

Capstone Project— Book-My-Show DevOps Lifecycle

BATCH 1 – YASHASWINI K L

Step 1: Jira Workflow

- Each candidate must assign themselves tasks in Jira.
- Track task progress using Jira board (To Do → In Progress → Done).
- Export the board as part of the final submission.

Step 2: GitHub Workflow

- Clone the GitHub repository: <https://github.com/akshu20791/Book-My-Show/>
- Create a feature branch and make changes.
- Push changes and raise a Pull Request (PR).
- Review peers' PRs and approve before merging into the main branch.
- Submit the final PR link as deliverable.

Step 3: Jenkins CI/CD Pipeline

Pipeline stages to implement:

1. Clean Workspace
2. Checkout Code from GitHub
3. SonarQube Analysis (Quality Gate)
4. Install Dependencies (NPM)
5. Trivy FS Scan (Optional)
6. OWASP Dependency Check (Optional)
7. Docker Build & Push to DockerHub (via Jenkins)
8. Deploy to Docker Container
9. (Optional) Deploy to Kubernetes (EKS)
10. Email Notification on build result

Note: Candidates must implement this pipeline using a Jenkinsfile and submit the Jenkinsfile as a deliverable.

Step 4: Docker Deployment

- Candidates must write their own Dockerfile to build the BMS app image.
- Build Docker image and push to DockerHub via Jenkins.
- Run container locally and validate accessibility on port 3000.

Step 5: Kubernetes Deployment (EKS)

- Candidates must write their own Kubernetes manifests (`deployment.yaml`, `service.yaml`).
- Deploy the application on EKS cluster.
- Expose service using NodePort or LoadBalancer.
- Validate deployment using 'kubectl get pods' and 'kubectl get svc'.

Step 6: Monitoring & Observability

- Install Prometheus and Node Exporter to collect metrics.
- Integrate Jenkins metrics into Prometheus.
- Install Grafana, configure Prometheus as a data source.
- Add dashboards for Node health and Jenkins performance.
- Submit Grafana screenshots in deliverables.

Step 1: Jira Workflow

1. Create a Jira Project and Assign Tasks

- Log in to Jira: Access your Jira instance.
- Create a New Project:
 - Click on "Projects" in the top navigation bar.
 - Click "Create project."
 - Choose a Project Template: Select "Scrum" and click "Use template."
 - Select "Company-managed project"
 - Name your project: "Book-My-Show Project"
 - Key: Jira will automatically generate a short key ("BMSP").
 - Click "Create project."
- Define Epics
 - Click on "Projects" in the top navigation bar.
 - Go to your project's "Backlog" or "Board."
 - Click "Create" or the "+" button.
 - Issue Type: Select "Epic."
 - Summary:
 - "GitHub Workflow Setup"
 - "Jenkins CI/CD Pipeline"
 - "Docker Deployment"
 - "Kubernetes Deployment EKS"
 - "Monitoring & Observability Setup"
 - "Project Deliverables & Reporting"

The screenshot shows the Jira interface with the following details:

- Top Navigation:** Jira logo, back arrow, forward arrow, search icon, and user icon.
- Side Navigation:** For you, Recent, Starred, Apps, Plans, Projects (selected), Recent, Book-my-show ..., BMSP board (selected), More projects, Teams, and More.
- Project Header:** Projects / Book-my-show project, BMSP board, Summary, Timeline, Backlog (selected).
- Search Bar:** Search backlog, filter icons (Y, person, version).
- Epics List:** Epic (header), No epic, GitHub Workflow, Jenkins CI/CD Pipeline, Docker Deployment, Kubernetes Deployment (EKS), Monitoring & Observability, Final Deliverables, and a Create epic button.

- To create a task under an Epic:
 - Click "Create."
 - Issue Type: Select "Task."
 - Summary: Detail the specific action.
 - Description: Add any relevant notes or sub-steps.
 - Assignee: Assign it to yourself.
 - Reporter: Set yourself as the reporter.
 - Epic Link: Make sure to link it to the appropriate Epic.
 - Priority: Set a priority (e.g., Medium, High).

Epics, Stories & Tasks

Epic 1: GitHub Workflow

Story 1.1: Set up GitHub Workflow

- Task 1.1.1: Clone the GitHub repository <https://github.com/akshu20791/Book-My-Show/>
- Task 1.1.2: Create a feature branch and make required code changes
- Task 1.1.3: Push changes to GitHub and raise a Pull Request (PR)
- Task 1.1.4: Review PR (self/peer) and approve before merging
- Task 1.1.5: Submit final PR link as deliverable

Epic 2: Jenkins CI/CD Pipeline

Story 2.1: Set up Jenkins Environment

- Task 2.1.1: Install Jenkins and configure global tools (JDK, Git, NodeJS)
- Task 2.1.2: Install required plugins (Git, Pipeline, Blue Ocean, SonarQube, Docker, Email Extension, etc.)
- Task 2.1.3: Configure Jenkins credentials (GitHub token, DockerHub credentials, SonarQube token)

Story 2.2: Implement Pipeline Stages (Jenkinsfile)

- Task 2.2.1: Stage 1 – Clean Workspace
- Task 2.2.2: Stage 2 – Checkout Code from GitHub
- Task 2.2.3: Stage 3 – SonarQube Analysis (Quality Gate)
- Task 2.2.4: Stage 4 – Install Dependencies (NPM)
- Task 2.2.5: Stage 5 – Trivy FS Scan (Optional)
- Task 2.2.6: Stage 6 – OWASP Dependency Check (Optional)
- Task 2.2.7: Stage 7 – Docker Build & Push to DockerHub
- Task 2.2.8: Stage 8 – Deploy to Docker Container
- Task 2.2.9: Stage 9 – (Optional) Deploy to Kubernetes (EKS)

Epic 3: Docker Deployment

Story 3.1: Containerize BMS App

- Task 3.1.1: Write Dockerfile for BMS App
- Task 3.1.2: Build Docker image locally and test
- Task 3.1.3: Configure Jenkins to build & push Docker image to DockerHub
- Task 3.1.4: Run container locally on port 3000 & validate app accessibility

Epic 4: Kubernetes Deployment (EKS)

Story 4.1: Deploy BMS App to EKS

- Task 4.1.1: Write deployment.yaml and service.yaml manifests
- Task 4.1.2: Deploy manifests to EKS cluster
- Task 4.1.3: Expose service using NodePort/LoadBalancer
- Task 4.1.4: Validate deployment with kubectl get pods & kubectl get svc
- Task 4.1.5: Document deployment proof (screenshots & outputs)

Epic 5: Monitoring & Observability

Story 5.1: Install Monitoring Tools

- Task 5.1.1: Install & configure Prometheus + Node Exporter
- Task 5.1.2: Configure Jenkins metrics scraping for Prometheus
- Task 5.1.3: Install Grafana and configure Prometheus as data source
- Task 5.1.4: Create dashboards for Node Health & Jenkins Performance
- Task 5.1.5: Take screenshots of dashboards

Epic 6: Final Deliverables

Story 6.1: Submit Deliverables

- Task 6.1.1: Export Jira Board progress
- Task 6.1.2: Submit GitHub PR link
- Task 6.1.3: Attach Jenkins logs & SonarQube report
- Task 6.1.4: Attach DockerHub repo link & Dockerfile
- Task 6.1.5: Attach Kubernetes manifests & kubectl output screenshots
- Task 6.1.6: Attach Grafana screenshots
- Task 6.1.7: Attach Email notification screenshot
- Task 6.1.8: Prepare & submit Final Summary Report (Word/PPT)

