

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 & Project Setup
2. GitHub Workflow
3. Jenkins Setup
4. CI/CD Pipeline
5. Docker Deployment
6. Kubernetes Deployment
7. Monitoring & Observability
8. Email Notifications
9. Final Deliverables
10. Links

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

The screenshot shows the Jira login page with the URL 'akramsk2828.atlassian.net/jira/for-you'. The dashboard includes a sidebar with links for 'Recent', 'Starred', 'Apps', 'Plans', 'Projects', 'Teams', and 'More'. The main area displays 'Recent projects' with cards for 'QA Automation', 'QA Automation project', and 'TechFlow Solutions'. Below this is a section titled 'Worked on' showing tasks assigned to the user, such as 'assigning' and 'QAP-4', all created less than a month ago. On the right, there's a profile sidebar for 'Waseem Sk' with options for 'Profile', 'Account settings', 'Build ri', 'Prioritiz delivery', 'Theme', 'Switch account', 'Try it', and 'Log out'.

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

The screenshot shows the 'Project settings' page for the 'Book-My-Show-project'. The URL is 'akramsk2828.atlassian.net/jira/software/c/projects/BMSP/settings/details'. The left sidebar has sections for 'Project settings', 'Book-My-Show-project (Software project)', and 'Details' (which is currently selected). Other sections include 'Summary', 'People', 'Permissions', 'Notifications', 'Automation', 'Features', 'Toolchain', 'Workflows', 'Work items', 'Versions', 'Components', and 'Apps'. The main content area shows the 'Details' tab with fields for 'Name' (Book-My-Show-project), 'Project key' (BMSP), 'URL' (empty), and 'Project type' (empty). There's also a 'Change icon' button with a placeholder image of a sad face. A note at the bottom says 'Required fields are marked with an asterisk *'.

- **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.

The screenshot shows the Jira interface in list view for the 'BMSP board'. The left sidebar shows the project navigation. The main area displays a table of tasks:

| | Type | Key | Summary | Status | Comments | Sprint | Assignee |
|--------------------------|------|--------|----------------------------|--------|-----------------------------|---------------|-----------|
| <input type="checkbox"/> | ⚡ | BMSP-1 | Jira & Project Setup | TO DO | Add comment | BMSP Sprint 1 | Waseem Sk |
| <input type="checkbox"/> | ⚡ | BMSP-2 | GitHub Workflow | TO DO | Add comment | BMSP Sprint 1 | Waseem Sk |
| <input type="checkbox"/> | ⚡ | BMSP-3 | Jenkins Setup | TO DO | Add comment | BMSP Sprint 1 | Waseem Sk |
| <input type="checkbox"/> | ⚡ | BMSP-4 | CI/CD Pipeline | TO DO | Add comment | BMSP Sprint 1 | Waseem Sk |
| <input type="checkbox"/> | ⚡ | BMSP-5 | Docker Deployment | TO DO | Add comment | BMSP Sprint 1 | Waseem Sk |
| <input type="checkbox"/> | ⚡ | BMSP-6 | Kubernetes Deployment | TO DO | Add comment | BMSP Sprint 1 | Waseem Sk |
| <input type="checkbox"/> | ⚡ | BMSP-7 | Monitoring & Observability | TO DO | Add comment | BMSP Sprint 1 | Waseem Sk |
| <input type="checkbox"/> | ⚡ | BMSP-8 | Final Deliverables | TO DO | Add comment | BMSP Sprint 1 | Waseem Sk |

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

The screenshot shows the Jira interface in the 'Active sprints' board view for the 'BMSP board'. The main area displays tasks categorized by status:

- TO DO:** 58 tasks, including 'Jira & Project Setup' (BMSP-1), 'GitHub Workflow' (BMSP-2), 'Jenkins Setup' (BMSP-3), and 'CI/CD Pipeline' (BMSP-4).
- IN PROGRESS:** 0 tasks.
- DONE:** 0 tasks.

- **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.

| Column | Item | Status |
|-------------|---|--------|
| TO DO | Github Workflow (BMSP-2) Jenkins Setup (BMSP-3) CI/CD Pipeline (BMSP-4) Docker Deployment (BMSP-5) Kubernetes Deployment (BMSP-6) | |
| IN PROGRESS | Github Workflow (BMSP-2) | |
| DONE | Jira & Project Setup (BMSP-1) | |

2. GITHUB WORKFLOW:

GitHub serves as the source code management system. A **branching strategy** was used:

main branch → stable production code

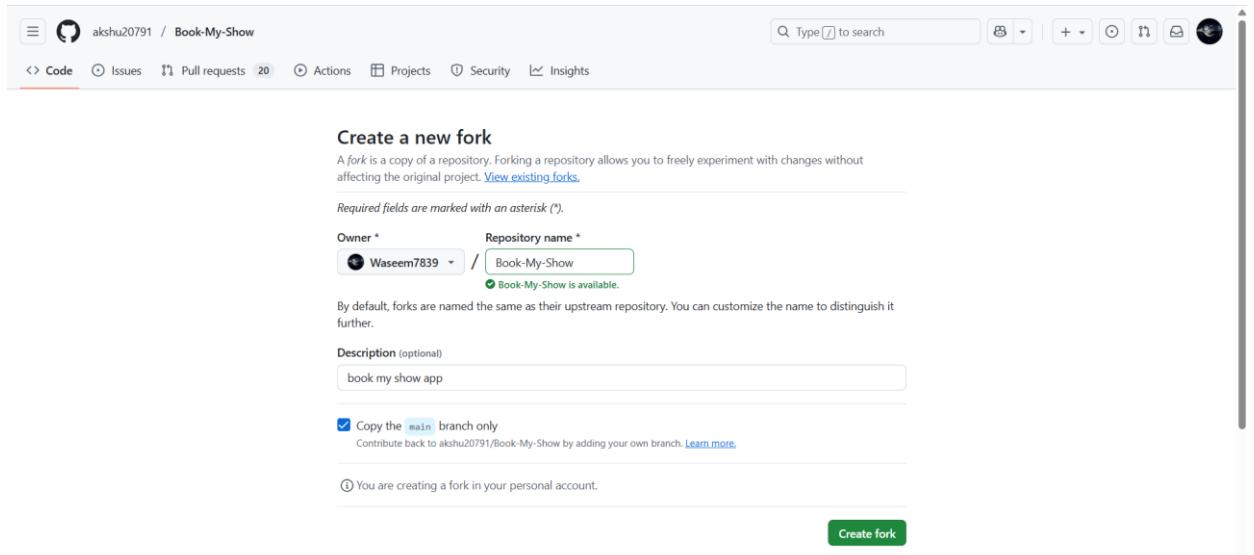
feature-* branches → new changes

Changes were merged using Pull Requests (PRs) after review.

