

# Capstone Project – Book-My-Show DevOps Lifecycle

Name: ARJU Kumar

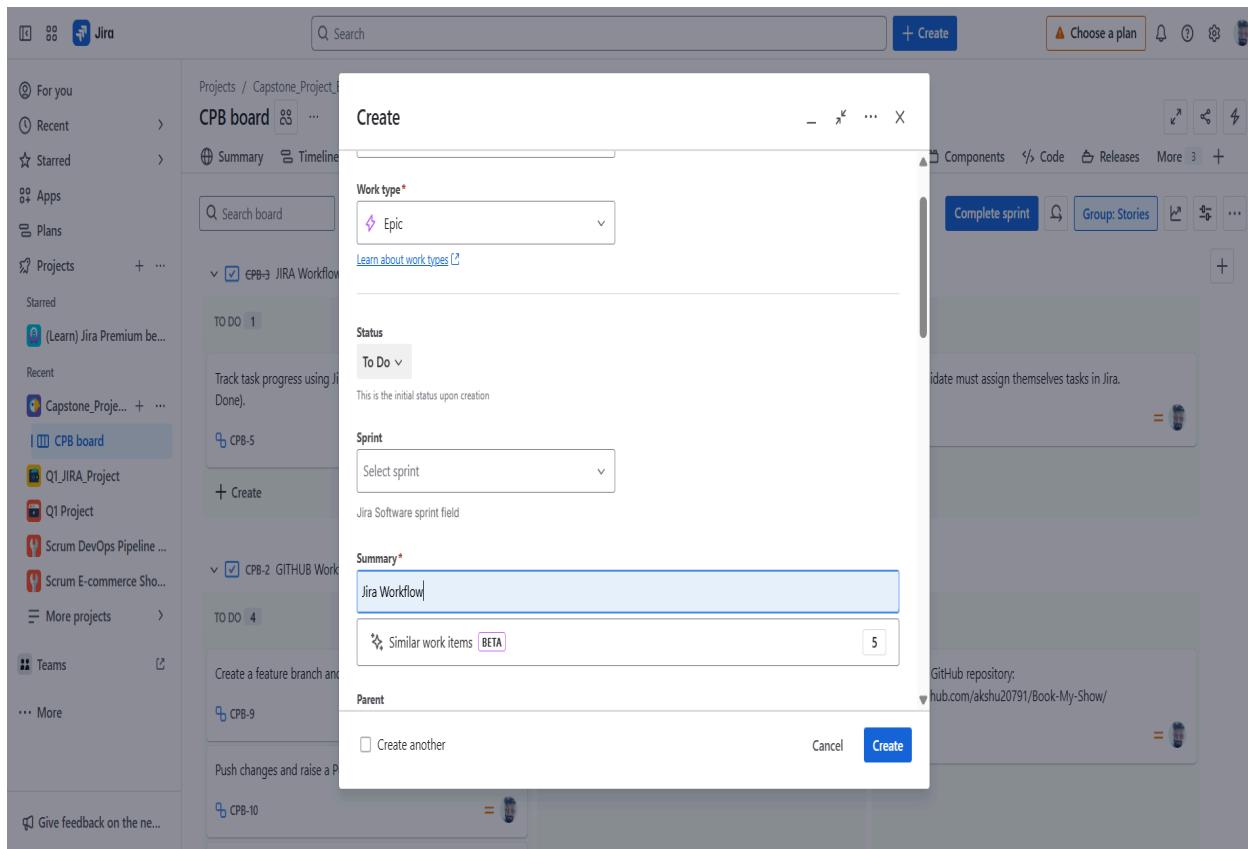
Github Repo: <https://github.com/arju7jha/Book-My-Show.git>

## Stage 1: Jira Workflow

**Step 1:** Log in to Jira workspace.

**Step 2:** Go to the **project board** for creating the tasks and subtasks for the Capstone Project and to be assigned to myself.

**Step 3:** Create “**Epic**” for each Steps of “Book-My-Show DevOps Lifecycle - Capstone Project ” and for each epic assign the subtask of the Capstone Project Steps and open each task , Click **Assign** → Choose my name to be assigned tasks to me.



**Step 4:** Now got to the “**Backlog**” tab and start the sprint and move all the tasks to the started Sprint.

Projects / Capstone\_Project\_BMS

CPB board ...

Summary Timeline Backlog Active sprints Calendar Reports List Forms Goals All work Components Code Releases More ...

Search backlog Version Epic Quick filters ...

CPB Sprint 1 13 Sep – 20 Sep (6 work items)

CPB-3 JIRA Workflow

- CPB-4 Each candidate must assign themselves tasks in Jira. (DONE)
- CPB-5 Track task progress using Jira board (To Do → In Progress → Done). (TO DO)
- CPB-6 Export the board as part of the final submission. (IN PROGRESS)

CPB-2 GITHUB Workflow

- CPB-2 GitHub Workflow (TO DO)
- CPB-13 Jenkins CI/CD Pipeline (TO DO)
- CPB-22 Docker Deployment (TO DO)
- CPB-27 Kubernetes Deployment (EKS) (TO DO)
- CPB-33 Monitoring & Observability (TO DO)

Backlog (0 work items)

0 0 0 Complete sprint ...

0 0 0 Create sprint

**Step 5:** Now got to the “**Active Sprints**” and check all the tasks and Subtasks and according to progress of the tasks and subtasks update the status from **ToDo** → **In-progress** → **Done**

For you Recent > Starred > Apps > Plans > Projects + ... Started (Learn) Jira Premium beta... Recent Capstone\_Project\_Board + ... Q1 JIRA Project Q1 Project Scrum DevOps Pipeline ... Scrum E-commerce Show... More projects > Teams ... More

Projects / Capstone\_Project\_BMS

CPB board ...

Summary Timeline Backlog Active sprints Calendar Reports List Forms Goals All work Components Code Releases Archived work items More ...

Search board Version Epic Type Quick filters ...

CPB-3 JIRA Workflow (3 subtasks) JIRA WORKFLOW DONE

TO DO	IN PROGRESS	DONE
Track task progress using Jira board (To Do → In Progress → Done). CPB-5	Export the board as part of the final submission. CPB-6	Each candidate must assign themselves tasks in Jira. CPB-4

CPB-2 GITHUB Workflow (5 subtasks) GITHUB WORKFLOW TO DO

TO DO	IN PROGRESS	DONE
Create a feature branch and make changes. CPB-9	Push changes and raise a Pull Request (PR). CPB-10	Clone the GitHub repository: https://github.com/akshu20791/Book-My-Show/ CPB-8

Give feedback on the new...

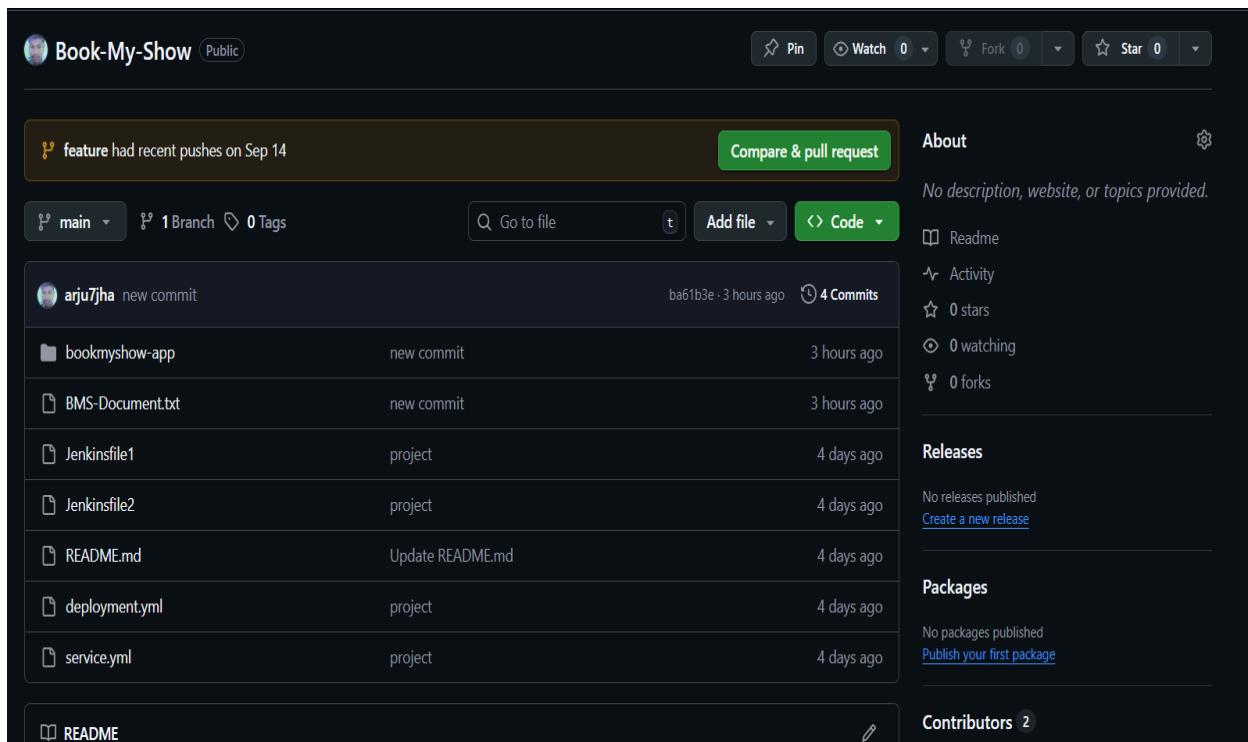
## Stage 2: GitHub Workflow

**Step 1:** Clone the GitHub repository: <https://github.com/arju7jha/Book-My-Show.git>  
using below command :

→ “**git clone <https://github.com/arju7jha/Book-My-Show.git>** ”  
→ “**cd Book-My-Show**”