2. Assign Tasks to Self: As you create each task, make sure to set the "Assignee" field to your own user account.

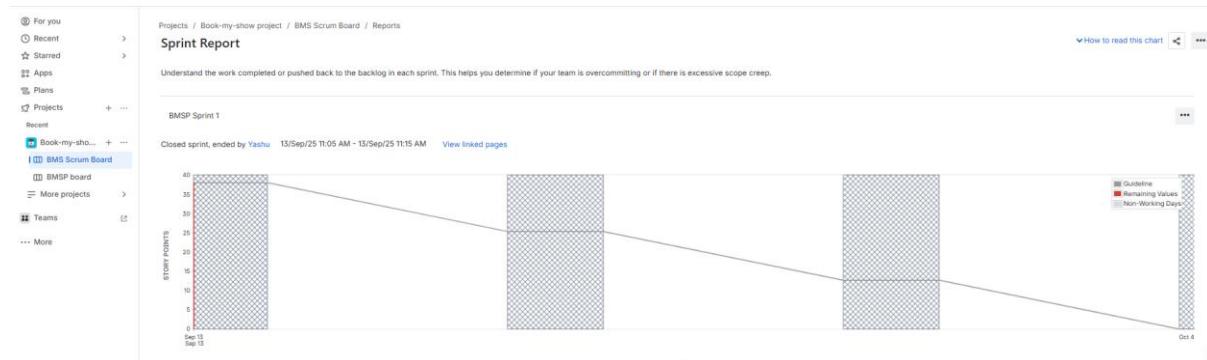
3. Create a Sprint

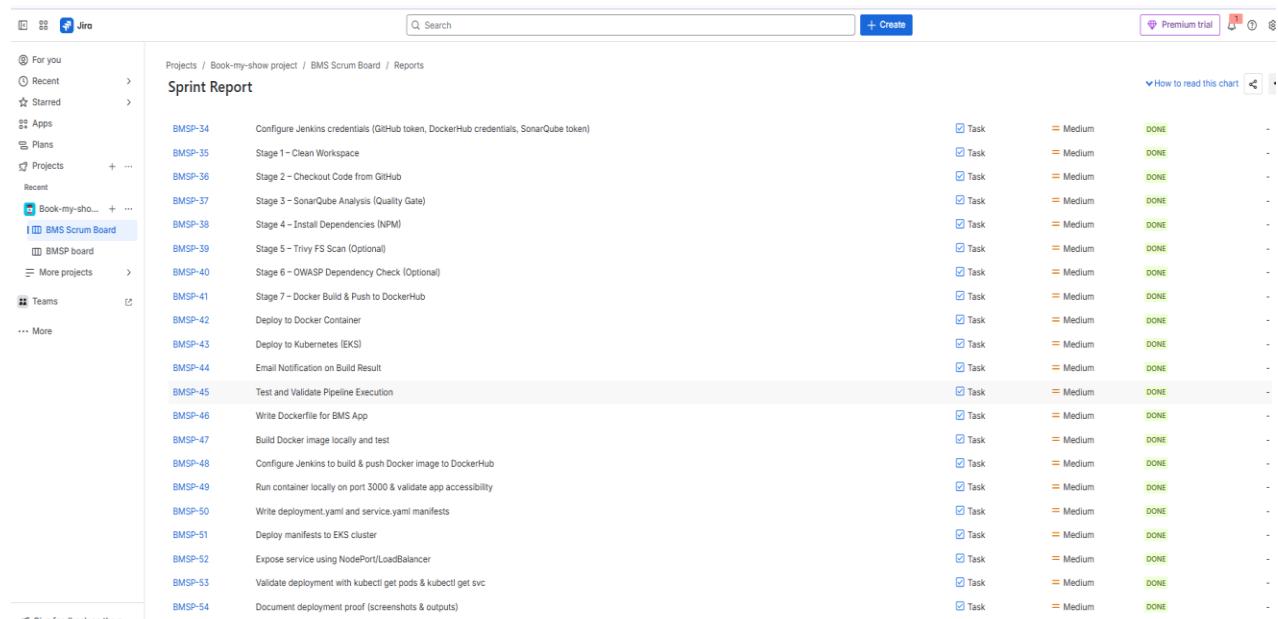
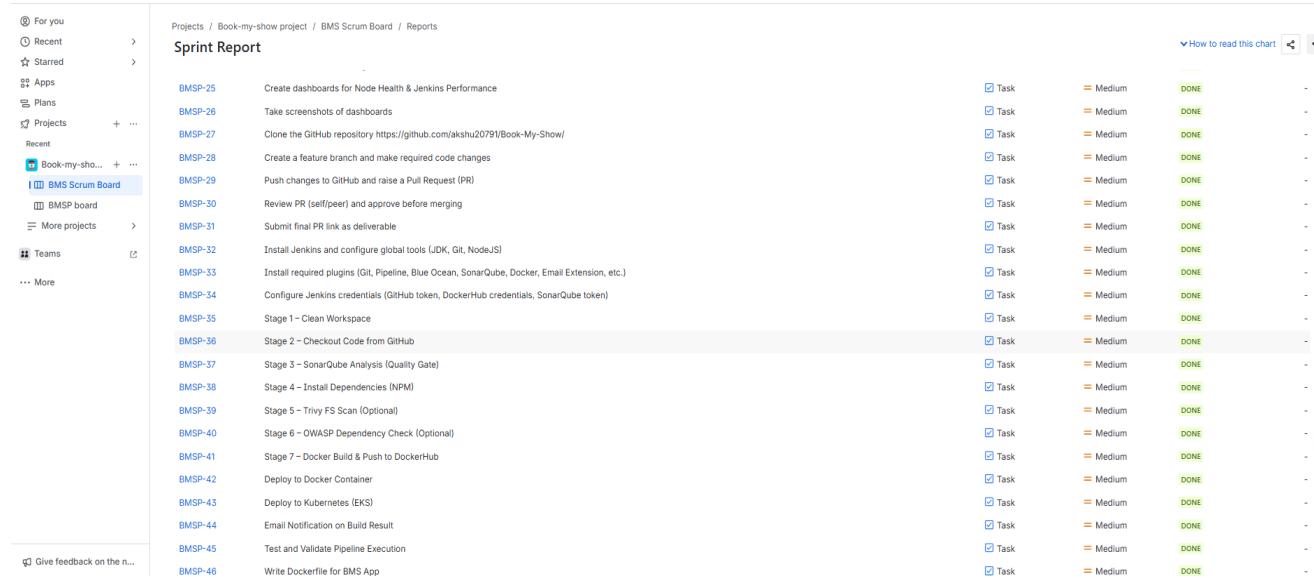
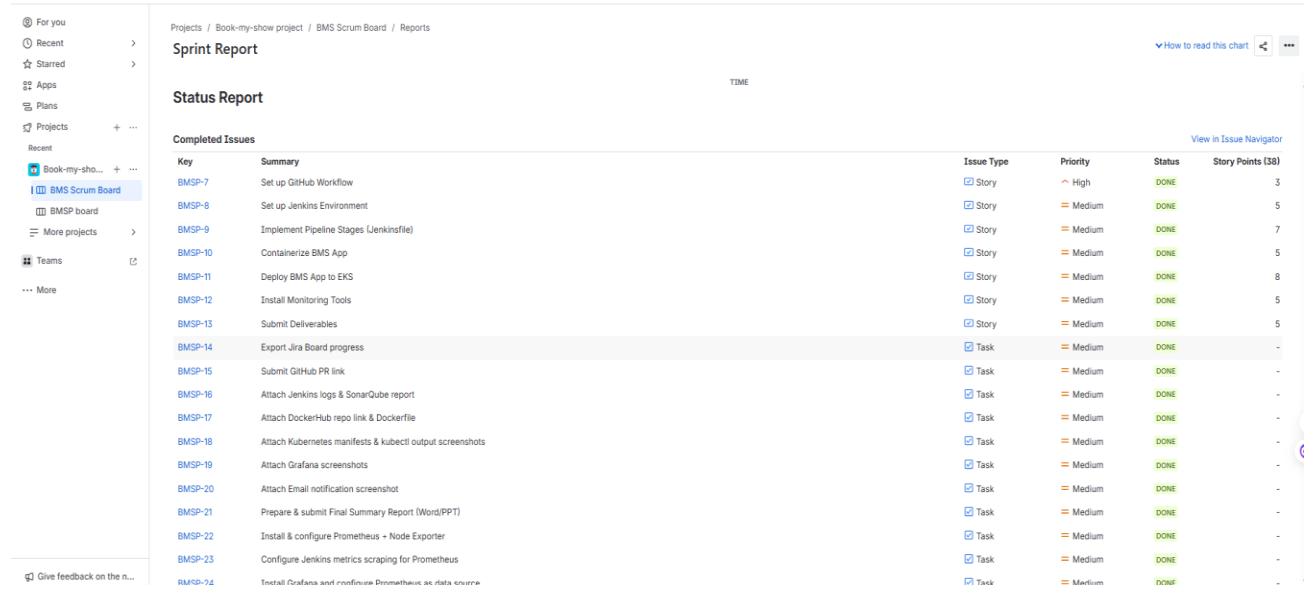
- On your "Backlog" screen, click "Create sprint."
- You can drag tasks from the backlog into the sprint.
- Once you've added some tasks you plan to work on, click "Start sprint."
- Define the sprint duration (e.g., 3 weeks).

