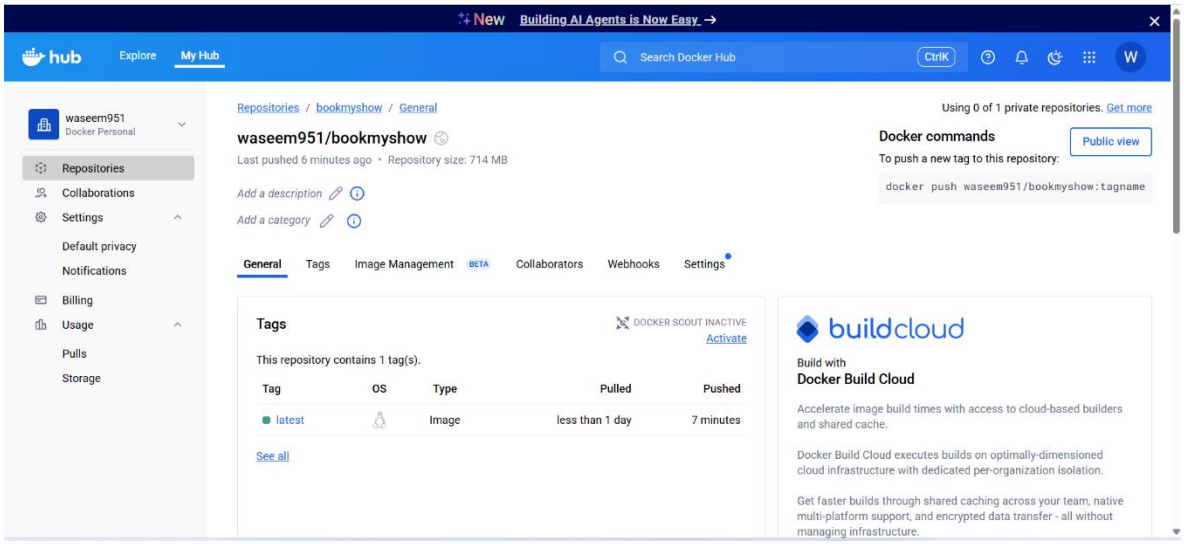
```
# Install dependencies
RUN npm install

# Copy the entire project
COPY . .

# Expose port 3000
EXPOSE 3000

# Set environment variable to prevent OpenSSL errors
ENV NODE_OPTIONS=--openssl-legacy-provider
ENV PORT=3000

# Start the application
CMD ["npm", "start"]
```

- **Docker Build:** The Docker image was built using Jenkins pipeline

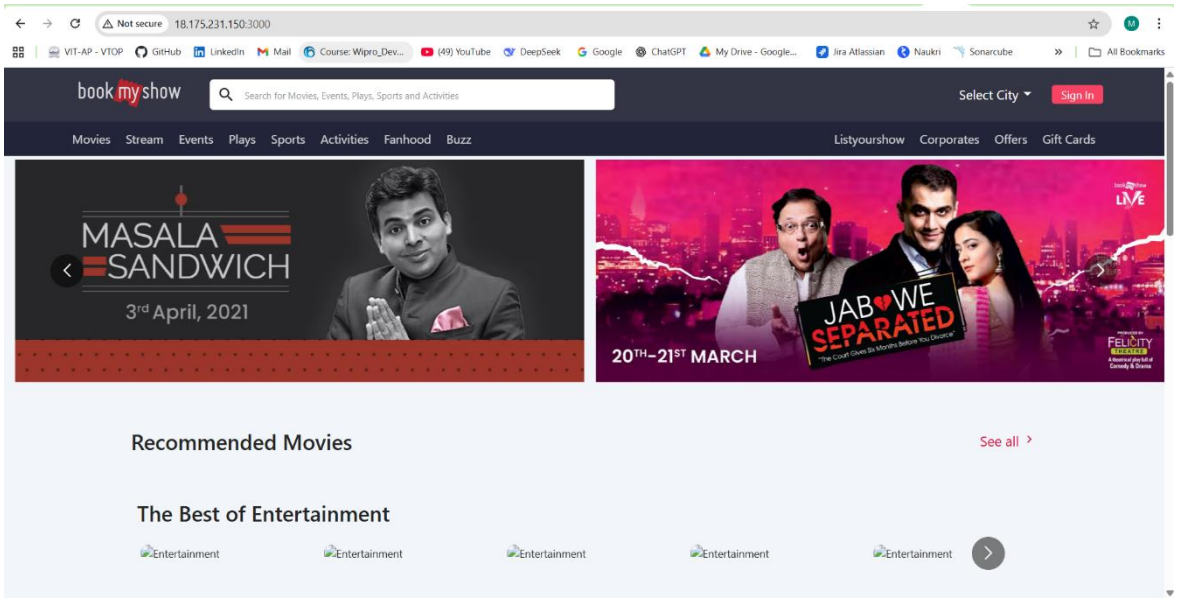
```
60            stage('Docker Build & Push') {
61                steps {
62                    script {
63                        withDockerRegistry(credentialsId: 'docker_creds', toolName: 'docker') {
64                            sh '''
65                                echo "Building Docker image..."
66                                docker build --no-cache -t waseem951/bookmyshow:latest -f bookmyshow-app/Dockerfile bookmyshow-app
67
68                                echo "Pushing Docker image to registry..."
69                                docker push waseem951/bookmyshow:latest
70                            '''
71                        }
72                    }
73                }
74            }
75
```

- **Docker Push to Docker Hub:** The newly built image was pushed to Docker Hub
  docker run -d -p 3000:3000 waseem951/bookmyshow:latest

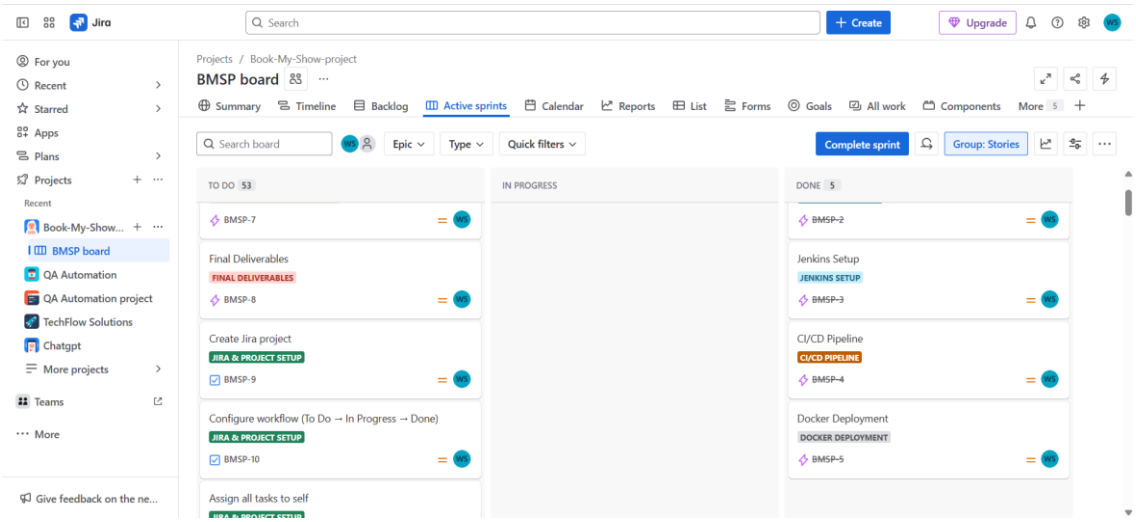
Dockerhub Repository Link: https://hub.docker.com/repositories/waseem951



- **Application Deployment Completed:** The BookMyShow clone UI is accessible at http://18.175.231.150:3000
  Homepage loads with movie/event banners and navigation options.

