

# Project: Deployment of Online Ticketing and Event Management Applications and Infrastructure

*By: Harshika*

*LinkedIn: [Harshika](#)*

*Portfolio: [harshika-devops.online](#)*

## PHASE 1 (Foundation)

- Launch 3 EC2 instances
- (Server 1 "Jenkins" → Build + Docker", Server 2 "SonarQube" → Code Analysis", Server 3 "Amazon EKS" → Deployment")
- Setup:
  - Jenkins
  - Docker
  - Trivy

## PHASE 2 (Pipeline)

- Jenkins pipeline (without Kubernetes)
- Sonarqube
- Build + Push Docker image
- Deploy container

## PHASE 3 (Kubernetes / EKS)

- Create cluster
- Deploy via Jenkinsfile2

## **PHASE 4 (Monitoring)**

- Prometheus
- Grafana

### **Phase 1:**

#### **STEP 1: Launch FIRST EC2 Instance (BMS-Server)**

This will be the main server (**Jenkins + Docker**)

##### **1. Go to EC2 Dashboard**

- Login to AWS
- Go to EC2
- Click “Launch Instance”

##### **2. Name the Instance**

Name: **Jenkins-bms-Server**

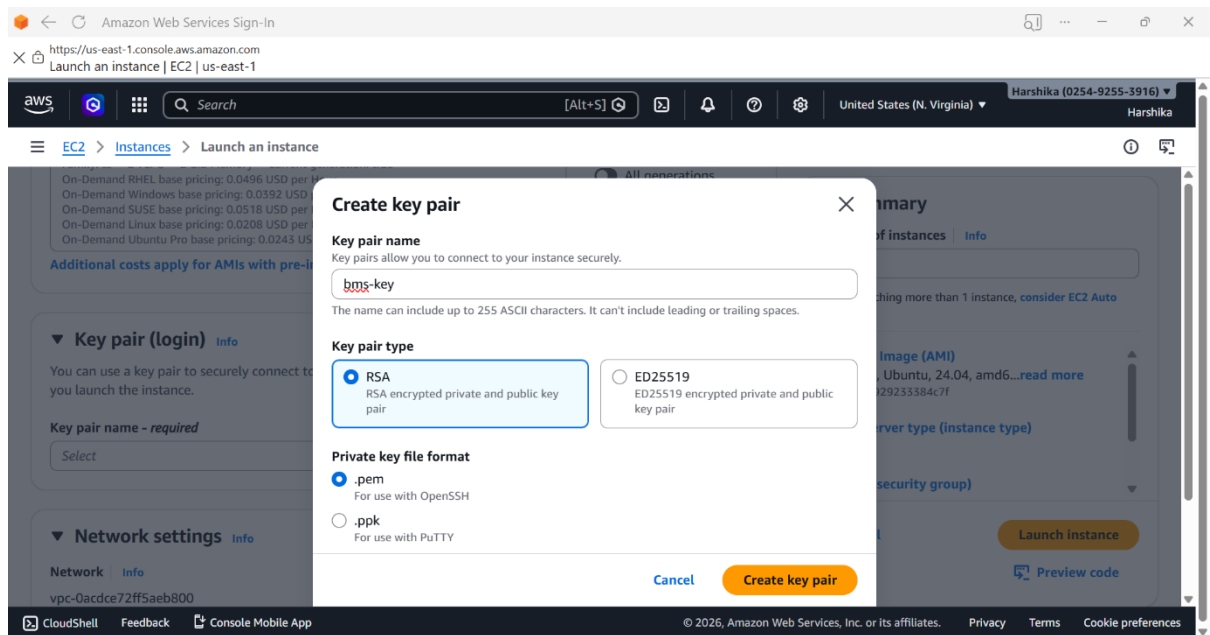
##### **3. Ubuntu Server 22.04 LTS (HVM), SSD Volume Type**

##### **4. Choose Instance Type**

t3.small

##### **5. Under “Key pair”**

- Click Create new key pair
- Name: **bms-key**
- Type: RSA
- Format: .pem
- Click Create



## 6. Network Settings (VERY IMPORTANT)

Add Inbound Rules: **bms-sg**

Click “Add security group rule” and add ALL below:

| Type       | Port | Source   |
|------------|------|----------|
| SSH        | 22   | Anywhere |
| HTTP       | 80   | Anywhere |
| HTTPS      | 443  | Anywhere |
| Custom TCP | 8080 | Anywhere |
| Custom TCP | 3000 | Anywhere |

This is IMPORTANT for:

- Jenkins (8080)
- App (3000)
- Other services

Amazon Web Services Sign-In

https://us-east-1.console.aws.amazon.com

SecurityGroup | EC2 | us-east-1

aws [Search] [Alt+S] United States (N. Virginia) Harshika (0254-9255-3916) Harshika

EC2 > Security Groups > sg-0f56798f0ccdf3856 - bms-sg

Inbound security group rules successfully modified on security group (sg-0f56798f0ccdf3856 | bms-sg) Details

Inbound rules (7)

Search

| <input type="checkbox"/> | Name | Security group rule ID | IP version | Type       | Protocol | Port range   | Source    |
|--------------------------|------|------------------------|------------|------------|----------|--------------|-----------|
| <input type="checkbox"/> | -    | sgr-07b76251d12503ccb  | IPv4       | HTTP       | TCP      | 80           | 0.0.0.0/0 |
| <input type="checkbox"/> | -    | sgr-0afe9d6409ace4298  | IPv4       | Custom TCP | TCP      | 8080         | 0.0.0.0/0 |
| <input type="checkbox"/> | -    | sgr-048c4d9599a2c3a4f  | IPv4       | Custom TCP | TCP      | 3000         | 0.0.0.0/0 |
| <input type="checkbox"/> | -    | sgr-05c27f243170f080c  | IPv4       | SSH        | TCP      | 22           | 0.0.0.0/0 |
| <input type="checkbox"/> | -    | sgr-0b2160e8b7fc742cb  | IPv4       | Custom TCP | TCP      | 9000         | 0.0.0.0/0 |
| <input type="checkbox"/> | -    | sgr-0c007d06cf89387d1  | IPv4       | Custom TCP | TCP      | 3000 - 10000 | 0.0.0.0/0 |
| <input type="checkbox"/> | -    | sgr-097bb7cb4e941638a  | IPv4       | HTTPS      | TCP      | 443          | 0.0.0.0/0 |

CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## 7. Configure Storage

Size: 20 GB

(Default is 8 → increase it)

Amazon Web Services Sign-In

https://us-east-1.console.aws.amazon.com

Launch an instance | EC2 | us-east-1

aws [Search] [Alt+S] United States Harshika (0254-9255-3916) Harshika

EC2 > Instances > Launch an instance

reach your instance.

☐ Create security group ☒ Select existing security group

Common security groups Info

Select security groups

bms-sg sg-0f56798f0ccdf3856 X  
VPC: vpc-0acdce72ff5aeb800

Compare security group rules

Security groups that you add or remove here will be added to or removed from all your network interfaces.

▼ Configure storage Info Advanced

1x 20 GiB gp3 Root volume, 3000 IOPS,  
Not encrypted

Add new volume

CloudShell Feedback Console Mobile App Privacy Terms Cookie preferences  
© 2026, Amazon Web Services, Inc. or its affiliates.

## 8. Launch Instance

Click:

Launch Instance

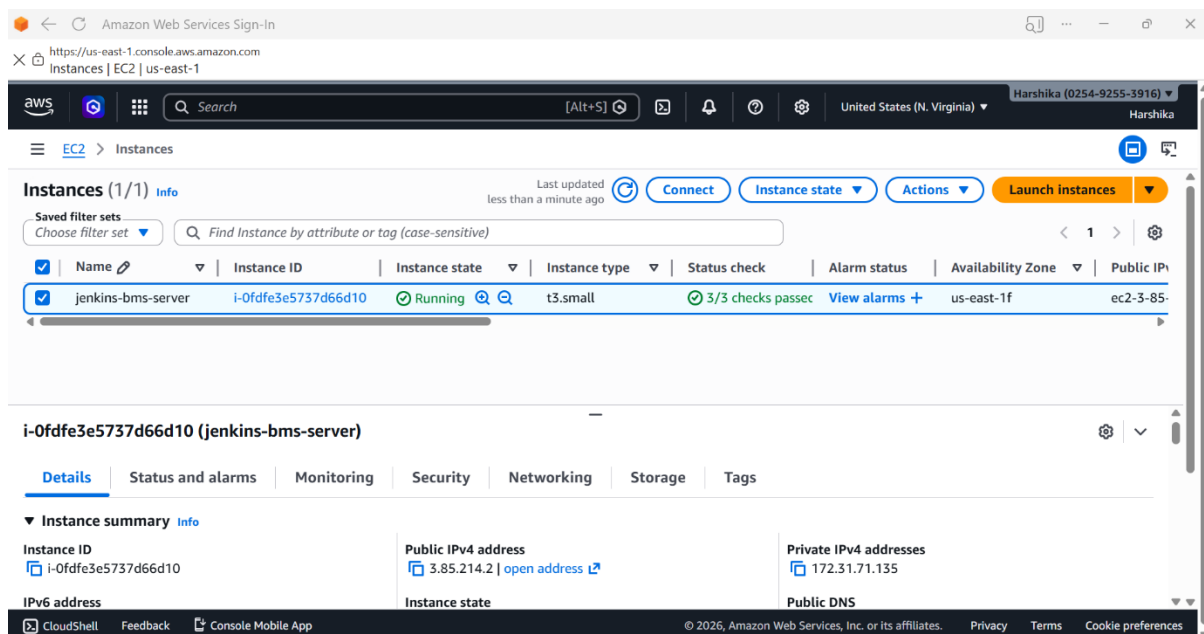
## 9. Wait & Verify

After launching:

- Go to Instances
- Wait until:

Instance state = Running

Status check = 3/3 checks passed



## STEP 2: Connect to EC2 using PowerShell

### 2.1 Connect to Server

Cd Downloads

Fix permission

Run this:

```
icac ls bms-key.pem /inheritance:r  
icac ls bms-key.pem /grant:r "%username%:R"
```

This avoids SSH permission error (very common)

```
ssh -i bms-key.pem ubuntu@3.85.214.2
```

First-time prompt

You will see:

Are you sure you want to continue connecting (yes/no)?

Type:

yes

## **STEP 2.2: Add SWAP MEMORY (VERY IMPORTANT)**

Run these commands

```
sudo falloccate -l 2G /swapfile
```

```
sudo chmod 600 /swapfile
```

```
sudo mkswap /swapfile
```

```
sudo swapon /swapfile
```

Make it permanent (IMPORTANT)

```
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Verify swap

```
free -h
```

You should see something like:

Swap: 2.0Gi

```
ubuntu@ip-172-31-71-135: ~  
-en [658 kB]  
Get:52 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]  
Get:53 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [28.8 kB]  
Get:54 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [7204 B]  
Get:55 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]  
Get:56 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [396 B]  
Fetched 41.6 MB in 9s (4649 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
48 packages can be upgraded. Run 'apt list --upgradable' to see them.  
ubuntu@ip-172-31-71-135:~$ sudo fallocate -l 2G /swapfile  
ubuntu@ip-172-31-71-135:~$ sudo chmod 600 /swapfile  
ubuntu@ip-172-31-71-135:~$ sudo mkswap /swapfile  
Setting up swspace version 1, size = 2 GiB (2147479552 bytes)  
no label, UUID=77717caf-50fb-487f-beae-f6e0ff558cf6  
ubuntu@ip-172-31-71-135:~$ sudo swapon /swapfile  
ubuntu@ip-172-31-71-135:~$ echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab  
/swapfile none swap sw 0 0  
ubuntu@ip-172-31-71-135:~$ free -h  
              total        used        free      shared  buff/cache   available  
Mem:           1.9Gi       387Mi       1.0Gi         2.7Mi        654Mi        1.5Gi  
Swap:          2.0Gi          0B         2.0Gi
```

## STEP 3: Install Java + Jenkins using the official link- [Linux](#)

### 3.1 Install Java 21

sudo apt update

sudo apt install fontconfig openjdk-21-jdk -y

### 3.2 Verify Java

java -version

```
ubuntu@ip-172-31-71-135: ~  
Replacing debian:emSign_ECC_Root_CA_-_G3.pem  
Replacing debian:emSign_Root_CA_-_C1.pem  
Replacing debian:emSign_Root_CA_-_G1.pem  
Replacing debian:vTrus_ECC_Root_CA.pem  
Replacing debian:vTrus_Root_CA.pem  
done.  
Setting up openjdk-21-jre:amd64 (21.0.10+7-1~24.04) ...  
Processing triggers for libgdk-pixbuf-2.0-0:amd64 (2.42.10+dfsg-3ubuntu3.3) ..  
.  
Scanning processes...  
Scanning linux images...  
  
Running kernel seems to be up-to-date.  
  
No services need to be restarted.  
  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
openjdk version "21.0.10" 2026-01-20  
OpenJDK Runtime Environment (build 21.0.10+7-Ubuntu-124.04)  
OpenJDK 64-Bit Server VM (build 21.0.10+7-Ubuntu-124.04, mixed mode, sharing)  
ubuntu@ip-172-31-71-135:~$ sudo wget -O /etc/apt/keyrings/jenkins-keyring.asc  
https://pkg.jenkins.io/debian-stable/jenkins.io-2026.key  
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]" \  
https://pkg.jenkins.io/debian-stable binary/ | sudo tee \  
/etc/apt/sources.list.d/jenkins.list > /dev/null  
sudo apt update  
sudo apt install jenkins
```

### 3.3 Create keyring directory

```
sudo mkdir -p /etc/apt/keyrings
```

### 3.4 Add Jenkins Key

```
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2026.key | sudo tee  
/etc/apt/keyrings/jenkins-keyring.asc > /dev/null
```

### 3.5 Add Jenkins Repo

```
echo "deb [signed-by=/etc/apt/keyrings/jenkins-keyring.asc]  
https://pkg.jenkins.io/debian-stable binary/" | sudo tee  
/etc/apt/sources.list.d/jenkins.list > /dev/null
```

### 3.6 Update packages

```
sudo apt update
```

### 3.7 Install Jenkins

```
sudo apt install jenkins -y
```

### 3.8 Start Jenkins

```
sudo systemctl start jenkins  
sudo systemctl enable jenkins
```

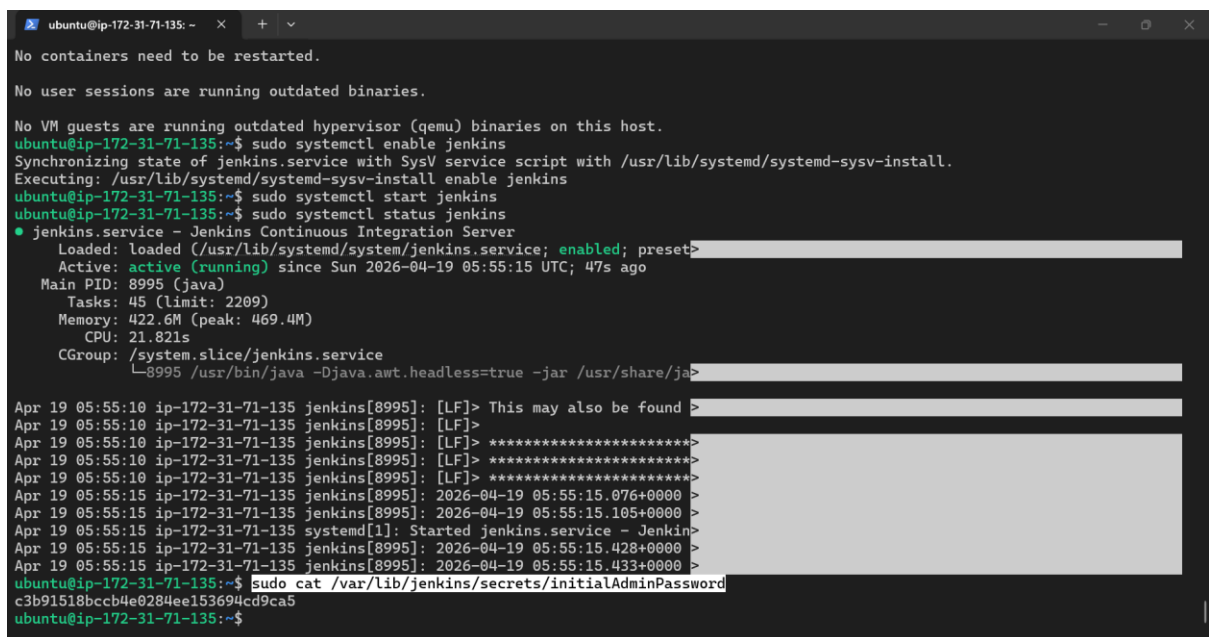
### 3.9 Check status

```
sudo systemctl status jenkins
```

Expect:

active (running)

Press q to exit



```
ubuntu@ip-172-31-71-135: ~  
No containers need to be restarted.  
  