4. Manage Your Board:

- Go to your project's "Active sprints" (Scrum).
- Here you will see your tasks in columns like "To Do," "In Progress," and "Done."
- Drag and drop tasks between columns as you work on them.

The screenshot shows the Jira Software interface for the 'BMSP board'. The top navigation bar includes 'Search', '+ Create', and 'Premium trial' options. The left sidebar shows 'For you', 'Recent', 'Starred', 'Apps', 'Plans', 'Projects', 'Recent', 'Book-my-show...', 'BMSP board', and 'More projects'. The main area displays the 'Active sprints' board with three columns: 'TO DO', 'IN PROGRESS', and 'DONE'. The 'IN PROGRESS' column contains tasks such as 'Containerize BMS App', 'Configure Jenkins to build & push Docker image to DockerHub', 'Write Dockerfile for BMS App', 'Run container locally on port 3000 & validate app accessibility', 'Build Docker image locally and test', and 'Deploy to Docker Container'. The 'DONE' column contains tasks like 'Test and Validate Pipeline Execution', 'Set up GitHub Workflow', 'Submit final PR link as deliverable', 'Deploy to Docker Container', 'Stage 1 - Clean Workspace', 'Email Notification on Build Result', and 'Show in CI/CD Pipeline Dependence Check (Optional)'. Each task has its status, assignee (Yashu), and due date.





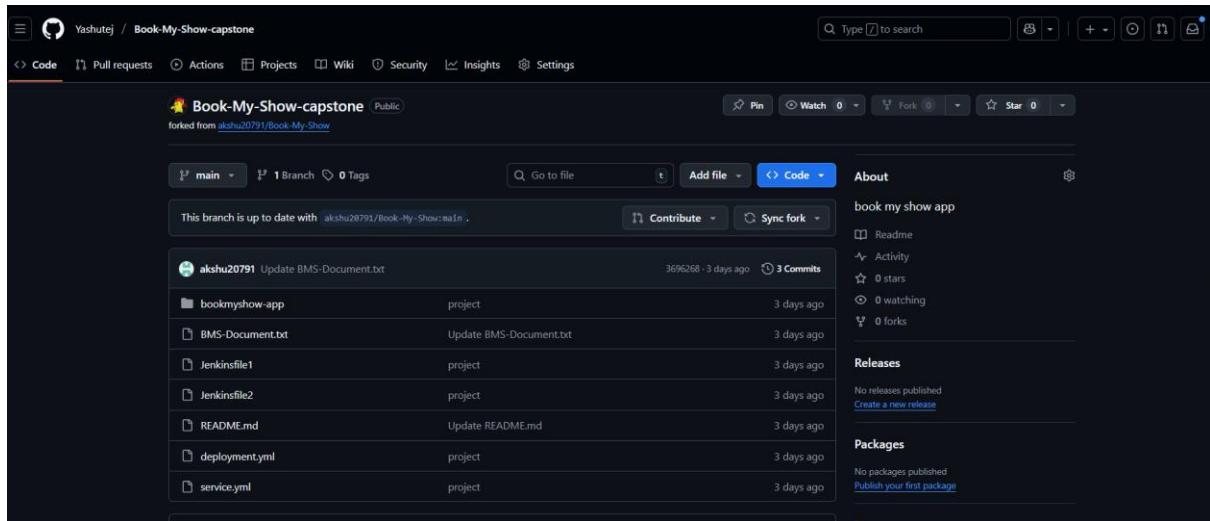
Step 2: GitHub Workflow

Workflow:

Clone Repo → Create Feature Branch → Make Changes → Commit & Push →
Raise Pull Request → Peer Review & Approval → Merge to Main

Step 1 – Fork Repository

- Go to: <https://github.com/akshu20791/Book-My-Show>
- Click Fork (top-right corner).
- Select your GitHub account → wait for fork to complete.



Step 2 – Clone Your Fork

The screenshot shows the VS Code interface with a terminal window open. The terminal output shows the command 'git clone https://github.com/Yashutej/Book-My>Show-capstone.git' being run, followed by the cloning process and a successful 'ls' command showing the directory contents. The Explorer sidebar on the left lists various local projects and files. The bottom status bar indicates the path 'D:\Book-My>Show-capstone'.

```
PS D:\> git clone https://github.com/Yashutej/Book-My>Show-capstone.git
Cloning into 'Book-My>Show-capstone'...
remote: Enumerating objects: 136, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 136 (delta 3), reused 1 (delta 1), pack-reused 129 (from 1)
Receiving objects: 100% (136/136), 1.58 MiB | 932.00 KiB/s, done.
Resolving deltas: 100% (18/18), done.
PS D:\> cd .\Book-My>Show-capstone\
PS D:\Book-My>Show-capstone> ls

Directory: D:\Book-My>Show-capstone

Mode LastWriteTime Length Name
---- -- -a--- 13-09-2025 11:42 bookmyshow-app
-a--- 13-09-2025 11:42 30439 BMS-Document.txt
-a--- 13-09-2025 11:42 439 deployment.yml
-a--- 13-09-2025 11:42 3781 Jenkinsfile1
-a--- 13-09-2025 11:42 4125 Jenkinsfile2
-a--- 13-09-2025 11:42 2 README.md
-a--- 13-09-2025 11:42 232 service.yml
```

Step 3 – Add Upstream

```
● PS D:\Book-My-Show-capstone> git remote add upstream https://github.com/akshu20791/Book-My-Show.git
PS D:\Book-My-Show-capstone> git remote -v
● origin https://github.com/Yashutej/Book-My-Show-capstone.git (fetch)
origin https://github.com/Yashutej/Book-My-Show-capstone.git (push)
upstream https://github.com/akshu20791/Book-My-Show.git (fetch)
upstream https://github.com/akshu20791/Book-My-Show.git (push)
```

Step 4 – Create Feature Branch

```
● PS D:\Book-My-Show-capstone> git checkout -b feature/add-login-page
Switched to a new branch 'feature/add-login-page'
```

Step 5 – Make Changes, Commit, Push

```
● PS D:\Book-My-Show-capstone> git add .
PS D:\Book-My-Show-capstone> git commit -m "Added login page feature"
● [feature/add-login-page 28a5914] Added login page feature
  2 files changed, 2 insertions(+)
PS D:\Book-My-Show-capstone> git push origin feature/add-login-page
● Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 420 bytes | 420.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: Create a pull request for 'feature/add-login-page' on GitHub by visiting:
remote:     https://github.com/Yashutej/Book-My-Show-capstone/pull/new/feature/add-login-page
remote:
remote: To https://github.com/Yashutej/Book-My-Show-capstone.git
 * [new branch]      feature/add-login-page -> feature/add-login-page
💡 PS D:\Book-My-Show-capstone> █
```

Step 6 – Open Pull Request

- Go to your fork → feature branch → click Compare & Pull Request.
- Base repo: akshu20791/Book-My-Show (main branch)
Head repo: Yashutej/Book-My-Show-capstone (feature branch)
- Add title + description → Submit PR.

The screenshot shows a GitHub repository page for 'Book-My-Show-capstone'. The repository is public and was forked from 'akshu20791/Book-My-Show'. The main branch is 'main', there is 1 branch, and 0 tags. The repository is up-to-date with the original 'main' branch. A recent push to 'feature/add-login-page' occurred 2 seconds ago. The commit history shows three commits from 'akshu20791' updating BMS-related files like 'BMS-Document.txt', 'Jenkinsfile1', 'Jenkinsfile2', 'README.md', and 'deployment.yml'. The commits were made 3 days ago.