**Step 2:** Create a feature branch and make changes, push changes and raise a Pull Request (PR). using below command :

→ “**git branch**”  
→ “**git checkout -b feature**”  
→ “**git branch**”  
→ “**git add .**”  
→ “**git commit -m "new feature branch"** ”  
→ “**git push -u origin feature**”



The screenshot shows the GitHub repository page for 'Book-My-Show'. At the top, there are buttons for Pin, Watch (0), Fork (0), and Star (0). A banner at the top right says 'About' with the note 'No description, website, or topics provided.' Below the banner, there are sections for 'Code' (with a dropdown menu), 'About' (with a note 'No description, website, or topics provided.'), 'Readme', 'Activity', 'Stars', 'Watching', 'Forks', 'Releases' (with a note 'No releases published' and a link 'Create a new release'), 'Packages' (with a note 'No packages published' and a link 'Publish your first package'), and 'Contributors' (with a note '2'). The main content area shows a list of recent commits:

Author	Commit Message	Time Ago	Commits
arju7jha	new commit	ba61b3e · 3 hours ago	4 Commits
bookmyshow-app	new commit	3 hours ago	
BMS-Document.txt	new commit	3 hours ago	
Jenkinsfile1	project	4 days ago	
Jenkinsfile2	project	4 days ago	
README.md	Update README.md	4 days ago	
deployment.yml	project	4 days ago	
service.yml	project	4 days ago	

**Step 3:** Review peer's PRs and approve before merging into the main branch.

#### A). Review Peers' PRs

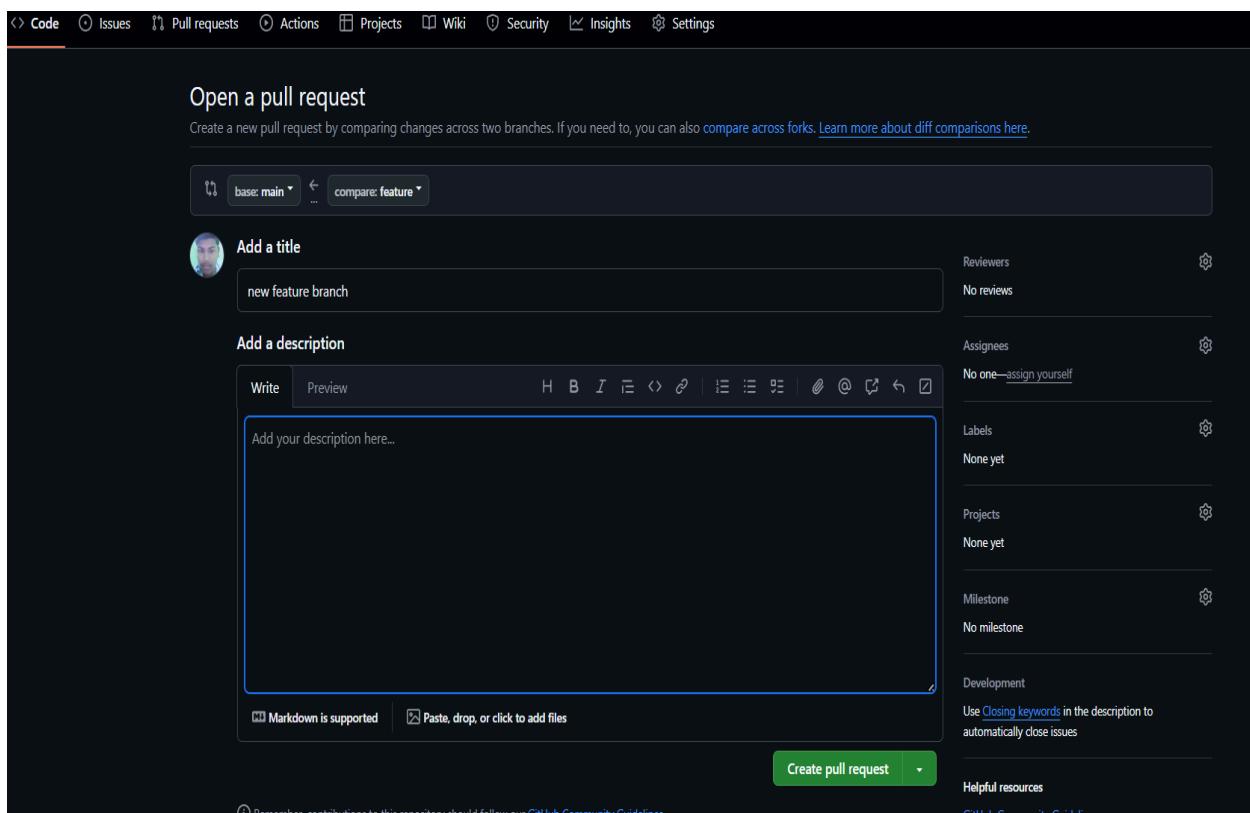
Go to the Pull Requests tab.

Open your peers' PRs, review their code, and leave comments or approve.

Make sure everything passes tests and meets requirements.

#### B). Merge PR into Main Branch

Once PRs are approved, merge into main branch using the Merge button on GitHub.



new feature branch #1

arju7jha wants to merge 1 commit into `main` from `feature`

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

**arju7jha** commented now

No description provided.

**new feature branch** a6db5b5

**No conflicts with base branch**  
Merging can be performed automatically.

**Merge pull request** You can also merge this with the command line. [View command line instructions](#).

**Add a comment**

Write Preview

Add your comment here...

Reviewers  
No reviews  
Still in progress? [Convert to draft](#)

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Development  
Successfully merging this pull request may close these

new feature branch #1

Merged arju7jha merged 1 commit into `main` from `feature` now

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

**arju7jha** commented 1 minute ago

No description provided.

**new feature branch** a6db5b5

**arju7jha** merged commit `5a51b64` into `main` now

**Revert**

**Pull request successfully merged and closed**  
You're all set — the `feature` branch can be safely deleted.

**Delete branch**

**Add a comment**

Write Preview

Add your comment here...

Reviewers  
No reviews  
Still in progress? [Convert to draft](#)

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Development  
Successfully merging this pull request may close these issues.

None yet

## Step 4: Submit the Final PR Link

→ URL of your Pull Request : <https://github.com/arju7jha/Book-My-Show/tree/feature>

## Stage 3: Jenkins CI/CD Pipeline

**Step 1:** Cleared Jenkins' workspace before starting the build so there were **no leftover files** from previous runs.

**Step 2:** Use Jenkins to pull the latest code from our GitHub repository. This ensures the pipeline always works with the latest code changes.



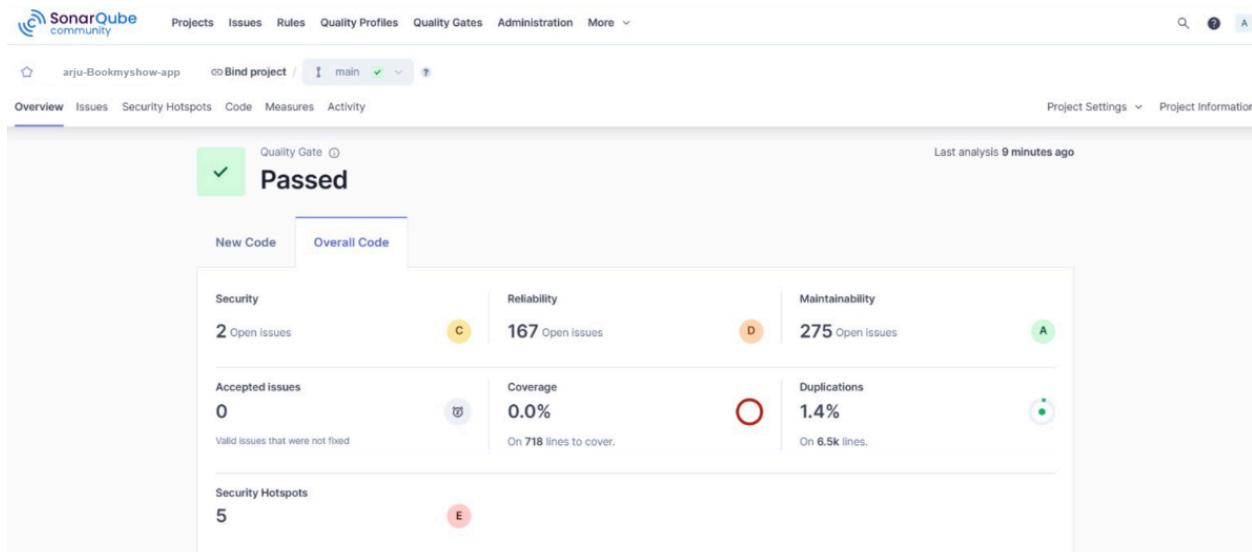
The screenshot shows the Jenkins pipeline stage view for the 'arju-capstone-project'. The stage view displays the following stages and their average times:

Stage	Average Time
Declarative: Tool Install	104ms
Clean Workspace	189ms
Checkout from Git	2s
SonarQube Analysis	21s
Quality Gate	311ms
Build Docker Image	4s
Push to Docker Hub	9s
Deploy Container	1s
Declarative: Post Actions	4s

Additional details shown in the screenshot include:

- Build date: Sep 14 21:12
- Changes: No Changes
- Permalinks: Builds (Today: #1 3:42PM)

**Step 3:** Run the SonarQube to analyze the code for bugs, security issues, and code smells. The pipeline only continued if the quality gate passed (good, clean code).