No user sessions are running outdated binaries.  
  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-71-135:~$ sudo systemctl enable jenkins  
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins  
ubuntu@ip-172-31-71-135:~$ sudo systemctl start jenkins  
ubuntu@ip-172-31-71-135:~$ sudo systemctl status jenkins  
● jenkins.service - Jenkins Continuous Integration Server  
   Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset>  
   Active: active (running) since Sun 2026-04-19 05:55:15 UTC; 47s ago  
   Main PID: 8995 (java)  
     Tasks: 45 (Limit: 2209)  
    Memory: 422.6M (peak: 469.4M)  
       CPU: 21.821s  
   CGroup: /system.slice/jenkins.service  
           └─8995 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/ja  
  
Apr 19 05:55:10 ip-172-31-71-135 jenkins[8995]: [LF]> This may also be found >  
Apr 19 05:55:10 ip-172-31-71-135 jenkins[8995]: [LF]>  
Apr 19 05:55:10 ip-172-31-71-135 jenkins[8995]: [LF]> *****>  
Apr 19 05:55:10 ip-172-31-71-135 jenkins[8995]: [LF]> *****>  
Apr 19 05:55:10 ip-172-31-71-135 jenkins[8995]: [LF]> *****>  
Apr 19 05:55:15 ip-172-31-71-135 jenkins[8995]: 2026-04-19 05:55:15.076+0000 >  
Apr 19 05:55:15 ip-172-31-71-135 jenkins[8995]: 2026-04-19 05:55:15.105+0000 >  
Apr 19 05:55:15 ip-172-31-71-135 systemd[1]: Started jenkins.service - Jenkin>  
Apr 19 05:55:15 ip-172-31-71-135 jenkins[8995]: 2026-04-19 05:55:15.428+0000 >  
Apr 19 05:55:15 ip-172-31-71-135 jenkins[8995]: 2026-04-19 05:55:15.433+0000 >  
ubuntu@ip-172-31-71-135:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword  
c3b91518bccb4e0284ee153694cd9ca5  
ubuntu@ip-172-31-71-135:~$
```

### Unlocking Jenkins

Browse to <http://localhost:8080> (or the port you configured for Jenkins during installation) and wait until the **Unlock Jenkins** page appears.

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

### 3.10 Plugin Installation

On your Jenkins UI:

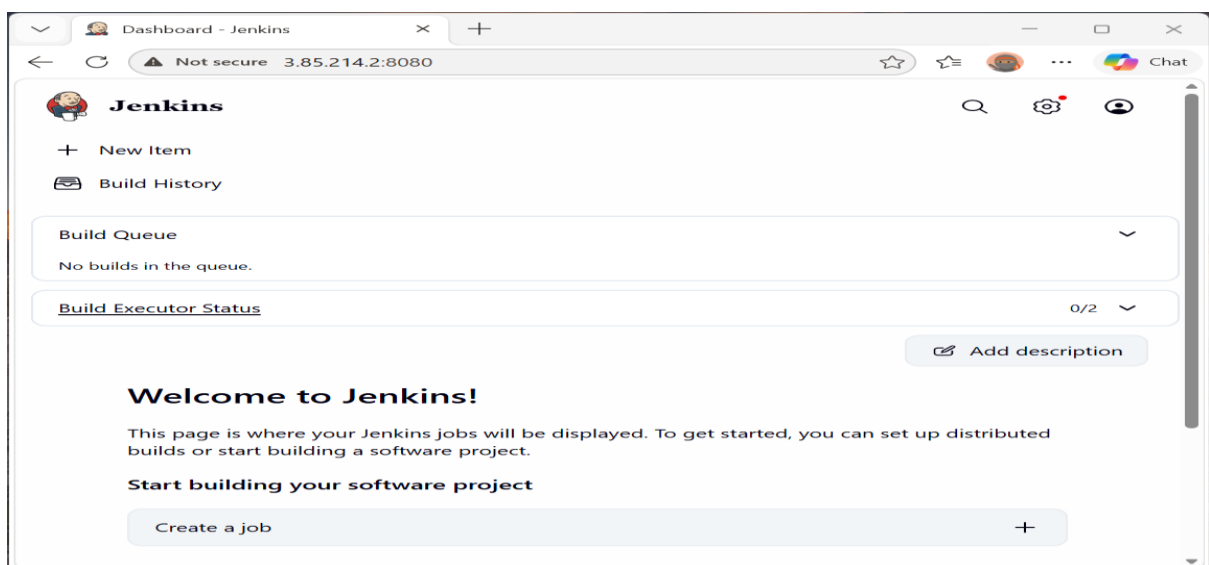
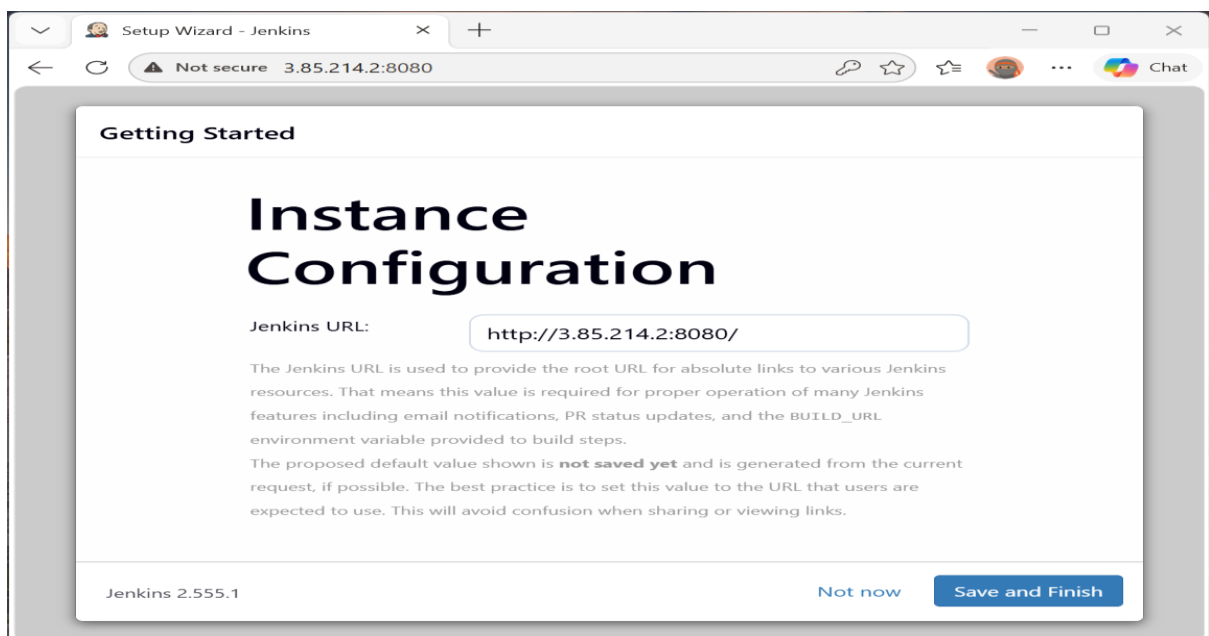
Click: **“Install suggested plugins”**

### 3.11 Create Admin User

Fill:

- Username → admin
- Password → (set something strong, remember it)
- Full name → anything
- Email → your email

Start Using Jenkins



**Now open Jenkins and install plugins:**

manage Jenkins-plugins-available plugins

Install below plugins;

Eclipse Temurin Installer,

SonarQube scanner,

NodeJS,

Docker,

Docker Commons,

Docker Pipeline,

Docker API,

docker-build-step,

OWASP dependency check,

Pipeline stage view,

Email Extension Template,

Kubernetes,

Kubernetes CLI,

Kubernetes Client API,

Kubernetes Credentials,

Config File Provider,

Prometheus metrics.

Prometheus metrics requires restart check the restart Jenkins box and re-login into Jenkins.

Update Center - Manage Jenkins - X

Not secure 3.85.214.2:8080/manage/pluginMana... Summarize

Manage Jenkins / Plugins

## Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

|                           |           |
|---------------------------|-----------|
| Config File Provider      | ✓ Success |
| NodeJS                    | ✓ Success |
| Authentication Tokens API | ✓ Success |
| Docker Commons            | ✓ Success |
| Docker Pipeline           | ✓ Success |
| Loading plugin extensions | ✓ Success |
| Pipeline Graph Analysis   | ✓ Success |
| Pipeline: REST API        | ✓ Success |
| Pipeline: Stage View      | ✓ Success |
| Groovy                    | ✓ Success |
| Loading plugin extensions | ✓ Success |

Update Center - Manage Jenkins - X Book My Show Security - My account - SonarQube - X

Not secure 3.85.214.2:8080/manage/pluginManager/updates/ Summarize

Jenkins / Manage Jenkins / Plugins

## Plugins

- Updates
- Available plugins
- Installed plugins
- Advanced settings
- Download progress

## Download progress

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

|                                      |           |
|--------------------------------------|-----------|
| SonarQube Scanner                    | ✓ Success |
| Cloud Statistics                     | ✓ Success |
| Apache HttpComponents Client 5.x API | ✓ Success |
| Commons Compress API                 | ✓ Success |
| Docker API                           | ✓ Success |
| Docker                               | ✓ Success |
| Kubernetes Client API                | ✓ Success |
| Kubernetes Credentials               | ✓ Success |
| Kubernetes                           | ✓ Success |
| Kubernetes CLI                       | ✓ Success |
| DataTables.net API                   | ✓ Success |
| OWASP Dependency-Check               | ✓ Success |
| Email Extension Template             | ✓ Success |

## **STEP 5: Install Git + Docker (on Jenkins server)**

We'll install:

- Git → for pulling code
- Docker → for building & running apps
- Give Jenkins permission to use Docker

### **5.1 Install Git**

`sudo apt install git -y`

Verify:

`git --version`

```
ubuntu@ip-172-31-71-135:~$ sudo apt install git -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
git is already the newest version (1:2.43.0-1ubuntu7.3).
git set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 48 not upgraded.
ubuntu@ip-172-31-71-135:~$ git --version
git version 2.43.0
ubuntu@ip-172-31-71-135:~$
```

### **5.2 Install Docker**

`sudo apt install apt-transport-https ca-certificates curl software-properties-c`

`url -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo tee  
/etc/apt/keyrings/docker.asc > /dev/null`

`echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.asc]  
https://download.docker.com/linux/ubuntu noble stable" | sudo tee  
/etc/apt/sources.list.d/docker.list > /dev/null`

`sudo apt update`

`sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin  
docker-compose-plugin -y`

```
ubuntu@ip-172-31-71-135: ~  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-71-135:~$ curl -fsSL https://download.docker.com/linux/ubuntu  
/gpg | sudo tee /etc/apt/keyrings/docker.asc > /dev/null  
ubuntu@ip-172-31-71-135:~$ echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/d  
ocker.asc] https://download.docker.com/linux/ubuntu noble stable" | sudo tee /  
etc/apt/sources.list.d/docker.list > /dev/null  
ubuntu@ip-172-31-71-135:~$ sudo apt update  
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble InRelease  
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease  
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease  
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease  
Get:5 https://download.docker.com/linux/ubuntu noble InRelease [48.5 kB]  
Hit:6 https://pkg.jenkins.io/debian-stable binary/ Release  
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease  
Get:8 https://download.docker.com/linux/ubuntu noble/stable amd64 Packages [50  
.8 kB]  
Fetched 99.3 kB in 1s (138 kB/s)  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
48 packages can be upgraded. Run 'apt list --upgradable' to see them.  
ubuntu@ip-172-31-71-135:~$ sudo apt install docker-ce docker-ce-cli containerd  
.io docker-buildx-plugin docker-compose-plugin -y  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
The following additional packages will be installed:  
  docker-ce-rootless-extras libslirp0 pigz slirp4netns  
Suggested packages:  
  cgroupfs-mount | cgroup-lite docker-model-plugin  
The following NEW packages will be installed:
```

### 5.3 Start Docker

sudo systemctl start docker

sudo systemctl enable docker

### 5.4 Verify Docker

docker --version

sudo docker run hello-world

Must print:

Hello from Docker!

```
ubuntu@ip-172-31-71-135: ~  
No user sessions are running outdated binaries.  
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-71-135:~$ sudo systemctl start docker  
sudo systemctl enable docker  
Synchronizing state of docker.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable docker  
ubuntu@ip-172-31-71-135:~$ docker --version  
Docker version 20.10.0, build 9d7ad9f  
ubuntu@ip-172-31-71-135:~$ sudo docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
4f55986f7dd0: Pull complete  
d5e71e642bf5: Download complete  
Digest: sha256:f9878146db2e05e794366b1bfe584a14ea6317f4027d10ef7dad65279026885  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
   (amd64)  
3. The Docker daemon created a new container from that image which runs the  
   executable that produces the output you are currently reading.  
4. The Docker daemon streamed that output to the Docker client, which sent it  
   to your terminal.  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
ubuntu@ip-172-31-71-135:~$
```

## 5.5 Give Jenkins access to Docker (VERY IMPORTANT)

sudo usermod -aG docker jenkins

sudo usermod -aG docker ubuntu

For more examples and ideas, visit:  
<https://docs.docker.com/get-started/>

```
ubuntu@ip-172-31-71-135:~$ sudo usermod -aG docker jenkins  
ubuntu@ip-172-31-71-135:~$ sudo usermod -aG docker ubuntu  
ubuntu@ip-172-31-71-135:~$ sudo systemctl restart jenkins  
ubuntu@ip-172-31-71-135:~$ newgrp docker  
ubuntu@ip-172-31-71-135:~$
```

## 5.6 Restart Jenkins

sudo systemctl restart jenkins

## 5.7 (IMPORTANT) Reload groups for current user

newgrp docker

## STEP 6 — Install Node.js + Trivy on Jenkins Server

### PART A — Install Node.js

The repo (Book-My-Show) is Node-based, so:

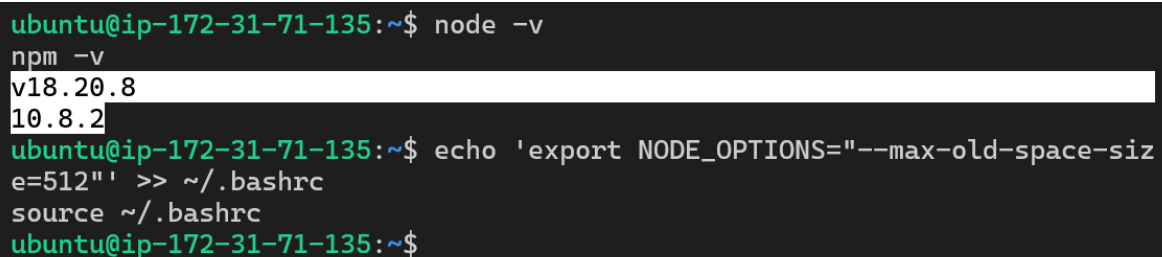
Run this (Node 18 LTS — stable & lightweight)

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -  
sudo apt install -y nodejs
```

Verify

```
node -v
```

```
npm -v
```



```
ubuntu@ip-172-31-71-135:~$ node -v  
v18.20.8  
ubuntu@ip-172-31-71-135:~$ npm -v  
10.8.2  
ubuntu@ip-172-31-71-135:~$ echo 'export NODE_OPTIONS="--max-old-space-size=512"' >> ~/.bashrc  
source ~/.bashrc  
ubuntu@ip-172-31-71-135:~$
```

### Set memory limit (VERY IMPORTANT)

```
echo 'export NODE_OPTIONS="--max-old-space-size=512"' >> ~/.bashrc  
source ~/.bashrc
```

### IMPORTANT (Memory optimisation)

Node builds can crash the t3.small if not controlled.

Set this:

```
export NODE_OPTIONS="--max-old-space-size=512"
```

// This limits RAM usage (VERY IMPORTANT)

### PART B — Install Trivy (for security scan)

Run:

```
sudo apt install wget apt-transport-https gnupg lsb-release -y
```

```
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | \  
gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
```

```
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg]
```

```
https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | \
sudo tee /etc/apt/sources.list.d/trivy.list
```

```
sudo apt update
```

```
sudo apt install trivy -y
```

Verify

```
trivy --version
```

```
ubuntu@ip-172-31-71-135: ~  
2026-04-19T06:16:22Z      INFO      [secret] If your scanning is slow, please  
try '--scanners vuln' to disable secret scanning  
2026-04-19T06:16:22Z      INFO      [secret] Please see https://trivy.dev/doc  
s/v0.70/guide/scanner/secret#recommendation for faster secret detection  
2026-04-19T06:16:23Z      INFO      Number of language-specific files      n  
um=0  
2026-04-19T06:16:23Z      WARN      [report] Supported files for scanner(s) n  
ot found.      scanners=[vuln]  
2026-04-19T06:16:23Z      INFO      [report] No issues detected with scanner(  
s).      scanners=[secret]  
  
Report Summary  
  


| Target | Type | Vulnerabilities | Secrets |
|--------|------|-----------------|---------|
| -      | -    | -               | -       |

  
Legend:  
- '-': Not scanned  
- '0': Clean (no security findings detected)  
  
ubuntu@ip-172-31-71-135:~$ trivy --version  
Version: 0.70.0  
Vulnerability DB:  
Version: 2  
UpdatedAt: 2026-04-16 18:52:29.28557025 +0000 UTC  
NextUpdate: 2026-04-17 18:52:29.285569899 +0000 UTC  
DownloadedAt: 2026-04-19 06:16:22.789185476 +0000 UTC  
ubuntu@ip-172-31-71-135:~$
```

## VERY IMPORTANT (for low RAM)

First time Trivy runs, it downloads DB → heavy (~100MB)

So run once manually:

```
trivy image hello-world
```

```
ubuntu@ip-172-31-71-135: ~$ trivy image hello-world
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
2026-04-19T06:16:12Z INFO [vuln] Need to update DB
2026-04-19T06:16:12Z INFO [vuln] Downloading vulnerability DB...
2026-04-19T06:16:12Z INFO [vuln] Downloading artifact... repo="mirror.gcr.io/aquasec/trivy-db:2"
90.40 MiB / 90.40 MiB [-----] 100.00% 9.94 MiB p/s 9.3s
2026-04-19T06:16:22Z INFO [vuln] Artifact successfully downloaded repo="mirror.gcr.io/aquasec/trivy-db:2"
2026-04-19T06:16:22Z INFO [vuln] Vulnerability scanning is enabled
2026-04-19T06:16:22Z INFO [secret] Secret scanning is enabled
2026-04-19T06:16:22Z INFO [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2026-04-19T06:16:22Z INFO [secret] Please see https://trivy.dev/docs/v0.70/guide/scanner/secret#recommendation for fas
ter secret detection
2026-04-19T06:16:23Z INFO Number of language-specific files num=0
2026-04-19T06:16:23Z WARN [report] Supported files for scanner(s) not found. scanners=[vuln]
2026-04-19T06:16:23Z INFO [report] No issues detected with scanner(s). scanners=[secret]

Report Summary



| Target | Type | Vulnerabilities | Secrets |
|--------|------|-----------------|---------|
| -      | -    | -               | -       |



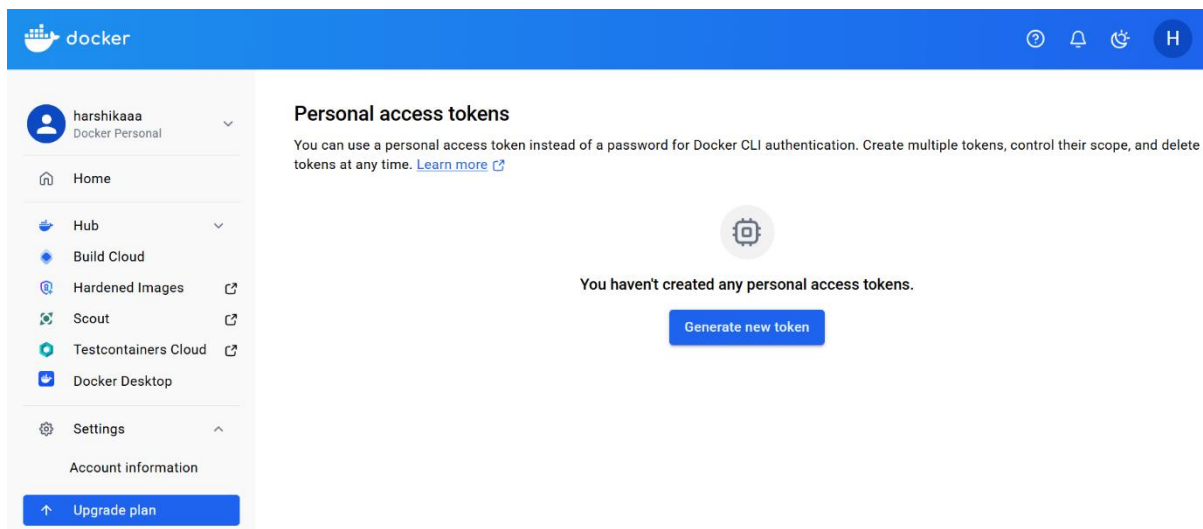
Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)

ubuntu@ip-172-31-71-135:~$
```

## Phase 2:

### Step 7- Create PAT in DockerHub:

Log in to the DockerHub Account



## Create Personal Access Token

harshikaaa  
Docker Personal

Home

Hub

Build Cloud

Hardened Images

Scout

Testcontainers Cloud

Docker Desktop

Settings

Account information

Email

Password

Upgrade plan

Personal access tokens / New access token

### Create access token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more](#)

Access token description

jenkins-token

Expiration date

30 days

Optional

Access permissions

Read & Write

Read & Write tokens allow you to push images to any repository managed by your account.

Cancel

Generate

## Generate

Access token description

jenkins-token

Expires on

May 17, 2026 at 23:59:59

Access permissions

Read & Write

To use the access token from your Docker CLI client:

1. Run

\$

Copy

2. At the prompt, paste the personal access token.

Copy

Back to access tokens

Generated!