File	Commit Message	Time Ago
BMS-Document.txt	Update BMS-Document.txt	3 days ago
Jenkinsfile1	project	3 days ago
Jenkinsfile2	project	3 days ago
README.md	Update README.md	3 days ago
deployment.yml	project	3 days ago

Step 7 – Copy the PR Link

- After submitting, you will land on the **PR details page**.
 - PR link: <https://github.com/akshu20791/Book-My-Show/pull/25>

Added new branch "feature" and made changes #25

Yashutej wants to merge 1 commit into akshu20791/main from Yashutej:feature/add-login-page

Conversation 0 · Commits 1 · Checks 0 · Files changed 2

Yashutej commented 1 minute ago

No description provided.

Added login page feature

No conflicts with base branch

Changes can be easily merged.

Add a comment

Write Preview

Add your comment here...

Markdown is supported

Paste, drop, or click to add files

Close pull request Comment

Successfully merging this pull request may close these issues.

Unsubscribe

Step 3: Jenkins CI/CD Pipeline

1. Jenkins and EKS Cluster Setup

Prerequisites:

Infrastructure:

- AWS EC2 instance (Ubuntu 20.04+) with Jenkins installed
- Docker installed and configured
- AWS CLI installed and configured
- kubectl installed and configured (or installed at runtime by Jenkins)

Tools Installed on Jenkins:

- JDK 17
- NodeJS 23
- SonarQube Scanner
- Trivy
- OWASP Dependency Check

- Launch EC2 Instance to set up Jenkins Server

Name: Yashaswini_Jenkins

AMI: Ubuntu

Instance type: t2.medium

Select key pair, Security Group and Subnet

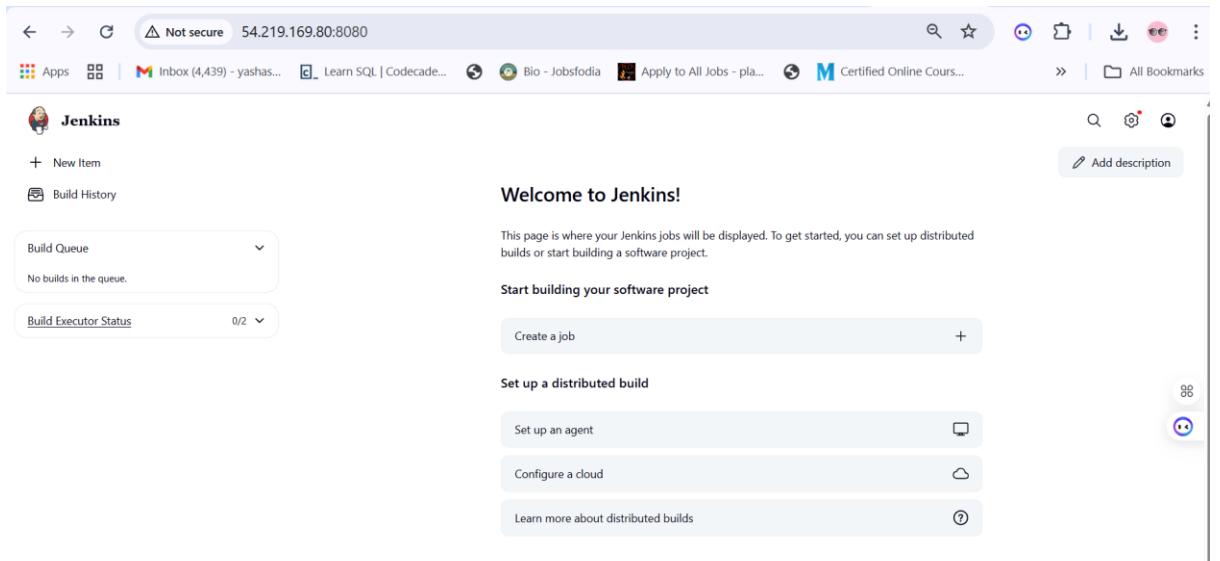
Click on “Launch instance”

The screenshot shows the AWS EC2 Instances page. In the left sidebar, under 'Instances', there is a section for 'Launch instances'. A new instance is being created with the following details:

- Name:** Yashaswini_Jenkins
- AMI:** Ubuntu
- Instance type:** t2.medium
- Key pair:** (not explicitly shown)
- Security group:** (not explicitly shown)
- Subnet:** (not explicitly shown)

The instance status is currently 'Pending'.

- Connect to putty server using SSH client.
- Install Java and Jenkins. Next Enable and start Jenkins on server. Make sure you have to install kubectl and aws cli on Jenkins server.
- Access your Jenkins server on browser using instance public IP address with port number
- Open port number 8080 in your security group for Jenkins



➤ Set up EKS Cluster

Launch EC2 Instance to set up EKS Cluster

Name: yashu-cluster

AMI: Ubuntu

Instance type: t2.medium

Select key pair, Security Group and Subnet, Config storage =22

Click on “Launch instance”

➤ Connect to putty server

Install Kubectl, AWS CLI & EKSCTL

Provide AWS access key and secret key

➤ Next, we need to copy Kube config file into the Jenkins server from Cluster

Provide AWS access keys on Jenkins server

Execute following command on Jenkins server

`>> aws eks update-kubeconfig --name yashu-cluster --region us-west-1`

➤ Kube-config file has copied to Jenkins serve from master node i.e. cluster.

Now execute following commands in Jenkins server one by one

`sudo mkdir -p /home/jenkins/.kube`

`sudo cp /root/.kube/config /home/jenkins/.kube/config`

`sudo chown -R jenkins:jenkins /home/jenkins/.kube`

`sudo su - jenkins`

`aws configure` (provide access keys on Jenkins server)

➤ Install following plugins in Jenkins

- Now we have to provide Kube-config file in Jenkins web server
- Firstly, we need to copy and save it in a document or file in your local
- Provide AWS access keys and kubeconfig file to credentials in Jenkins

2. Pipeline Creation

- Click on “New item” for creating pipeline
- Provide a name for pipeline and select pipeline then click on “ok”

- Next write pipeline script by adding all necessary stages

```

Book-My-Show > pipeline_script
1 pipeline {
2     agent any
3
4     tools {
5         jdk 'jdk21'
6         nodejs 'node-js'
7     }
8
9     environment {
10        SCANNER_HOME = tool 'sonar'
11        DOCKER_IMAGE = 'yashukl2002/bms:latest'
12        EKS_CLUSTER_NAME = 'yashu-cluster'
13        AWS_REGION = 'us-west-1'
14    }
15
16    stages {
17        stage('Clean Workspace') {
18            steps {
19                cleanWs()
20            }
21        }
22
23        stage('Checkout from Git') {
24            steps {
25                git branch: 'main', url: 'https://github.com/Yashutej/Book-My>Show-capstone.git'
26                sh 'ls -la' // Verify files after checkout
27            }
28        }
29
30        stage('SonarQube Analysis') {
31            steps {

```

```

Book-My-Show > pipeline_script
1 pipeline {
16     stages {
21
22         stage('SonarQube Analysis') {
23             steps {
24                 withSonarQubeEnv('sonar-server') {
25                     sh '''
26                         $SCANNER_HOME/bin/sonar-scanner \
27                         -Dsonar.projectName=BMS \
28                         -Dsonar.projectKey=BMS
29
30                 '''
31             }
32         }
33
34         stage('Quality Gate') {
35             steps {
36                 script {
37                     // Added timeout to avoid infinite waiting
38                     timeout(time: 1, unit: 'MINUTES') {
39                         def qg = waitForQualityGate abortPipeline: true, credentialsId: 'sonar-token'
40                         echo "Quality Gate status: ${qg.status}"
41                     }
42                 }
43             }
44         }
45     }
46 }
47
48
49
50
51
52
53