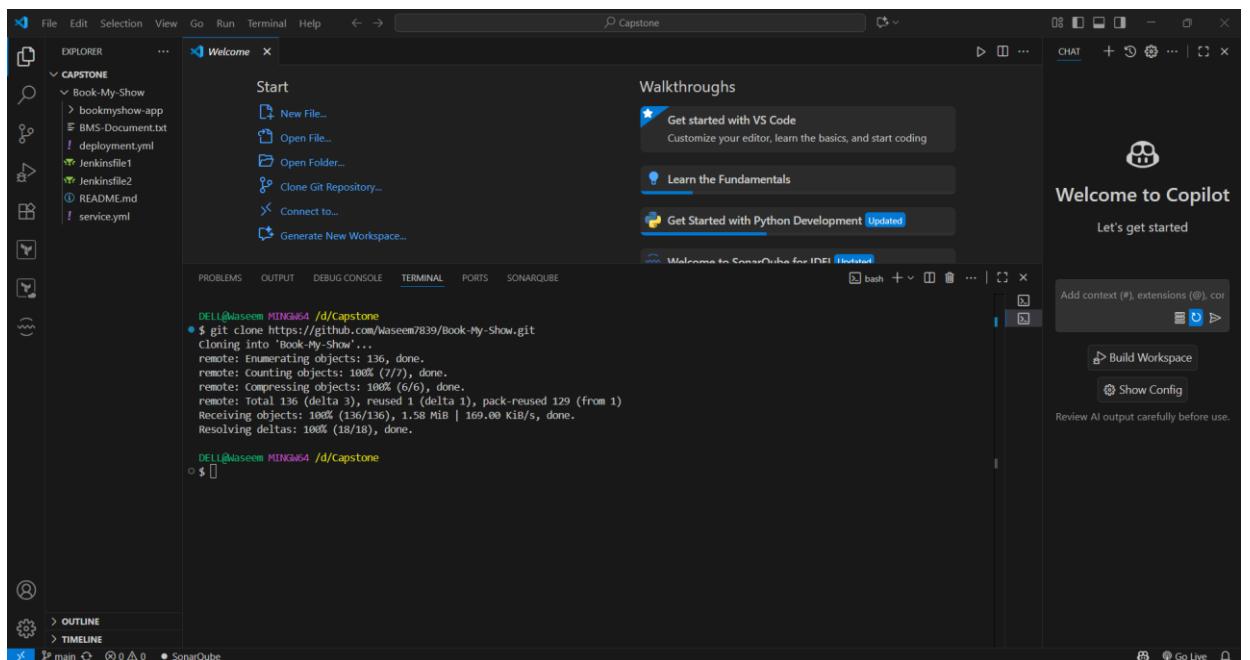
- **TO DO ->**

| Column | Item | Status |
|-------------|---|--------|
| TO DO | Jenkins Setup (BMSP-3) GitHub Workflow (BMSP-2) CI/CD Pipeline (BMSP-4) Docker Deployment (BMSP-5) Kubernetes Deployment (BMSP-6) Jenkins Setup (BMSP-3) | |
| IN PROGRESS | Github Workflow (BMSP-2) | |
| DONE | Jira & Project Setup (BMSP-1) | |

- **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

Book-My-Show Public
forked from [akshu20791/Book-My>Show](#)

This branch is up to date with [akshu20791/Book-My>Show:main](#).

| File | Type | Last Commit |
|------------------|---------|-------------|
| BMS-Document.txt | project | 3 days ago |
| Jenkinsfile1 | project | 3 days ago |
| Jenkinsfile2 | project | 3 days ago |
| README.md | project | 3 days ago |
| deployment.yml | project | 3 days ago |
| service.yml | project | 3 days ago |

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

```

1 # Book My Show - DevOps CI/CD Project
2
3 This project is a **Book My Show clone application** used for learning and demonstrating **DevOps practices** including:
4
5 ## Features
6 - Basic movie ticket booking functionality
7 - Frontend built with Node.js/React
8 - Backend APIs for managing movies, shows, and bookings
9 - Runs on Docker container
10
11 ## DevOps Implementation
12
13 $ git add .
14
15 DELL@Aseem MINIGAM6 /d/Capstone/Book-My>Show (feature)
16 $ git commit -m "Read me file updated"
17 [feature 95abae] Read me file updated
18 1 file changed, 34 insertions(+)
19
20 DELL@Aseem MINIGAM6 /d/Capstone/Book-My>Show (feature)
21 $ git push origin feature
22Enumerating objects: 5, done.
23Counting objects: 100% (5/5), done.
24Delta compression using up to 8 threads
25Compressing objects: 100% (3/3), done.
26Writing objects: 100% (3/3), 1017 bytes | 508.00 KiB/s, done.
27Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
28remote: Resolving deltas: 100% (1/1), completed with 1 local object.
29remote:
30remote: Create a pull request for 'feature' on GitHub by visiting:
31remote:   https://github.com/Waseem7839/Book-My>Show/pull/new/feature
32remote:
33To https://github.com/Waseem7839/Book-My>Show.git
34 * [new branch]      feature -> feature
  
```

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

The screenshot shows a GitHub interface for comparing branches. At the top, it says "Comparing changes" and "base: main" vs "compare: feature". It indicates 1 commit, 1 file changed, and 1 contributor. Below this, a commit from "Shak Mohammad Waseem Akram" on Sep 12, 2025, is shown. The commit message is "Read me file updated". The diff shows one addition and zero deletions to the README.md file. The file content includes "# Book My Show - DevOps CI/CD Project". There are buttons for "Split" and "Unified" view.

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

The screenshot shows a GitHub pull request merge interface. The title is "Read me file updated #1". It shows "Waseem7839 wants to merge 1 commit into main from feature". The commit message is "Read me file updated". A note says "No conflicts with base branch" and "Merging can be performed automatically". A "Merge pull request" button is present. To the right, there are sections for "Reviewers" (no reviews), "Assignees" (no assignees), "Labels" (none yet), "Projects" (none yet), and "Milestone" (no milestone). There is also an "Edit" button and a "Code" dropdown.

➤ **Merged PR confirmation.**

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

The screenshot shows a GitHub pull request page for a repository named 'Book-My-Show'. The pull request is titled 'Read me file updated #1' and has been merged by 'Waseem7839' 7 minutes ago. The merge commit hash is 964d50c. The pull request was successfully merged and closed. On the right side, there are sections for Reviewers, Assignees, Labels, Projects, and Milestone, all of which are currently empty. The bottom of the page has a button to 'Add a comment'.

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

The screenshot shows a Jira board for the project 'Book-My-Show-project'. The board has three columns: TO DO, IN PROGRESS, and DONE. The 'TO DO' column contains stories for Jenkins Setup, CI/CD Pipeline, Docker Deployment, and Kubernetes Deployment. The 'IN PROGRESS' column contains stories for BMSP-3, BMSP-4, BMSP-5, and BMSP-6. The 'DONE' column contains stories for JIRA & PROJECT SETUP (BMSP-1) and GITHUB WORKFLOW (BMSP-2). The Jira sidebar shows the user has access to the 'BMSP board'.

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.

The screenshot shows the AWS EC2 Instances page. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances (selected), Images, and Elastic Block Store. The main area displays a table of instances. The columns include Name, Instance ID, Instance state, Instance type, Status check, Alarm status, Availability Zone, and Public IPv4. Three instances are listed:

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 |
|------------------|---------------------|----------------|---------------|-------------------|-----------------------------|-------------------|-------------|
| jenkins-veera | i-04c505ad932f2a5cb | Running | t2.large | 2/2 checks passed | View alarms | eu-west-2b | ec2-13-41-1 |
| Srilatha-jenkins | i-073b623f503666cd6 | Running | t2.medium | 2/2 checks passed | View alarms | eu-west-2b | ec2-52-56-2 |
| Waseem-jenkins | i-05a4ea68e9ad5024b | Running | t2.large | 2/2 checks passed | View alarms | eu-west-2b | ec2-18-175 |

Below the table, a specific instance named "i-05a4ea68e9ad5024b (Waseem-jenkins)" is selected. The "Security" tab is active, showing details like IAM Role (none), Owner ID (909688465000), and Launch time (Sat Sep 13 2025 12:55:46 GMT+0530 (India Standard Time)).