### Personal access tokens

You can use a personal access token instead of a password for Docker CLI authentication. Create multiple tokens, control their scope, and delete tokens at any time. [Learn more](#)

Generate new token

| Description   | Scope        | Status | Source <span>i</span> | Created                  | Last used | Expiration date       |   |
|---------------|--------------|--------|-----------------------|--------------------------|-----------|-----------------------|---|
| jenkins-token | Read & Write | Active | Manual                | Apr 17, 2026 at 11:41:21 | Never     | May 17, 2026 at 23:59 | ⋮ |

Rows per page: 10 ▾ 1-1 of 1 < >

## Step 8- Add DockerHub Credentials in Jenkins

Go to:

Manage Jenkins → Credentials → Global → Add Credentials

Add:

- Username: your Docker username **(harshikaaa)**
- Password: NEW token
- ID: **docker-creds**

Credentials - Manage Jenkins - Jen X

Not secure 3.85.214.2:8080/manage/credenti... Summarize

### Add Username with password

Global (Jenkins, nodes, items, all child items, etc)

Username

harshikaaa

☐ Treat username as secret ?

Password

.....

ID ?

docker-creds

Description ?

Create

Credentials - Manage Jenkins - Jen X

Not secure 3.85.214.2:8080/manage/credenti... Summarize

Manage Jenkins / Credentials

## Credentials

+ Add Credentials

?

docker-creds harshikaaa/\*\*\*\*\*

System - Global

Stores scoped to Jenkins

System Domains: Global

## Step 9- Add Tool Config

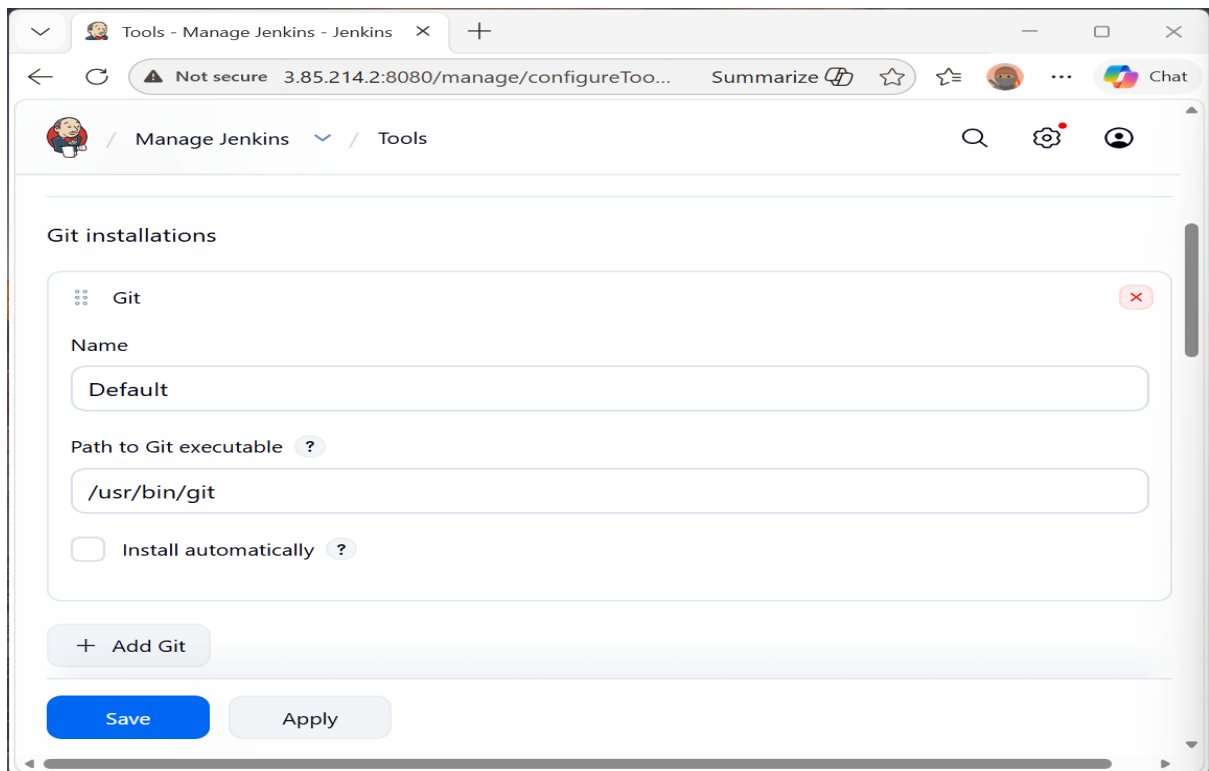
Go to manage Jenkins and tools:

### **Under Git**

Name: Default

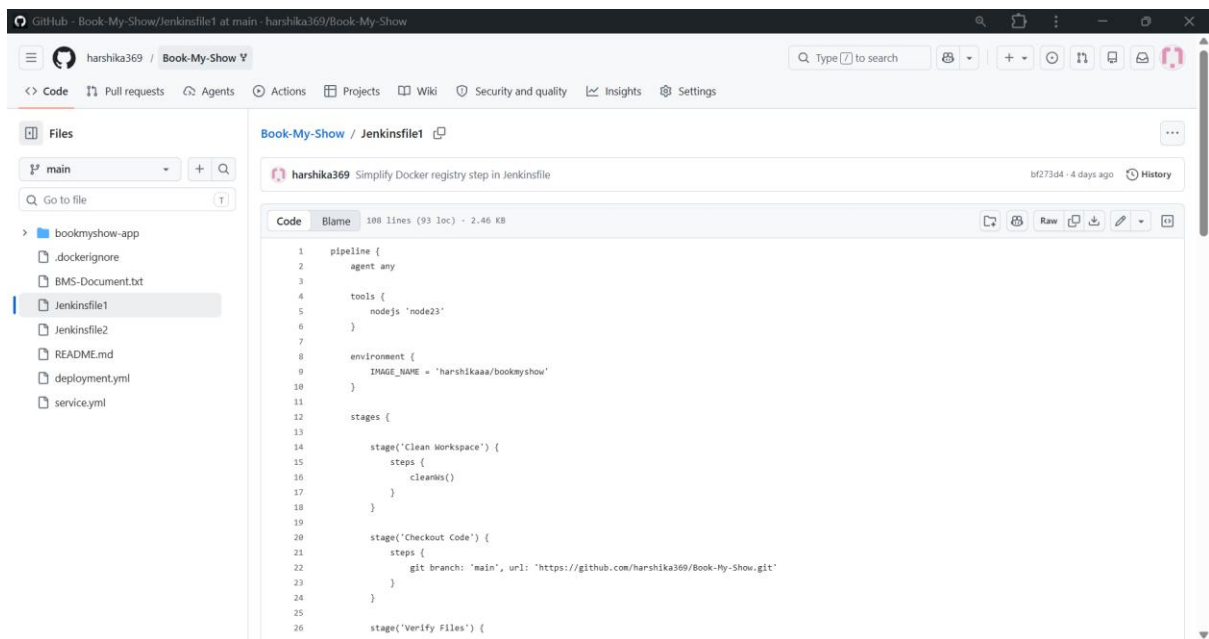
Path: /usr/bin/git

Click on apply and save



Now go to your GitHub repo that I have forked, and update the script of the file named **Jenkinsfile1** with this script and save commits.

<https://github.com/harshika369/Book-My-Show.git>



## file- JenkinsFile1

```
pipeline {
```

```
    agent any
```

```
    tools {
```

```
        nodejs 'node18'
```

```
    }
```

```
    environment {
```

```
        IMAGE_NAME = 'harshikaaa/bookmyshow'
```

```
    }
```

```
    stages {
```

```
stage('Clean Workspace') {  
    steps {  
        cleanWs()  
    }  
}
```

```
stage('Checkout Code') {  
    steps {  
        git branch: 'main', url: 'https://github.com/harshika369/Book-My-  
Show.git'  
    }  
}
```

```
stage('Verify Files') {  
    steps {  
        sh '''  
        echo "Root folder:"  
  
        ls -la  
  
        echo "App folder:"  
  
        ls -la bookmyshow-app  
        '''  
    }  
}
```

```
stage('Install Dependencies') {  
    steps {  
        dir('bookmyshow-app') {  
            sh '''  
            if [ -f package.json ]; then  
                rm -rf node_modules package-lock.json  
                npm install  
            else  
                echo "package.json not found!"  
                exit 1  
            fi  
            '''  
        }  
    }  
}
```

```
stage('Trivy FS Scan') {  
    steps {  
        sh 'trivy fs . > trivyfs.txt'  
    }  
}
```

```
stage('Build Docker Image') {  
    steps {  
        sh '''
```

```
        docker build -t $IMAGE_NAME:latest -f bookmyshow-app/Dockerfile
bookmyshow-app
```

```
        ""
    }
}
```

```
stage('Trivy Image Scan') {
    steps {
        sh 'trivy image $IMAGE_NAME:latest > trivyimage.txt'
    }
}
```

```
stage('Docker Push') {
    steps {
        script {
            withDockerRegistry(credentialsId: 'docker-creds') {
                sh ""
                docker push $IMAGE_NAME:latest
                ""
            }
        }
    }
}
```

```
stage('Deploy Container') {
```

```
steps {
  sh ""

  docker stop bms || true

  docker rm bms || true


  docker run -d \
    --name bms \
    -p 3000:3000 \
    --restart=always \
    $IMAGE_NAME:latest


  docker ps -a
  ""
}

}

}

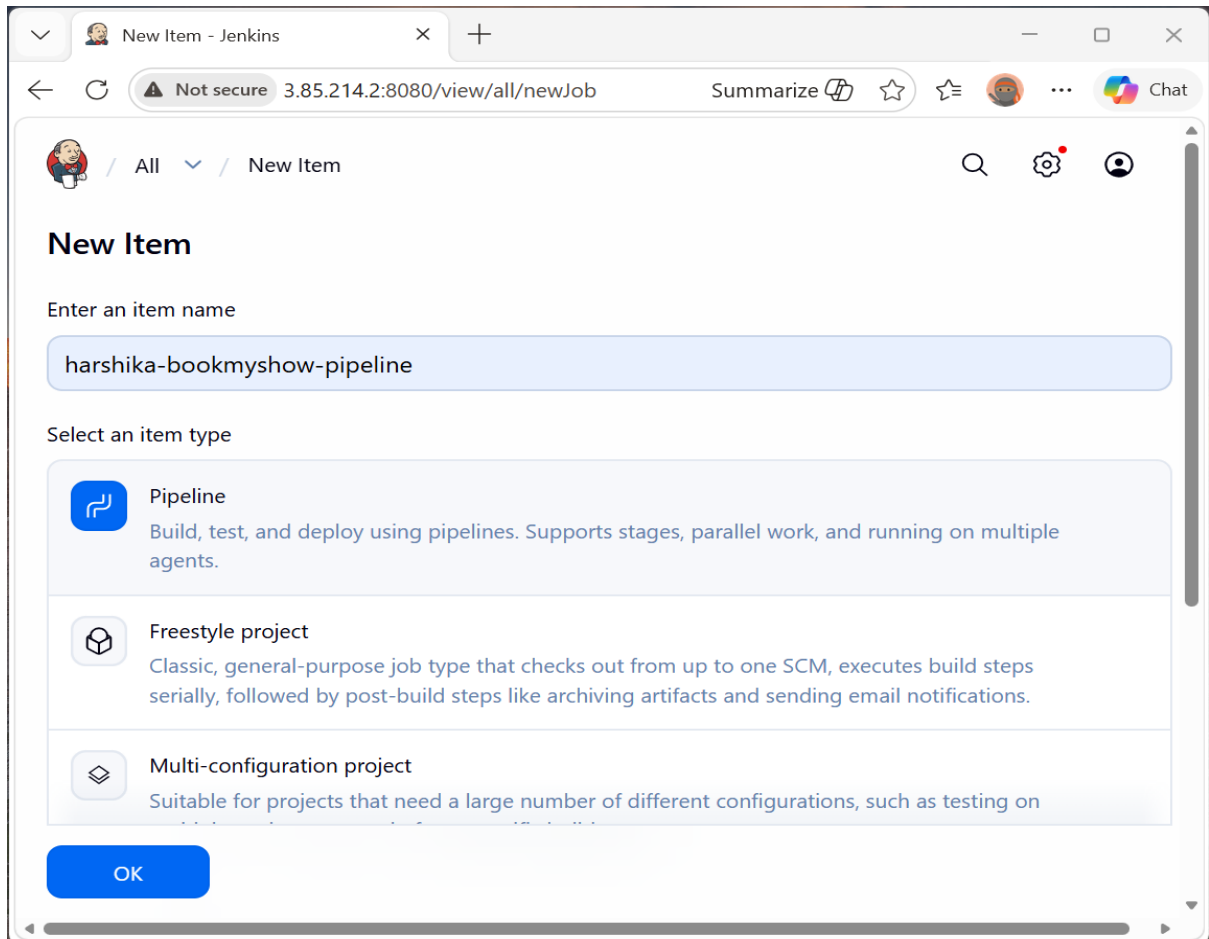
post {
  always {
    archiveArtifacts artifacts: 'trivyfs.txt,trivyimage.txt', fingerprint: true
  }
}

}
```

## **STEP 10: Create Jenkins Pipeline Job**

**Go to Jenkins Dashboard:**

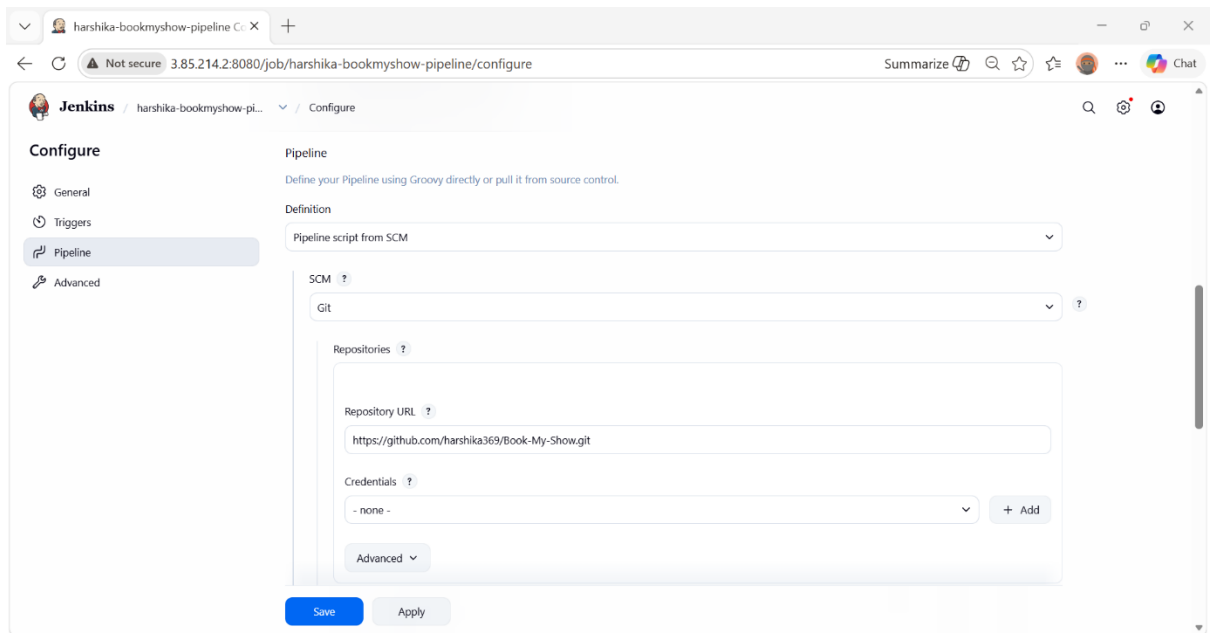
1. Click New Item
2. Name: **Harshika-bookmyshow-pipeline**
3. Select Pipeline



**Configure:**

Under the Pipeline section:

- Definition → Pipeline script from SCM
- SCM → Git
- Repo URL:
- <https://github.com/harshika369/Book-My-Show.git>



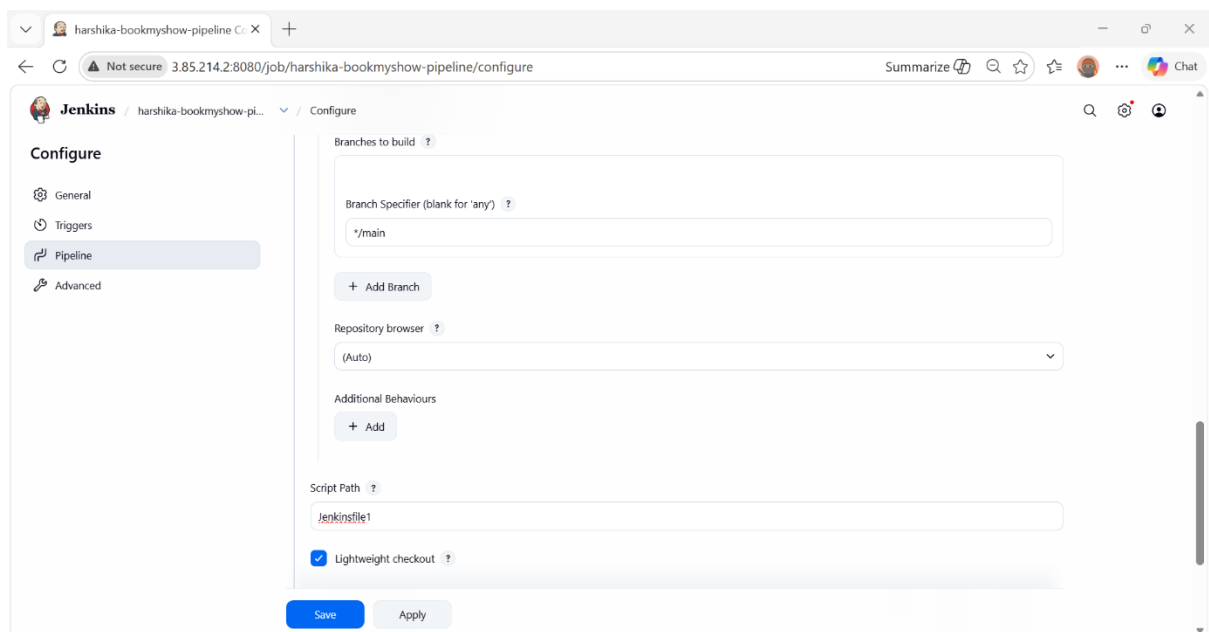
- Branch:

**\*/main**

- Script Path:

**Jenkinsfile1**

Save



## STEP 11: Run Pipeline

Click:

Build Now

Stages will run:

1. Clean Workspace ✓
2. Checkout Code ✓
3. Install Dependencies (Node works) ✓
4. Trivy Scan ✓
5. Docker Build ✓
6. Docker Push ✓
7. Deploy Container ✓

harshika-bookmyshow-pipeline

Stage View

| Stage                     | Duration |
|---------------------------|----------|
| Declarative: Checkout SCM | 630ms    |
| Declarative: Tool Install | 12s      |
| Clean Workspace           | 573ms    |
| Checkout Code             | 1s       |
| Verify Files              | 703ms    |
| Install Dependencies      | 2min 45s |
| Trivy FS Scan             | 20s      |
| Build Docker Image        | 8min 3s  |
| Trivy Image Scan          | 1min 30s |
| Docker Push               | 16s      |
| Deploy Container          | 1s       |
| Declarative: Post Actions | 476ms    |