```

```
Book-My-Show > pipeline_script
1 pipeline {
16   stages {
54     stage('Install Dependencies') {
55       steps {
56         sh ...
57         cd bookmyshow-app
58         ls -la # Verify package.json exists
59         if [ -f package.json ]; then
60           rm -rf node_modules package-lock.json # Remove old dependencies
61           npm install # Install fresh dependencies
62         else
63           echo "Error: package.json not found in bookmyshow-app!"
64           exit 1
65         fi
66         ...
67     }
68   }
69
70   stage('OWASP FS Scan') {
71     steps {
72       dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', options: []
73       dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
74     }
75   }
76 }
```

```
Book-My-Show > pipeline_script
1 pipeline {
16   stages {
77     stage('Trivy FS Scan') {
78       steps {
79         sh ''
80         if ! command -v trivy &> /dev/null
81         then
82           echo "Trivy not installed! Skipping scan."
83           exit 0
84         fi
85
86         echo "Running Trivy FS scan..."
87         trivy fs . > trivyfs.txt
88         ...
89     }
90   }
91   stage('Docker Build & Push') {
92     steps {
93       script {
94         withDockerRegistry(credentialsId: 'docker', toolName: 'docker') {
95           sh ''
96           echo "Building Docker image..."
97           docker build --no-cache -t $DOCKER_IMAGE -f bookmyshow-app/Dockerfile bookmyshow-
98
99           echo "Pushing Docker image to Docker Hub..."
100          docker push $DOCKER_IMAGE
101          ...
102        }
103      }
104    }
105  }
106 }
```

```
Book-My-Show > pipeline_script
90  }
91
92  stage('Deploy to EKS Cluster') {
93      steps {
94          script {
95              sh ''
96              echo "Verifying AWS credentials..."
97              aws sts get-caller-identity
98
99
100             echo "Configuring kubectl for EKS cluster..."
101            aws eks update-kubeconfig --name $EKS_CLUSTER_NAME --region $AWS_REGION
102
103             echo "Verifying kubeconfig..."
104            kubectl config view
105
106
107             echo "Deploying application to EKS..."
108            kubectl apply -f deployment.yml
109            kubectl apply -f service.yml
110
111             echo "Verifying deployment..."
112            kubectl get pods
113            kubectl get svc
114            ''
115
116        }
117    }
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132 }
```

```
! deployment.yml 1, M   ! service.yml M   ! prometheus.yml U   Jenkinsfile2   pipeline_script U X
Book-My-Show > pipeline_script
90  }
91
92
93  post {
94      always {
95          emailext attachLog: true,
96          subject: "${currentBuild.result}",
97          body: "Project: ${env.JOB_NAME}<br/>" +
98              "Build Number: ${env.BUILD_NUMBER}<br/>" +
99              "URL: ${env.BUILD_URL}<br/>",
100             to: 'klyashu632@gmail.com',
101             attachmentsPattern: 'trivyfs.txt'
102     }
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
```

The screenshot shows the Jenkins Pipeline configuration page for a pipeline named 'bms-app'. The 'Pipeline' tab is selected in the sidebar. The main area contains a code editor with a Groovy script:

```

19
20~     stage('Checkout from Git') {
21~         steps {
22~             git branch: 'main', url: 'https://github.com/Yashutej/Book-My-Show-capstone.git'
23~             sh 'ls -la' // Verify files after checkout
24~         }
25~     }
26
27~     stage('Install Dependencies') {
28~         steps {
29~             sh '''
30~             cd bookmyshow-app
31~             ls -la # Verify package.json exists
32~             if [ ! -f package.json ]; then
33~                 rm -rf node_modules package-lock.json
34~             fi
35~         }
36~     }

```

Use Groovy Sandbox ?

Pipeline Syntax

- After saving your script come back to your pipeline and click on “Build Now” to running your pipeline.

Outputs:

Jenkins UI

The screenshot shows the Jenkins dashboard. The 'Build History' section displays information for the 'bms-app' pipeline:

S	W	Name ↓	Last Success	Last Failure	Last Duration
		bms-app	13 hr #11	14 hr #10	11 min

Build Queue: No builds in the queue.

Build Executor Status: 0/2

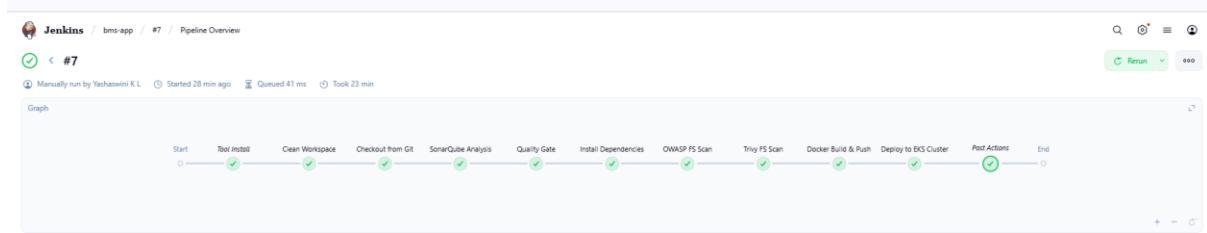
Icon: S M L

The screenshot shows the Jenkins Stage View for the 'bms-app' pipeline. The left sidebar includes links for Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Favorite, SonarQube, Open Blue Ocean, Stages, Rename, Pipeline Syntax, and Credentials.

The Stage View table shows the following data:

	Declarative: Tool Install	Clean Workspace	Checkout from Git	SonarQube Analysis	Quality Gate	Install Dependencies	OWASP FS Scan	Trivy FS Scan	Docker Build & Push	Deploy to EKS Cluster	Declarative: Post Actions
Average stage times: (full run time: ~23min 44s)	198ms	355ms	2s	20s	398ms	1min 46s	14min 46s	523ms	1min 44s	1s	1s
Sept 13 21:15	321ms	571ms	2s	26s	527ms (paused for 6s)	2min 20s	11min 52s	629ms	8min 42s	7s	2s

Dependency-Check Trend chart showing the trend of dependency checks over seven stages (#4 to #7). The legend indicates Unassigned (grey), Low (green), Medium (yellow), High (orange), and Critical (red).



Search Post Actions

Tool Install 0.28s
 Clean Workspace 0.5s
 Checkout from Git 2.6s
 SonarQube Analysis 26s
 Quality Gate 6.9s
 Install Dependencies 2m 20s
 OWASP FS Scan 11m
 Trivy FS Scan 0.6s
 Docker Build & Push 8m 42s
 Deploy to EKS Cluster 7.5s
 Post Actions 2.0s

Extended Email
 Sending email to: klyashu632@gmail.com

2.0s Started 5m 5s ago Jenkins

Jenkins / bms-app #7

```

+ kubectl get svc
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP     PORT(S)      AGE
bms-service LoadBalancer 10.100.202.177 <pending>     80:31168/TCP 1s
kubernetes   ClusterIP  10.100.0.1    <none>        443/TCP    3h41m
[Pipeline] 
[Pipeline] // script
[Pipeline] 
[Pipeline] // withEnv
[Pipeline] 
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] mailer
Sending email to: klyashu632@gmail.com
[Pipeline] 
[Pipeline] // stage
[Pipeline] 
[Pipeline] // withEnv
[Pipeline] 
[Pipeline] // withEnv
[Pipeline] 
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

Jenkins / bms-app

Changes 19:38 169ms 290ms 1s 2s failed 69ms failed 68ms failed 69ms failed 67ms failed

SonarQube Quality Gate

BMS Passed server-side processing: Success

 Latest Dependency-Check

Permalinks

- Last build (#7), 30 min ago
- Last stable build (#7), 30 min ago
- Last successful build (#7), 30 min ago
- Last failed build (#6), 51 min ago
- Last unsuccessful build (#6), 51 min ago
- Last completed build (#7), 30 min ago

SonarQube Analysis

The screenshot shows the SonarQube main dashboard for the project 'main'. The dashboard is green and labeled 'Passed'. It displays various metrics: Security (2 Open issues), Reliability (167 Open issues), Maintainability (275 Open issues), Accepted Issues (0), Coverage (0.0%), and Duplications (1.4%). There are also sections for Security Hotspots (5) and Activity. A note at the top states: '⚠️ 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](#)'.

Project Settings and **Project Information** are visible in the top right corner.

The screenshot shows the SonarQube Webhooks configuration page. It lists a single webhook entry for Jenkins, which is triggered by the URL <http://54.177.74.78:8080/sonarqube-webhook/>. The webhook was last delivered on September 13, 2025, at 9:15 PM. A 'Create' button is available in the top right.

Dependency Check

The screenshot shows the Jenkins Dependency-Check results page for build #7. The left sidebar includes links for Status, Changes, Console Output, Edit Build Information, Delete build #7, Timings, Git Build Data, Dependency-Check (which is selected), Open Blue Ocean, Pipeline Overview, Restart from Stage, Replay, Pipeline Steps, Workspaces, and Previous Build.

The main content area is titled 'Dependency-Check Results' and shows the 'SEVERITY DISTRIBUTION' chart. Below the chart is a table of vulnerabilities:

File Name	Vulnerability	Severity	Weakness
ansi-html:0.7	NVD CVE-2021-23424	High	NVD-CWE-noInfo
axios.js	RETIRED CVE-2025-27152	High	
axios.js	RETIRED CVE-2025-58754	High	
axios.js	NVD CVE-2023-45857	Medium	CWE-352
axios.js	RETIRED Versions before 1.6.8 depends on follow-redirects before 1.15.6 which could leak the proxy authentication credentials	Medium	
axios.min.js	RETIRED CVE-2025-27152	High	
axios.min.js	RETIRED CVE-2025-58754	High	
axios.min.js	NVD CVE-2023-45857	Medium	CWE-352
axios.min.js	RETIRED Versions before 1.6.8 depends on follow-redirects before 1.15.6 which could leak the proxy authentication credentials	Medium	
axios.0.21.4	OSSINDEX CVE-2025-58754	High	CWE-770

Docker Build & push

Jenkins / bms-app / #7