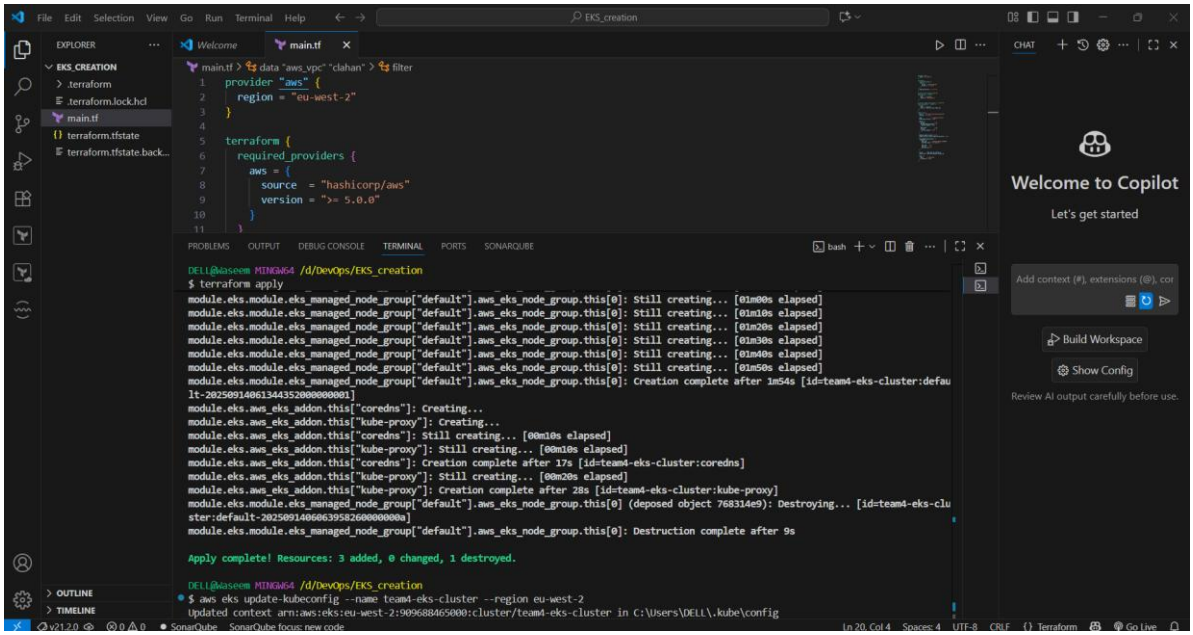


- **Task completion:** The Docker Deployment task has been marked as Done on the Jira board.

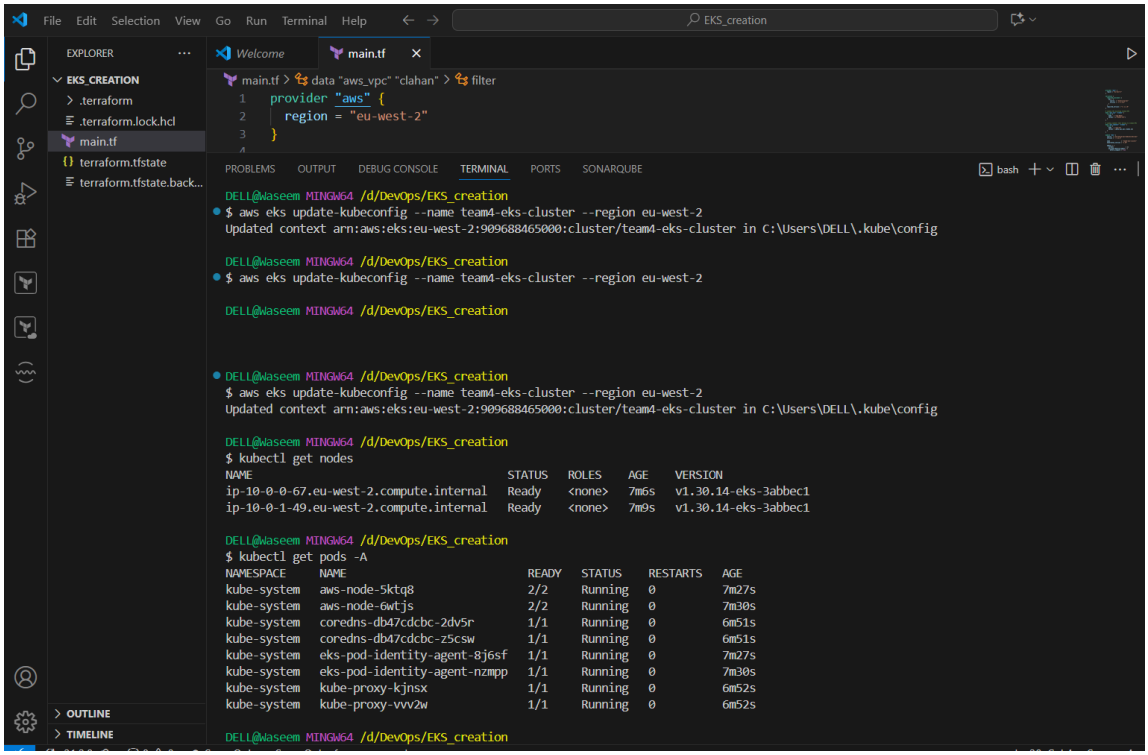


6. [KUBERNETES DEPLOYMENT(EKS):](#) The Kubernetes cluster was provisioned on AWS EKS using **Terraform**.

- **EKS Creation:** Terraform was applied to create the EKS cluster along with managed node groups and core add-ons like coredns and kube-proxy. The apply completed successfully, confirming cluster creation.



- **EKS Cluster Verification:** The kubeconfig was updated, and verification with kubectl get nodes and kubectl get pods -A showed that worker nodes are in **Ready** state and all system pods are running smoothly.