Ports

The screenshot shows the AWS Security Groups page. The left sidebar includes options for EC2, Instances, Images, and Elastic Block Store. The main area shows a success message: "Inbound security group rules successfully modified on security group (sg-0332712a7dbebba34 | launch-wizard-1)". Below this, the "Inbound rules" tab is selected, showing a table of 8 rules. The columns are Name, Security group rule ID, IP version, Type, Protocol, and Port range.

| Name | Security group rule ID | IP version | Type | Protocol | Port range |
|------|------------------------|------------|------------|----------|---------------|
| - | sgr-071cd76e50546f085 | IPv4 | Custom TCP | TCP | 3000 - 10000 |
| - | sgr-0e96ef9214a89164c | IPv4 | SMTPS | TCP | 465 |
| - | sgr-0faaa87c41b7f47bb | IPv4 | Custom TCP | TCP | 30000 - 32767 |
| - | sgr-051a4bbaf2b487ed8 | IPv4 | SSH | TCP | 22 |
| - | sgr-0dec5544cc54ef258 | IPv4 | Custom TCP | TCP | 6443 |
| - | sgr-090ebb9a3a3103ef3 | IPv4 | HTTP | TCP | 80 |
| - | sgr-0bd10fffd3c14984f | IPv4 | HTTPS | TCP | 443 |
| - | sgr-020a9cc94bd487f83 | IPv4 | SMTP | TCP | 25 |

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

The screenshot shows the Jenkins user profile interface. At the top, there's a navigation bar with links like 'VIT-AP - VTOP', 'GitHub', 'LinkedIn', 'Mail', 'Course: Wipro_Dev...', 'YouTube', 'DeepSeek', 'Google', 'ChatGPT', 'My Drive - Google...', 'Jira Atlassian', 'Naukri', 'Sonarcube', and 'All Bookmarks'. Below the bar, the user's profile picture and name 'Shaik Mohammad Waseem Akram' are displayed. A sidebar on the left lists account management options: Status (selected), Builds, My Views, Account, Appearance, Preferences, Security, Experiments, and Credentials. The main content area shows the Jenkins User ID: Waseem. On the right, there are buttons for 'Add description', a search icon, and a gear icon. At the bottom right, it says 'REST API' and 'Jenkins 2.516.2'.

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

The screenshot shows the Jenkins plugin manager interface. The top navigation bar includes links to 'Manage Jenkins' and 'Plugins'. The main content area is titled 'Download progress' and shows a table comparing 'Preparation' steps with 'Download progress' for various Jenkins plugins. The preparation steps listed are: Checking internet connectivity, Checking update center connectivity, and Success. The download progress table shows green checkmarks next to each plugin name, indicating success for all listed items. The table includes columns for Preparation and Download progress, and rows for commons-lang3 v3.x Jenkins API, Ionicons API, Folders, OWASP Markup Formatter, ASM API, JSON Path API, Structs, Pipeline: Step API, commons-text API, Token Macro, Build Timeout, bouncycastle API, Credentials, Plain Credentials, Variant, SSH Credentials, and Credentials Binding.

➤ Credential Configuration:

Configured credentials for:

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

The screenshot shows the Jenkins 'Credentials' management interface. At the top, there's a header with tabs like 'Manage Jenkins' and 'Credentials'. Below the header is a table titled 'Credentials' with columns: Type, Provider, Store, Domain, ID, and Name. Three entries are listed:

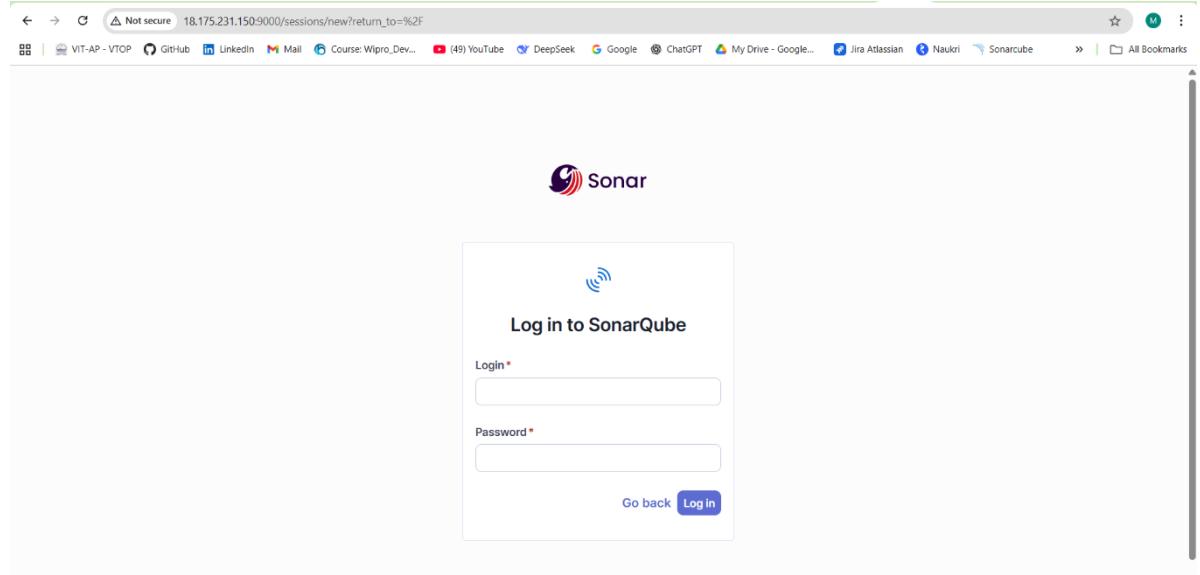
| Type | Provider | Store | Domain | ID | Name |
|--------|----------|----------|----------------------------------|-----------------|--|
| System | System | (global) | waseem951/***** | docker_creds | waseem951/***** |
| System | System | (global) | sonarqube-token | sonarqube-token | sonarqube-token |
| System | System | (global) | akramwaseem78690@gmail.com/***** | email-creds | akramwaseem78690@gmail.com/***** (email-creds) |

Below the table, there's a section titled 'Stores scoped to Jenkins' which lists 'System' and 'Kubernetes' under 'Domains'. At the bottom right, it says 'REST API Jenkins 2.516.2'.

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

The screenshot shows a Gmail inbox with one unread email. The subject of the email is 'Test email #1'. The message body contains the text: 'address not configured yet <akramwaseem78690@gmail.com> to me'. Below the message are standard Gmail interaction buttons: Reply, Forward, and a reply icon.

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



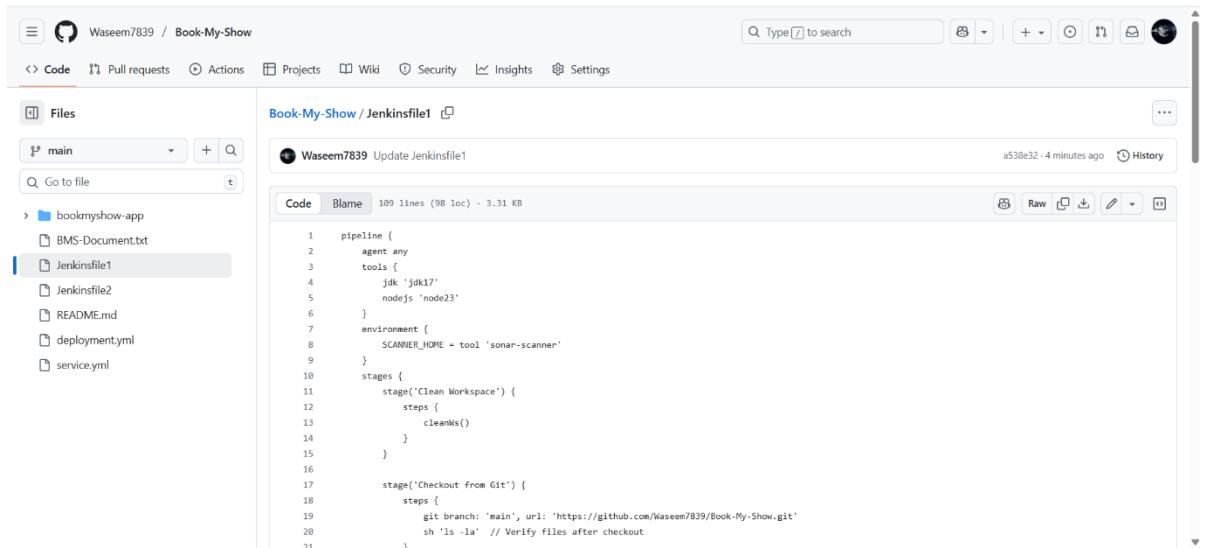
- **Task completion:** The *Jenkins Setup* task (BMSP-3) is now marked as **Done** on the Jira

| Column | Task | Status |
|-------------|----------------------------|-------------|
| TO DO | CI/CD Pipeline | Pending |
| TO DO | Docker Deployment | Pending |
| TO DO | Kubernetes Deployment | Pending |
| TO DO | Monitoring & Observability | Pending |
| IN PROGRESS | | In Progress |
| DONE | Jenkins Setup | Completed |
| DONE | CI/CD Pipeline | Completed |
| DONE | Docker Deployment | Completed |
| DONE | Kubernetes Deployment | Completed |
| DONE | Monitoring & Observability | Completed |

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.