```
--> e7bde9b36dae
Step 7/10 : EXPOSE 3000
--> Running in f87a332cc99b
--> Removed intermediate container f87a332cc99b
--> a69323140d43
Step 8/10 : ENV NODE_OPTIONS=--openssl-legacy-provider
--> Running in 887de853dd28
--> Removed intermediate container 887de853dd28
--> 9dc67353c454
Step 9/10 : ENV PORT=3000
--> Running in 584f5b9d8bd2
--> Removed intermediate container 584f5b9d8bd2
--> f477370fd80e
Step 10/10 : CMD ["npm", "start"]
--> Running in 5df1f8b08dd7
--> Removed intermediate container 5df1f8b08dd7
--> a5a132475186
Successfully built a5a132475186
Successfully tagged yashukl2002/bms:latest
+ echo Pushing Docker image to Docker Hub...
Pushing Docker image to Docker Hub...
+ docker push yashukl2002/bms:latest
The push refers to repository [docker.io/yashukl2002/bms]
8ddf2b46566: Preparing
33e332e4067d: Preparing
3234ac9a82dc: Preparing
5c62565b78f4: Preparing
```

EKS Deployment

Jenkins / bms-app / #7

```
- --cluster-name
- yashu-cluster
- --output
- json
command: aws
env: null
+ echo Deploying application to EKS...
Deploying application to EKS...
+ kubectl apply -f deployment.yml
deployment.apps/bms-app created
+ kubectl apply -f service.yml
service/bms-service created
+ echo Verifying deployment...
Verifying deployment...
+ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
bms-app-5c987585fc-2m8gz  0/1     ContainerCreating   0      1s
bms-app-5c987585fc-9bh7t  0/1     ContainerCreating   0      1s
+ kubectl get svc
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP   PORT(S)      AGE
bms-service   LoadBalancer  10.100.202.177  <pending>   80:31168/TCP  1s
kubernetes   ClusterIP   10.100.0.1    <none>       443/TCP     3h41m
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
```

root@ip-172-31-22-136:/home/ubuntu# kubectl get svc > svc_status.txt

```
root@ip-172-31-22-136:/home/ubuntu# kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
bms-app-5c987585fc-2m8gz  1/1     Running   0      16m
bms-app-5c987585fc-9bh7t  1/1     Running   0      16m
root@ip-172-31-22-136:/home/ubuntu# kubectl get svc
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP   PORT(S)      AGE
bms-service   LoadBalancer  10.100.202.177  ae95b4efd3970497d8afb06f97ca6100-361306071.us-west-1.elb.amazonaws.com  80:31168/TCP  16m
kubernetes   ClusterIP   10.100.0.1    <none>       443/TCP     3h57m
```

Email notification

The screenshot shows the Gmail web interface. On the left, the sidebar includes a 'Compose' button, an 'Inbox' section with 11,590 messages, and other navigation options like Starred, Snoozed, Sent, Drafts (8), and More. Below that is a 'Labels' section with an 'Upgrade' link. The main area displays an email message. The subject is 'SUCCESS' and it's from 'address not configured yet <klyashu632@gmail.com> to me'. The message was sent at 21:38 (1 hour ago). It contains the text: 'Project: bms-app
Build Number: 7
URL: <http://54.177.74.78:8080/job/bms-app/7/>
'. Below the message, it says 'One attachment • Scanned by Gmail' and shows a thumbnail for 'build.log'.

Step 4: Docker Deployment

- Docker file

```
! deployment.yml ! service.yml Jenkinsfile2 Dockerfile M X
Book-My-Show-capstone > bookmyshow-app > Dockerfile > ...
3
4 WORKDIR /app
5
6 COPY package.json package-lock.json ./
7
8 RUN npm install postcss@8.4.21 postcss-safe-parser@6.0.0 --legacy-peer-deps
9
10 RUN npm install
11
12 COPY . .
13
14 EXPOSE 3000
15
16 ENV NODE_OPTIONS=--openssl-legacy-provider
17 ENV PORT=3000
18
19 CMD ["npm", "start"]
20
```

- Build Docker image and push to DockerHub via Jenkins.

```
Jenkins / bms-app / #7
---> e7bde9b36dae
Step 7/10 : EXPOSE 3000
---> Running in f87a332cc99b
---> Removed intermediate container f87a332cc99b
---> a69323140d43
Step 8/10 : ENV NODE_OPTIONS=--openssl-legacy-provider
---> Running in 887de853dd28
---> Removed intermediate container 887de853dd28
---> 9dc67353c454
Step 9/10 : ENV PORT=3000
---> Running in 584f5b9d8bd2
---> Removed intermediate container 584f5b9d8bd2
---> f477370fd80e
Step 10/10 : CMD ["npm", "start"]
---> Running in 5df1f8b08dd7
---> Removed intermediate container 5df1f8b08dd7
---> a5a132475186
Successfully built a5a132475186
Successfully tagged yashukl2002/bms:latest
+ echo Pushing Docker image to Docker Hub...
Pushing Docker image to Docker Hub...
+ docker push yashukl2002/bms:latest
The push refers to repository [docker.io/yashukl2002/bms]
8dd2f2b46566: Preparing
33e332e4067d: Preparing
3234ac9a82dc: Preparing
5c62565b78f4: Preparing
```

- Image pushed to dockerhub

Screenshot of a Docker repository page for 'yashukl2002/bms' on Dockerhub.

The page shows a summary of the repository, including a Dockerfile icon, the repository name, the owner, and the last update time. It also displays the number of stars (0) and forks (29).

Below this, there are tabs for 'Overview' and 'Tags'. The 'Tags' tab is selected, showing a list of tags. The 'latest' tag is highlighted, indicating it is the default tag. Other tags listed include 'Digest' and '50c2be1a10ce'. The 'OS/ARCH' column shows 'linux/amd64' and the 'Last pull' column shows 'less than 1 day'. On the right side, there is a command line interface (CLI) section with the command 'docker pull yashukl2002/bms:latest' and a compressed size of '689.66 MB'.

<https://hub.docker.com/repository/docker/yashukl2002/bms>

- Dockerhub repository link
- Run container locally and validate accessibility

Screenshot of the BookMyShow website.

The header includes the BookMyShow logo, a search bar, and navigation links for 'Select City' and 'Sign In'. Below the header, there are links for 'Movies', 'Stream', 'Events', 'Plays', 'Sports', 'Activities', 'Fanhood', 'Buzz', 'Listyourshow', 'Corporates', 'Offers', and 'Gift Cards'.

The main content features two movie posters: 'JAB WE SEPARATED' (20TH-21ST MARCH) and 'WRONG NUMBER' (A COMEDY PLAY BY RAMAN KUMAR). Both posters include promotional text and images of the lead actors.

Below the posters, there is a section titled 'Recommended Movies' with a 'See all >' link. At the bottom of the page, there is a footer with the text 'The Best of Entertainment'.

Step 5: Kubernetes Deployment EKS

Manifest Files:

- deployment.yml



A screenshot of a terminal window titled "k8s-Cluster" showing a deployment YAML configuration. The configuration includes metadata like name and namespace, a template with labels, and a spec section defining two containers, each with an image and port mapping. The terminal also shows the file path as "deployment.yaml" and its size as 241.454B.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bms-app
  namespace: yashaswin
  labels:
    app: bms
spec:
  replicas: 2
  selector:
    matchLabels:
      app: bms
  template:
    metadata:
      labels:
        app: bms
    spec:
      containers:
        - name: bms-container
          image: abhishekrauthor21/bms:latest # Replace with your Docker image
          ports:
            - containerPort: 3000 # Replace with the port your app runs on
```