- **Deployment.yml:**

Code

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bms-app
  labels:
    app: bms
spec:
  replicas: 2
  selector:
    matchLabels:
      app: bms
  template:
    metadata:
      labels:
        app: bms
    spec:
      containers:
      - name: bms-container
        image: waseem951/bookmyshow:latest  # Replace with your Docker image
        ports:
        - containerPort: 3000  # Replace with the port your app runs on
```

- **Service.yml**

Code

```
apiVersion: v1
kind: Service
metadata:
  name: bms-service
  labels:
    app: bms
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 3000  # Replace with the port your app runs on
  selector:
    app: bms
```

- **Jenkins Pipeline Execution with Kubernetes Integration:** The Jenkins pipeline (Waseem-BMS-with-K8s) executed successfully, covering all stages including tool installation, code checkout, SonarQube analysis, Docker image build & push, container deployment, and EKS cluster deployment. The SonarQube Quality Gate also passed, confirming code quality.

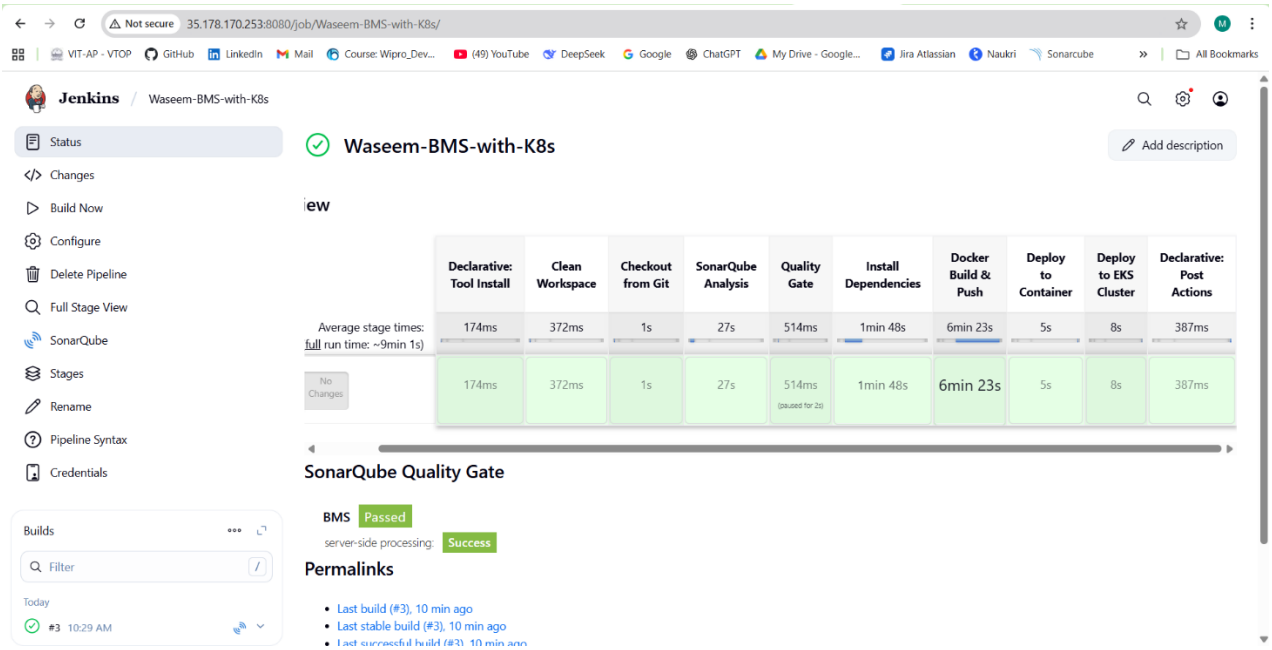
CODE: with Kubernetes

```
pipeline {
    agent any

    tools {
        jdk 'jdk17'
        nodejs 'node23'
    }

    environment {
        SCANNER_HOME = tool 'sonar-scanner'
        DOCKER_IMAGE = 'waseem951/bookmyshow:latest'
        EKS_CLUSTER_NAME = 'team4-eks-cluster'
        AWS_REGION = 'eu-west-2'
    }

    stages {
        stage('Clean Workspace') {
```

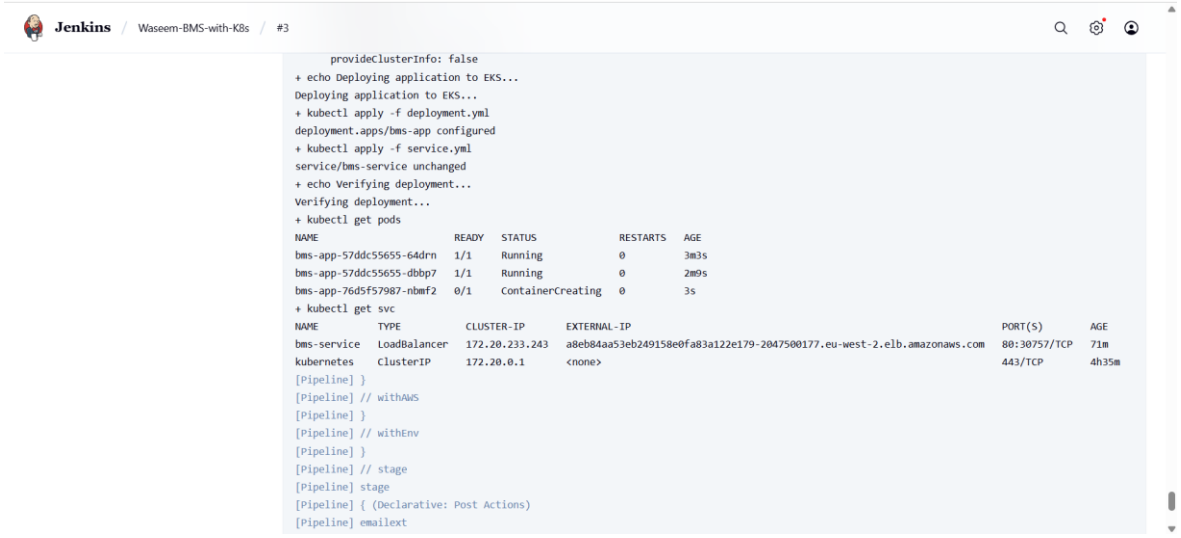
```
    steps {
        cleanWs()
    }
}

stage('Checkout from Git') {
    steps {
        git branch: 'main', url: 'https://github.com/Waseem7839/Book-My-Show.git'
        sh 'ls -la'  // Verify files after checkout
    }
}

stage('SonarQube Analysis') {
    steps {
        withSonarQubeEnv('sonar-server') {
            sh '''
            $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=BMS \
            -Dsonar.projectKey=BMS
            '''
        }
    }
}

stage('Quality Gate') {
    steps {
        script {
            waitForQualityGate abortPipeline: false, credentialsId: 'sonarqube-token'
        }
    }
}

stage('Install Dependencies') {
    steps {
        sh '''
        cd bookmyshow-app
        ls -la  # Verify package.json exists
        if [ -f package.json ]; then
            rm -rf node_modules package-lock.json
            npm install
        else
            echo "Error: package.json not found in bookmyshow-app!"
            exit 1
        fi
        '''
    }
}

stage('Docker Build & Push') {
```

```
        steps {
            script {
                withDockerRegistry(credentialsId: 'docker_creds', toolName: 'docker') {
                    sh '''
                    echo "Building Docker image..."
                    docker build --no-cache -t $DOCKER_IMAGE -f bookmyshow-app/Dockerfile
bookmyshow-app

                    echo "Pushing Docker image to registry..."
                    docker push $DOCKER_IMAGE
                    '''
                }
            }
        }
    }

    stage('Deploy to Container') {
        steps {
            sh '''
            echo "Stopping and removing old container..."
            docker stop bms || true
            docker rm bms || true

            echo "Running new container on port 3000..."
            docker run -d --restart=always --name bms -p 3005:3000 $DOCKER_IMAGE