Average stage times: 630ms, 12s, 573ms, 1s, 703ms, 2min 45s, 20s, 8min 3s, 1min 30s, 16s, 1s, 476ms

Total run time: 13min 21s

Builds

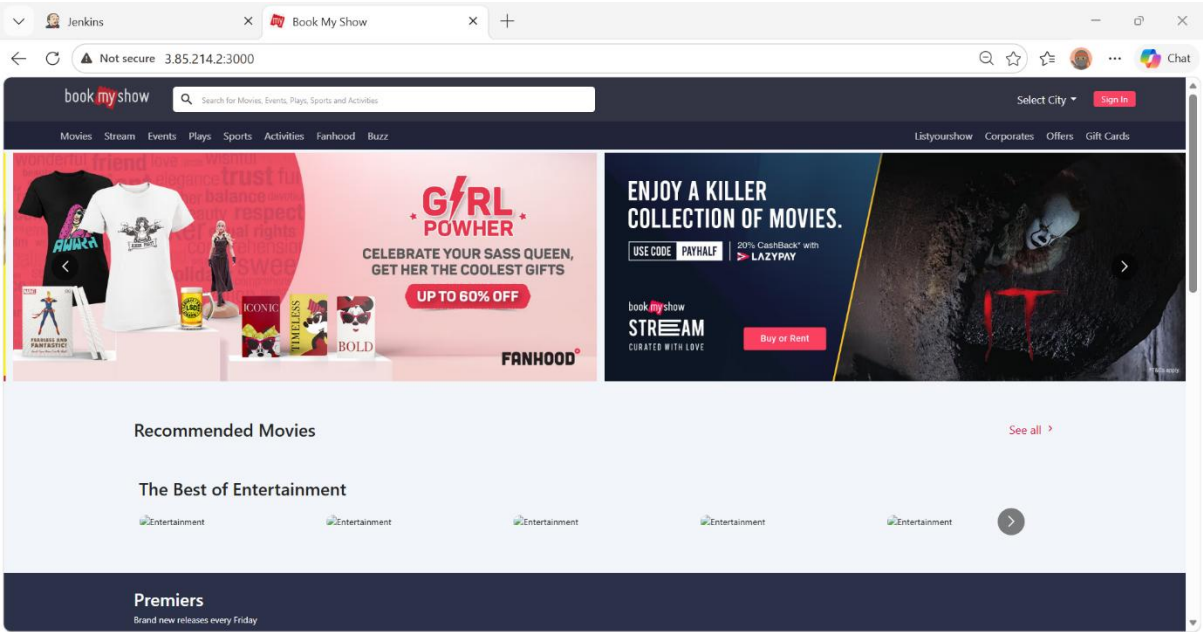
- Today
- #1 7:56 AM

REST API Jenkins 2.555.1



Open in browser:

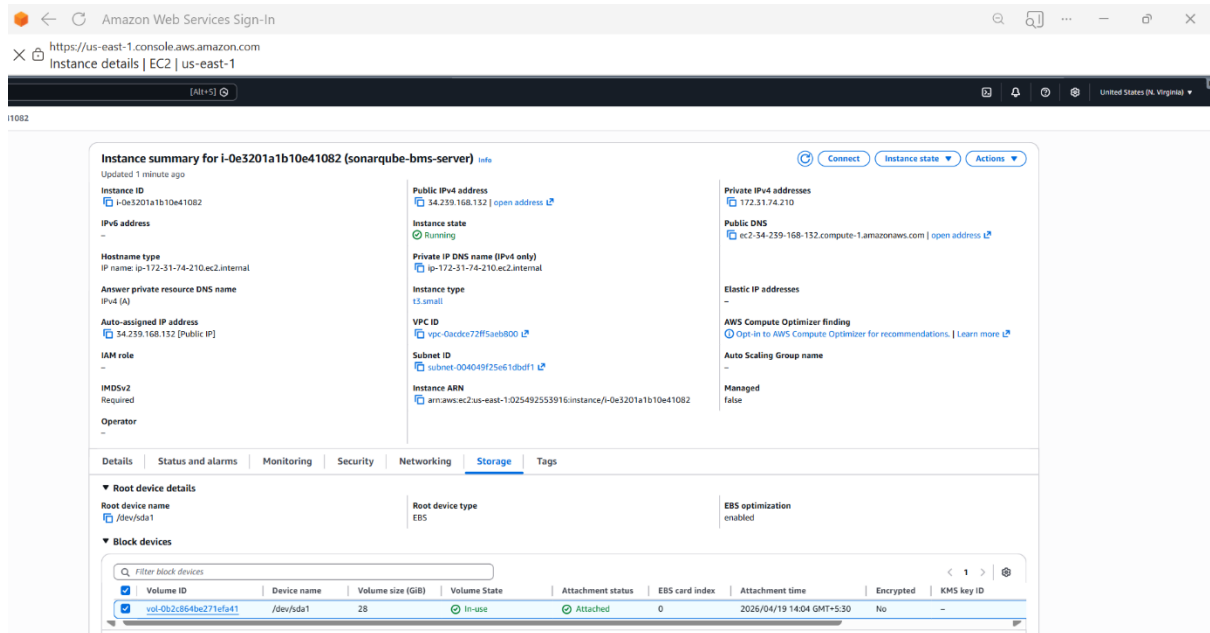
http://<EC2-PUBLIC-IP>:3000



## STEP 13: Set up SonarQube Server (Separate VM)

### 1. Create VM:

- Ubuntu 22.04 / 24.04
- Instance: t3.small
- Open port: **9000**



### 2. Connect to the Sonarqube server:

cd Downloads

icacs "your-key.pem" /inheritance:r

icacs "your-key.pem" /grant:r "\$(\$env:USERNAME):R"

ssh -i "your-key.pem" ubuntu@<SONAR\_PUBLIC\_IP>

### 3. Once Connected (Inside Server)

Now run these **exact commands for Sonar setup**

#### **Add SWAP (VERY IMPORTANT)**

sudo fallocate -l 2G /swapfile

sudo chmod 600 /swapfile

sudo mkswap /swapfile

```
sudo swapon /swapfile
```

```
free -h
```

**Make permanent:**

```
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

```
ubuntu@ip-172-31-74-210:~$ sudo sysctl -w vm.max_map_count=262144
sudo sysctl -w fs.file-max=65536
vm.max_map_count = 262144
fs.file-max = 65536
ubuntu@ip-172-31-74-210:~$ echo "vm.max_map_count=262144" | sudo tee -a /etc/s
ysctl.conf
echo "fs.file-max=65536" | sudo tee -a /etc/sysctl.conf
vm.max_map_count=262144
fs.file-max=65536
ubuntu@ip-172-31-74-210:~$ docker --version
Docker version 29.1.3, build 29.1.3-0ubuntu3~24.04.1
ubuntu@ip-172-31-74-210:~$ sudo chmod 666 /var/run/docker.sock
ubuntu@ip-172-31-74-210:~$ free -h
```

|       | total | used  | free  | shared | buff/cache | availab |
|-------|-------|-------|-------|--------|------------|---------|
| Mem:  | 1.9Gi | 427Mi | 667Mi | 2.8Mi  | 987Mi      | 1.4     |
| Swap: | 2.0Gi | 0B    | 2.0Gi |        |            |         |

```
ubuntu@ip-172-31-74-210:~$
```

## 4. Install Docker

```
sudo apt update
```

```
sudo apt install docker.io -y
```

```
sudo systemctl start docker
```

```
sudo systemctl enable docker
```

```
sudo chmod 666 /var/run/docker.sock
```

```
ubuntu@ip-172-31-74-210:~$ sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
sudo chmod 666 /var/run/docker.sock
```

Then check logs:

```
docker logs sonar
```

Make sure you see:

SonarQube is up

```
ubuntu@ip-172-31-74-210:~$ docker ps
```

| CONTAINER ID | IMAGE                   | COMMAND                  | CREATED       | STATUS       | PORTS                                       |
|--------------|-------------------------|--------------------------|---------------|--------------|---------------------------------------------|
| c16ef4f3df8a | sonarqube:lts-community | "/opt/sonarqube/dock..." | 9 seconds ago | Up 9 seconds | 0.0.0.0:9000->9000/tcp, [::]:9000->9000/tcp |

```
ubuntu@ip-172-31-74-210:~$
```

## 5. Kernel settings (MANDATORY for Sonar)

```
sudo sysctl -w vm.max_map_count=262144
```

```
sudo sysctl -w fs.file-max=65536
```

### Permanent:

```
echo "vm.max_map_count=262144" | sudo tee -a /etc/sysctl.conf
```

```
echo "fs.file-max=65536" | sudo tee -a /etc/sysctl.conf
```

```
ubuntu@ip-172-31-74-210:~$ sudo sysctl -w vm.max_map_count=262144
sudo sysctl -w fs.file-max=65536
vm.max_map_count = 262144
fs.file-max = 65536
ubuntu@ip-172-31-74-210:~$ echo "vm.max_map_count=262144" | sudo tee -a /etc/s
ysctl.conf
echo "fs.file-max=65536" | sudo tee -a /etc/sysctl.conf
vm.max_map_count=262144
fs.file-max=65536
ubuntu@ip-172-31-74-210:~$
```

```
docker run -d --name sonar -p 9000:9000 sonarqube:its-community
```

```
ubuntu@ip-172-31-74-210:~$ docker run -d --name sonar \
-p 9000:9000 \
sonarqube:community
Unable to find image 'sonarqube:community' locally
community: Pulling from library/sonarqube
c235bc109909: Pull complete
5f6278b6d85f: Pull complete
b40150c1c271: Pull complete
4f4fb700ef54: Pull complete
69b037921f81: Pull complete
16992be7d806: Pull complete
456eee9f1cc2: Pull complete
c25dbfa2b07e: Download complete
19f6168e631a: Download complete
Digest: sha256:177e3708ef0c205ad7f9564377fe974757ea588a1e2a2e25c4050bcf1958718
8
Status: Downloaded newer image for sonarqube:community
74f5f0c8e9673af7b3ee34dd767a41e33abcad55f2bb9260e3b3635213bb3e4b
ubuntu@ip-172-31-74-210:~$
```

## 6. Permissions for Docker

Run this (important):

```
sudo chmod 666 /var/run/docker.sock
```

```
ubuntu@ip-172-31-74-210:~$ sudo chmod 666 /var/run/docker.sock
ubuntu@ip-172-31-74-210:~$ free -h
              total        used        free      shared  buff/cache   availab
le
Mem:           1.9Gi        427Mi        667Mi        2.8Mi        987Mi        1.4
Gi
Swap:           2.0Gi          0B          2.0Gi
```

## 6. SWAP memory

Check: free -h