- service.yml



The screenshot shows a CloudWatch Log Stream titled 'k8s-Cluster' with a single log entry. The log entry is a JSON-formatted Kubernetes Service specification:

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

```

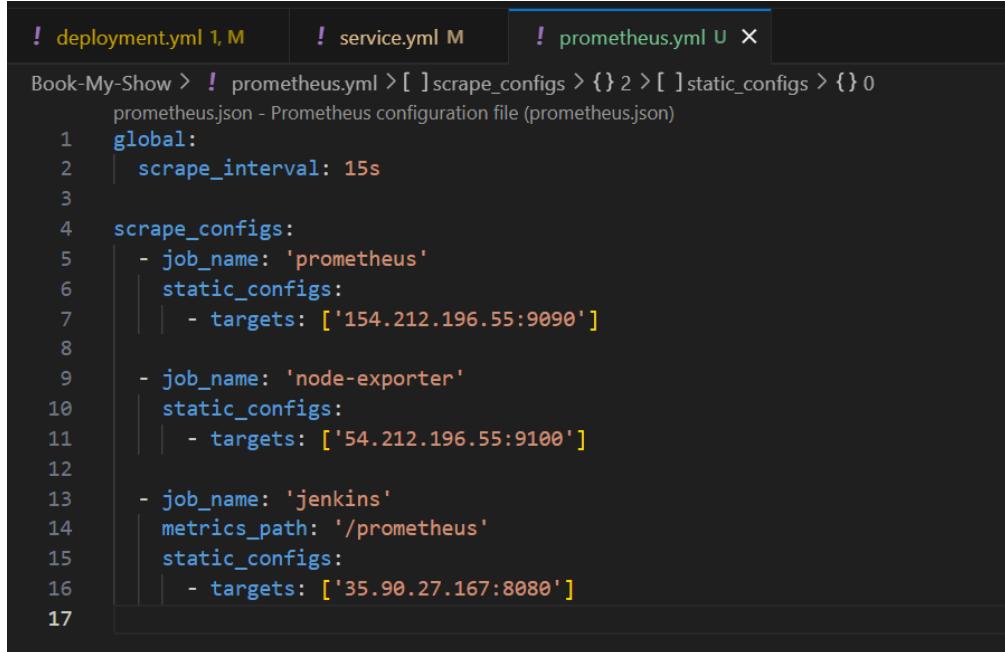
root@ip-172-31-22-61:/home/ubuntu# kubectl create namespace yashaswini
namespace/yashaswini created
root@ip-172-31-22-61:/home/ubuntu# vi deployment.yml
root@ip-172-31-22-61:/home/ubuntu# vi service.yml
root@ip-172-31-22-61:/home/ubuntu# vi deployment.yml
root@ip-172-31-22-61:/home/ubuntu# kubectl apply -f deployment.yml
deployment.apps/bms-app created
root@ip-172-31-22-61:/home/ubuntu# kubectl apply -f service.yml
service/bms-service created
root@ip-172-31-22-61:/home/ubuntu# kubectl get pods -n yashaswini
NAME          READY   STATUS    RESTARTS   AGE
bms-app-67c4b84589-cn8zn  1/1     Running   0          45s
bms-app-67c4b84589-mb9hh  1/1     Running   0          45s
root@ip-172-31-22-61:/home/ubuntu# kubectl get svc -n yashaswini
NAME        TYPE      CLUSTER-IP      EXTERNAL-IP                                     PORT(S)           AGE
bms-service LoadBalancer  10.100.1.240   a3d4a25dc77f746cf3c8a379e61899f-1845912715.us-west-1.elb.amazonaws.com  80:30987/TCP   51s

```

Step 6: Monitoring & Observability

- Launch an AWS EC2 Instance
- Install Prometheus, Grafana and Node Exporter on the server
- Add Jenkins Performance Dashboard
 - Install Jenkins Prometheus plugin on Jenkins server (if not already installed):
Go to Manage Jenkins → Plugins → Available → Search "Prometheus metrics" → Install
 - Enable metrics endpoint:
Go to Manage Jenkins → Configure System
Scroll to Prometheus section → Check Enable Prometheus metrics
Save

In Prometheus prometheus.yml, add:



The screenshot shows a terminal window with three tabs: deployment.yml, service.yml, and prometheus.yml. The prometheus.yml tab is active and displays the following YAML configuration:

```
! deployment.yml 1, M ! service.yml M ! prometheus.yml U X
Book-My-Show > ! prometheus.yml > [ ] scrape_configs > {} 2 > [ ] static_configs > {} 0
    prometheus.json - Prometheus configuration file (prometheus.json)
1   global:
2     |   scrape_interval: 15s
3
4   scrape_configs:
5     - job_name: 'prometheus'
6       static_configs:
7         |   - targets: ['154.212.196.55:9090']
8
9     - job_name: 'node-exporter'
10    static_configs:
11      |   - targets: ['54.212.196.55:9100']
12
13    - job_name: 'jenkins'
14      metrics_path: '/prometheus'
15      static_configs:
16        |   - targets: ['35.90.27.167:8080']
```

- Configure Grafana dashboards to visualize node health (CPU, memory, disk usage) and Jenkins performance metrics using Prometheus as a data source
- Open Prometheus UI:
<http://18.236.253.59:9090>

The screenshot shows the Prometheus web interface at <http://18.236.253.59:9090/targets>. It displays two sections: 'node_exporter' and 'prometheus'. Each section lists an endpoint (http://localhost:9100/metrics) with its labels (instance="localhost:9100", job="node_exporter" for node_exporter; instance="localhost:9090", job="prometheus" for prometheus). The 'Last scrape' column shows the time of the most recent scrape (14.338s ago for node_exporter, 12.054s ago for prometheus), and the 'State' column indicates they are both 'UP'.

➤ Access Grafana UI → http://<server-ip>:3000

Default Login: admin / admin → Change password after first login.

Configure Prometheus as a Grafana Data Source

In Grafana:

Go to Connections → Data Sources → Add data source

Select Prometheus

URL → <http://<prometheus-server>:9090>

Click Save & Test

➤ Node Exporter metrics

The screenshot shows the Node Exporter metrics endpoint at <http://18.236.253.59:9100/metrics>. The page displays a large amount of Prometheus metric definitions, including various disk, memory, and system metrics. Some examples include:
HELP node_disk_rw_time_seconds_total The weighted average of seconds spent doing I/Os.
TYPE node_disk_rw_time_seconds_total counter
node_disk_rw_time_seconds_total{device="xvda"} 125.672
HELP node_disk_read_bytes_total The total number of bytes read successfully.
TYPE node_disk_read_bytes_total counter
node_disk_read_bytes_total{device="xvda"} 3.40461056e+08
HELP node_disk_read_time_seconds_total The total number of seconds spent by all reads.
TYPE node_disk_read_time_seconds_total counter
node_disk_read_time_seconds_total{device="xvda"} 15.151
HELP node_disk_reads_completed_total The total number of reads completed successfully.
TYPE node_disk_reads_completed_total counter
node_disk_reads_completed_total{device="xvda"} 7763
HELP node_disk_reads_merged_total The total number of reads merged.
TYPE node_disk_reads_merged_total counter
node_disk_reads_merged_total{device="xvda"} 2548
HELP node_disk_write_time_seconds_total This is the total number of seconds spent by all writes.
TYPE node_disk_write_time_seconds_total counter
node_disk_write_time_seconds_total{device="xvda"} 110.521
HELP node_disk_writes_completed_total The total number of writes completed successfully.
TYPE node_disk_writes_completed_total counter
node_disk_writes_completed_total{device="xvda"} 17478
HELP node_disk_writes_merged_total The number of writes merged.
TYPE node_disk_writes_merged_total counter
node_disk_writes_merged_total{device="xvda"} 5969
HELP node_disk_written_bytes_total The total number of bytes written successfully.
TYPE node_disk_written_bytes_total counter
node_disk_written_bytes_total{device="xvda"} 1.280104448e+09
HELP node_dmi_info A metric with a constant '1' value labeled by bios_date, bios_release, bios_vendor, bios_version, board_asset_tag, board_name, board_serial, board_vendor, board_version, chassis_asset_tag, chassis_serial, chassis_vendor, chassis_version, product_family, product_name, product_serial, product_sku, product_uuid, product_version, system_vendor if provided by DM¹
TYPE node_dmi_info gauge
node_dmi_info{bios_date="08/24/2006", bios_release="4.11", bios_vendor="Xen", bios_version="4.11.amazon", chassis_asset_tag="", chassis_serial="", chassis_vendor="Xen", chassis_version="", product_fam
ly=""},product_name="VM domU",product_serial="ec27144e-7834-d965-6bbe-60b6ae9e981",product_sku="",product_uuid="ec27144e-7834-d965-6bbe-60b6ae9e981",product_version="4.11.amazon",system_vendor="Xen" } 1
HELP node_entropy_available_bits Bits of available entropy.
TYPE node_entropy_available_bits gauge
node_entropy_available_bits 256
HELP node_entropy_pool_size_bits Bits of entropy pool.
TYPE node_entropy_pool_size_bits gauge
node_entropy_pool_size_bits 256
HELP node_exporter_build_info A metric with a constant '1' value labeled by version, revision, branch, goversion from which node_exporter was built, and the goos and goarch for the build.
TYPE node_exporter_build_info gauge