            echo "Checking running containers..."
            docker ps -a

            echo "Fetching logs..."
            sleep 5
            docker logs bms
            '''
        }
    }

    stage('Deploy to EKS Cluster') {
        steps {
            withAWS(credentials: 'aws_creds', region: "${AWS_REGION}") {
                sh '''
                echo "Verifying AWS credentials..."
                aws sts get-caller-identity

                echo "Configuring kubectl for EKS cluster..."
                aws eks update-kubeconfig --name $EKS_CLUSTER_NAME --region
$AWS_REGION

                echo "Verifying kubeconfig..."
```

```
                kubectl config view

                echo "Deploying application to EKS..."
                kubectl apply -f deployment.yml
                kubectl apply -f service.yml

                echo "Verifying deployment..."
                kubectl get pods
                kubectl get svc
                '''
            }
        }
    }
}

post {
    always {
        emailext attachLog: true,
            subject: "'${currentBuild.result}'",
            body: "Project: ${env.JOB_NAME}<br/>" +
                "Build Number: ${env.BUILD_NUMBER}<br/>" +
                "URL: ${env.BUILD_URL}<br/>",
            to: 'akramwaseem78690@gmail.com'
    }
}
```

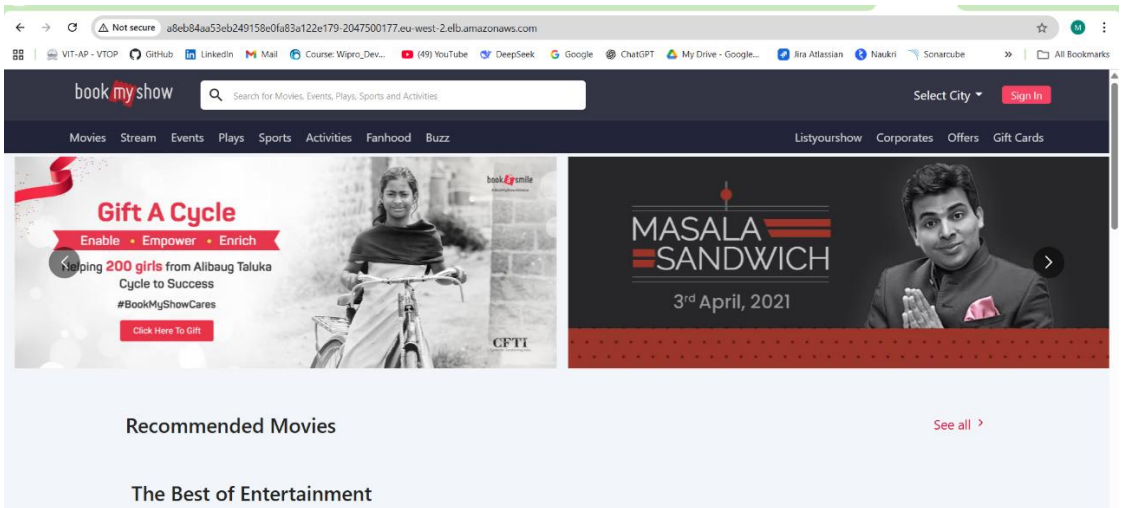


- **Application Deployment on EKS Cluster:** Deployment files (deployment.yml and service.yml) were applied to the EKS cluster via Jenkins.

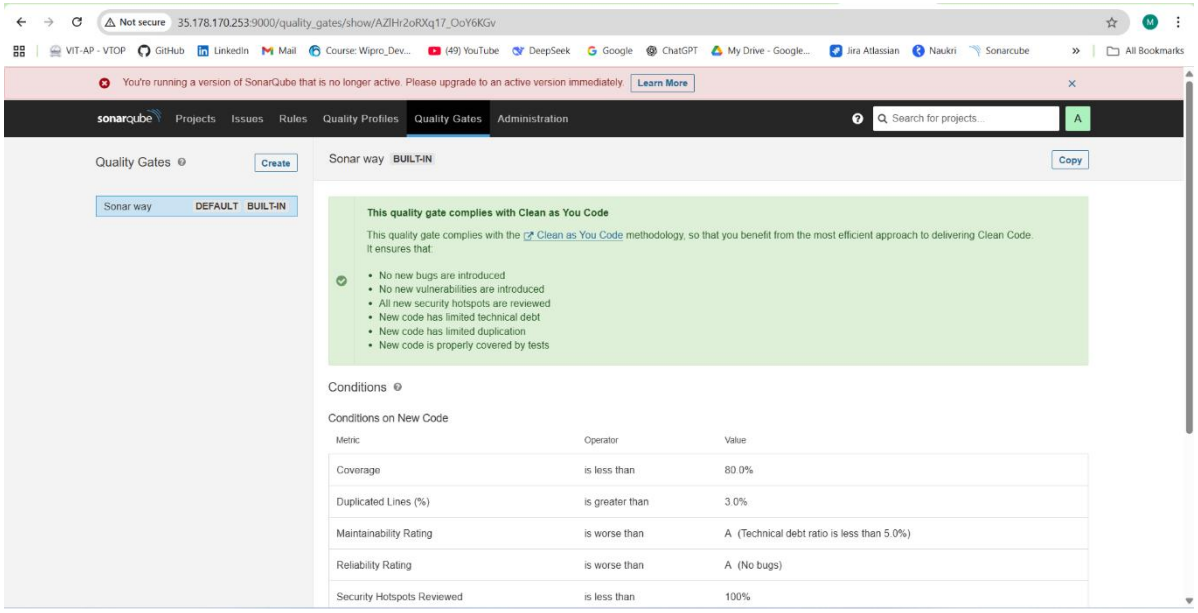


- **Kubernetes Service Verification:** Using kubectl get svc, the BMS service was verified with a **LoadBalancer EXTERNAL-IP** provisioned by AWS. This confirms that the application is accessible externally on the assigned IP and port.

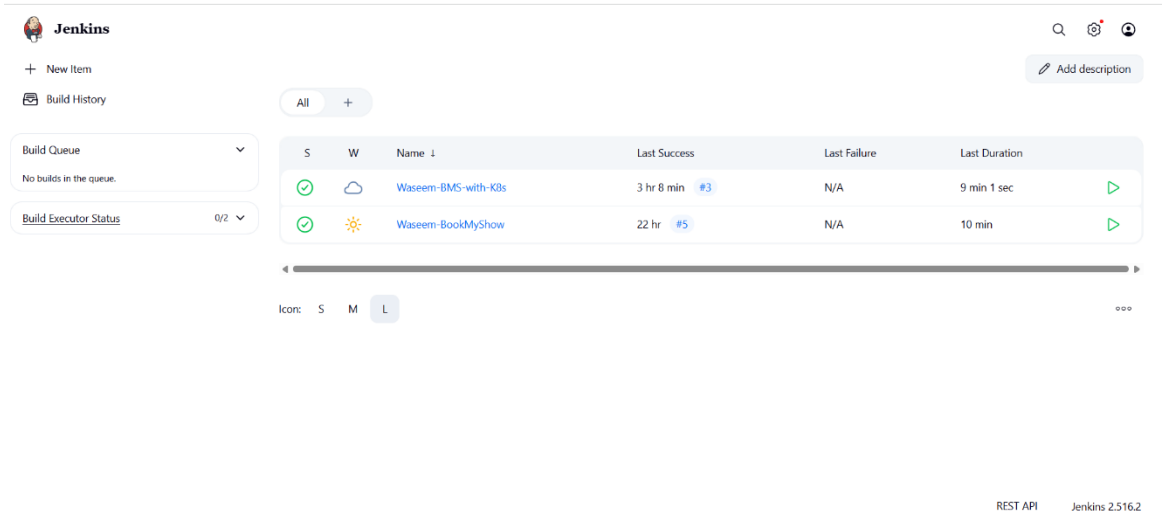
- **BookMyShow Application Running on Browser:** The BookMyShow application was accessed successfully using the LoadBalancer's public URL. This proves the deployment pipeline worked end-to-end, from Jenkins to Kubernetes.