The screenshot shows a GitHub repository named 'Book-My-Show' owned by 'Waseem7839'. The 'Jenkinsfile1' file is displayed. The code defines a Jenkins pipeline with the following stages:

```
1 pipeline {
2     agent any
3     tools {
4         jdk 'jdk17'
5         nodejs 'node23'
6     }
7     environment {
8         SCANNER_HOME = tool 'sonar-scanner'
9     }
10    stages {
11        stage('Clean Workspace') {
12            steps {
13                cleanWs()
14            }
15        }
16        stage('Checkout from Git') {
17            steps {
18                git branch: 'main', url: 'https://github.com/Waseem7839/Book-My-Show.git'
19                sh 'ls -la' // Verify files after checkout
20            }
21        }
22    }
23}
```

- **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'
        )
    }
}
}

```

The screenshot shows the Jenkins Pipeline interface for the 'Waseem-BookMyShow' pipeline. On the left, a sidebar lists various pipeline management options like Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, SonarQube, Stages, Rename, Pipeline Syntax, and Credentials. The main area displays the 'Stage View' which includes a timeline showing the execution of stages: 'Clean Workspace' (277ms), 'Checkout from Git' (1s), 'SonarQube Analysis' (28s), 'Quality Gate' (452ms), 'Install Dependencies' (1min 56s), 'Docker Build & Push' (7min 33s), 'Deploy to Container' (7s), and 'Declarative: Post Actions' (351ms). A tooltip for the 'Logs' button indicates 'Declarative: Successfull'. Below the stage view is the 'SonarQube Quality Gate' section, which shows a green 'Passed' status with a 'Success' message. It also lists recent builds: 'Last build (#5), 10 min ago', 'Last stable build (#5), 10 min ago', and 'Last successful build (#5), 10 min ago'. A 'Builds' summary table on the left shows one build (#5) completed at 2:38 PM today.

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

The screenshot shows the Docker Hub repository page for `waseem951/bookmyshow`. The left sidebar includes links for Repositories, Collaborations, Settings, Default privacy, Notifications, Billing, Usage, Pulls, and Storage. The main content area shows the repository details: 'waseem951/bookmyshow' was last pushed 6 minutes ago with a repository size of 714 MB. It features tabs for General, Tags, Image Management (Beta), Collaborators, Webhooks, and Settings. Under the General tab, there's a 'Tags' section showing one tag: 'latest' (OS: Image, Type: Image, Pulled: less than 1 day, Pushed: 7 minutes). A note indicates 'DOCKER SCOUT INACTIVE' with a 'Activate' link. To the right, there's a 'Docker commands' section with a 'Public view' button and a command line snippet: `docker push waseem951/bookmyshow:tagname`. A 'buildcloud' sidebar promotes Docker Build Cloud, stating it accelerates image build times with cloud-based builders and shared cache. It also mentions Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation, and provides faster builds through shared caching across the team.

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

The screenshot shows the SonarQube Quality Gates page. At the top, there is a warning message: "Embedded database should be used for evaluation purposes only. It doesn't support scaling, upgrading to a new SonarQube Server version, or migration to another database engine. Learn more." Below this, a note says: "The way in which security, reliability, and maintainability counts and ratings are calculated has changed. Learn more in SonarQube documentation." The main content area displays the following information:

- Perspective:** Overall Status
- Sort by:** Name
- Projects:** 1 project(s) - BMS
- Status:** Passed
- Last analysis:** 11 minutes ago - 5.8k Lines of Code - JavaScript, CSS, ...
- Metrics:**
 - Security: 2 Open issues (C)
 - Reliability: 167 Open issues (D)
 - Maintainability: 275 Open issues (A)
 - Hotspots Reviewed: 0.0%
 - Coverage: 0.0%
 - Duplications: 1.4%
- Filters:**
 - Quality Gate:** Passed (1), Failed (0)
 - Security:**
 - A: ≥ 0 info issues (0)
 - B: ≥ 1 low issue (0)
 - C: ≥ 1 medium issue (1)
 - D: ≥ 1 high issue (0)
 - E: ≥ 1 blocker issue (0)

At the bottom, there are links for SonarQube technology, Community Build, LGPL v3, and various documentation and plugin links.

This screenshot shows the detailed view of the Quality Gate results. The main header indicates "Passed". The "Overall Code" tab is selected. The results are summarized as follows:

| Category | Count | Rating |
|-----------------|-----------------|--------|
| Security | 2 Open issues | C |
| Reliability | 167 Open issues | D |
| Maintainability | 275 Open issues | A |
| Accepted issues | 0 | |
| Coverage | 0.0% | |
| Duplications | 1.4% | |

Below this, there are sections for "New Code" and "Security Hotspots". The "Last analysis" was 19 minutes ago. Project settings and information are also visible at the top right.

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

The screenshot shows a Jira board titled "BMSP board". The board is divided into three columns: "TO DO", "IN PROGRESS", and "DONE".

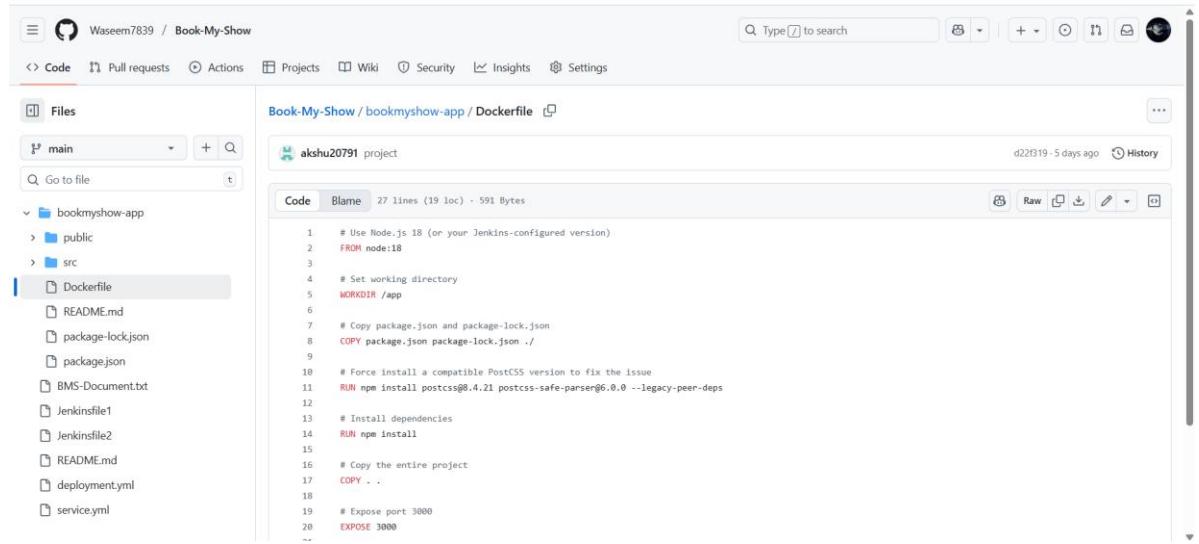
- TO DO:**
 - DOCKER DEPLOYMENT
 - Kubernetes Deployment
 - Monitoring & Observability
 - Final Deliverables
- IN PROGRESS:**
 - JIRA & PROJECT SETUP
 - GITHUB WORKFLOW
 - JENKINS SETUP
 - CI/CD PIPELINE
- DONE:**
 - BMSP-1
 - BMSP-2
 - BMSP-3
 - BMSP-4
 - BMSP-5
 - BMSP-6
 - BMSP-7
 - BMSP-8

All tasks in the "DONE" column are marked as 100% complete.

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.



A screenshot of a GitHub repository interface. The repository is named 'Book-My-Show'. The left sidebar shows a tree view of files: main, bookmyshow-app (which contains public, src, and Dockerfile), README.md, package-lock.json, package.json, BMS-Document.txt, Jenkinsfile1, Jenkinsfile2, README.md, deployment.yml, and service.yml. The right pane displays the 'Dockerfile' content under the 'bookmyshow-app' folder. The code is as follows:

```
1 # Use Node.js 18 (or your Jenkins-configured version)
2 FROM node:18
3
4 # Set working directory
5 WORKDIR /app
6
7 # Copy package.json and package-lock.json
8 COPY package.json package-lock.json ./ 
9
10 # Force install a compatible PostCSS version to fix the issue
11 RUN npm install postcss@8.4.21 postcss-safe-parser@6.0.0 --legacy-peer-deps
12
13 # Install dependencies
14 RUN npm install
15
16 # Copy the entire project
17 COPY . .
18
19 # Expose port 3000
20 EXPOSE 3000
~~
```

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

```

59
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>

The screenshot shows the Docker Hub interface for the repository `waseem951/bookmyshow`. The repository has one tag, `latest`, which was pushed 7 minutes ago. The Docker Build Cloud sidebar is present, advertising faster builds through shared caching across teams.

- **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.

The screenshot shows the BookMyShow clone application running locally. It features a search bar, navigation links like 'Movies', 'Stream', 'Events', etc., and a 'Select City' dropdown. Two movie banners are displayed: 'MASALA SANDWICH' (3rd April, 2021) and 'JAB WE SEPARATED' (20TH-21ST MARCH). Below the banners, there's a 'Recommended Movies' section and a 'The Best of Entertainment' section featuring news from various sources.

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

Jira Board Screenshot:

- TO DO:**
 - BMSP-7
 - BMSP-8
 - BMSP-9
- IN PROGRESS:**
 - BMSP-10
 - BMSP-5
- DONE:**
 - BMSP-2
 - Jenkins Setup
 - BMSP-3
 - CI/CD Pipeline
 - Docker Deployment

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.

```

provider "aws" {
  region = "eu-west-2"
}

terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "> 5.0.0"
    }
  }
}

module "eks" {
  module "managed_node_group"["default"] {
    aws_eks_node_group.this[0] = {
      name = "team4-eks-cluster"
      role_arn = "arn:aws:iam::123456789012:role/eks-node-group-role"
      subnet_ids = ["subnet-00000000", "subnet-00000001"]
      vpc_id = "vpc-00000000"
    }
  }
}

module "aws_eksAddon" {
  module "coredns" {
    provider = "aws"
    kube_proxy = true
  }
}

```

TERMINAL OUTPUT:

```

DELL@Waseem MINIGM64 /d/DevOps/EKS_creation
$ terraform apply
module.eks.module.eks_managed_node_group["default"].aws_eks_node_group.this[0]: Still creating... [0m00s elapsed]
module.eks.module.eks_managed_node_group["default"].aws_eks_node_group.this[0]: Creation complete after 1m54s [id=team4-eks-cluster:default-20250914061344352000000001]
module.eks.aws_eksAddon.this["coredns"]: Creating...
module.eks.aws_eksAddon.this["coredns"]: Still creating... [0m00s elapsed]
module.eks.aws_eksAddon.this["coredns"]: Still creating... [0m00s elapsed]
module.eks.aws_eksAddon.this["coredns": Creation complete after 17s [id=team4-eks-cluster:coredns]
module.eks.aws_eksAddon.this["coredns"]: Still creating... [0m00s elapsed]
module.eks.aws_eksAddon.this["coredns"]: Still creating... [0m00s elapsed]
module.eks.aws_eksAddon.this["coredns": Creation complete after 28s [id=team4-eks-cluster:kube-proxy]
module.eks.module.eks_managed_node_group["default"].aws_eks_node_group.this[0]: Destroying... [id=team4-eks-cluster:default-2025091406063958260000000a]
module.eks.module.eks_managed_node_group["default"].aws_eks_node_group.this[0]: Destruction complete after 9s

Apply complete! Resources: 3 added, 0 changed, 1 destroyed.

DELL@Waseem MINIGM64 /d/DevOps/EKS_creation
$ aws eks update-kubeconfig --name team4-eks-cluster --region eu-west-2
Updated context 'arn:aws:eks:eu-west-2:909688465000:cluster/team4-eks-cluster' in C:\Users\DELL\.kube\config

```

- **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.

```

File Edit Selection View Go Run Terminal Help ← → ⌘ EKS_creation
EXPLORER ... Welcome main.tf
EKS_CREATION > .terraform & .terraform.lock.hcl
main.tf
terraform.tfstate & terraform.state.back...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SONARQUBE
DELL@Waseem MINGW64 /d/DevOps/EKS_creation
$ aws eks update-kubeconfig --name team4-eks-cluster --region eu-west-2
Updated context: arn:aws:eks:eu-west-2:909688465000:cluster/team4-eks-cluster in C:\Users\DELL\.kube\config

DELL@Waseem MINGW64 /d/DevOps/EKS_creation
$ aws eks update-kubeconfig --name team4-eks-cluster --region eu-west-2

DELL@Waseem MINGW64 /d/DevOps/EKS_creation
$ kubectl get nodes
NAME STATUS ROLES AGE VERSION
ip-10-0-0-67.eu-west-2.compute.internal Ready <none> 7m6s v1.30.14-eks-3abbec1
ip-10-0-1-49.eu-west-2.compute.internal Ready <none> 7m9s v1.30.14-eks-3abbec1

DELL@Waseem MINGW64 /d/DevOps/EKS_creation
$ kubectl get pods -A
NAMESPACE NAME READY STATUS RESTARTS AGE
kube-system aws-node-5ktq8 2/2 Running 0 7m27s
kube-system aws-node-6wtjs 2/2 Running 0 7m30s
kube-system coredns-db47cdcbc-2dv5r 1/1 Running 0 6m51s
kube-system coredns-db47cdcbc-z5csw 1/1 Running 0 6m51s
kube-system eks-pod-identity-agent-8j6sf 1/1 Running 0 7m27s
kube-system eks-pod-identity-agent-nzmpp 1/1 Running 0 7m30s
kube-system kube-proxy-kjnsx 1/1 Running 0 6m52s
kube-system kube-proxy-vvv2w 1/1 Running 0 6m52s
DELL@Waseem MINGW64 /d/DevOps/EKS_creation