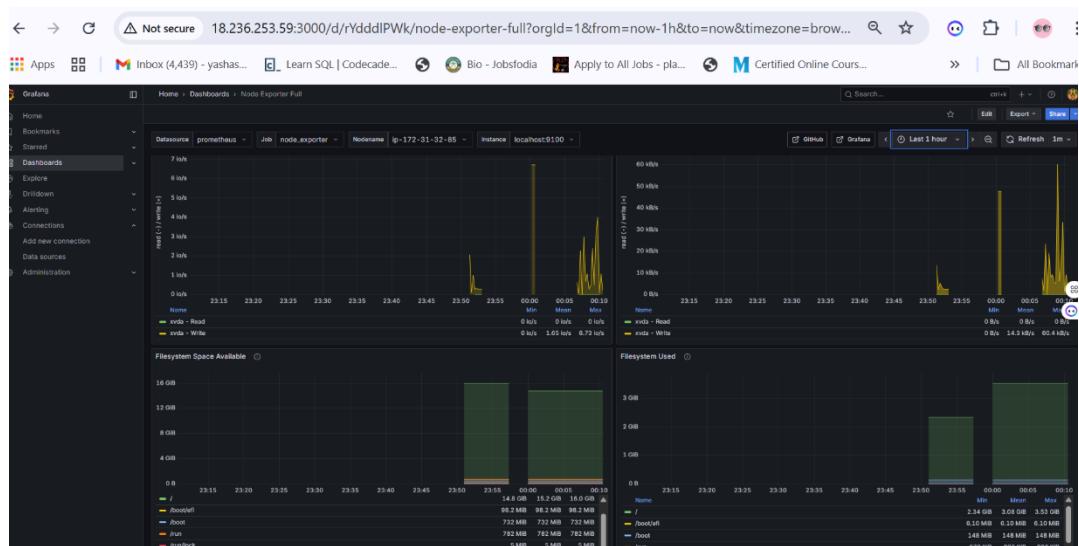
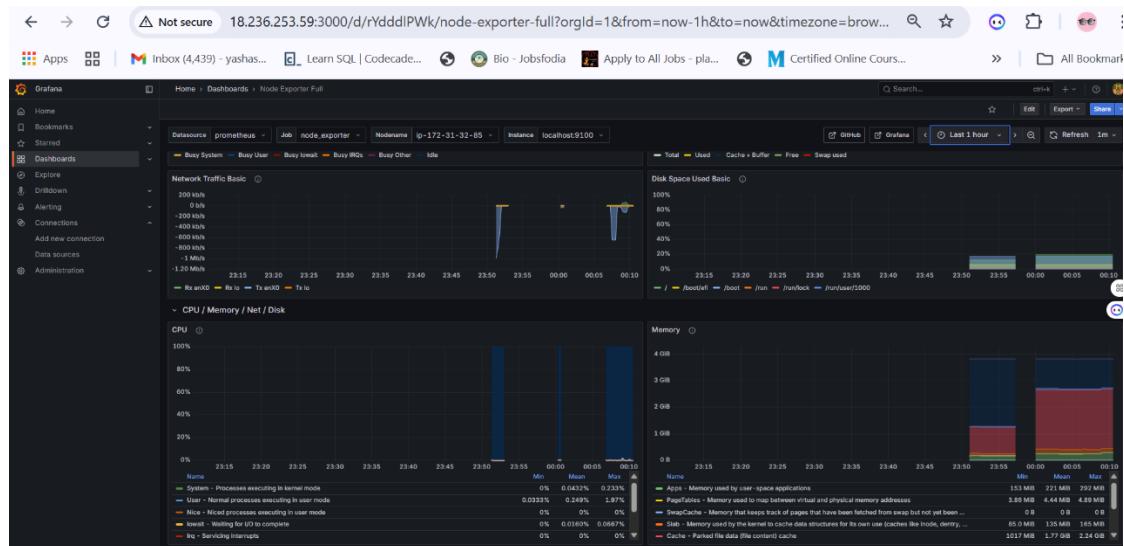
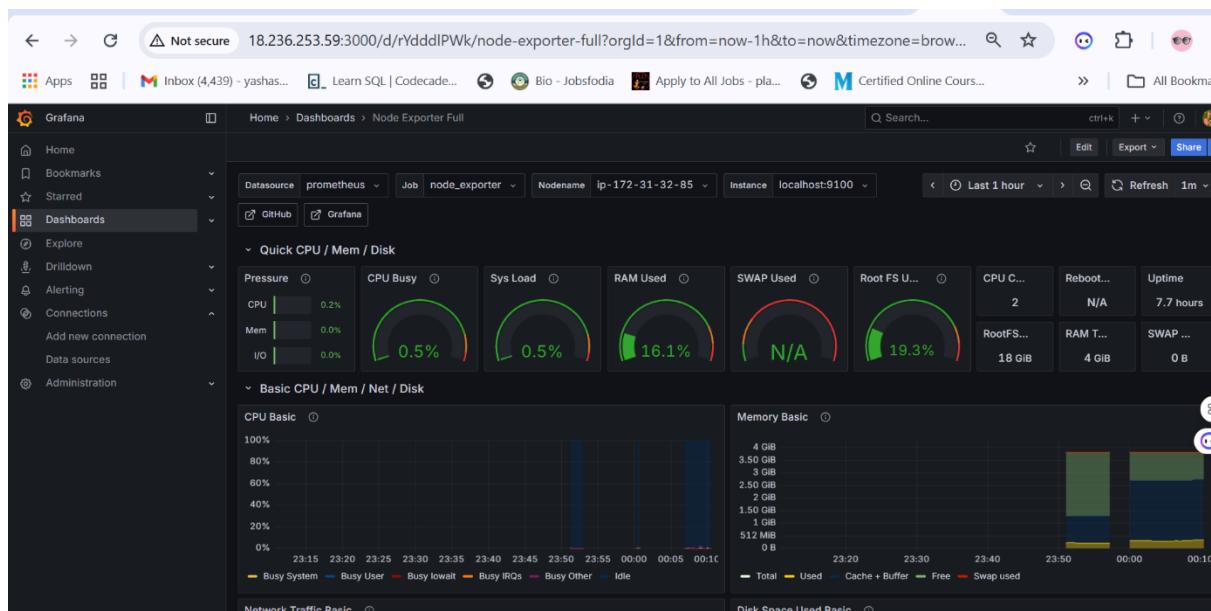
➤ Add Dashboards

Node Health Dashboard:

Import from Grafana's dashboard library.

Recommended dashboard ID: 1860 (Node Exporter Full)

Grafana.com Dashboard



Jenkins Performance Dashboard:

Import dashboard ID: 9964 (Jenkins Prometheus Exporter Dashboard)

Click Import