```
swapon --show
```

## 7. Access Sonar UI

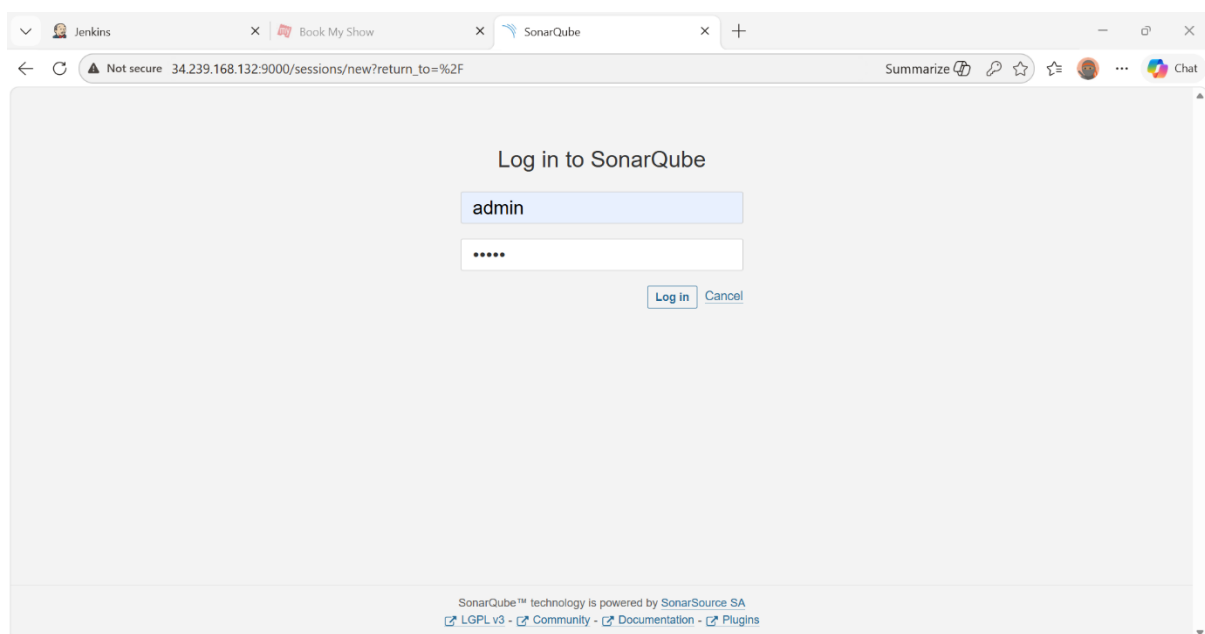
Open browser:

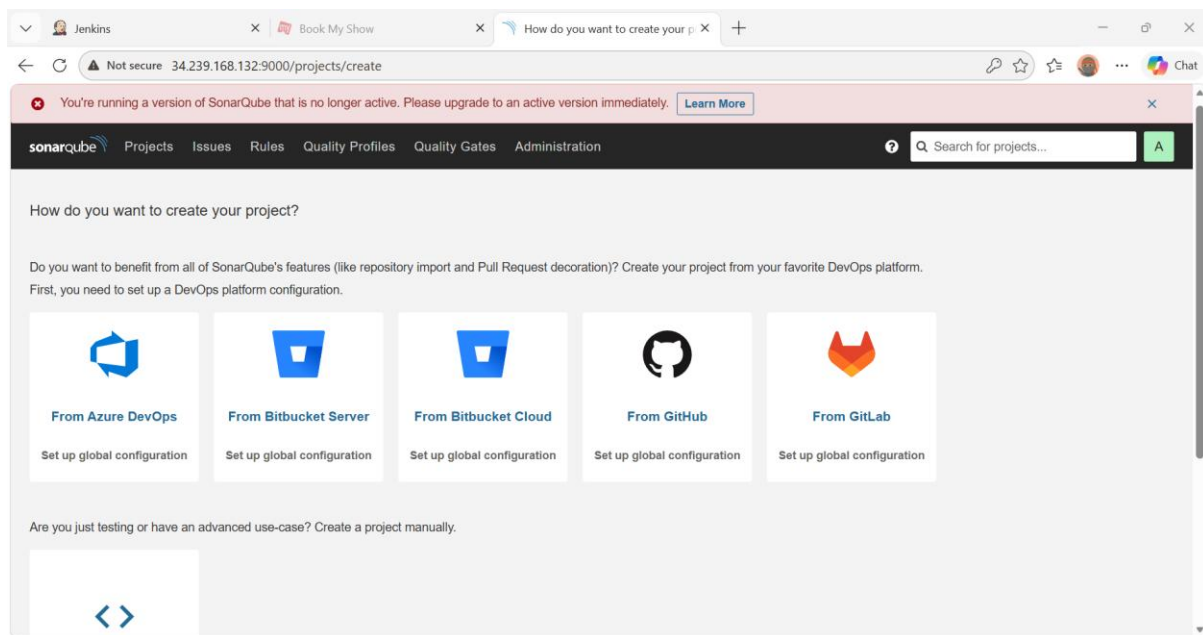
```
http://<SONAR_PUBLIC_IP>:9000
```

### Login

username: admin

password: admin





## 8. Connect SonarQube with Jenkins

### Generate Token in SonarQube

Open:

`http://<your-sonar-public-ip>:9000`

Login:

- username: admin
- password: (your new password)

## 9. Generate token

### Step 1:

Top right corner → click **your profile icon (A)**

### Step 2:

Click **“My Account”**

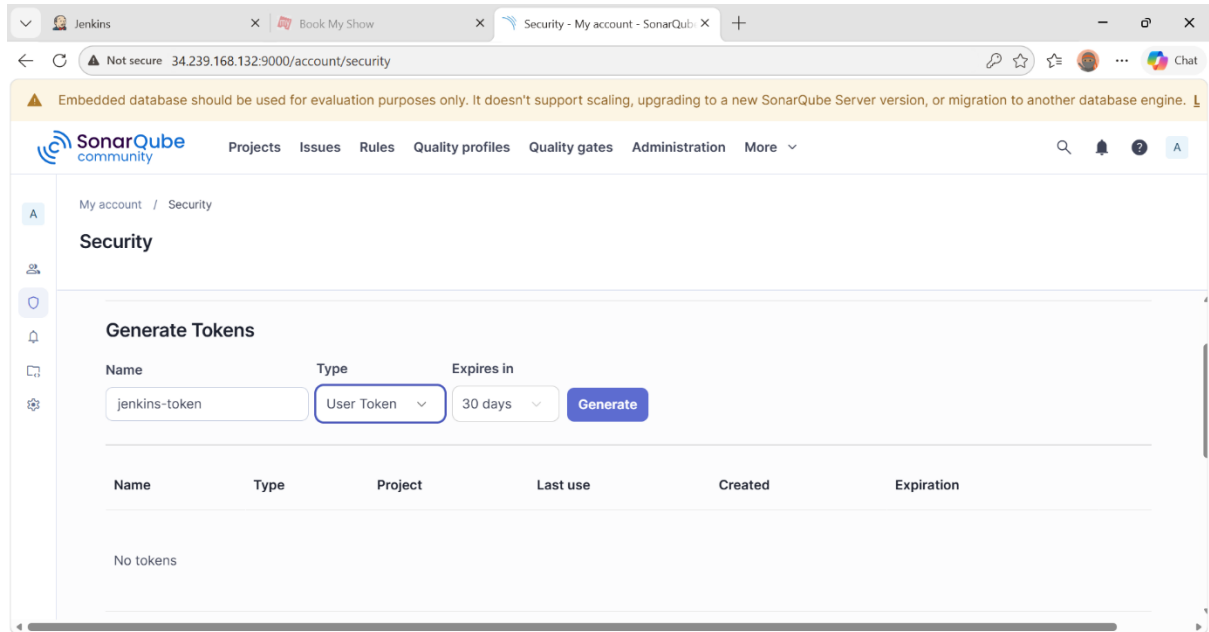
### Step 3:

Go to the **“Security”** tab

## Step 4:

Under **Tokens** section:

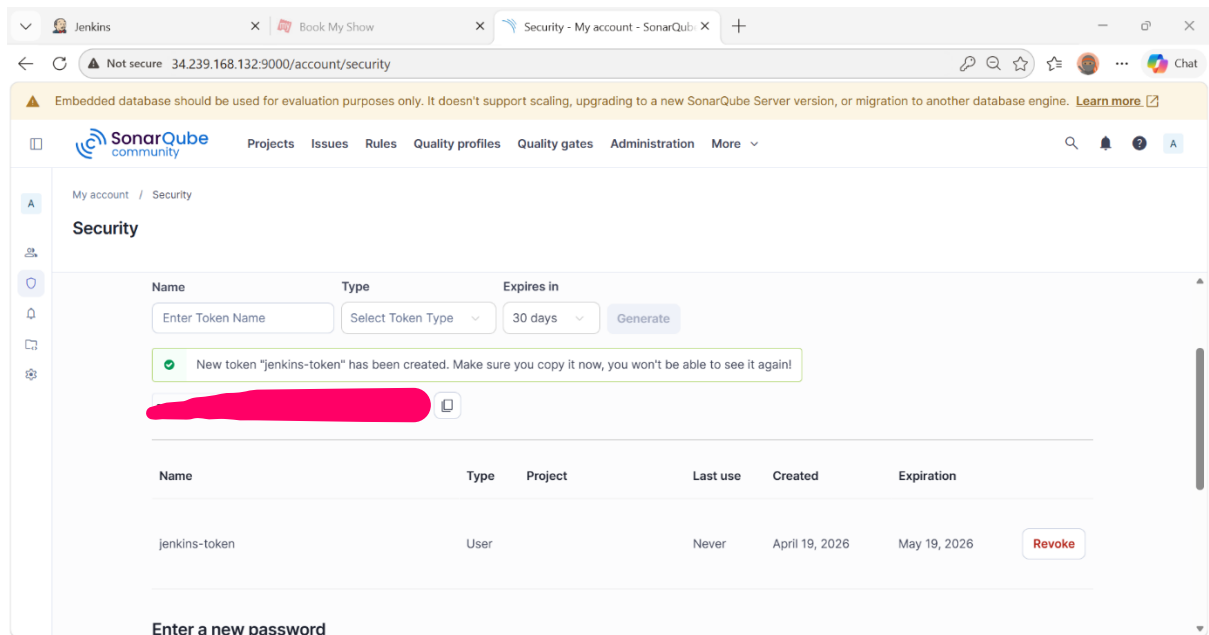
- Name: **jenkins-token**
- Click **Generate**



## Step 5:

IMPORTANT

- Copy the token immediately (you won't see it again)



## 10. Connect SonarQube with Jenkins

### Configure SonarQube in Jenkins

#### Manage Jenkins → System

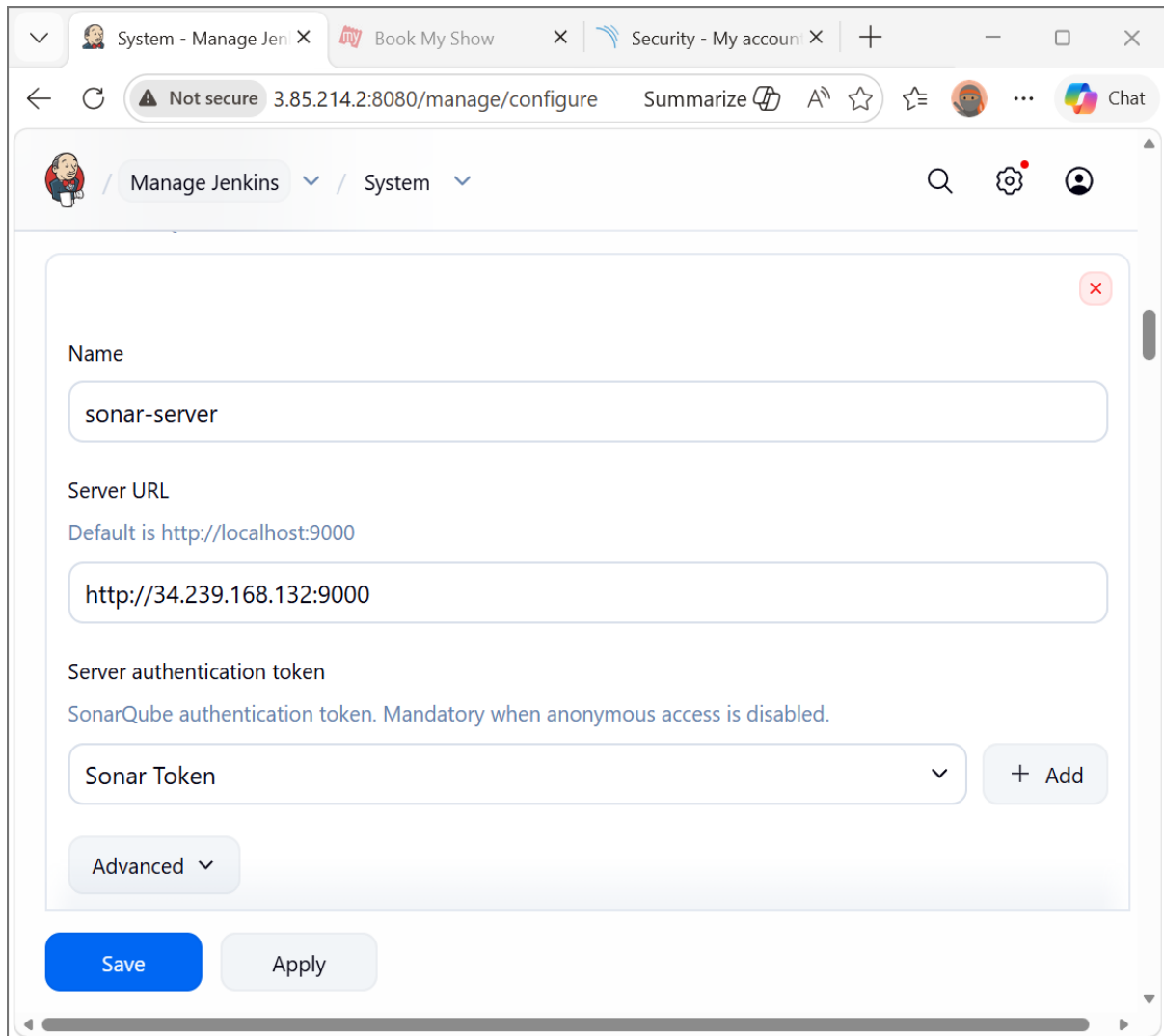
Scroll to: **SonarQube servers**

Click → **Add SonarQube**

Fill:

- Name: **sonar-server**
- Server URL:

`http://<sonar-private-ip>:9000`



The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `3.85.214.2:8080/manage/configure`. The page title is "Manage Jenkins" and the breadcrumb is "System". The "SonarQube servers" section is active, showing a configuration form for a new server named "sonar-server". The "Server URL" field is filled with `http://34.239.168.132:9000`. The "Server authentication token" field is labeled "Sonar Token" and has a dropdown arrow. A "+ Add" button is next to the token field. Below the token field is an "Advanced" dropdown menu. At the bottom of the form are "Save" and "Apply" buttons.

System - Manage Jen X Book My Show X Security - My account X +

← ↻ ⚠ Not secure 3.85.214.2:8080/manage/configure Summarize A ☆ ☆ Chat

Manage Jenkins / System

Name

sonar-server

Server URL

Default is `http://localhost:9000`

`http://34.239.168.132:9000`

Server authentication token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Sonar Token ▼ + Add

Advanced ▼

Save Apply

## Add Token:

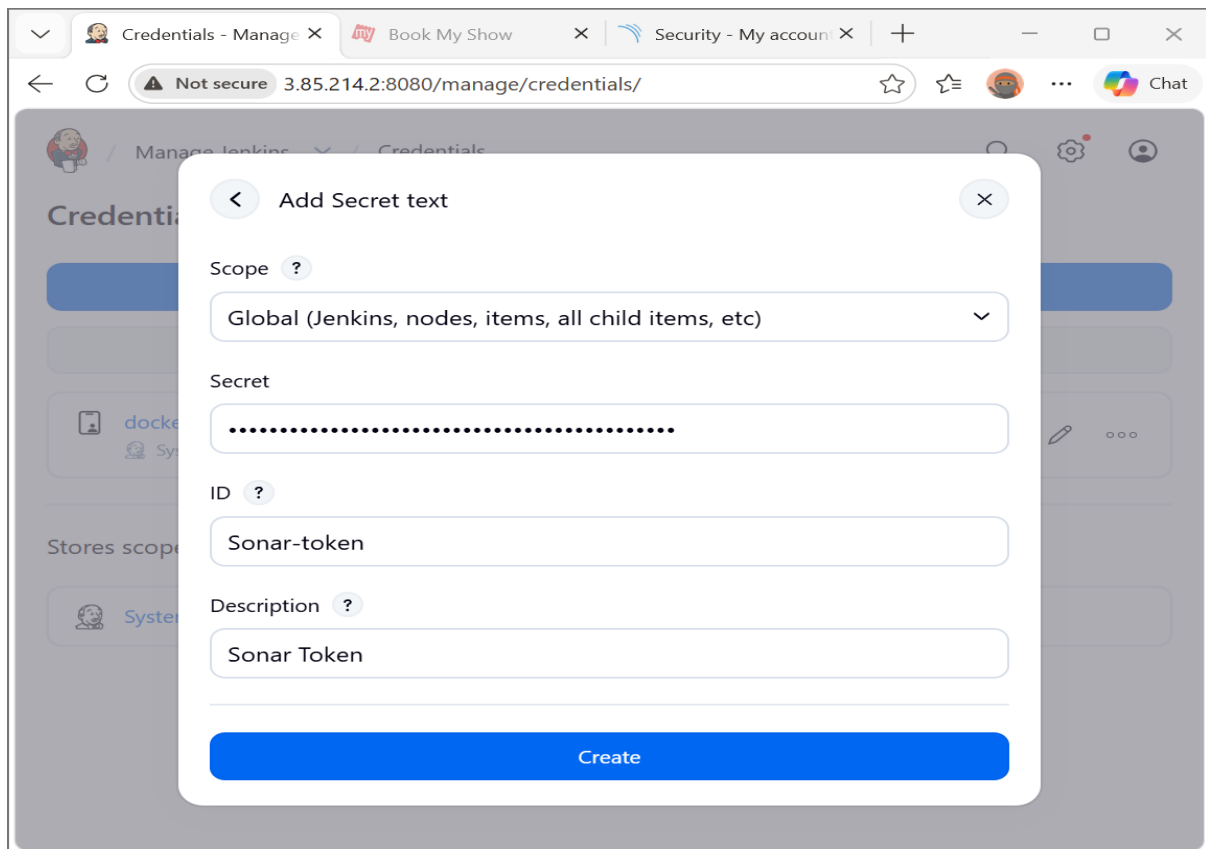
Click:

- **Add → Jenkins**

Fill:

- Kind → Secret Text
- Secret → *(paste your token)*
- ID → Sonar-token

Click **Add**

A screenshot of a web browser showing the Jenkins 'Add Secret text' dialog box. The browser's address bar shows '3.85.214.2:8080/manage/credentials/'. The dialog box has a title bar with a back arrow and 'Add Secret text'. It contains four fields: 'Scope' with a dropdown menu showing 'Global (Jenkins, nodes, items, all child items, etc)', 'Secret' with a masked input field, 'ID' with a text field containing 'Sonar-token', and 'Description' with a text field containing 'Sonar Token'. A blue 'Create' button is at the bottom.

Scope ?  
Global (Jenkins, nodes, items, all child items, etc) ▼

Secret  
.....

ID ?  
Sonar-token

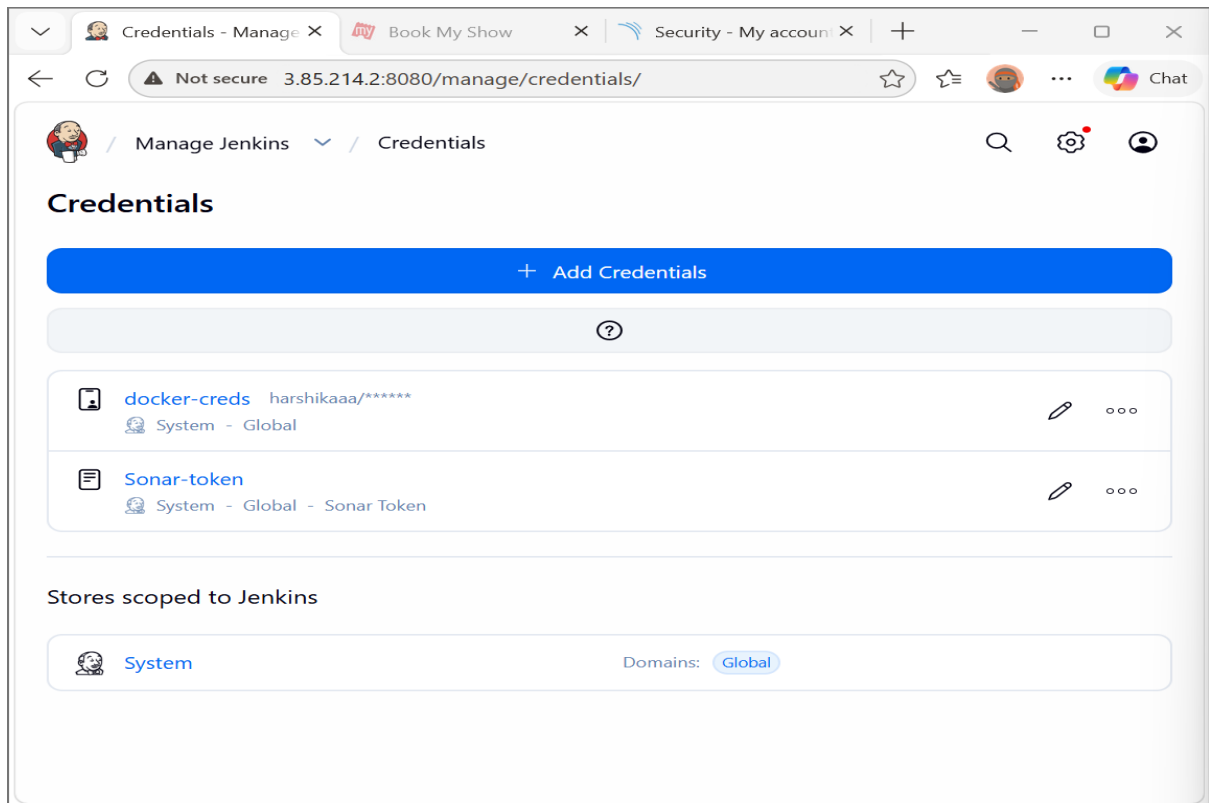
Description ?  
Sonar Token

Create

Now select:

- Credentials → Sonar-token

Click **Save**



## 11. Configure Sonar Scanner Tool

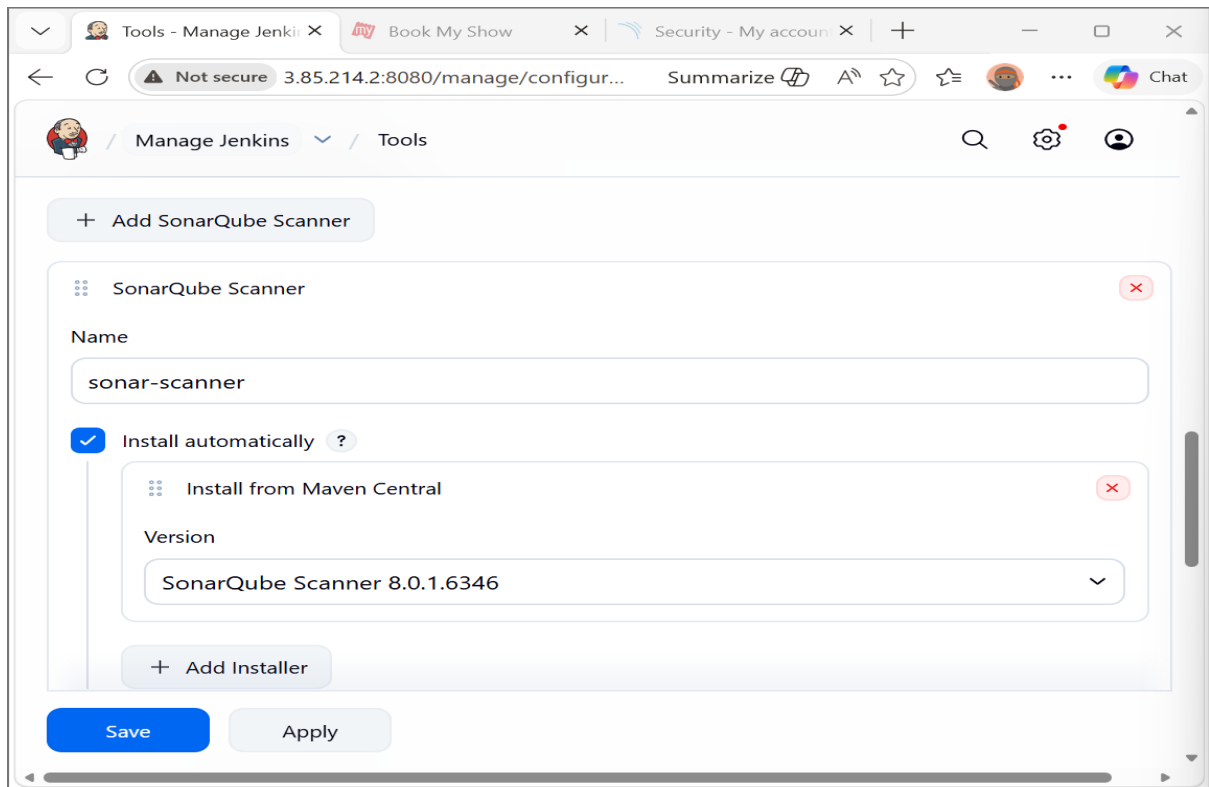
Manage Jenkins → Tools

SonarQube Scanner

Click **Add**

- Name → sonar-scanner
- Tick → Install automatically

Click **Save**



## Restart Jenkins (IMPORTANT)

Run on Jenkins server:

`sudo systemctl restart jenkins`