```

- **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/>" +
```

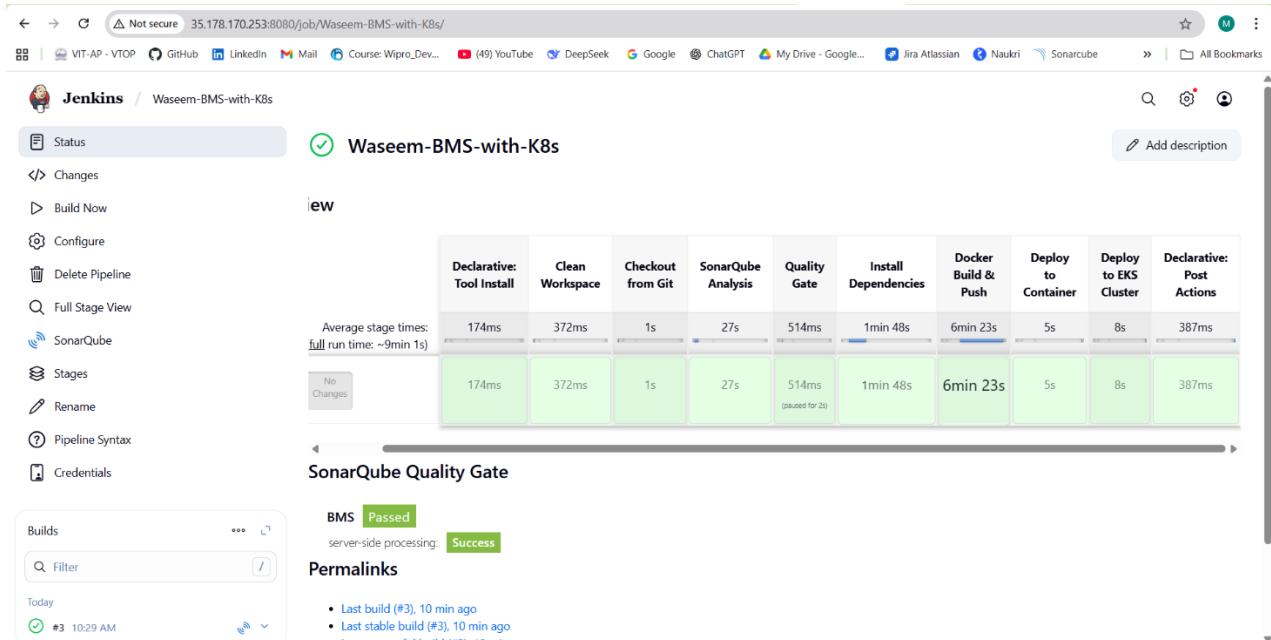
```
            "URL: ${env.BUILD_URL}<br/>,"
```

```
        to: 'akramwaseem78690@gmail.com'
```

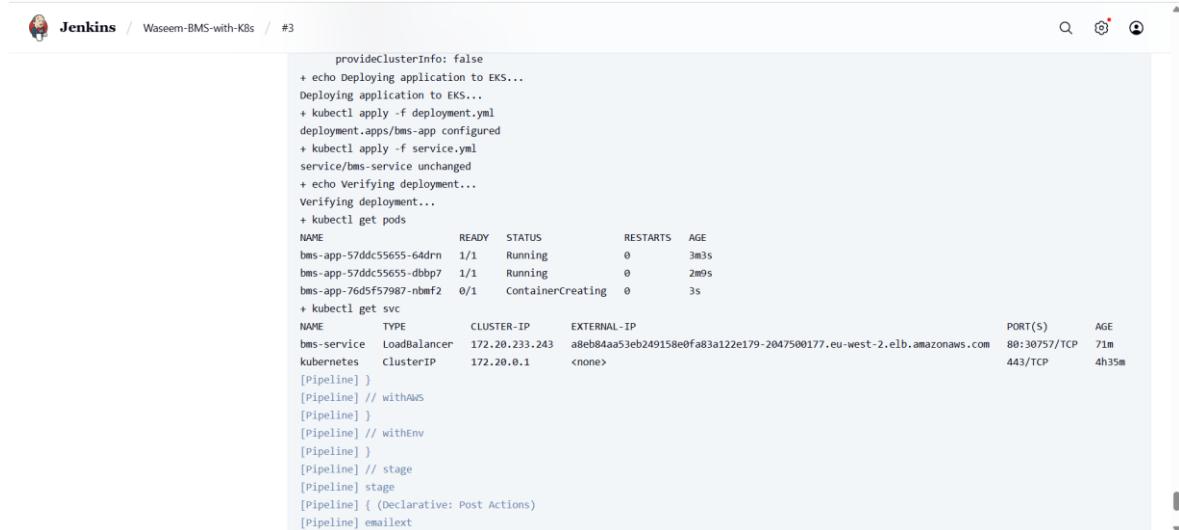
```
    }
```

```
}
```

```
}
```



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

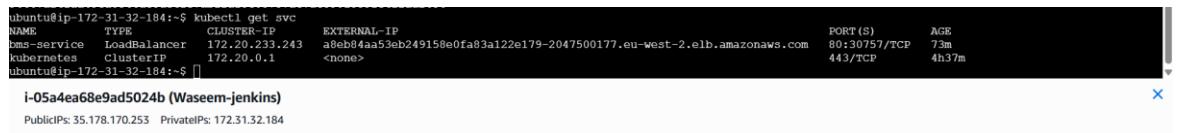


```

provideClusterInfo: false
+ echo Deploying application to EKS...
Deploying application to EKS...
+ kubectl apply -f deployment.yaml
deployment.apps/bms-app configured
+ kubectl apply -f service.yaml
service/bms-service unchanged
+ echo Verifying deployment...
Verifying deployment...
+ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
bms-app-57ddc55655-64drn  1/1     Running   0          3m3s
bms-app-57ddc55655-dbfp7  1/1     Running   0          2m9s
bms-app-76d5f57987-nbmf2  0/1     ContainerCreating   0          3s
+ kubectl get svc
NAME        TYPE      CLUSTER-IP      EXTERNAL-IP           PORT(S)        AGE
bms-service  LoadBalancer  172.20.233.243  a8eb84aa53eb249158e0fa83a122e179-2047500177.eu-west-2.elb.amazonaws.com  80:30757/TCP  71m
kubernetes   ClusterIP   172.20.0.1    <none>               443/TCP       4h35m
[Pipeline]
[Pipeline] // withAns
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] emailext

```

- **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.



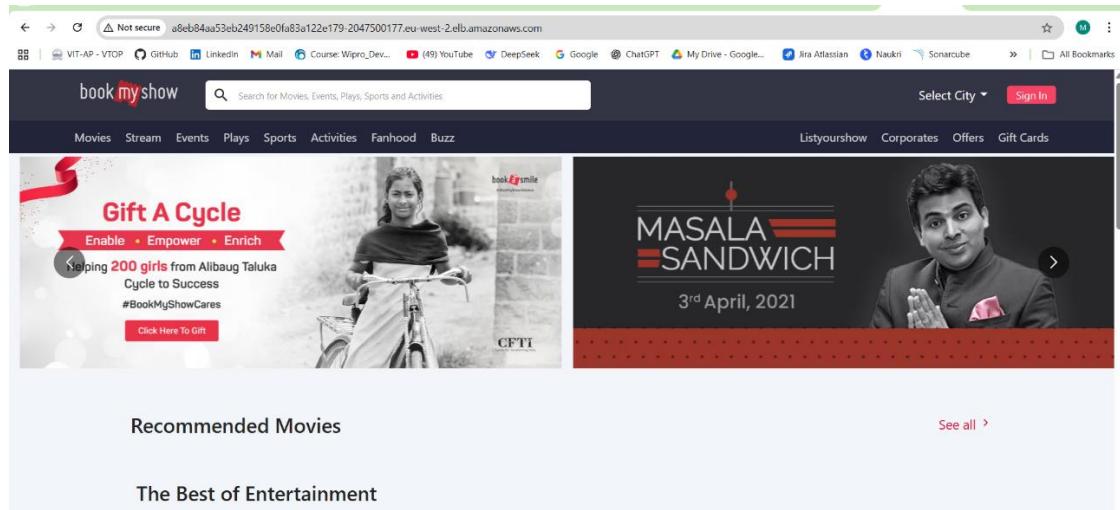
```