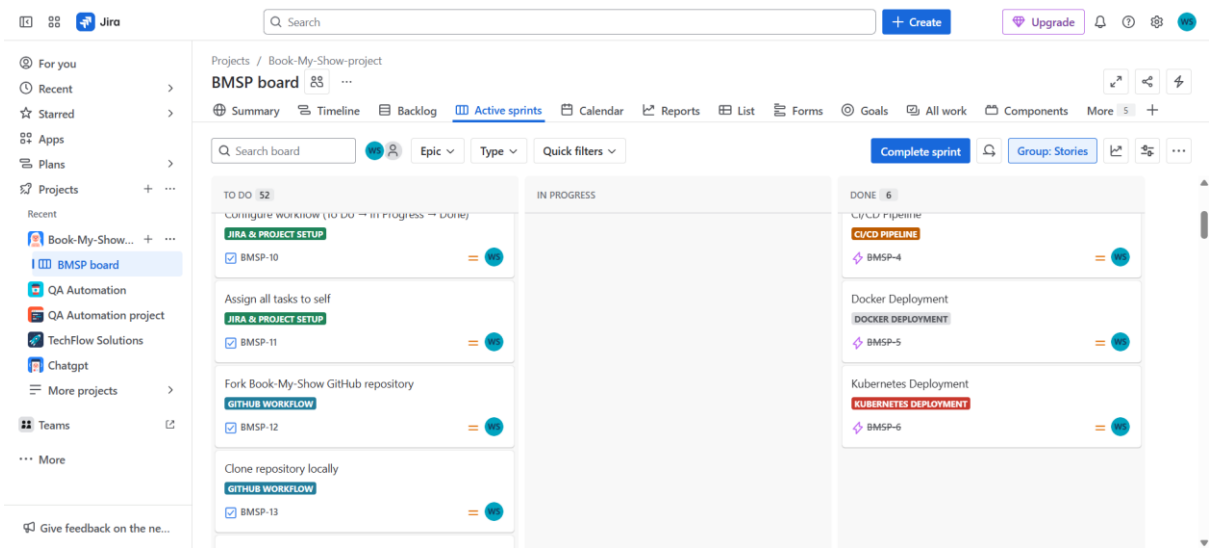
- **Quality Gate Details:** The default **Sonar way** Quality Gate was applied, ensuring no new bugs, vulnerabilities, or security hotspots are introduced. This maintains clean and maintainable code throughout deployments.



- **Jenkins Dashboard Overview:** The Jenkins dashboard shows both pipelines (Waseem-BMS-with-K8s and Waseem-BookMyShow) executed successfully without failures, highlighting consistent CI/CD health.

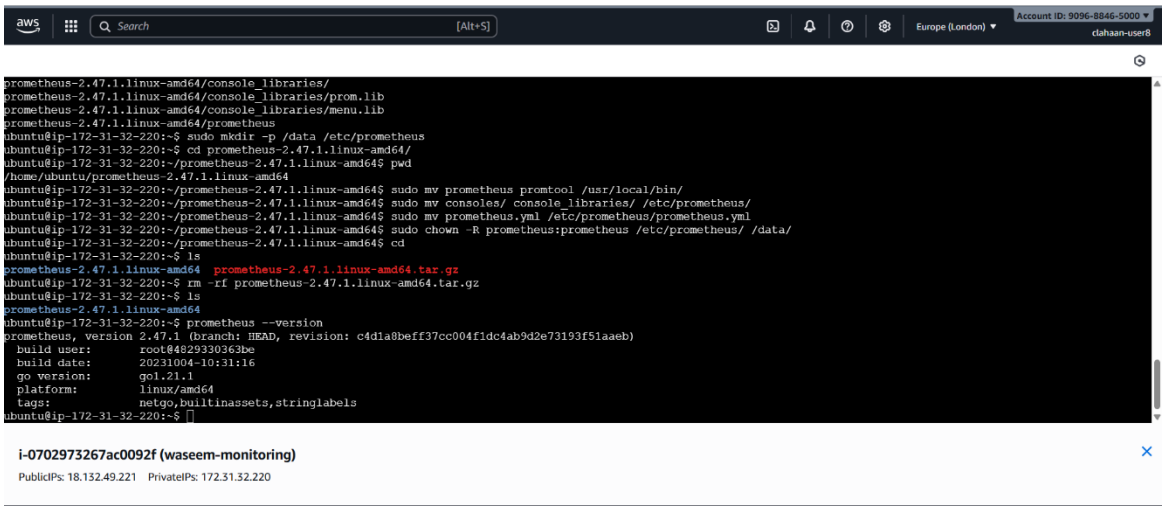


- **Task completion:** The Kubernetes Deployment task has been marked as Done on the Jira board.

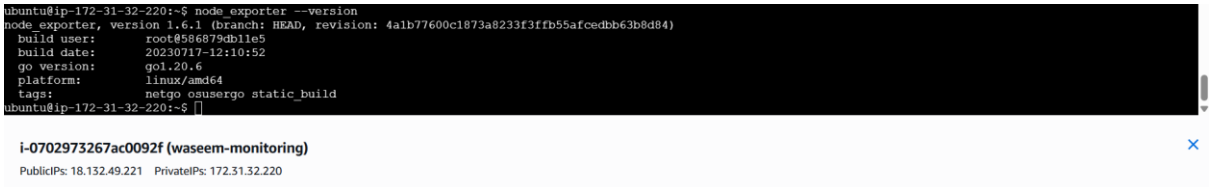


7. **Monitoring & Observability:** Monitoring and observability were implemented to track the health, performance, and reliability of the Kubernetes cluster and deployed applications.

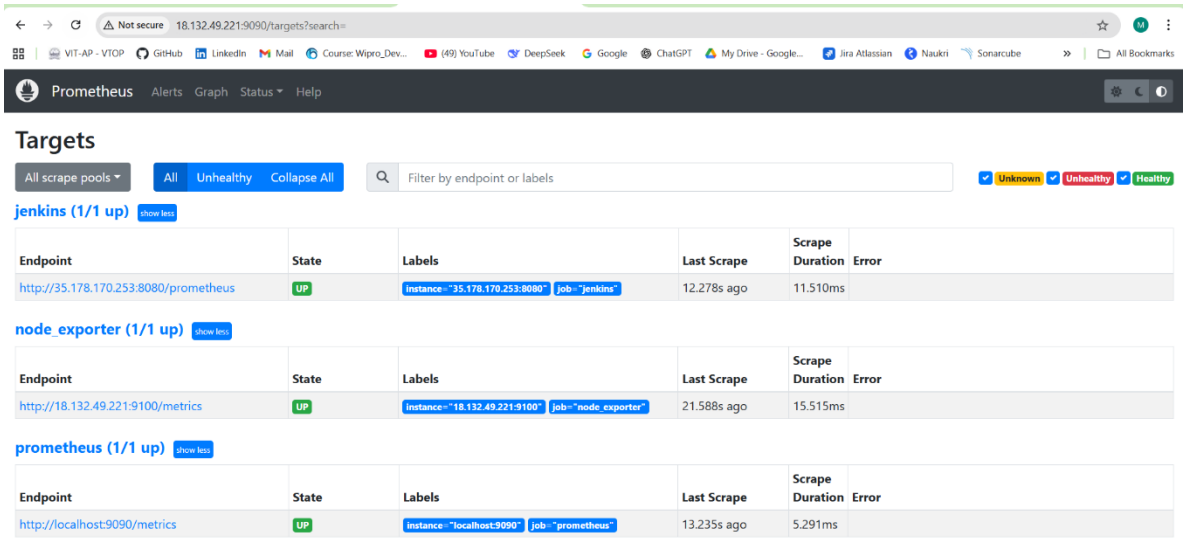
- **Prometheus** Installed successfully. It was used for collecting metrics