```
ubuntu@ip-172-31-71-135:~$ sudo systemctl restart jenkins
ubuntu@ip-172-31-71-135:~$
```

## 12. Other Tools Configuration in Jenkins:

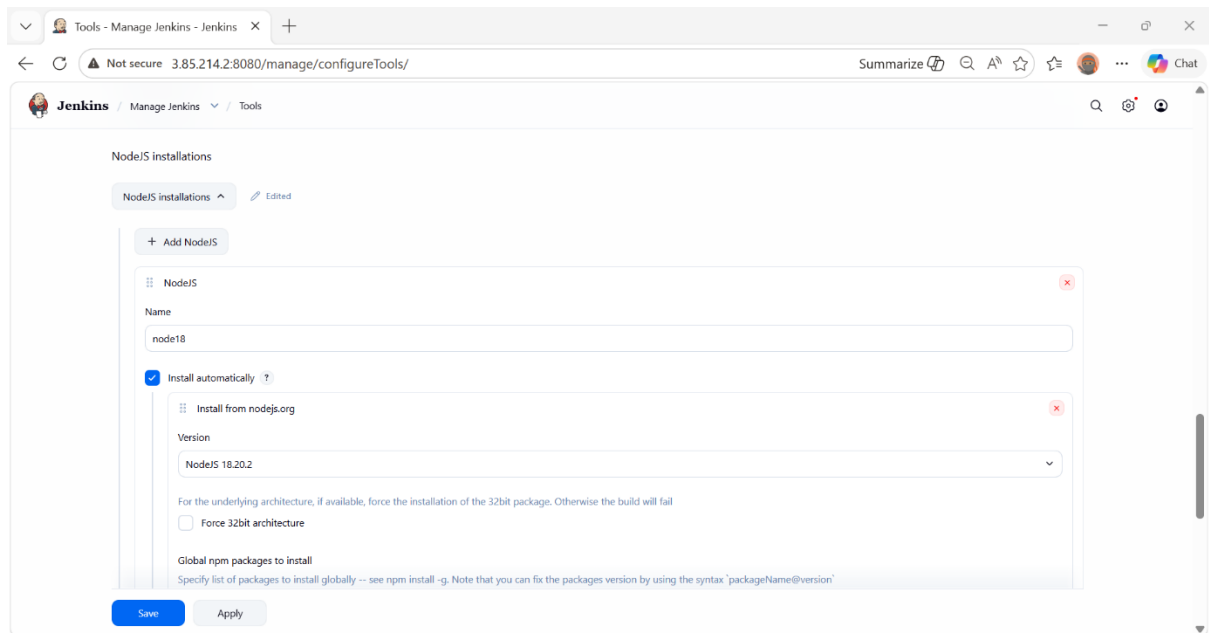
Go to manage Jenkins and tools

### Node.js installation

Name: **node18**

Select install automatically

Version?: NodeJS 23.7.0



## Dependency-check installations

Add Dependency-check

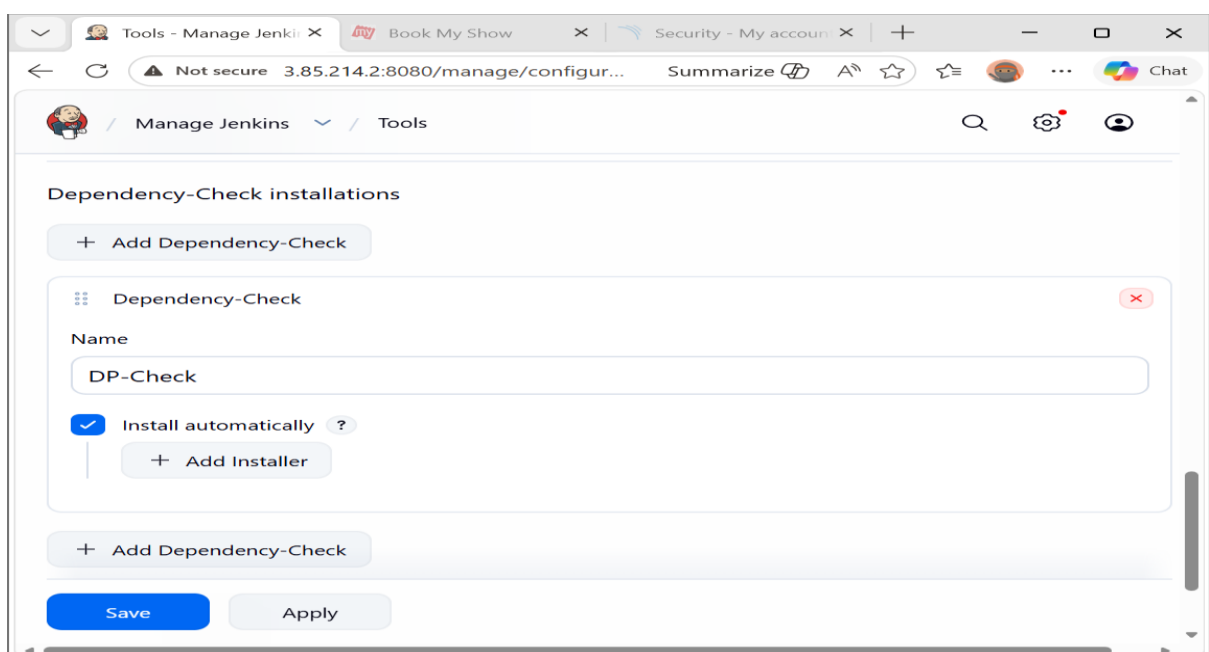
Name: DP-Check

Select install automatically

add installer

Install from github.com

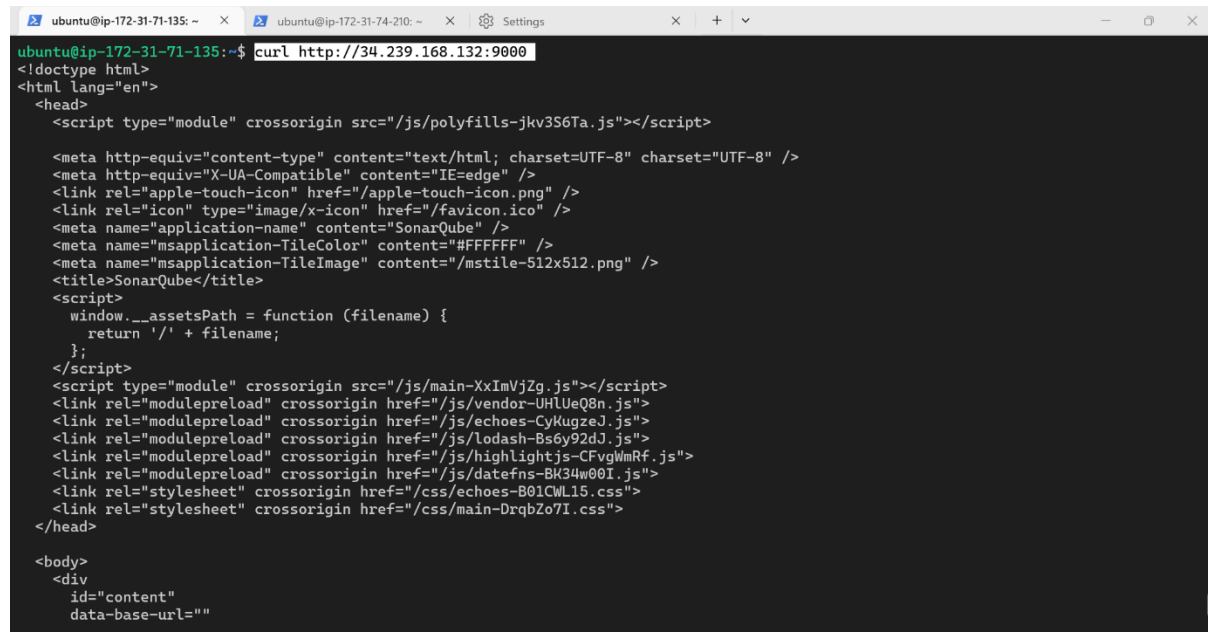
version?: dependency-check 12.0.2



### 13. To confirm the SonarQube setup in Jenkins

Replace with your actual SonarQube public IP:

curl [http://YOUR\\_SONAR\\_PUBLIC\\_IP:9000](http://YOUR_SONAR_PUBLIC_IP:9000)

A terminal window with a dark background. The prompt is 'ubuntu@ip-172-31-71-135: ~'. The command 'curl http://34.239.168.132:9000' has been executed. The output is an HTML document. The first line is '<!doctype html>'. The second line is '<html lang="en">'. The third line is '<head>'. The fourth line is '<script type="module" crossorigin src="/js/polyfills-jkv3S6Ta.js"></script>'. The fifth line is '<meta http-equiv="content-type" content="text/html; charset=UTF-8" charset="UTF-8" />'. The sixth line is '<meta http-equiv="X-UA-Compatible" content="IE=edge" />'. The seventh line is '<link rel="apple-touch-icon" href="/apple-touch-icon.png" />'. The eighth line is '<link rel="icon" type="image/x-icon" href="/favicon.ico" />'. The ninth line is '<meta name="application-name" content="SonarQube" />'. The tenth line is '<meta name="msapplication-TileColor" content="#FFFFFF" />'. The eleventh line is '<meta name="msapplication-TileImage" content="/mstile-512x512.png" />'. The twelfth line is '<title>SonarQube</title>'. The thirteenth line is '<script>'. The fourteenth line is 'window.\_\_assetsPath = function (filename) {'. The fifteenth line is 'return '/' + filename;'. The sixteenth line is '};'. The seventeenth line is '</script>'. The eighteenth line is '<script type="module" crossorigin src="/js/main-XxImVjZg.js"></script>'. The nineteenth line is '<link rel="modulepreload" crossorigin href="/js/vendor-UHLUeQ8n.js">'. The twentieth line is '<link rel="modulepreload" crossorigin href="/js/echoes-CyKugzeJ.js">'. The twenty-first line is '<link rel="modulepreload" crossorigin href="/js/lodash-Bs6y92dJ.js">'. The twenty-second line is '<link rel="modulepreload" crossorigin href="/js/highlightjs-CFvgWmRf.js">'. The twenty-third line is '<link rel="modulepreload" crossorigin href="/js/date-fns-BK34w00I.js">'. The twenty-fourth line is '<link rel="stylesheet" crossorigin href="/css/echoes-B01CWL15.css">'. The twenty-fifth line is '<link rel="stylesheet" crossorigin href="/css/main-DrqbZo7I.css">'. The twenty-sixth line is '</head>'. The twenty-seventh line is '<body>'. The twenty-eighth line is '<div'. The twenty-ninth line is 'id="content"'. The thirtieth line is 'data-base-url=""'.

```
ubuntu@ip-172-31-71-135: ~$ curl http://34.239.168.132:9000
<!doctype html>
<html lang="en">
<head>
  <script type="module" crossorigin src="/js/polyfills-jkv3S6Ta.js"></script>

  <meta http-equiv="content-type" content="text/html; charset=UTF-8" charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <link rel="apple-touch-icon" href="/apple-touch-icon.png" />
  <link rel="icon" type="image/x-icon" href="/favicon.ico" />
  <meta name="application-name" content="SonarQube" />
  <meta name="msapplication-TileColor" content="#FFFFFF" />
  <meta name="msapplication-TileImage" content="/mstile-512x512.png" />
  <title>SonarQube</title>
  <script>
    window.__assetsPath = function (filename) {
      return '/' + filename;
    };
  </script>
  <script type="module" crossorigin src="/js/main-XxImVjZg.js"></script>
  <link rel="modulepreload" crossorigin href="/js/vendor-UHLUeQ8n.js">
  <link rel="modulepreload" crossorigin href="/js/echoes-CyKugzeJ.js">
  <link rel="modulepreload" crossorigin href="/js/lodash-Bs6y92dJ.js">
  <link rel="modulepreload" crossorigin href="/js/highlightjs-CFvgWmRf.js">
  <link rel="modulepreload" crossorigin href="/js/date-fns-BK34w00I.js">
  <link rel="stylesheet" crossorigin href="/css/echoes-B01CWL15.css">
  <link rel="stylesheet" crossorigin href="/css/main-DrqbZo7I.css">
</head>
<body>
  <div
    id="content"
    data-base-url=""
```

### Expected Output

You should see HTML like:

```
<!doctype html>
<html ...
```

That means the Jenkins server **can reach SonarQube**

### 14. Generate Gmail App Password

#### Step 1: Open Google Account Security

In your browser, go to:

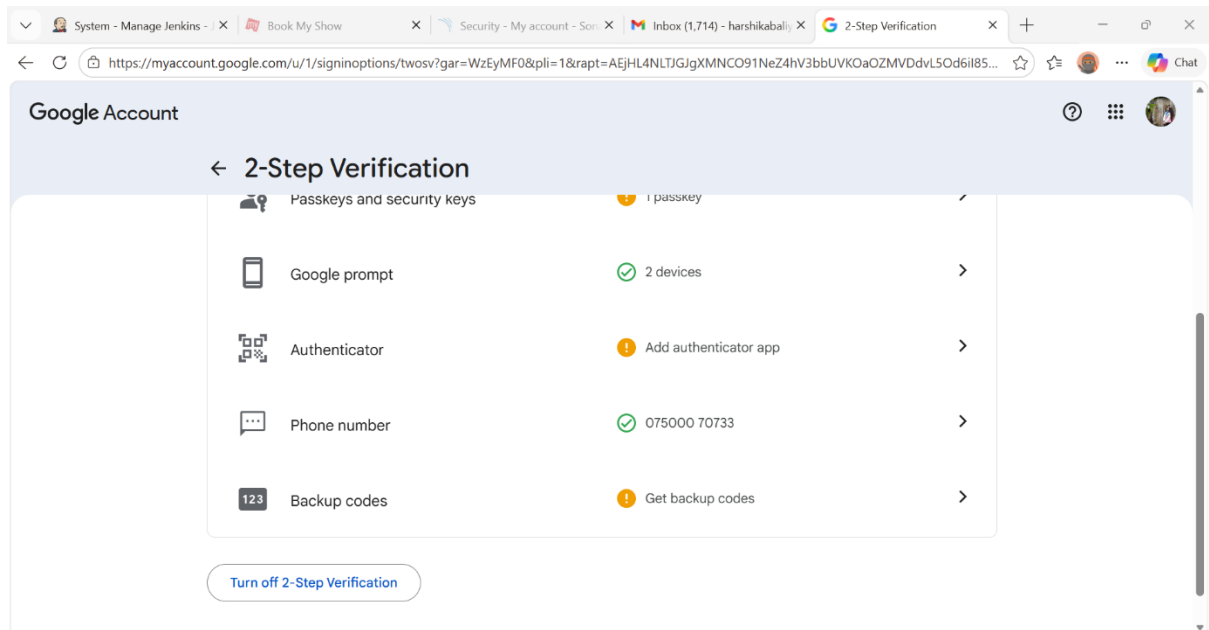
<https://myaccount.google.com/security>

#### Step 2: Enable 2-Step Verification (MANDATORY)

Scroll → find:

**“2-Step Verification”**

- Click it
- Click **Get Started**
- Enter your Gmail password
- Choose verification method (OTP / phone)
- Complete setup



### Step 3: Open App Passwords

Now on the same page:

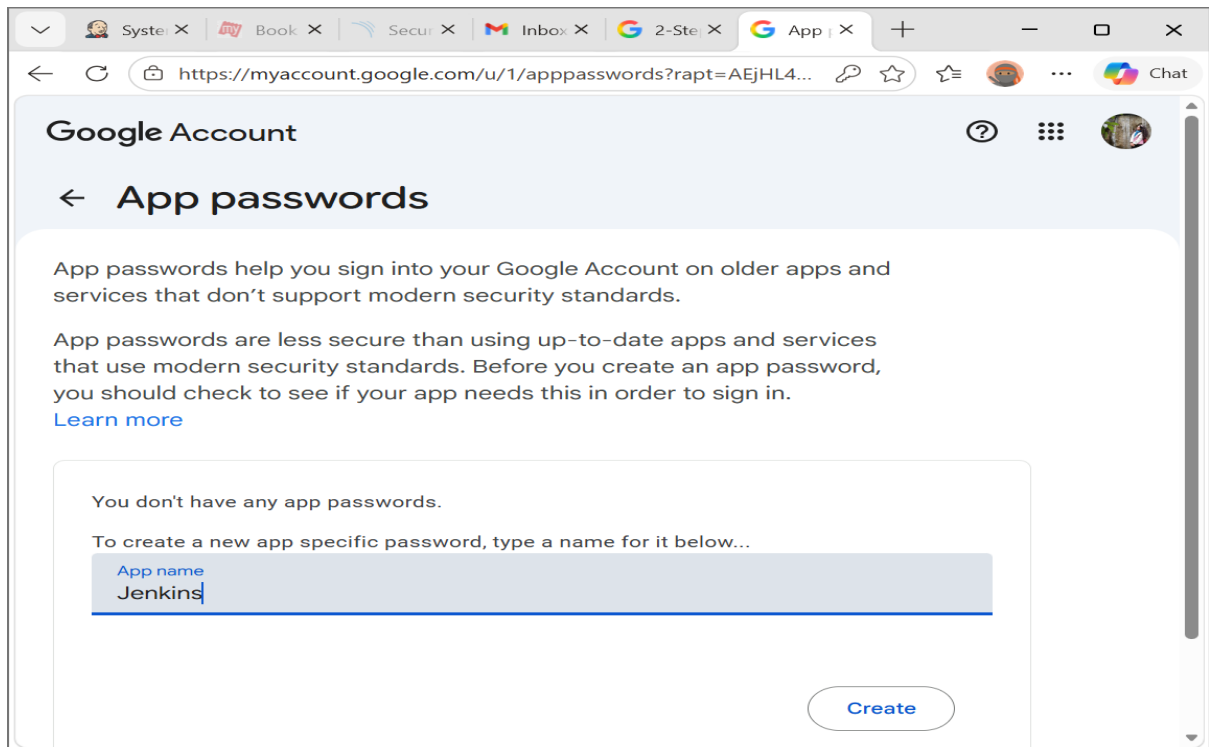
Search at top: **App Passwords**

### Step 4: Generate Password

Inside App Passwords:

- Select App → **Mail**
- Select Device → **Other**
- Enter name → Jenkins

Click **Generate**



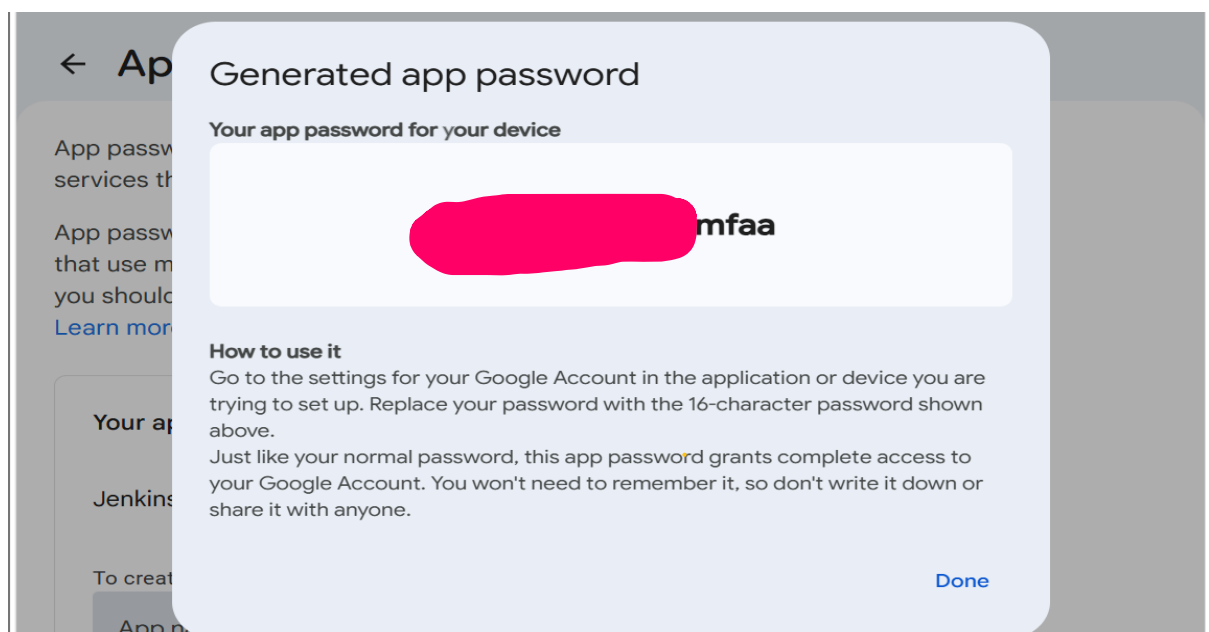
## Step 5: Copy Password

Copy it

REMOVE spaces → becomes:

abcdefghijklmnop

This is your **SMTP password**



## Step 6: Add Gmail SMTP Credentials In Jenkins

The screenshot shows the Jenkins 'Add Username with password' dialog box. The 'Scope' is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'Username' field contains 'harshika@gmail.com'. The 'Password' field is masked with dots. The 'ID' field contains 'gmail-smtp'. The 'Description' field contains 'Gmail SMTP'. The 'Create' button is at the bottom.

System - Manage Jenkins | Book My Show | Security - My account | Inbox (1,715) - harshika | 2-Step Verification | App passwords | + | - | Chat

Not secure 3.85.214.2:8080/manage/configure

Jenkins / Manage Jenkins / System

☐ Disable performance enhancements ?  
☐ Preserve second fetch during checkout ?  
☐ Add git tag action to jobs ?

Groovy  
☐ Allow token macro processing ?

Shell  
Shell executable ?

Extended E-mail Notification

Save Apply

Add Username with password

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)

Username  
harshika@gmail.com

☐ Treat username as secret ?

Password  
\*\*\*\*\*

ID ?  
gmail-smtp

Description ?  
Gmail SMTP

Create

The screenshot shows the Jenkins 'Credentials' page. At the top, there is a blue button labeled '+ Add Credentials'. Below this, there is a list of credentials. The first credential is 'docker-creds' with username 'harshika/\*\*\*\*\*' and scope 'System - Global'. The second credential is 'Sonar-token' with scope 'System - Global - Sonar Token'. The third credential is 'gmail-smtp' with username 'harshika@gmail.com/\*\*\*\*\*' and scope 'System - Global - Gmail SMTP'. Below the list, there is a section titled 'Stores scoped to Jenkins' with a single entry 'System' and domains 'Global'.

Cred | Book | Secur | Test e | 2-Ste | App | + | - | Chat

Not secure 3.85.214.2:8080/manage/credentials/ Summarize

Manage Jenkins / Credentials

### Credentials

+ Add Credentials

?

|              |                          |                               |   |   |
|--------------|--------------------------|-------------------------------|---|---|
| docker-creds | harshika/*****           | System - Global               | ✎ | ⋮ |
| Sonar-token  |                          | System - Global - Sonar Token | ✎ | ⋮ |
| gmail-smtp   | harshika@gmail.com/***** | System - Global - Gmail SMTP  | ✎ | ⋮ |

### Stores scoped to Jenkins

|        |                 |
|--------|-----------------|
| System | Domains: Global |
|--------|-----------------|

## 15. Configure SMTP in Jenkins

### Step 6: Go to Jenkins Config

Open Jenkins:

**Manage Jenkins → Configure System**

### Step 7: Find “Extended E-mail Notification”

Scroll down until you see:

#### Extended E-mail Notification

### Step 8: Fill SMTP Details

Enter EXACTLY this:

SMTP Server: smtp.gmail.com

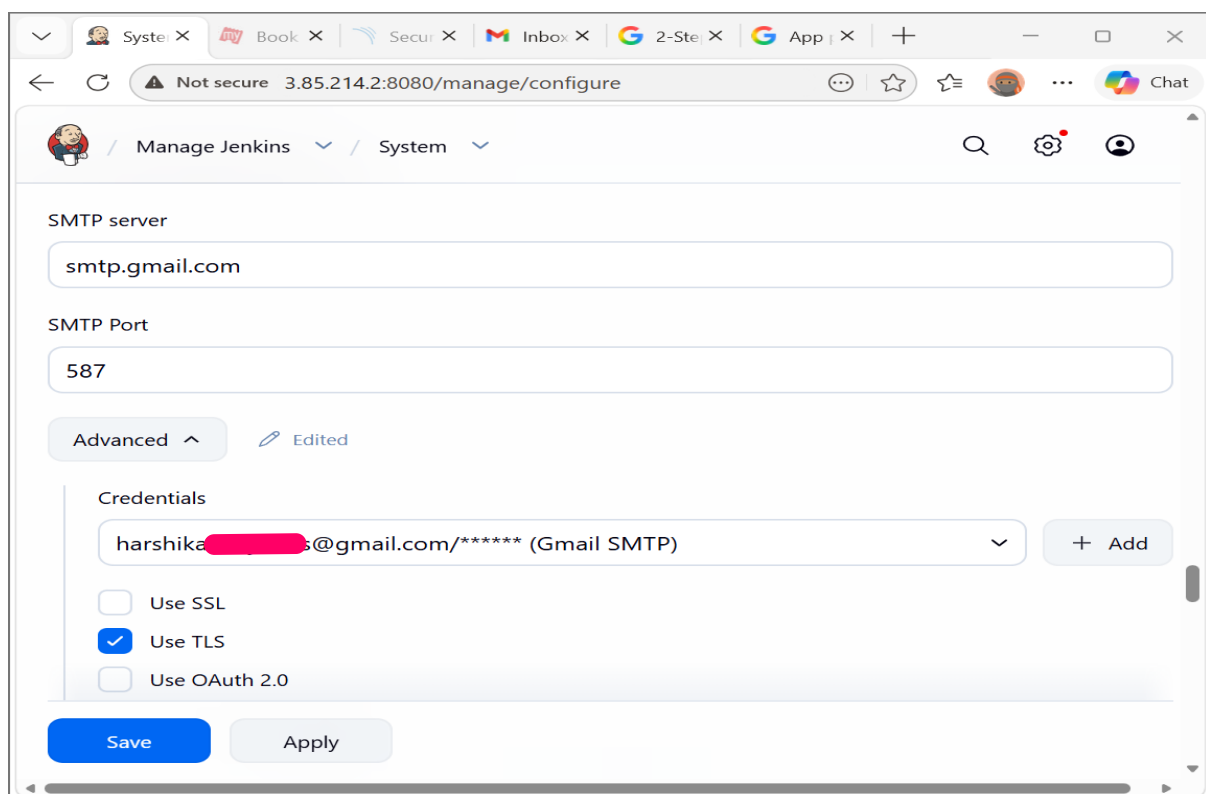
SMTP Port: 587

Use SMTP Authentication

Username: your-email@gmail.com

Password: (paste app password WITHOUT spaces)

Use TLS



The screenshot shows the Jenkins 'Configure System' page. The browser address bar indicates the URL is '3.85.214.2:8080/manage/configure'. The page title is 'Manage Jenkins / System'. The 'SMTP server' field is filled with 'smtp.gmail.com'. The 'SMTP Port' field is filled with '587'. Below these fields, there is an 'Advanced' section with a dropdown menu showing 'harshika@gmail.com/\*\*\*\*\* (Gmail SMTP)'. Under the 'Credentials' section, there are three checkboxes: 'Use SSL' (unchecked), 'Use TLS' (checked), and 'Use OAuth 2.0' (unchecked). At the bottom, there are 'Save' and 'Apply' buttons.

## Step 9: Advanced Settings

Click **Advanced**

Make sure:

Use TLS → checked

Use SSL → NOT checked

## Step 10: Test Email

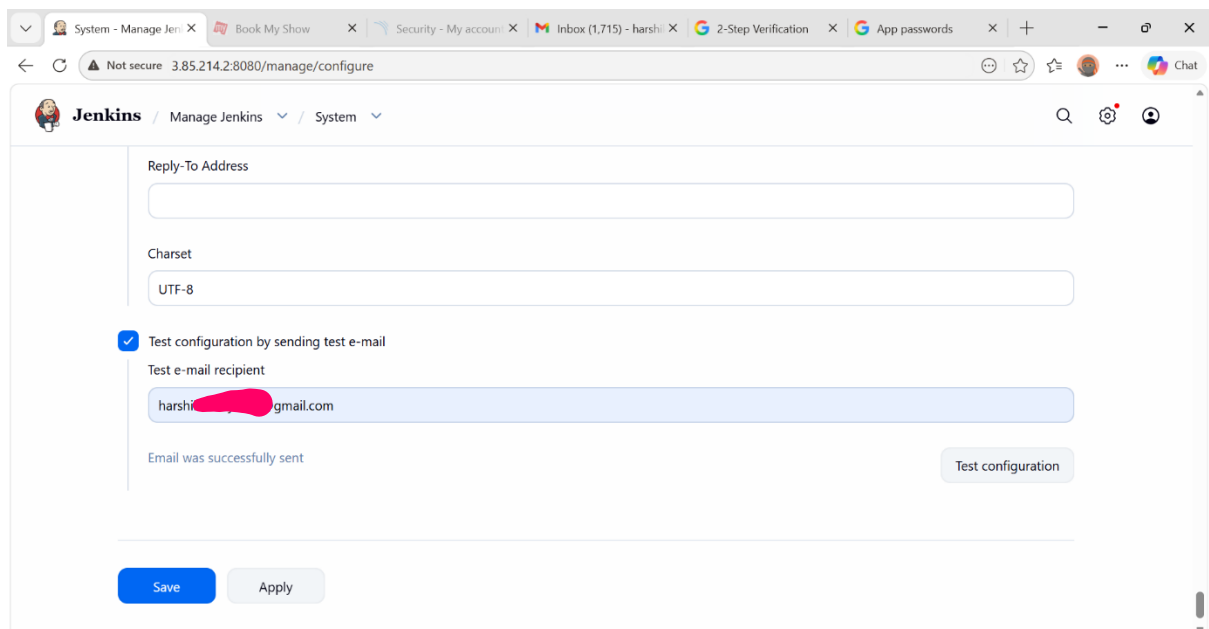
Scroll slightly down:

Find **“Test configuration by sending test e-mail”**

Enter your Gmail:

harshik[REDACTED]@gmail.com

Click **Test Configuration**



The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is 3.85.214.2:8080/manage/configure. The page title is 'Jenkins / Manage Jenkins / System'. The 'Test configuration by sending test e-mail' option is checked. The 'Test e-mail recipient' field contains 'harshik[REDACTED]@gmail.com'. A message at the bottom states 'Email was successfully sent'. There are 'Save' and 'Apply' buttons at the bottom left, and a 'Test configuration' button at the bottom right.

System - Manage Jen X Book My Show X Security - My account X Inbox (1,715) - harshi X 2-Step Verification X App passwords X + - Chat

Not secure 3.85.214.2:8080/manage/configure

Jenkins / Manage Jenkins / System

Reply-To Address

Charset

UTF-8

☒ Test configuration by sending test e-mail

Test e-mail recipient

harshik[REDACTED]@gmail.com

Email was successfully sent

Test configuration

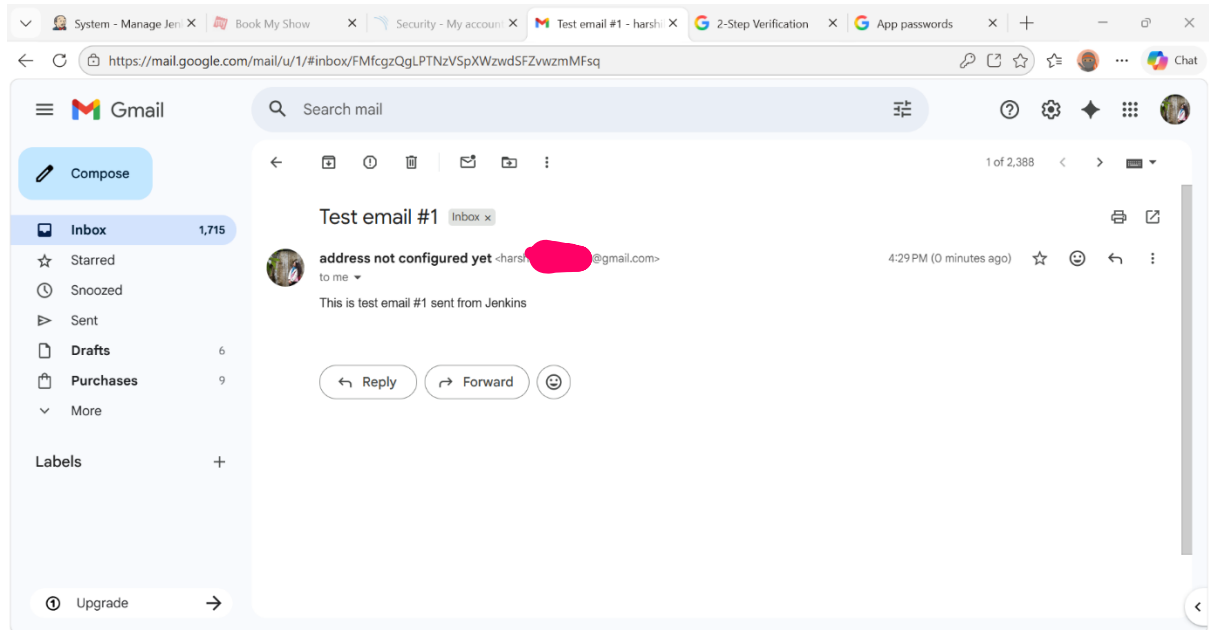
Save Apply

## Expected Result

You should see:

Email sent successfully

Also check inbox



**Now, create a webhook in Sonarcube:**

Go to Webhooks Section

- Click on Administration (top menu)
- Then go to:  
Configuration → Webhooks

Click “Create”

- On the right side, click the Create button

Fill Webhook Details

Enter the following:

- Name: Jenkins

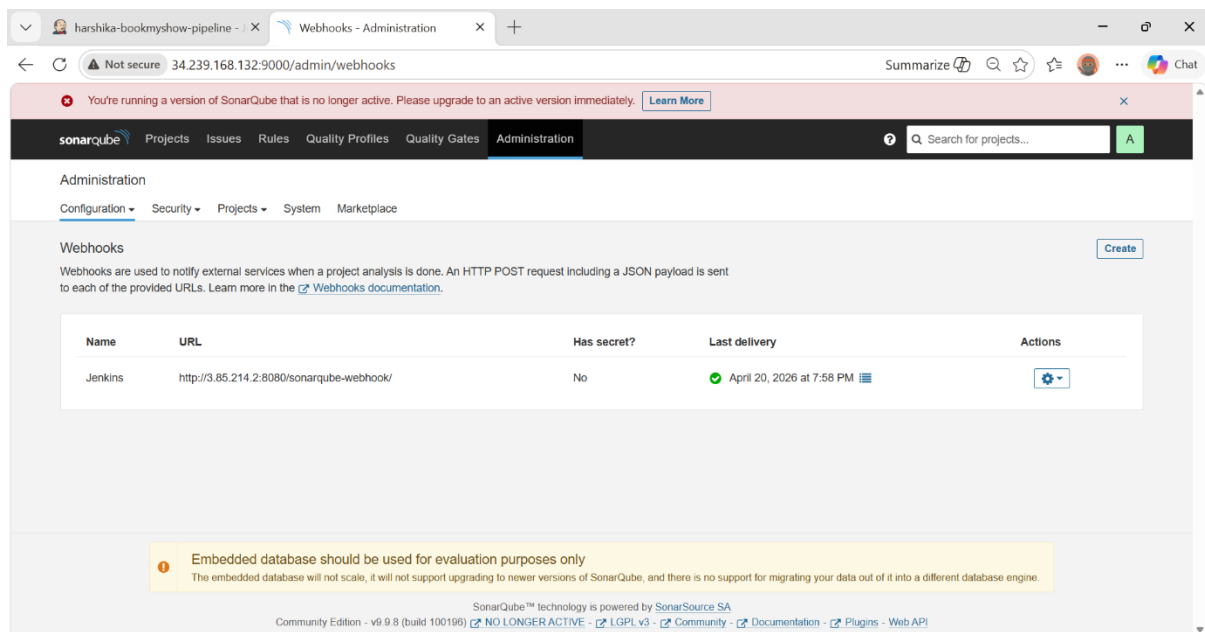
- URL:

http://<jenkins-ip>:8080/sonarqube-webhook/

**http://3.85.214.2:8080/sonarqube-webhook/**

**Click “Create”**

- After filling in the details, click Create



The screenshot shows the SonarQube Administration interface. At the top, there's a navigation bar with tabs for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration tab is selected. Below the navigation bar, there's a search bar and a 'Create' button. The main content area is titled 'Webhooks' and contains a table with the following data:

| Name    | URL                                       | Has secret? | Last delivery               | Actions |
|---------|-------------------------------------------|-------------|-----------------------------|---------|
| Jenkins | http://3.85.214.2:8080/sonarqube-webhook/ | No          | ✓ April 20, 2026 at 7:58 PM |         |

At the bottom of the page, there's a yellow warning box that says: 'Embedded database should be used for evaluation purposes only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.'

## PHASE 3: EKS SETUP (STEP-BY-STEP)

### STEP 1: Create IAM User

Amazon Web Services Console

→ Search **IAM**

→ Click **Users** → **Create user**

Amazon Web Services Sign-In

https://us-east-1.console.aws.amazon.com

Create user | IAM | Global

RWS IAM Users Create user

Set permissions

Step 3 Review and create

Step 4 Retrieve password

**User details**

**User name**

jenkins-user

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , @ \_ - (hyphen)

☒ **Provide user access to the AWS Management Console - optional**

In addition to console access, users with `SignInLocalDevelopmentAccess` permissions can use the same console credentials for programmatic access without the need for access keys.

**Console password**

☐ **Autogenerated password**

You can view the password after you create the user.

☒ **Custom password**

Enter a custom password for the user.

H

• Must be at least 8 characters long

• Must include at least three of the following mix of character types: uppercase letters (A-Z), lowercase letters (a-z), numbers (0-9), and symbols ! @ # \$ % ^ & \* ( ) \_ + - (hyphen) = [ ] { } ' "

☒ **Show password**

☐ **Users must create a new password at next sign-in - Recommended**

Users automatically get the `IAMUserChangePassword` policy to allow them to change their own password.

ⓘ If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. [Learn more](#)

Cancel Next

CloudShell Feedback Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**Fill details:**

**Username:**

jenkins-user

**Access:**

✓ Check **Programmatic access**

**Permissions:**

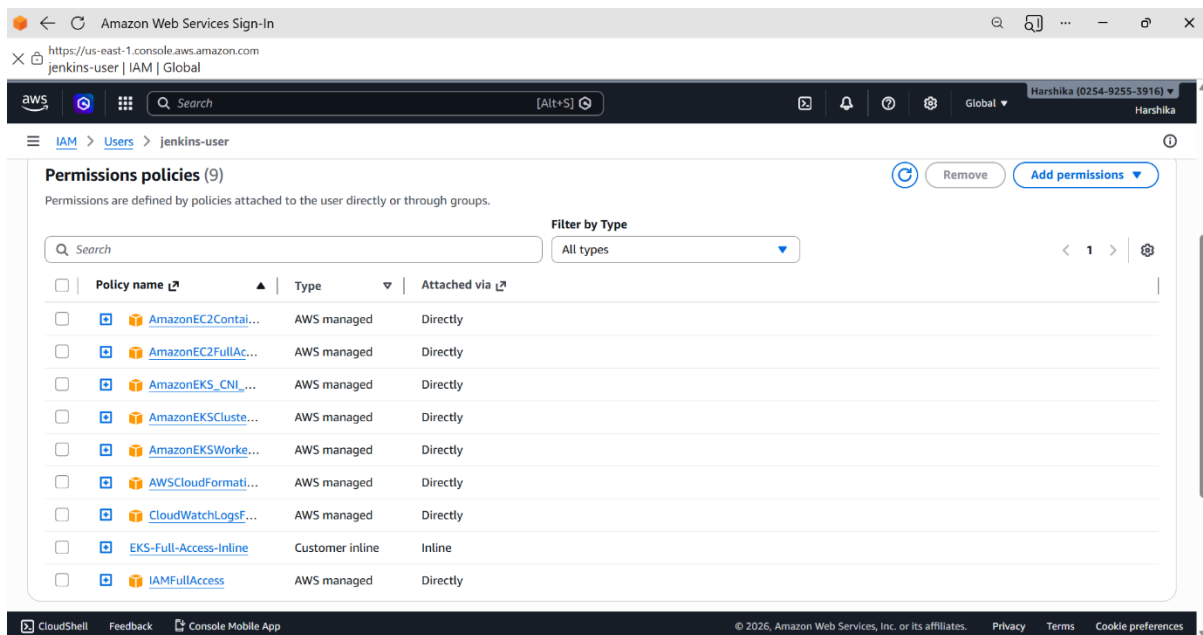
Select:

**Attach policies directly**

Add **ONLY** these:

- AmazonEKSClusterPolicy

- AmazonEKSWorkerNodePolicy
- AmazonEC2ContainerRegistryFullAccess
- AmazonEC2FullAccess ⚠️ (needed for EKS infra)
- CloudWatchLogsFullAccess
- AmazonEKS\_CNI\_Policy
- All other policy shown in below snapshot



**Finish:**

**Add Inline Policy**

**Click:**

**Add permissions → Create inline policy**

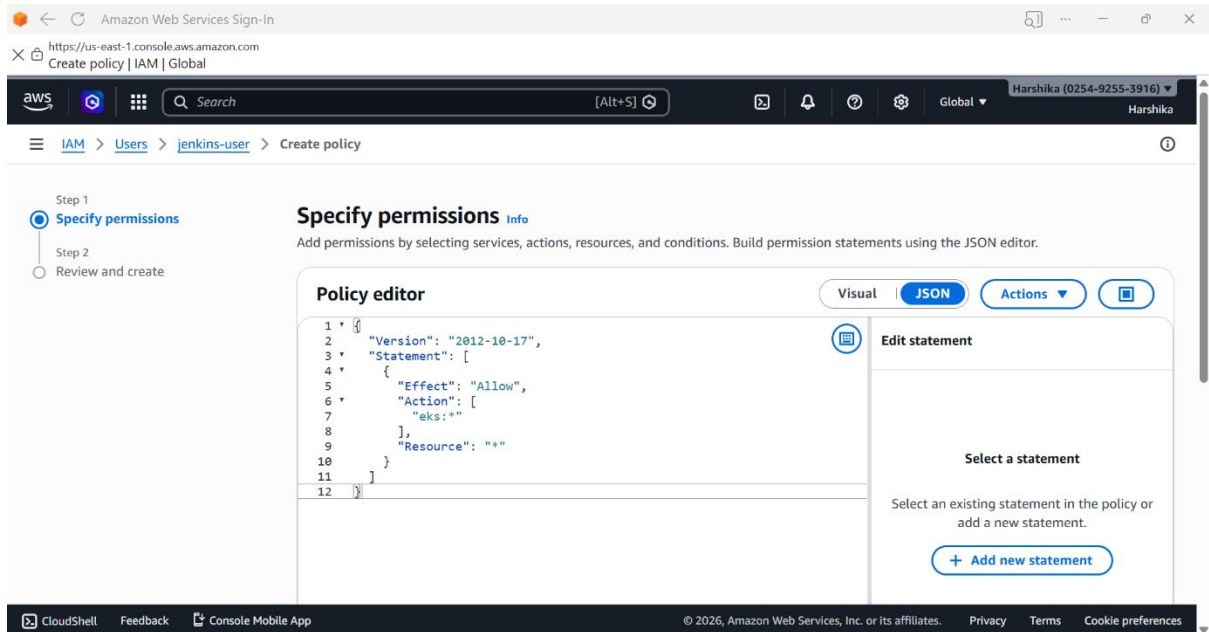
**Choose the JSON tab**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "eks:*"
      ]
    }
  ]
}
```

```

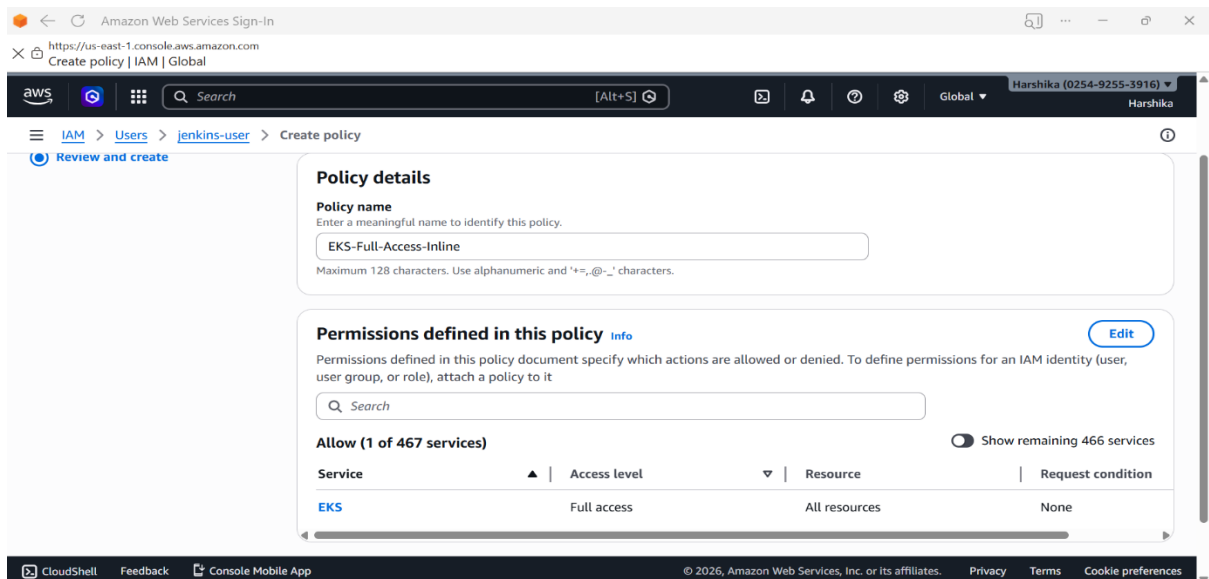
    },
    "Resource": "*"
  }
}
}