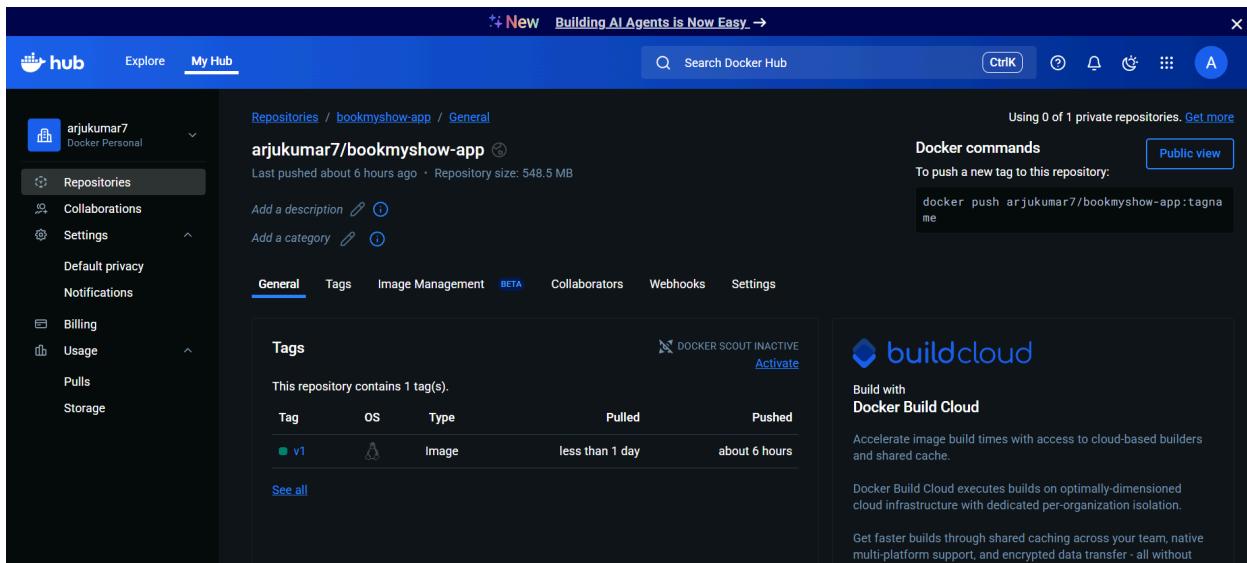
The screenshot shows the SonarQube Quality Gate analysis for the 'arju-Bookmyshow-app' project. The analysis is marked as 'Passed'. Key metrics displayed include:

Metric	Value	Grade
Security	2 Open issues	C
Reliability	167 Open issues	D
Maintainability	275 Open issues	A
Accepted issues	0	B
Coverage	0.0%	O
Duplications	1.4%	●
Security Hotspots	5	E

Other tabs visible in the SonarQube interface include: Overview, Issues, Security Hotspots, Code, Measures, Activity, Project Settings, and Project Information.

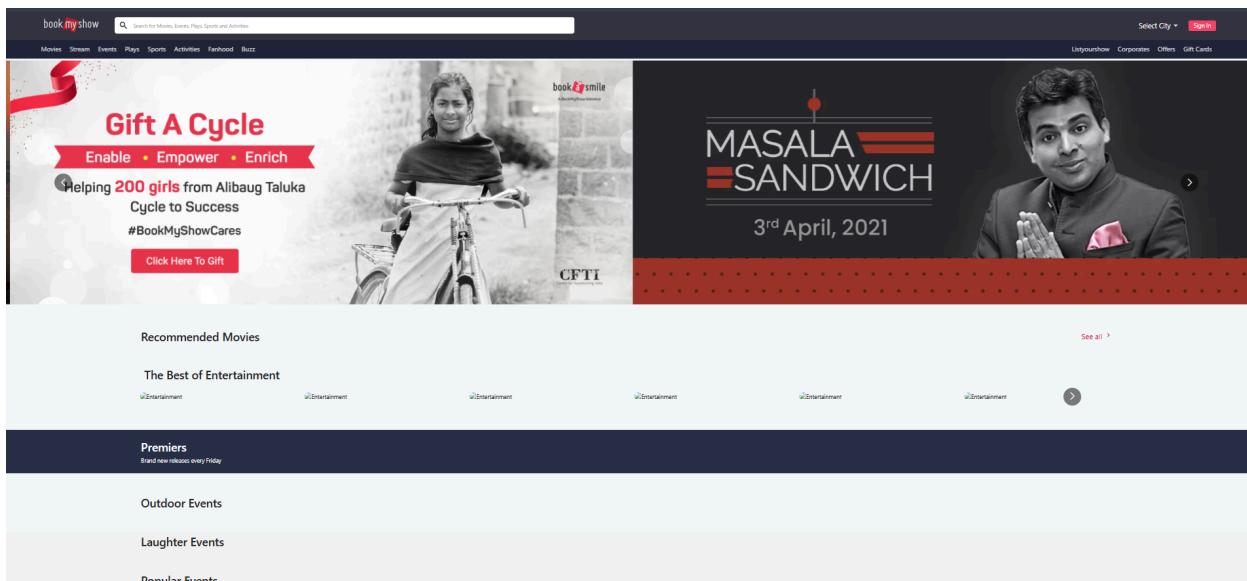
**Step 4:** Installed all the Node.js dependencies by running: “ **npm install** ”

**Step 5:** Created a Docker image of the app and pushed it to DockerHub with a version tag. This makes the app container image available anywhere for deployment.



The screenshot shows the DockerHub interface. On the left, a sidebar for the user 'arjukumar7' is visible with sections for Repositories, Collaborations, Settings, Default privacy, Notifications, Billing, Usage, Pulls, and Storage. The main content area shows a repository named 'bookmyshow-app' under the 'General' tab. It displays a single tag 'v1' with the following details: OS (None), Type (Image), Pulled (less than 1 day ago), and Pushed (about 6 hours ago). A Docker command box shows the command 'docker push arjukumar7/bookmyshow-app:v1'. A sidebar on the right for 'buildcloud' offers to build with Docker Build Cloud, mentioning faster builds through shared caching and optimized cloud infrastructure.

**Step 6:** Run the Docker container using the image we just built and pushed to dockerhub. The app is now running live on port 3000.



The screenshot shows the BookMyShow website homepage. The top navigation bar includes 'bookmyshow', a search bar, and a 'Sign In' button. The main banner features a woman on a bicycle with the text 'Gift A Cycle' and 'Helping 200 girls from Alibaug Taluka Cycle to Success'. To the right, there's a promotional image for 'MASALA SANDWICH' with the date '3rd April, 2021'. Below the banner, there are sections for 'Recommended Movies', 'The Best of Entertainment', 'Premiers' (new releases every Friday), 'Outdoor Events', 'Laughter Events', and 'Doodhla Events'. The footer includes links for 'Select City', 'About Us', 'Corporate', 'Offers', and 'Gift Cards'.

**Step 7:** Configure Jenkins to send an email after the build. The email tells us whether the build succeeded or failed to keep the team updated.

## Stage 4: Docker Deployment

**Step 1:** Create a **Dockerfile** at the root of the Book-My-Show project to build the BMS app image.

```
ne_Project > Book-My-Show > bookmyshow-app > Dockerfile
FROM node:18

WORKDIR /app

COPY package.json package-lock.json ./

# Install chokidar@3 explicitly (fix for missing dependency)
RUN npm install chokidar@3 --legacy-peer-deps

# Your existing postcss fix
RUN npm install postcss@8.4.21 postcss-safe-parser@6.0.0 --legacy-peer-deps

# Install rest of the dependencies
RUN npm install --legacy-peer-deps

COPY . .

EXPOSE 3000

ENV NODE_OPTIONS=--openssl-legacy-provider
ENV PORT=3000

CMD ["npm", "start"]
```

**Step 2:** To Build a **Docker Image** and push to DockerHub via Jenkins created a Jenkins file and build the script in jenkins.