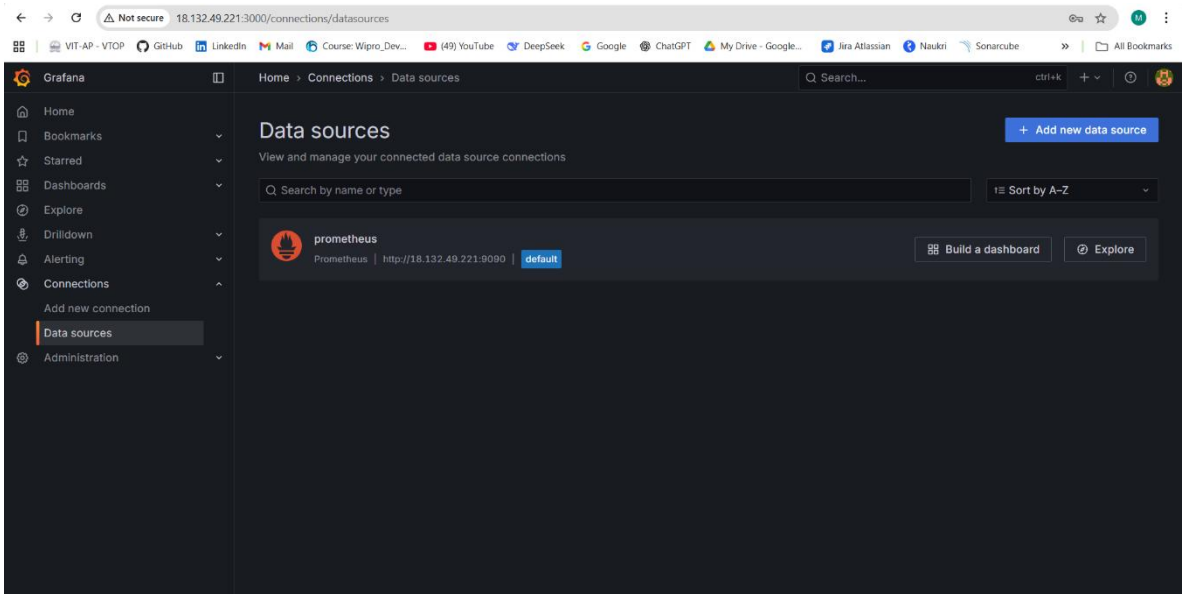
- **Node Exporter Installation done**



- **Prometheus Targets Monitoring:** Prometheus was successfully configured to scrape metrics from multiple sources. The active targets include **Jenkins**, **Node Exporter**, and **Prometheus itself**, all in an UP state.

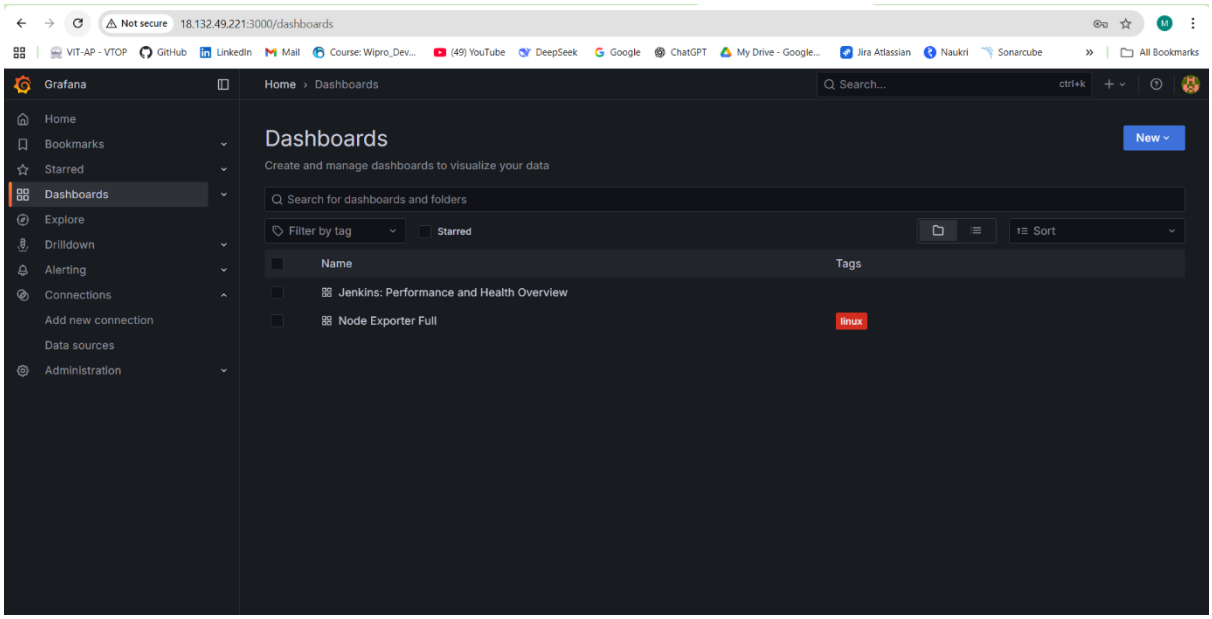


- **Grafana Data Source:** Prometheus (18.132.49.221:9090) has been configured as the **default data source** in Grafana.

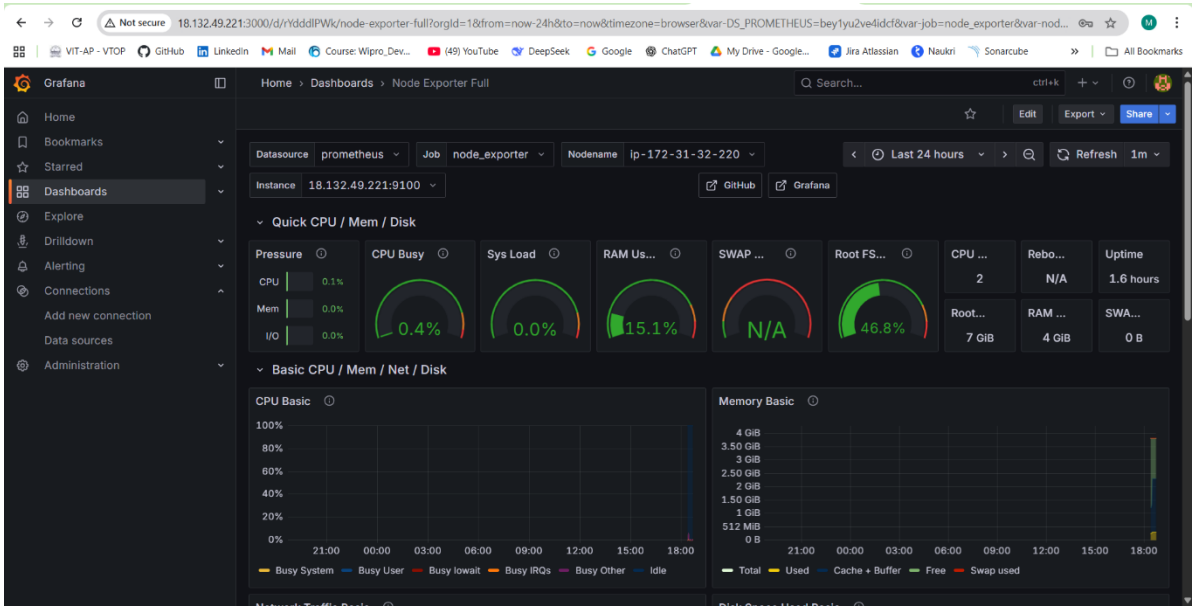


➢ **Grafana Dashboards:**

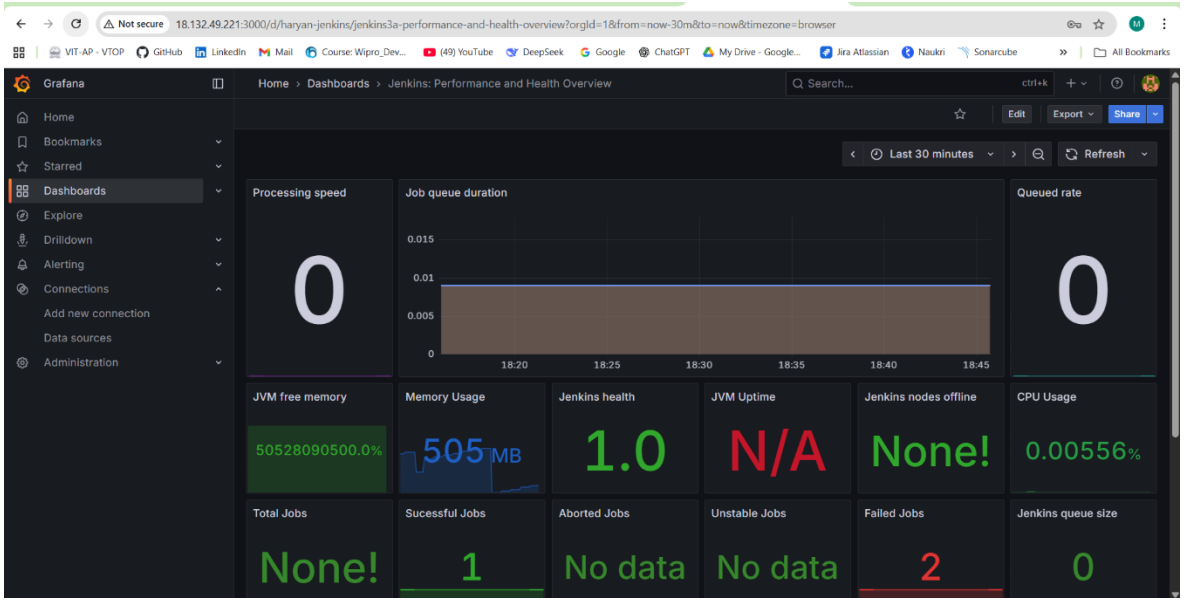
- **Jenkins: Performance and Health Overview** – for tracking Jenkins performance.
- **Node Exporter Full** – for system-level monitoring of CPU, memory, disk, and network.



➢ **Node Exporter Dashboard:** Displays system health metrics



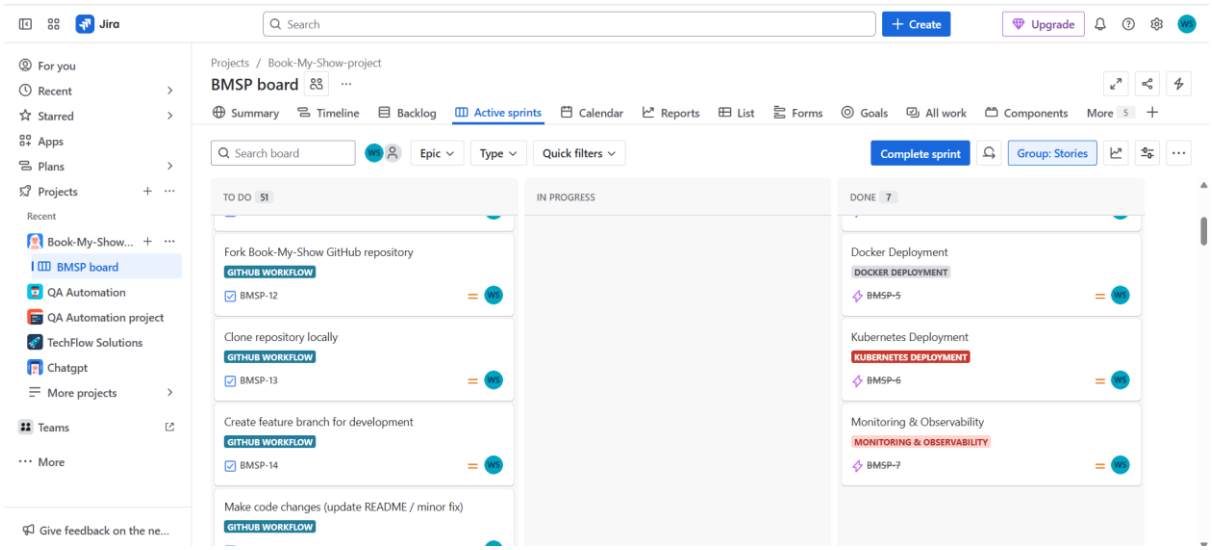
- **Jenkins Dashboard:** Monitors Jenkins performance and job activity



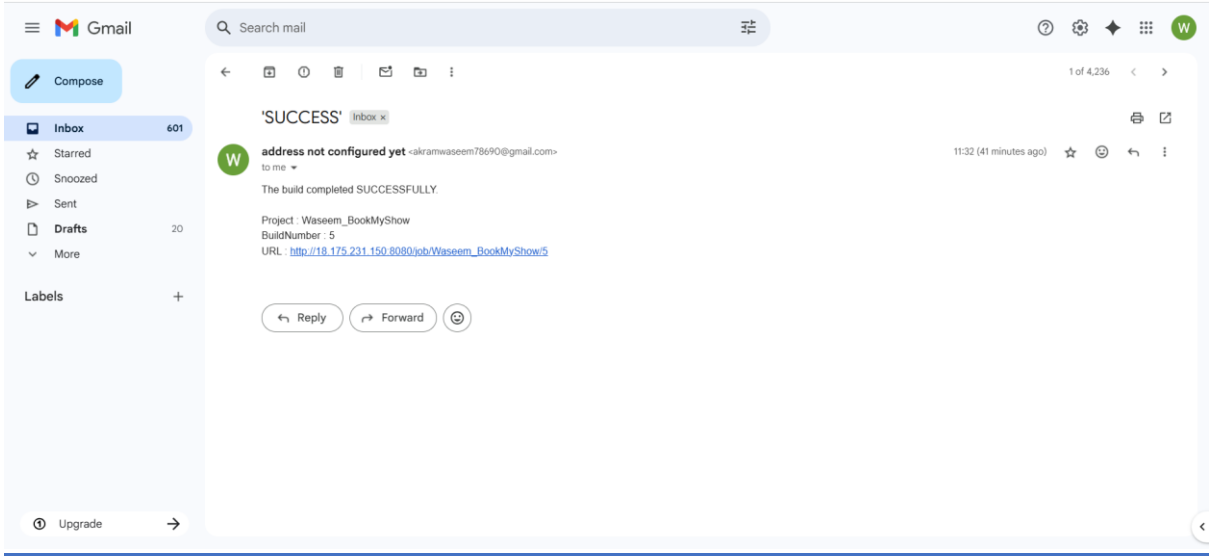
- **Node Exporter Metrics:**