ubuntu@ip-172-31-32-184:~$ kubectl get svc
NAME        TYPE      CLUSTER-IP      EXTERNAL-IP           PORT(S)        AGE
bms-service  LoadBalancer  172.20.233.243  a8eb84aa53eb249158e0fa83a122e179-2047500177.eu-west-2.elb.amazonaws.com  80:30757/TCP  73m
kubernetes   ClusterIP   172.20.0.1    <none>               443/TCP       4h37m
ubuntu@ip-172-31-32-184:~$ 

```

i-05a4ea68e9ad5024b (Waseem-jenkins)
PublicIPs: 35.178.170.253 PrivateIPs: 172.31.32.184

- **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.

The screenshot shows the SonarQube interface for managing quality gates. The URL is 35.178.170.253:9000/quality_gates/show/AZlHr2oRXq17_OoY6KGy. The top navigation bar includes links for VIT-AP - VTOP, GitHub, LinkedIn, Mail, Course: Wipro_Dev..., YouTube, DeepSeek, Google, ChatGPT, My Drive - Google..., Jira Atlassian, Naukri, Sonarcube, and All Bookmarks. A message at the top states: "You're running a version of SonarQube that is no longer active. Please upgrade to an active version immediately." Below this is a "Learn More" button.

The main content area shows the "Quality Gates" section for the "Sonar way" built-in quality gate. It has tabs for "Create", "Sonar way", "DEFAULT", and "BUILT-IN". The "Sonar way" tab is selected. A green box highlights the methodology: "This quality gate complies with Clean as You Code". It explains that it ensures no new bugs, vulnerabilities, or security hotspots are introduced. Below this, there's a "Conditions" section with a table defining various metrics and their operators:

| Metric | Operator | Value |
|----------------------------|-----------------|--|
| Coverage | is less than | 80.0% |
| Duplicated Lines (%) | is greater than | 3.0% |
| Maintainability Rating | is worse than | A (Technical debt ratio is less than 5.0%) |
| Reliability Rating | is worse than | A (No bugs) |
| Security Hotspots Reviewed | is less than | 100% |

- **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.

The screenshot shows the Jenkins dashboard. The top navigation bar includes links for New Item, Build History, and Add description. The main content area displays the "Build Queue" and "Build Executor Status". The "Build Queue" section shows "No builds in the queue." The "Build Executor Status" section shows 0/2 executors available. Below these are two pipeline entries in a table:

| S | W | Name ↓ | Last Success | Last Failure | Last Duration |
|---|---|---------------------|---------------|--------------|---------------|
| | | Waseem-BMS-with-K8s | 3 hr 8 min #3 | N/A | 9 min 1 sec |
| | | Waseem-BookMyShow | 22 hr #5 | N/A | 10 min |

At the bottom, there are icons for S, M, and L, and links for REST API and Jenkins 2.516.2.

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

The screenshot shows a Jira board for the 'Book-My-Show-project' under the 'BMSP board'. The board has three columns: TO DO, IN PROGRESS, and DONE. The TO DO column contains tasks like 'Configure workflow (TO DO → In Progress → Done)' and 'JIRA & PROJECT SETUP' (with sub-tasks BMSP-10 and BMSP-11). The IN PROGRESS column contains tasks like 'Fork Book-My-Show GitHub repository' (BMSP-12) and 'Clone repository locally' (BMSP-13). The DONE column contains tasks like 'Docker Deployment' (BMSP-5) and 'Kubernetes Deployment' (BMSP-6). The sidebar on the left shows recent projects like 'Book-My-Show-project' and 'BMSP 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

```

prometheus-2.47.1.linux-amd64/console_libraries/
prometheus-2.47.1.linux-amd64/console_libraries/prom.lib
prometheus-2.47.1.linux-amd64/console_libraries/menu.lib
prometheus-2.47.1.linux-amd64/prometheus
ubuntu@ip-172-31-32-220:~$ sudo mkdir -p /data /etc/prometheus
ubuntu@ip-172-31-32-220:~$ cd prometheus-2.47.1.linux-amd64/
ubuntu@ip-172-31-32-220:~/prometheus-2.47.1.linux-amd64$ pwd
/home/ubuntu/prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-32-220:~/prometheus-2.47.1.linux-amd64$ sudo mv prometheus promtool /usr/local/bin/
ubuntu@ip-172-31-32-220:~/prometheus-2.47.1.linux-amd64$ sudo mv consoles/ console_libraries/ /etc/prometheus/
ubuntu@ip-172-31-32-220:~/prometheus-2.47.1.linux-amd64$ sudo mv prometheus.yml /etc/prometheus/prometheus.yml
ubuntu@ip-172-31-32-220:~/prometheus-2.47.1.linux-amd64$ sudo chown -R prometheus:prometheus /etc/prometheus /data/
ubuntu@ip-172-31-32-220:~/prometheus-2.47.1.linux-amd64$ cd
ubuntu@ip-172-31-32-220:~$ ls
prometheus-2.47.1.linux-amd64  prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-32-220:~$ rm -rf prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-32-220:~$ ls
prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-32-220:~$ prometheus --version
prometheus version 2.47.1 (branch: HEAD, revision: c4dia8beff37cc004f1dc4ab9d2e73193f51aaeb)
  build user:          root@4829330363be
  build date:         2023/04/10 13:16
  go version:        go1.21.1
  platform:          linux/amd64
  tags:              netgo osusergo static_build
ubuntu@ip-172-31-32-220:~$ 

```

i-0702973267ac0092f (waseem-monitoring)
PublicIPs: 18.132.49.221 PrivateIPs: 172.31.32.220

- **Node Exporter Installation done**