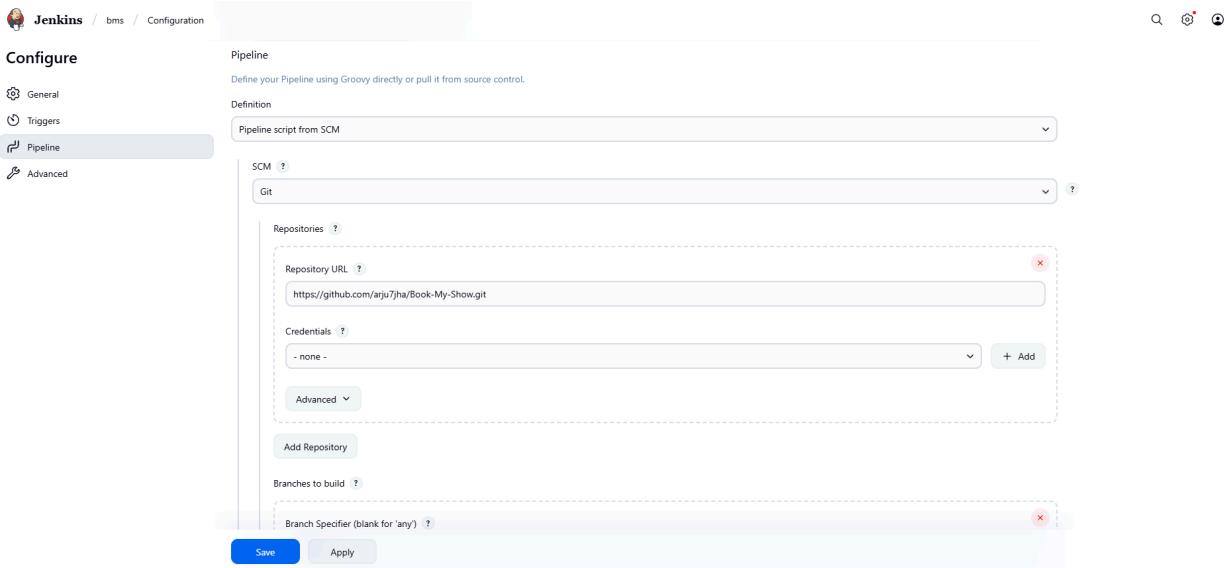
→ In Jenkins go to manage Jenkins and add DockerHub Credentials as: **dockerhub-cred**

→ To run Jenkins Pipeline and push to DockerHub via Jenkins :

- A). Create a new Jenkins Pipeline job.
- B). Under **Definition**, choose: **Pipeline script from SCM**
- C). In **SCM**: Select **Git** and the **Repository URL**:
- D). Enter the repo branch.

E). Provide the path of the jenkinsfile and click one apply and save .

F). Now build the pipeline .



Jenkins / bms / Configuration

Configure

Pipeline

General

Triggers

Pipeline

Advanced

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/arju7jha/Book-My-Show.git

Credentials

- none -

Advanced

Add Repository

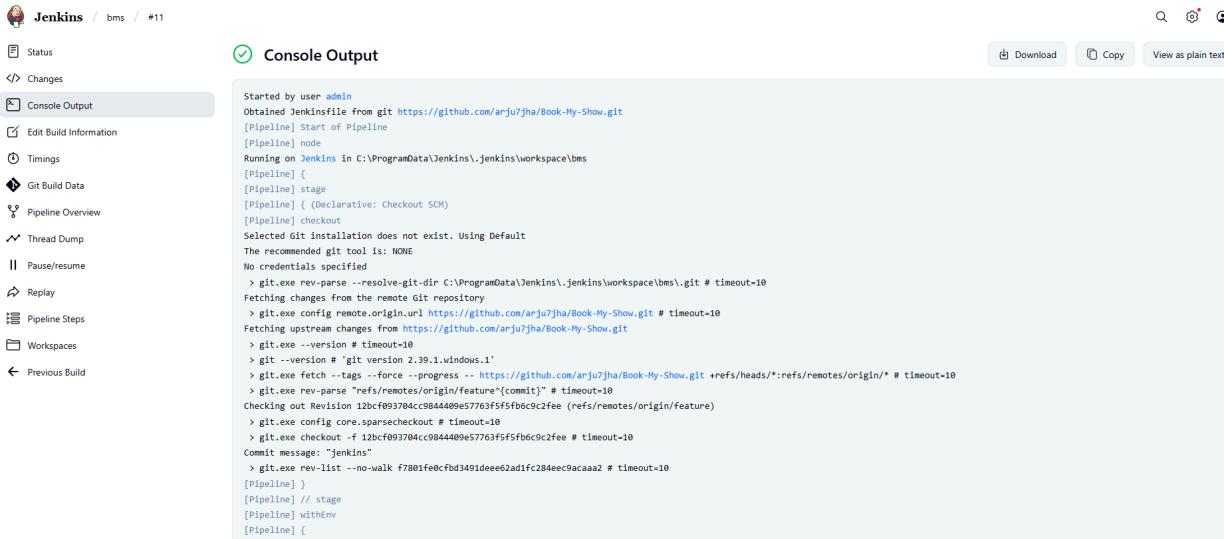
Branches to build

Branch Specifier (blank for 'any')

Save

Apply

Build Succeeded:



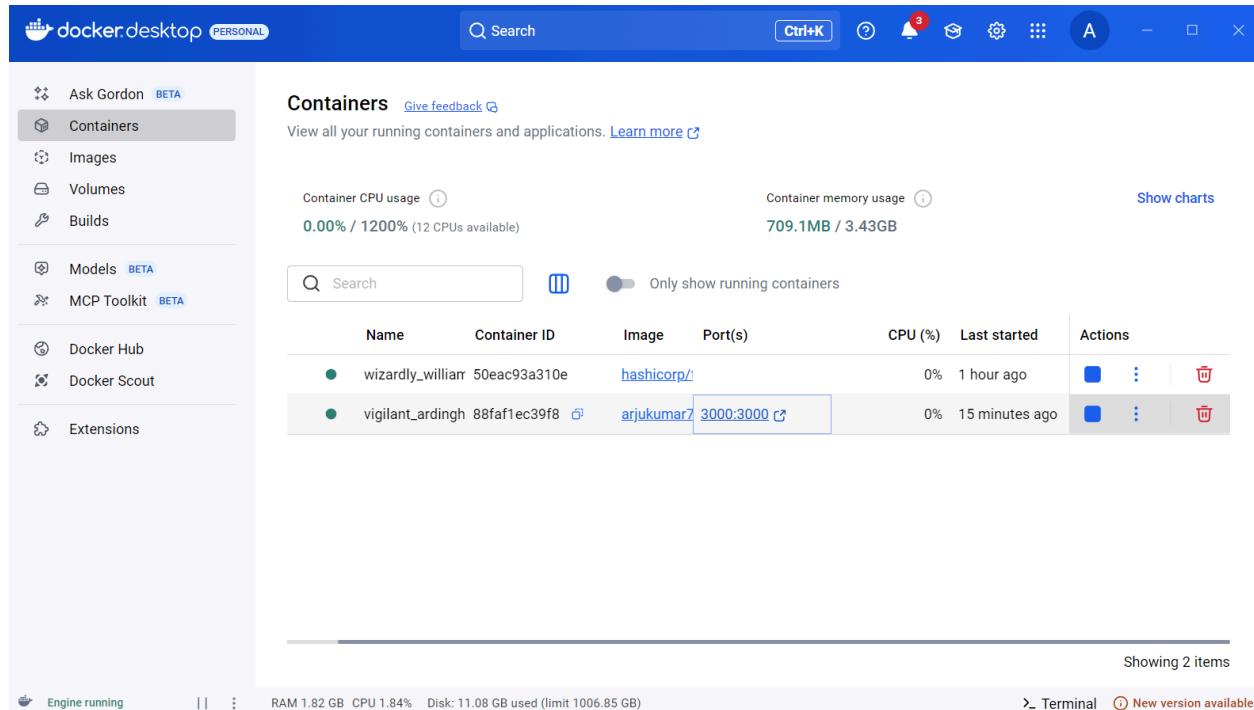
**Step 3:** To run container locally and validate accessibility on port 3000.

→ Run the container locally after pushing the image, run the container locally to verify it works:

**A)** use command: “**docker run -d -p 3000:3000 arjukumar7/bookmyshow-app:v1**”

→ Then, open your browser and visit: “ <http://localhost:3000> ”

→ Below attached the BMS application running proof on port 3000.



Docker Desktop PERSONAL

Containers [Give feedback](#) [Learn more](#)

View all your running containers and applications.