```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 4.0766e-05
go_gc_duration_seconds{quantile="0.25"} 4.5254e-05
go_gc_duration_seconds{quantile="0.5"} 4.7664e-05
go_gc_duration_seconds{quantile="0.75"} 5.5351e-05
go_gc_duration_seconds{quantile="1"} 0.000127506
go_gc_duration_seconds_sum 0.030359422
go_gc_duration_seconds_count 604
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.20.6"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 3.086248e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 1.120348568e+09
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 1.600224e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 1.640227e+07
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 8.424792e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 3.086248e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 3.60448e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 4.292608e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 28122
```

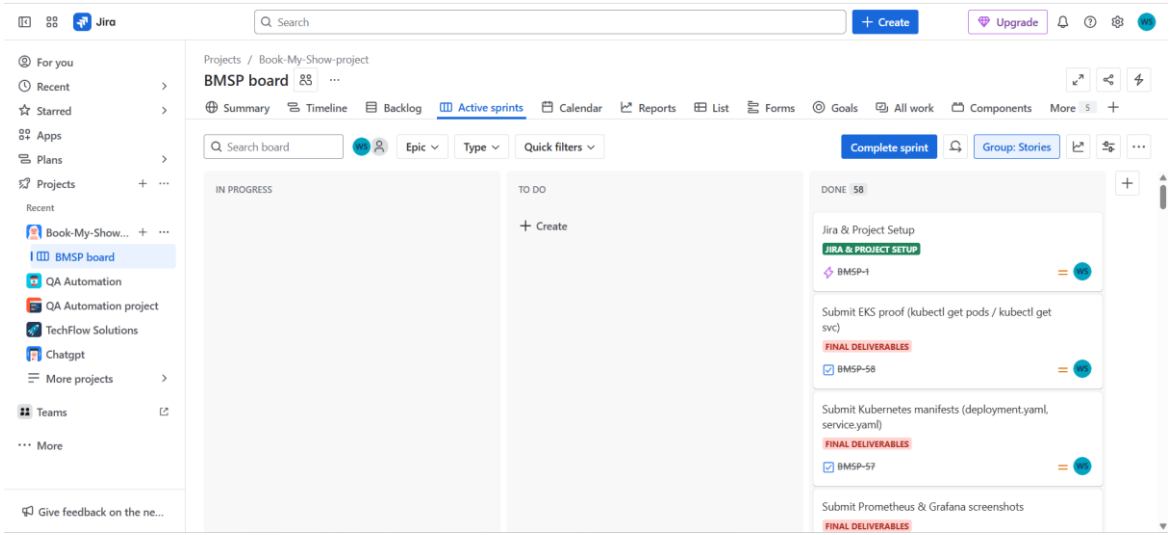
- **Task Completion:** The Monitoring & Observability task has been marked as Done on the Jira board.



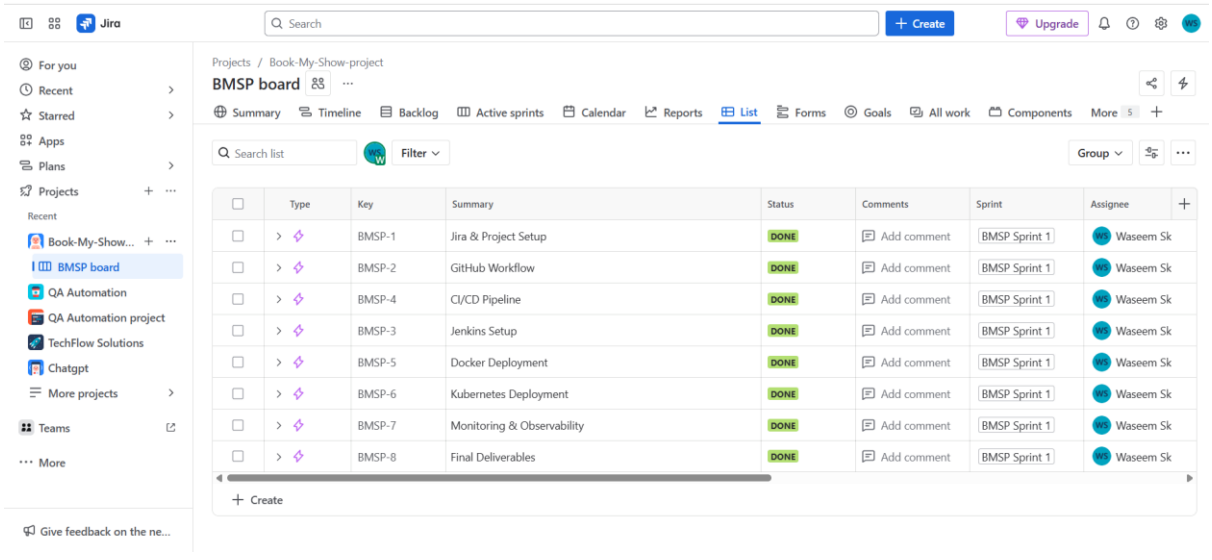
8. Email notification:



9. **Final Deliverables (BMSP-8):** The Final Deliverables task is in **To Do**, pending completion on the Jira board.



Done:- All tasks in the **Book-My-Show project** were completed and marked as **DONE** in Sprint 1.



10. Links:

- **Github PR Link:** https://github.com/Waseem7839/Book-My-Show/pull/1
- **Github_Repository_Link:** https://github.com/Waseem7839/Book-My-Show
- **Docker Repo Link:** https://hub.docker.com/repositories/waseem951