```

ubuntu@ip-172-31-32-220:~$ node exporter --version
node_exporter, version 1.6.1 (branch: HEAD, revision: 4a1b77600c1873a0233f3ffb55afcedbb63b8d84)
  build user:          root@586879db1e5
  build date:         2023/07/12 10:52
  go version:        go1.20.6
  platform:          linux/amd64
  tags:              netgo osusergo static_build
ubuntu@ip-172-31-32-220:~$ 

```

i-0702973267ac0092f (waseem-monitoring)
PublicIPs: 18.132.49.221 PrivateIPs: 172.31.32.220

- **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.

The screenshot shows the Prometheus Targets page at the URL `18.132.49.221:9090/targets?search=`. The page displays three tables of targets, each with columns for Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error.

| Targets | | All scrape pools ▾ | All Unhealthy Collapse All | Filter by endpoint or labels | <input checked="" type="checkbox"/> Unknown <input checked="" type="checkbox"/> Unhealthy <input checked="" type="checkbox"/> Healthy |
|--|-------|--|----------------------------|------------------------------|---|
| jenkins (1/1 up) <small>show less</small> | | | | | |
| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
| <code>http://35.178.170.253:8080/prometheus</code> | UP | <code>instance="35.178.170.253:8080"</code> job="jenkins" | 12.278s ago | 11.510ms | |
| node_exporter (1/1 up) <small>show less</small> | | | | | |
| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
| <code>http://18.132.49.221:9100/metrics</code> | UP | <code>instance="18.132.49.221:9100"</code> job="node exporter" | 21.588s ago | 15.515ms | |
| prometheus (1/1 up) <small>show less</small> | | | | | |
| Endpoint | State | Labels | Last Scrape | Scrape Duration | Error |
| <code>http://localhost:9090/metrics</code> | UP | <code>instance="localhost:9090"</code> job="prometheus" | 13.235s ago | 5.291ms | |

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

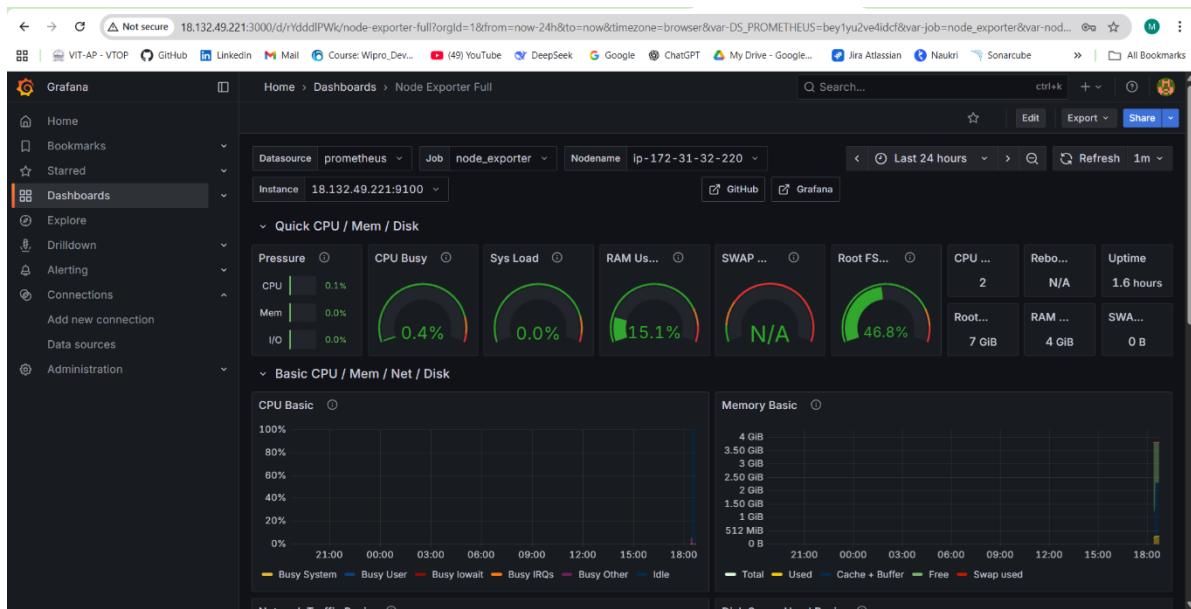
The screenshot shows the Grafana Connections > Data sources page at the URL `18.132.49.221:3000/connections/datasources`. The page lists a single data source named "prometheus" which is a Prometheus connection to `http://18.132.49.221:9090`. The "default" button is highlighted, indicating it is the selected default data source.

➤ Grafana Dashboards:

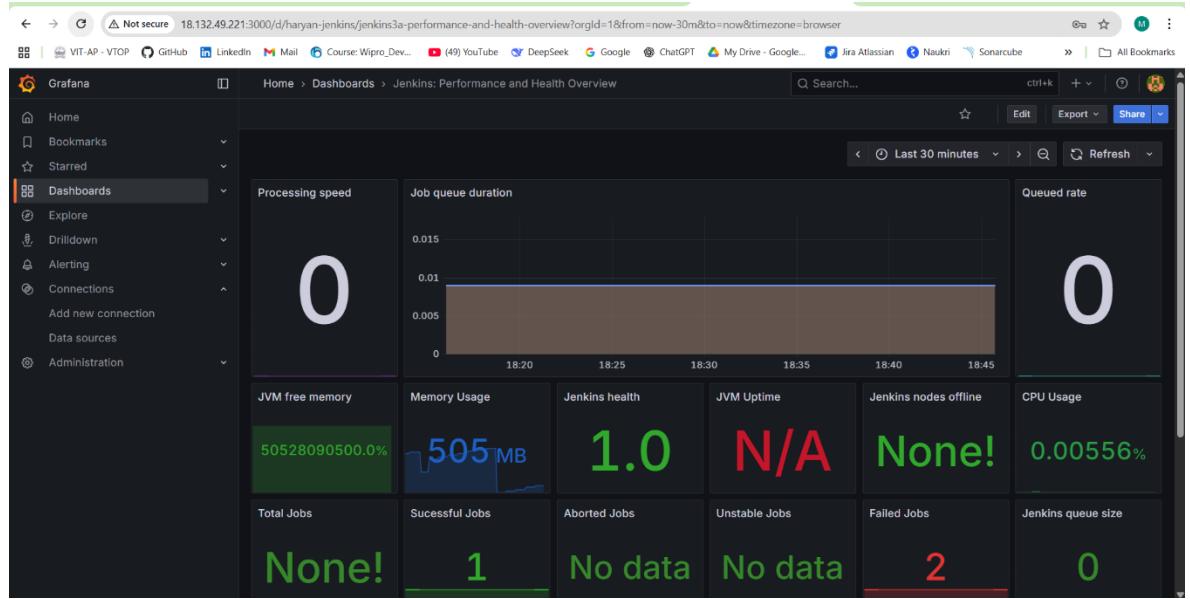
- **Jenkins: Performance and Health Overview** – for tracking Jenkins performance.
- **Node Exporter Full** – for system-level monitoring of CPU, memory, disk, and network.

The screenshot shows the Grafana web interface. The left sidebar is dark-themed and includes links for Home, Bookmarks, Starred, Dashboards (which is selected), Explore, Drilldown, Alerting, Connections, Data sources, and Administration. The main content area has a light background and displays the 'Dashboards' section. At the top of this section is a search bar with placeholder text 'Search for dashboards and folders'. Below it is a filter section with 'Filter by tag' and a 'Starred' checkbox. A table lists two dashboards: 'Jenkins: Performance and Health Overview' and 'Node Exporter Full'. The 'Node Exporter Full' dashboard is associated with the 'linux' tag. The right side of the interface includes standard browser controls like back, forward, and refresh, along with a search bar and a 'ctrl+k' keyboard shortcut.

➤ 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 5
# 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 3.7924e+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.

The screenshot shows a Jira board titled "BMSP board". The board has three columns: "TO DO", "IN PROGRESS", and "DONE".

- TO DO:**
 - Fork Book-My-Show GitHub repository (GITHUB WORKFLOW)
 - Clone repository locally (GITHUB WORKFLOW)
 - Create feature branch for development (GITHUB WORKFLOW)
 - Make code changes (update README / minor fix) (GITHUB WORKFLOW)
- IN PROGRESS:**
 - BMSP-12
 - BMSP-13
 - BMSP-14
- DONE:**
 - Docker Deployment (DOCKER DEPLOYMENT)
 - Kubernetes Deployment (KUBERNETES DEPLOYMENT)
 - Monitoring & Observability (MONITORING & OBSERVABILITY)

8. Email notification:

The screenshot shows a Gmail inbox with one unread email. The email is from "address not configured yet <akramwaseem78690@gmail.com>" with the subject "SUCCESS". The message body contains the following text:

```

address not configured yet <akramwaseem78690@gmail.com>
to me

The build completed SUCCESSFULLY.

Project: Waseem_BookMyShow
BuildNumber: 5
URL: http://18.175.231.150:8080/job/Waseem_BookMyShow/5
  
```

Below the message are standard Gmail interaction buttons: Reply, Forward, and a smiley face icon.

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

The screenshot shows the Jira board interface for the 'BMSP board'. The board has three columns: 'IN PROGRESS', 'TO DO', and 'DONE'. In the 'TO DO' column, there is a task titled 'BMSP-8' which is currently marked as 'In Progress'. Other tasks in the 'DONE' column include 'Jira & Project Setup', 'Submit EKS proof', 'Submit Kubernetes manifests', and 'Submit Prometheus & Grafana screenshots', all of which are completed.

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

The screenshot shows the Jira list view for the 'BMSP board'. The table displays eight tasks, each with a green 'DONE' status indicator. The tasks are: BMSP-1 (Jira & Project Setup), BMSP-2 (GitHub Workflow), BMSP-4 (CI/CD Pipeline), BMSP-3 (Jenkins Setup), BMSP-5 (Docker Deployment), BMSP-6 (Kubernetes Deployment), BMSP-7 (Monitoring & Observability), and BMSP-8 (Final Deliverables). All tasks are assigned to 'Waseem Sk' and are part of 'BMSP Sprint 1'.

| Type | Key | Summary | Status | Comments | Sprint | Assignee |
|------|--------|----------------------------|--------|-------------|---------------|-----------|
| | BMSP-1 | Jira & Project Setup | DONE | Add comment | BMSP Sprint 1 | Waseem Sk |
| | BMSP-2 | Github Workflow | DONE | Add comment | BMSP Sprint 1 | Waseem Sk |
| | BMSP-4 | CI/CD Pipeline | DONE | Add comment | BMSP Sprint 1 | Waseem Sk |
| | BMSP-3 | Jenkins Setup | DONE | Add comment | BMSP Sprint 1 | Waseem Sk |
| | BMSP-5 | Docker Deployment | DONE | Add comment | BMSP Sprint 1 | Waseem Sk |
| | BMSP-6 | Kubernetes Deployment | DONE | Add comment | BMSP Sprint 1 | Waseem Sk |
| | BMSP-7 | Monitoring & Observability | DONE | Add comment | BMSP Sprint 1 | Waseem Sk |
| | BMSP-8 | Final Deliverables | DONE | Add comment | BMSP Sprint 1 | Waseem Sk |

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>