Container CPU usage: 0.00% / 1200% (12 CPUs available)

Container memory usage: 709.1MB / 3.43GB

Show charts

Search

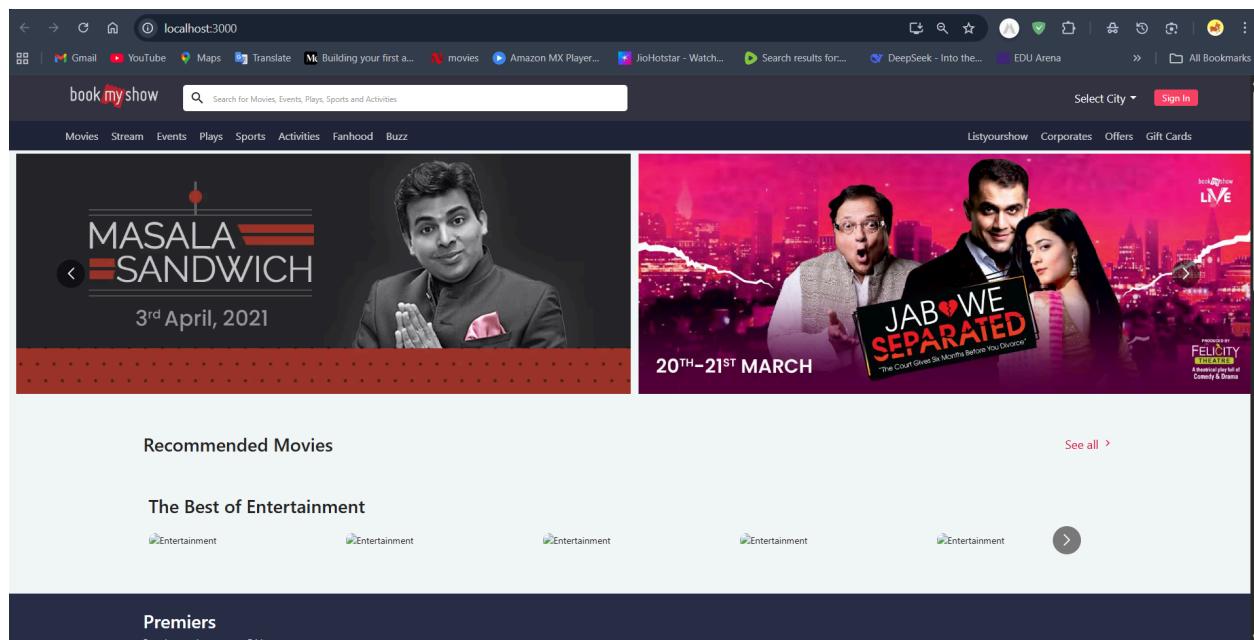
Only show running containers

Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
wizardly_william	50eac93a310e	hashicorp/		0%	1 hour ago	<a href="#">Stop</a> <a href="#">More</a> <a href="#">Delete</a>
vigilant_ardingh	88faf1ec39f8	arjukumar7	3000:3000	0%	15 minutes ago	<a href="#">Stop</a> <a href="#">More</a> <a href="#">Delete</a>

Showing 2 items

Engine running | RAM 1.82 GB CPU 1.84% Disk: 11.08 GB used (limit 1006.85 GB) | Terminal | [New version available](#)

→ Here you can see the BMS application running successfully locally on port 3000.



localhost:3000

bookmyshow

MASALA SANDWICH

3rd April, 2021

JAB WE SEPARATED

20TH-21ST MARCH

Recommended Movies

The Best of Entertainment

Premiers

## Stage 5: Kubernetes Deployment (EKS)

**Step 1:** Install all the necessary tools and setup the configuration to run the Cluster

```
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/WHEEL
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/RECORD
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/INSTALLER
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/top_level.txt
inflating: aws/dist/wheel-0.45.1.dist-info/entry_points.txt
inflating: aws/dist/wheel-0.45.1.dist-info/METADATA
inflating: aws/dist/wheel-0.45.1.dist-info/RECORD
inflating: aws/dist/wheel-0.45.1.dist-info/REQUESTED
inflating: aws/dist/wheel-0.45.1.dist-info/direct_url.json
inflating: aws/dist/wheel-0.45.1.dist-info/LICENSE.txt
inflating: aws/dist/wheel-0.45.1.dist-info/WHEEL
inflating: aws/dist/wheel-0.45.1.dist-info/INSTALLER
root@ip-10-0-0-55:/home/ubuntu# sudo ./aws/install
You can now run: /usr/local/bin/aws --version
root@ip-10-0-0-55:/home/ubuntu# aws --version
aws-cli/2.30.1 Python/3.13.7 Linux/6.14.0-1011-aws exe/x86_64.ubuntu.24
root@ip-10-0-0-55:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIA5HTN1M2UD67M3WUP
AWS Secret Access Key [None]: 4tWe1BtKqv6YxiDyx8c8LKkhCMvBnMmPGmtpkVm9
default region name [None]: us-east-1
default output format [None]:
root@ip-10-0-0-55:/home/ubuntu# kubectl get ns
The connection to the server localhost:8080 was refused - did you specify the right host or port?
root@ip-10-0-0-55:/home/ubuntu# eksctl create cluster \
  --name user1-cluster \
  --region us-east-1 \
  --zones us-east-1a,us-east-1b \
  --nodegroup-name user1-node-gp
2025-09-15 04:57:54 [I] eksctl version 0.214.0
2025-09-15 04:57:54 [I] using region us-east-1
2025-09-15 04:57:54 [I] subnets for us-east-1a - public:192.168.0.0/19 private:192.168.64.0/19
i-00bc368d8869a57fd (arju-eks)
```

→ Successfully set up the EKS cluster .

```
2025-09-15 05:06:56 [I] successfully created addon: metrics-server
2025-09-15 05:06:56 [I] recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl cannot configure the requested permissions: the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identity associations; after addon creation is completed, add all recommended policies to the config file, under 'addon.PodIdentityAssociations', and run 'eksctl update addon'
2025-09-15 05:06:56 [I] creating addon: vpc-cni
2025-09-15 05:06:56 [I] successfully created addon: vpc-cni
2025-09-15 05:06:57 [I] creating addon: kube-proxy
2025-09-15 05:06:57 [I] successfully created addon: kube-proxy
2025-09-15 05:06:57 [I] creating addon: coredns
2025-09-15 05:06:58 [I] successfully created addon: coredns
2025-09-15 05:06:58 [I] building managed nodegroup stack "eksctl-user1-cluster-nodegroup-user1-node-gp"
2025-09-15 05:06:58 [I] deploying stack "eksctl-user1-cluster-nodegroup-user1-node-gp"
2025-09-15 05:06:58 [I] waiting for CloudFormation stack "eksctl-user1-cluster-nodegroup-user1-node-gp"
2025-09-15 05:09:28 [I] waiting for CloudFormation stack "eksctl-user1-cluster-nodegroup-user1-node-gp"
2025-09-15 05:11:56 [I] waiting for CloudFormation stack "eksctl-user1-cluster-nodegroup-user1-node-gp"
2025-09-15 05:11:56 [I] waiting for the control plane to become ready
2025-09-15 05:11:57 [V] saved kubeconfig as "/root/.kube/config"
2025-09-15 05:11:57 [I] no tasks
2025-09-15 05:11:57 [I] all EKS cluster resources for "user1-cluster" have been created
2025-09-15 05:11:57 [I] nodegroup "user1-node-gp" has 2 node(s)
2025-09-15 05:11:57 [I] node "ip-192-168-26-201.ec2.internal" is ready
2025-09-15 05:11:57 [I] node "ip-192-168-43-68.ec2.internal" is ready
2025-09-15 05:11:57 [I] waiting for at least 2 node(s) to become ready in "user1-node-gp"
2025-09-15 05:11:57 [I] nodegroup "user1-node-gp" has 2 node(s)
2025-09-15 05:11:57 [I] node "ip-192-168-26-201.ec2.internal" is ready
2025-09-15 05:11:57 [I] node "ip-192-168-43-68.ec2.internal" is ready
2025-09-15 05:11:57 [I] created 1 managed nodegroup(s) in cluster "user1-cluster"
2025-09-15 05:11:58 [I] kubectl command should work with "/root/.kube/config", try 'kubectl get nodes'
2025-09-15 05:11:58 [V] EKS cluster "user1-cluster" in "us-east-1" region is ready
root@ip-10-0-0-55:/home/ubuntu#
```

**Step 2:** Created two files of Kubernetes manifests (`deployment.yaml`, `service.yaml`):

→ **deployment.yaml** – In this file defines how many replicas (pods) i want, what container image to use, and other configurations.

i-00bc368d8869a57fd (arju-eks)

→ **service.yaml** – In this file exposes the application to use by using the NodePort or LoadBalancer.

```
apiVersion: v1
kind: Service
metadata:
  name: bookmyshow-service
  namespace: arju
spec:
  type: LoadBalancer
  selector:
    app: bookmyshow
  ports:
    - protocol: TCP
      port: 80
      targetPort: 3000
```

**Step 3:** To deploy on EKS Cluster used the commands “**kubectl apply**” to deploy the app, and to create the pods and service inside the EKS cluster by using below commands:.

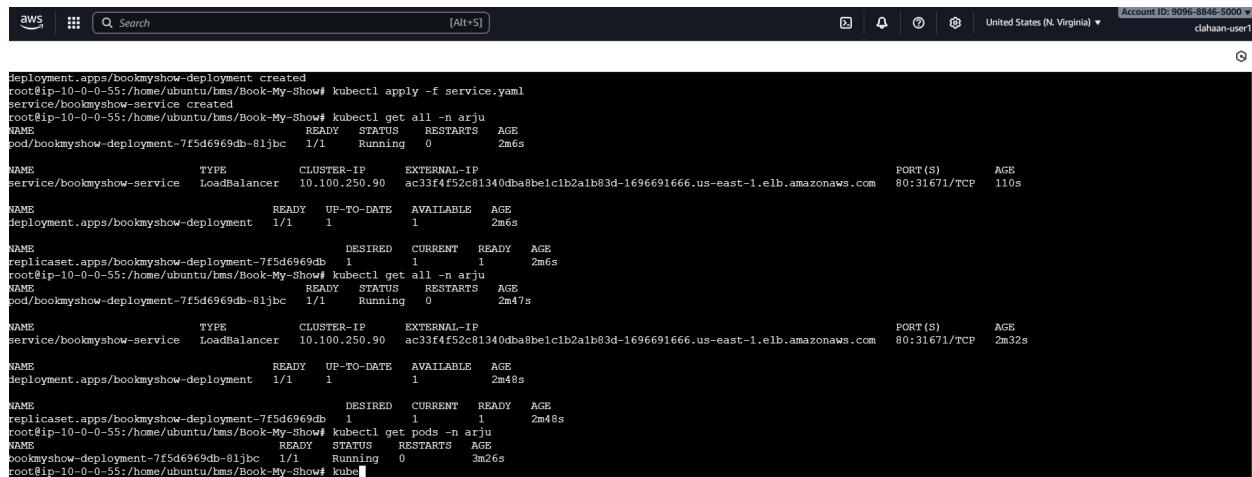
- “ **kubectl apply -f deployment.yaml** ”
- “ **kubectl apply -f service.yaml** ”

```
remote: Enumerating objects: 188, done.
remote: Counting objects: 100% (188/188), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 188 (delta 46), reused 168 (delta 33), pack-reused 0 (from 0)
Receiving objects: 100% (188/188), 1.60 MiB | 30.83 MiB/s, done.
Resolving deltas: 100% (46/46), done.
root@ip-10-0-0-55:/home/ubuntu/bms# cd ..
root@ip-10-0-0-55:/home/ubuntu# kubectl create ns arju
namespace/arju created
root@ip-10-0-0-55:/home/ubuntu# ls
aws awscli2.zip bms
root@ip-10-0-0-55:/home/ubuntu# kubectl get ns
NAME          STATUS   AGE
arju          Active   20s
default       Active   25m
kube-node-lease Active   25m
kube-public   Active   25m
kube-system   Active   25m
root@ip-10-0-0-55:/home/ubuntu# cd bms/
root@ip-10-0-0-55:/home/ubuntu/bms# ls
Book-My-Show
root@ip-10-0-0-55:/home/ubuntu/bms# cd Book-My-Show/
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# ls
Jenkinsfile Jenkinsfile Jenkinsfile2 README.md bookmyshow-app deployment.yaml deployment.yaml service.yaml service.yaml
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# vi deployment.yaml
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# vi service.yaml
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl apply -f deployment.yaml
deployment.apps/bookmyshow-deployment created
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl apply -f service.yaml
service/bookmyshow-service created
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show#
```

i-00bc368d8869a57fd (arju-eks)

**Step 4:** Exposed service using a NodePort or LoadBalancer in **service.yaml** to make the app accessible.

- **NodePort:** Opens a specific port on each node.
- **LoadBalancer:** Creates an AWS ELB (Elastic Load Balancer) to route traffic automatically.



```
aws | Search [Alt+S] Account ID: 9096-8846-5000
Deployment.apps/bookmyshow-deployment created
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl apply -f service.yaml
service/bookmyshow-service created
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get all -n arju
NAME          READY   STATUS    RESTARTS   AGE
pod/bookmyshow-deployment-7f5d6969db-8ljbc  1/1    Running   0          2m6s
NAME          TYPE    CLUSTER-IP   EXTERNAL-IP
service/bookmyshow-service  LoadBalancer  10.100.250.90  ac33f4f52c81340dba8be1c1b2a1b83d-1696691666.us-east-1.elb.amazonaws.com  PORT(S)        AGE
service/bookmyshow-service  LoadBalancer  10.100.250.90  ac33f4f52c81340dba8be1c1b2a1b83d-1696691666.us-east-1.elb.amazonaws.com  80:31671/TCP  110s
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bookmyshow-deployment  1/1     1           1           2m6s
NAME          DESIRED  CURRENT   READY   AGE
replicaset.apps/bookmyshow-deployment-7f5d6969db  1       1       1       2m6s
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get all -n arju
NAME          READY   STATUS    RESTARTS   AGE
pod/bookmyshow-deployment-7f5d6969db-8ljbc  1/1    Running   0          2m47s
NAME          TYPE    CLUSTER-IP   EXTERNAL-IP
service/bookmyshow-service  LoadBalancer  10.100.250.90  ac33f4f52c81340dba8be1c1b2a1b83d-1696691666.us-east-1.elb.amazonaws.com  PORT(S)        AGE
service/bookmyshow-service  LoadBalancer  10.100.250.90  ac33f4f52c81340dba8be1c1b2a1b83d-1696691666.us-east-1.elb.amazonaws.com  80:31671/TCP  2m32s
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bookmyshow-deployment  1/1     1           1           2m48s
NAME          DESIRED  CURRENT   READY   AGE
replicaset.apps/bookmyshow-deployment-7f5d6969db  1       1       1       2m48s
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get pods -n arju
NAME          READY   STATUS    RESTARTS   AGE
bookmyshow-deployment-7f5d6969db-8ljbc  1/1    Running   0          3m26s
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl
```

**Step 5:** To validate Deployment to make sure that everything is running correctly by using:

- “**kubectl get pods**” : Shows a **list of all pods** running in cluster
- “**kubectl get svc**” : Shows a **list of all services** running in a cluster and gives us the external IP or NodePort to access the app.

```
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get pods -n arju
NAME          READY   STATUS    RESTARTS   AGE
bookmyshow-deployment-7f5d6969db-8ljqbc   1/1     Running   0          11m
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get svc -n arju
NAME          TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
bookmyshow-service   LoadBalancer  10.100.250.90  ac33f4f52c8b1340d8a8b1c1b2a1b83d-1696691666.us-east-1.elb.amazonaws.com  80:31671/TCP   12m
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get all -n arju
NAME                                         READY   STATUS    RESTARTS   AGE
pod/bookmyshow-deployment-7f5d6969db-8ljqbc   1/1     Running   0          14m
NAME                                         TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/bookmyshow-service   LoadBalancer  10.100.250.90  ac33f4f52c8b1340d8a8b1c1b2a1b83d-1696691666.us-east-1.elb.amazonaws.com  80:31671/TCP   13m
NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bookmyshow-deployment   1/1     1           1           14m
NAME                                         DESIRED  CURRENT   READY   AGE
replicaset.apps/bookmyshow-deployment-7f5d6969db   1        1        1        14m
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl describe pod bookmyshow-deployment-7f5d6969db-8ljqbc -n arju
Name:          bookmyshow-deployment-7f5d6969db-8ljqbc
Namespace:    arju
Priority:     0
Node:         ip-192-168-26-201.ec2.internal/192.168.26.201
Start Time:   Mon, 15 Sep 2025 05:35:47 +0000
Labels:        app=bookmyshow
               pod-template-hash=7f5d6969db
Annotations:  <none>
Status:       Running
```

```
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get pods -n arju
NAME          READY   STATUS    RESTARTS   AGE
bookmyshow-deployment-7f5d6969db-8ljqbc   1/1     Running   0          11m
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get svc -n arju
NAME          TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
bookmyshow-service   LoadBalancer  10.100.250.90  ac33f4f52c8b1340d8a8b1c1b2a1b83d-1696691666.us-east-1.elb.amazonaws.com  80:31671/TCP   12m
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl get all -n arju
NAME                                         READY   STATUS    RESTARTS   AGE
pod/bookmyshow-deployment-7f5d6969db-8ljqbc   1/1     Running   0          14m
NAME                                         TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/bookmyshow-service   LoadBalancer  10.100.250.90  ac33f4f52c8b1340d8a8b1c1b2a1b83d-1696691666.us-east-1.elb.amazonaws.com  80:31671/TCP   13m
NAME                                         READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/bookmyshow-deployment   1/1     1           1           14m
NAME                                         DESIRED  CURRENT   READY   AGE
replicaset.apps/bookmyshow-deployment-7f5d6969db   1        1        1        14m
root@ip-10-0-0-55:/home/ubuntu/bms/Book-My-Show# kubectl describe pod bookmyshow-deployment-7f5d6969db-8ljqbc -n arju
Name:          bookmyshow-deployment-7f5d6969db-8ljqbc
Namespace:    arju
Priority:     0
Node:         ip-192-168-26-201.ec2.internal/192.168.26.201
Start Time:   Mon, 15 Sep 2025 05:35:47 +0000
Labels:        app=bookmyshow
               pod-template-hash=7f5d6969db
Annotations:  <none>
Status:       Running
```

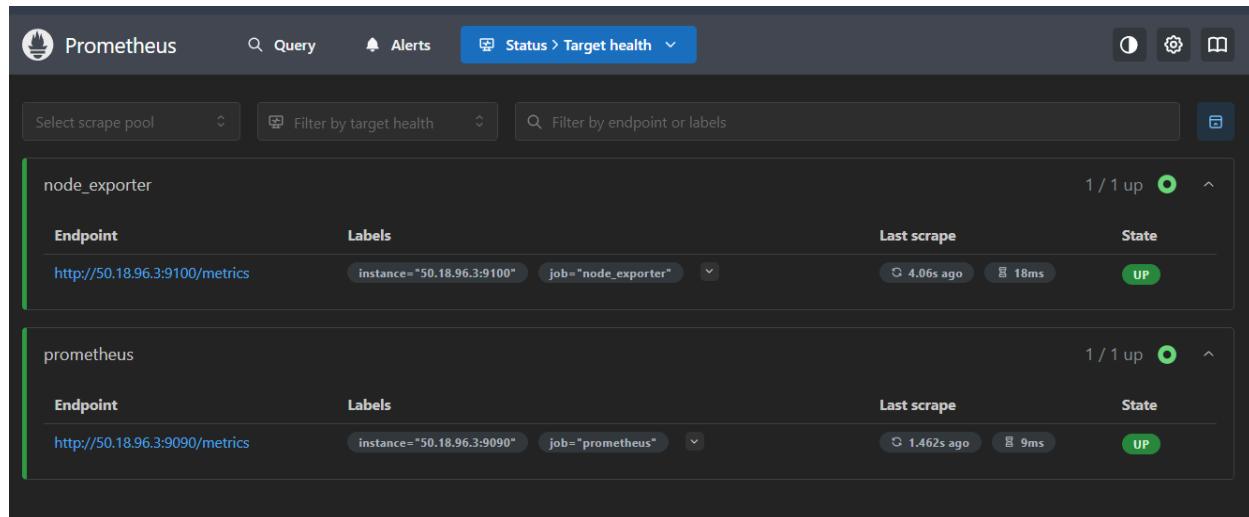
## Stage 6: Monitoring & Observability

### Step 1: Install Prometheus & Node Exporter

- Deployed Prometheus on our Kubernetes cluster using Helm.
- Installed Node Exporter on each node to collect system-level metrics (CPU, Memory, Disk, Network).
- Verified Prometheus was running by checking pods and services with:
  - “**kubectl get pods -n monitoring**”
  - “**kubectl get svc -n monitoring**”

```
service/grafana-external exposed
root@ip-10-0-0-55:/home/ubuntu# kubectl get svc -n monitoring
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP
grafana       ClusterIP  10.100.98.219  <none>
grafana-external LoadBalancer 10.100.190.221  a1fc1f3f66167400886e7eec77151ab2-444174145.us-east-1.elb.amazonaws.com
prometheus-alertmanager ClusterIP  10.100.62.168  <none>
prometheus-alertmanager-headless ClusterIP  None  <none>
prometheus-kube-state-metrics ClusterIP  10.100.196.207  <none>
prometheus-prometheus-node-exporter ClusterIP  10.100.197.246  <none>
prometheus-prometheus-pushgateway ClusterIP  10.100.235.184  <none>
prometheus-server ClusterIP  10.100.206.230  <none>
root@ip-10-0-0-55:/home/ubuntu# kubectl expose deployment prometheus-server \
  --type=LoadBalancer \
  --name=prometheus-external \
  --port=80 \
  --target-port=9090 \
  -n monitoring
service/prometheus-external exposed
root@ip-10-0-0-55:/home/ubuntu# kubectl get svc -n monitoring
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP
grafana       ClusterIP  10.100.98.219  <none>
grafana-external LoadBalancer 10.100.190.221  a1fc1f3f66167400886e7eec77151ab2-444174145.us-east-1.elb.amazonaws.com
prometheus-alertmanager ClusterIP  10.100.62.168  <none>
prometheus-alertmanager-headless ClusterIP  None  <none>
prometheus-kube-state-metrics LoadBalancer 10.100.120.3  a76b048369f941139b820e6447e5ab9-750060204.us-east-1.elb.amazonaws.com
prometheus-prometheus-node-exporter ClusterIP  10.100.196.207  <none>
prometheus-prometheus-pushgateway ClusterIP  10.100.197.246  <none>
prometheus-server ClusterIP  10.100.235.184  <none>
prometheus-grafana ClusterIP  10.100.206.230  <none>
root@ip-10-0-0-55:/home/ubuntu# ``
```

### Step 2: Enabled Jenkins Prometheus plugin to expose metrics at /prometheus.



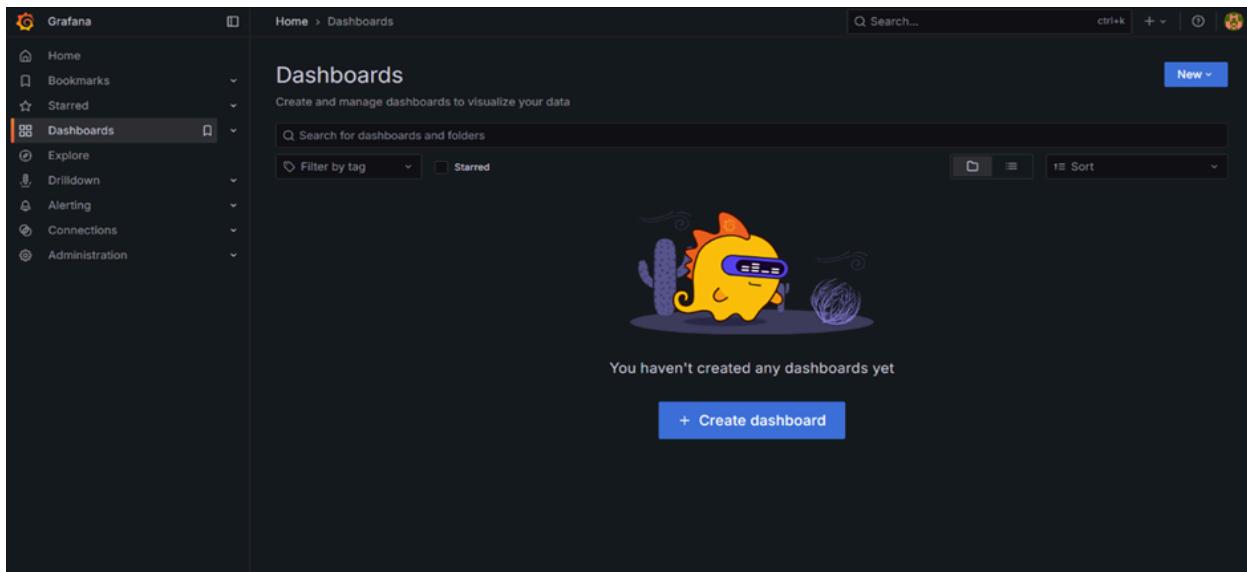
The screenshot shows the Prometheus web interface with two targets listed:

- node\_exporter**: An endpoint at <http://50.18.96.3:9100/metrics> with labels `instance="50.18.96.3:9100"` and `job="node_exporter"`. Last scraped 4.06s ago, 18ms response time, and status **UP**.
- prometheus**: An endpoint at <http://50.18.96.3:9090/metrics> with labels `instance="50.18.96.3:9090"` and `job="prometheus"`. Last scraped 1.462s ago, 9ms response time, and status **UP**.

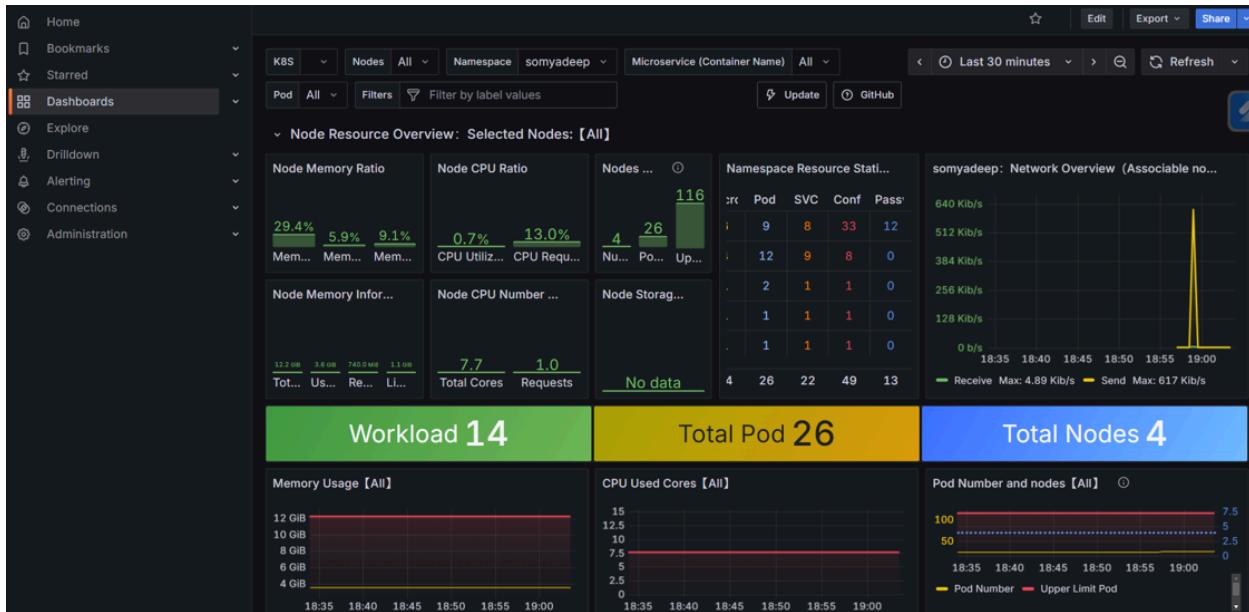
**Step 3:** Verified that Prometheus was successfully gathering metrics data Source from the application, Node Exporter, and Jenkins, confirming that the monitoring stack was fully functional and ready to be integrated with Grafana.

```
# HELP default_jenkins_version_info Jenkins Application Version
# TYPE default_jenkins_version_info gauge
default_jenkins_version_info{version="2.516.2"} 1.0
# HELP default_jenkins_up Is Jenkins ready to receive requests
# TYPE default_jenkins_up gauge
default_jenkins_up 1.0
# HELP default_jenkins_uptime Since Jenkins machine was initialized
# TYPE default_jenkins_uptime gauge
default_jenkins_uptime 1018880.0
# HELP default_jenkins_nodes_online Jenkins nodes online status
# TYPE default_jenkins_nodes_online gauge
# HELP default_jenkins_quietdown Is Jenkins in quiet mode
# TYPE default_jenkins_quietdown gauge
default_jenkins_quietdown 0.0
# HELP jvm_memory_pool_allocated_bytes_total Total bytes allocated in a given JVM memory pool. Only updated after GC, not continuously.
# TYPE jvm_memory_pool_allocated_bytes_total counter
jvm_memory_pool_allocated_bytes_total{pool="CodeHeap 'profiled methods'"} 1.3273216E7
jvm_memory_pool_allocated_bytes_total{pool="CodeHeap 'non-profiled methods'"} 7.42736E6
jvm_memory_pool_allocated_bytes_total{pool="G1 Eden Space"} 1.048576E8
jvm_memory_pool_allocated_bytes_total{pool="G1 Survivor Space"} 3844608.0
jvm_memory_pool_allocated_bytes_total{pool="Compressed Class Space"} 718552.0
jvm_memory_pool_allocated_bytes_total{pool="Metaspace"} 9639352.0
jvm_memory_pool_allocated_bytes_total{pool="CodeHeap 'non-nmethods'"} 3179648.0
# HELP process_cpu_seconds_total Total user and system CPU time spent in seconds.
# TYPE process_cpu_seconds_total counter
process_cpu_seconds_total 18.58
# HELP process_start_time_seconds Start time of the process since unix epoch in seconds.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.757784007786E9
# HELP process_open_fds Number of open file descriptors.
# TYPE process_open_fds gauge
process_open_fds 310.0
# HELP process_max_fds Maximum number of open file descriptors.
# TYPE process_max_fds gauge
process_max_fds 524288.0
# HELP process_virtual_memory_bytes Virtual memory size in bytes.
# TYPE process_virtual_memory_bytes gauge
process_virtual_memory_bytes 3.786080256E9
# HELP process_resident_memory_bytes Resident memory size in bytes.
# TYPE process_resident_memory_bytes gauge
process_resident_memory_bytes 3.5270656E8
# HELP jvm_buffer_pool_used_bytes Used bytes of a given JVM buffer pool.
# TYPE jvm_buffer_pool_used_bytes gauge
jvm_buffer_pool_used_bytes{pool="mapped"} 8.0
jvm_buffer_pool_used_bytes{pool="direct"} 69097.0
```

**Step 3:** Add dashboards for Node health and Jenkins performance.

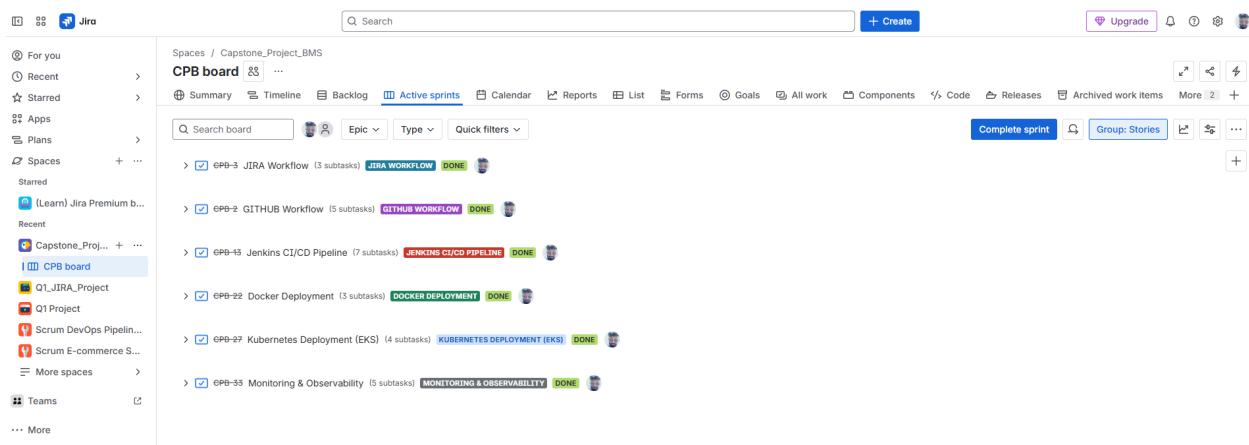


**Step 4:** Used Grafana Dashboard ID 15661, which is built for monitoring Linux system metrics through Prometheus Node Exporter. It offers a comprehensive, real-time view of system performance, helping you easily monitor resource usage and overall system health.



## Jira Submission: Task completion proof

→ Completed all the assigned tasks and updated them in the jira dashboard also .



Jira

Spaces / Capstone\_Project\_BMS

**CPB board** ...

Summary Timeline Backlog Active sprints Calendar Reports List Forms Goals All work Components Code Releases Archived work items More 2 +

Search backlog Version Epic Quick filters

Search backlog Version Epic Quick filters

Epics

- GPB-3 JIRA Workflow DONE
- GPB-2 GITHUB Workflow DONE
- GPB-13 Jenkins CI/CD Pipeline DONE
- GPB-22 Docker Deployment DONE
- GPB-27 Kubernetes Deployment (EKS) DONE
- GPB-29 Candidates must write their own Kubernetes manifests ('deployment...' DONE)
- GPB-30 Deploy the application on EKS cluster. DONE
- GPB-31 Expose service using NodePort or LoadBalancer. DONE
- GPB-32 Validate deployment using 'kubectl get pods' and 'kubectl get svc'. DONE
- GPB-33 Monitoring & Observability DONE
- GPB-35 Install Prometheus and Node Exporter to collect metrics. DONE
- GPB-36 Integrate Jenkins metrics into Prometheus. DONE
- GPB-37 Install Grafana, configure Prometheus as a data source. DONE
- GPB-38 Add dashboards for Node health and Jenkins performance. DONE
- GPB-39 Submit Grafana screenshots in deliverables. DONE

Subtasks

Work	P...	St...	A...	Status
GPB-35 Install Prom...	=			DONE
GPB-36 Integrat...	=			DONE
GPB-37 Instal...	=			DONE
GPB-38 Add dash...	=			DONE
GPB-39 Submit Graf...	=			DONE

Monitoring & Observability

CPB-34 / CPB-33

Done ✓ Done

Description

Edit description

Subtasks

100% Done

Jira

Spaces / Capstone\_Project\_BMS

**CPB board** ...

Summary Timeline Backlog Active sprints

Search board Version Epic Type

Epics

- GPB-3 JIRA Workflow (3 subtasks) JIRA WORKFLOW
- GPB-2 GITHUB Workflow (5 subtasks) GITHUB WORKFLOW
- GPB-13 Jenkins CI/CD Pipeline (7 subtasks) JENKINS CI/CD PIPELINE
- GPB-22 Docker Deployment (3 subtasks) DONE
- GPB-27 Kubernetes Deployment (EKS) (4 subtasks) DONE
- GPB-33 Monitoring & Observability (5 subtasks) DONE

Complete CPB Sprint 1

Sprint cannot be completed as there are incomplete subtasks on the following:

- GPB-13

Cancel Complete sprint