```

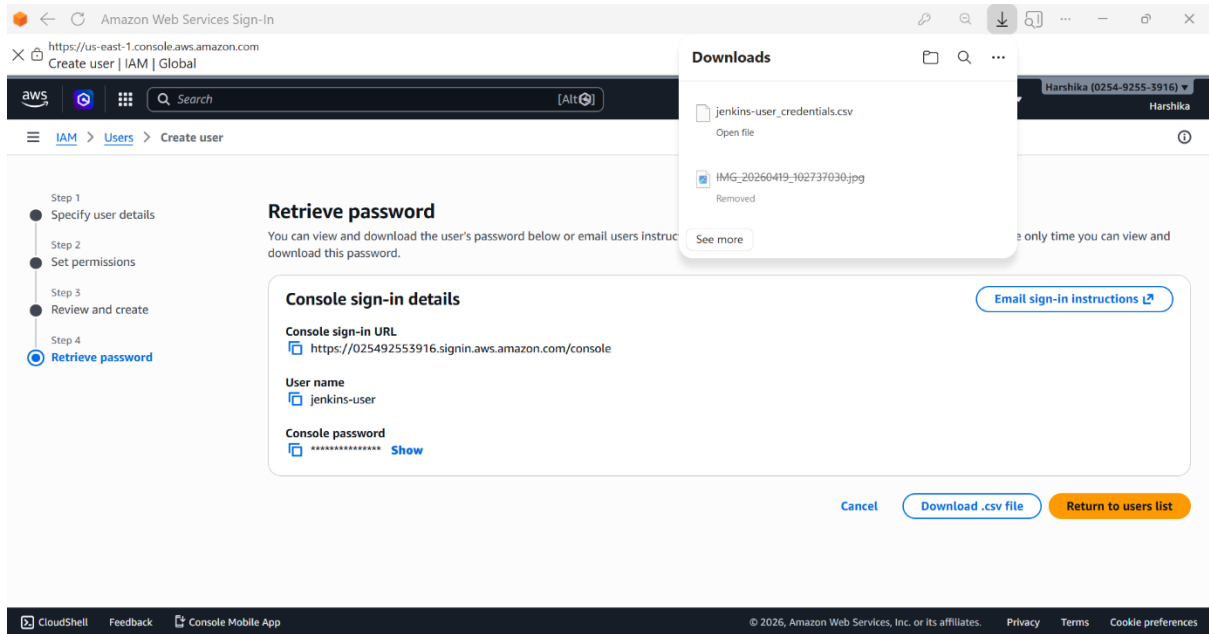


Then:

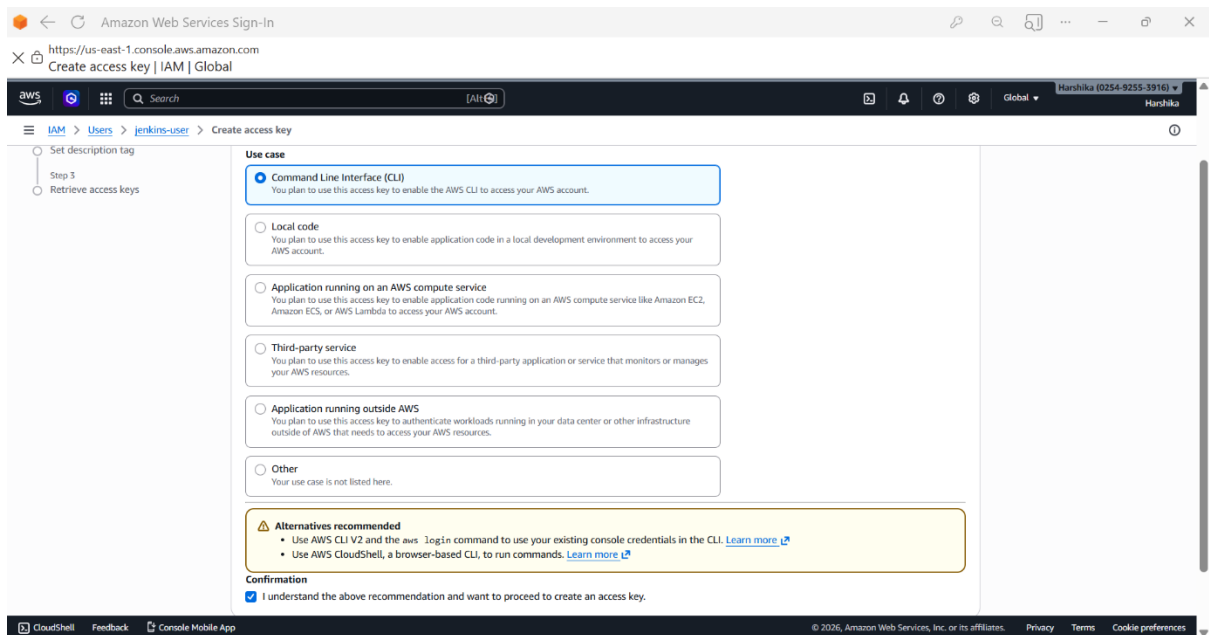
- Click Next
- Name: **EKS-Full-Access-Inline**
- Click Create policy



## Click Create User



## Click on jenkins-user and create access key:



## IMPORTANT

After creation, you will see:

- ✓ Access Key
- ✓ Secret Key

Download or copy them NOW (you won't see them again)

The first screenshot shows the 'Create access key' page for the 'jenkins-user' in the AWS IAM console. A green banner at the top states: 'This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key.' The page displays the 'Access key' section with the Access key ID 'AKIAQL33ZQC6OZVTQQON' and the Secret access key masked with asterisks. A 'Show' button is next to the secret key. Below this, 'Access key best practices' are listed, including: 'Never store your access key in plain text, in a code repository, or in code.', 'Disable or delete access key when no longer needed.', 'Enable least-privilege permissions.', and 'Rotate access keys regularly.' A 'Download .csv file' button is at the bottom right. A 'Downloads' window is open on the right, showing files 'jenkins-user\_accessKeys.csv' and 'jenkins-user\_credentials.csv' with 'Open file' buttons, and a removed image file 'IMG\_20260419\_102737030.jpg'.

The second screenshot shows the 'Access keys (1)' page for the 'jenkins-user'. It includes a 'Create access key' button. Below, a table lists the access key details:

| AKIAQL33ZQC6OZVTQQON |                     | Actions           |               |
|----------------------|---------------------|-------------------|---------------|
| Description          | jenkins-user access | Status            | Active        |
| Last used            | None                | Created           | 2 minutes ago |
| Last used region     | N/A                 | Last used service | N/A           |

## Step 2: Add AWS Credentials in Jenkins

Go to:

Jenkins → Manage Jenkins → Credentials → Global → Add Credentials

Add:

Type

## AWS Credentials

Fill:

- **Access Key ID** → (copy)
- **Secret Access Key** → (copy)
- **ID** → aws-creds

Click **Save**

The screenshot shows the Jenkins 'Add AWS Credentials' dialog box. The dialog is titled 'Add AWS Credentials' and has a close button (X) in the top right corner. It contains the following fields and options:

- Global (Jenkins, nodes, items, all child items, etc)**: A dropdown menu.
- ID**: A text input field containing 'aws-creds'.
- Description**: A text input field containing 'aws creds'.
- Access Key ID**: A text input field containing 'AKIAQL33ZQC6OZVTQQON'.
- Secret Access Key**: A text input field containing a masked string '.....'.
- These credentials are valid and have access to 6 availability zones**: A status message.
- IAM Role Support**: A section with an 'Advanced' dropdown menu.
- Create**: A blue button at the bottom.

The background shows the Jenkins 'Credentials' page with a list of existing credentials: 'docker-creds', 'Sonar-token', and 'gmail-smtp'.

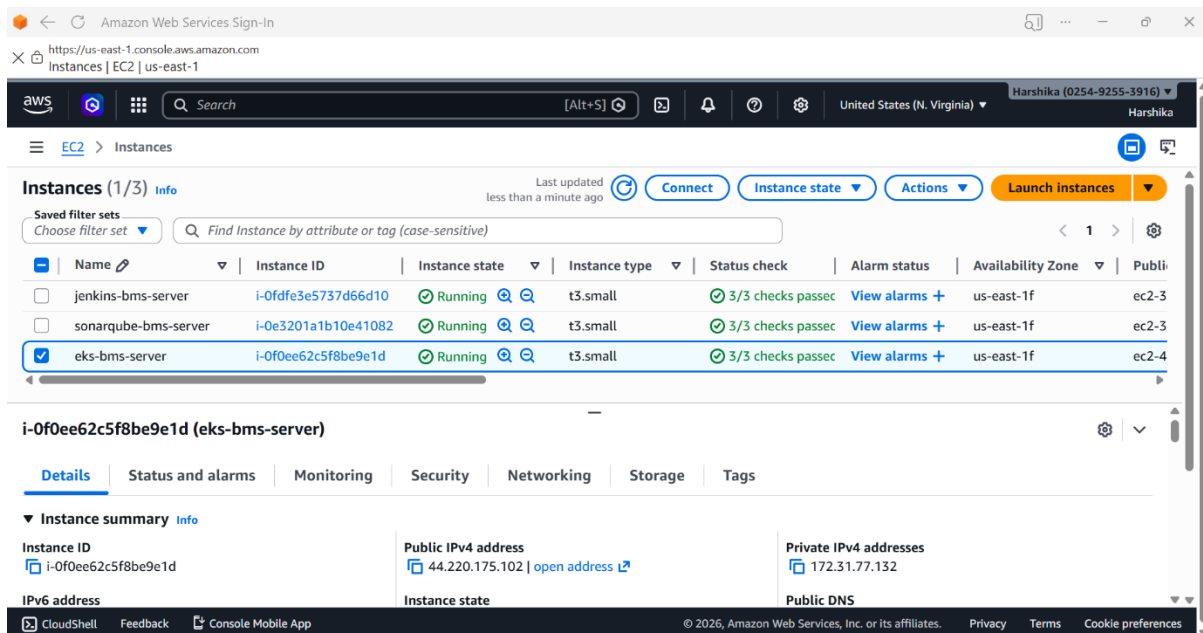
The screenshot shows the Jenkins 'Credentials' page. The page title is 'Credentials'. There is a blue button labeled '+ Add Credentials' and a help icon (?) in the top right corner. The page displays a list of credentials:

| Icon | Name         | Description                                              | Actions |
|------|--------------|----------------------------------------------------------|---------|
| 🔑    | docker-creds | harshika/*****<br>System - Global                        | ✎ ...   |
| 🔑    | Sonar-token  | System - Global - Sonar Token                            | ✎ ...   |
| 🔑    | gmail-smtp   | harshika@gmail.com/*****<br>System - Global - Gmail SMTP | ✎ ...   |
| 🔑    | aws-creds    | AKIAQL33ZQC6OZVTQQON<br>System - Global - aws creds      | ✎ ...   |

The 'aws-creds' entry is highlighted, indicating it is the newly added credential.

## STEP 3: Create a new server for the EKS Cluster (Launch EC2 (t3.small))

Launch the instance with the same configuration as the Jenkins-server



Connect via SSH from PowerShell

### 1. Install AWS CLI v2 (Ubuntu 24.04)

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

```
sudo apt update
```

```
sudo apt install unzip -y
```

```
unzip awscliv2.zip
```

```
sudo ./aws/install
```

### VERIFY

```
aws --version
```

Expected:

```
aws-cli/2.x.x Python/3.x ...
```

```
ubuntu@ip-172-31-71-135: ~
ubuntu@ip-172-31-77-132: ~
+
inflating: aws/dist/awscli/data/metadata.json
creating: aws/dist/awscli/customizations/sso/
creating: aws/dist/awscli/customizations/wizard/
creating: aws/dist/awscli/customizations/wizard/wizards/
creating: aws/dist/awscli/customizations/wizard/wizards/configure/
creating: aws/dist/awscli/customizations/wizard/wizards/dynamodb/
creating: aws/dist/awscli/customizations/wizard/wizards/iam/
creating: aws/dist/awscli/customizations/wizard/wizards/lambda/
inflating: aws/dist/awscli/customizations/wizard/wizards/configure/_main.yml

inflating: aws/dist/awscli/customizations/wizard/wizards/iam/new-role.yml
inflating: aws/dist/awscli/customizations/wizard/wizards/lambda/new-function
.yml
inflating: aws/dist/awscli/customizations/wizard/wizards/events/new-rule.yml
inflating: aws/dist/awscli/customizations/wizard/wizards/dynamodb/new-table.
.yml
inflating: aws/dist/awscli/customizations/sso/index.html
creating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/top_level.txt
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/METADATA
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/WHEEL
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/INSTALLER
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/RECORD
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/AUTHORS.rst
inflating: aws/dist/prompt_toolkit-3.0.51.dist-info/licenses/LICENSE
ubuntu@ip-172-31-77-132:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
ubuntu@ip-172-31-77-132:~$ aws --version
aws-cli/2.34.32 Python/3.14.4 Linux/6.17.0-1007-aws exe/x86_64.ubuntu.24
ubuntu@ip-172-31-77-132:~$
```

## 2. Configure AWS

aws configure

### 1. Access Key

Access Key ID

### 2. Secret Key

Secret Access Key

### 3. Region

us-east-1

(important — same region we'll use for EKS)

### 4. Output format

json

## AFTER THIS (VERIFY)

Run:

aws sts get-caller-identity

```

ubuntu@ip-172-31-77-132:~$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
ubuntu@ip-172-31-77-132:~$ aws --version
aws-cli/2.34.32 Python/3.14.4 Linux/6.17.0-1007-aws exe/x86_64.ubuntu.24
ubuntu@ip-172-31-77-132:~$ aws configure

Tip: You can deliver temporary credentials to the AWS CLI using your AWS Console session by running the command 'aws login'.

AWS Access Key ID [None]: AKIAQL33ZQC60ZVTQQON
AWS Secret Access Key [None]: XplmMRmZPPujcldl3Dinx2I60Mvgztk+Un85sHCo
Default region name [None]: us-east-1
Default output format [None]: json
ubuntu@ip-172-31-77-132:~$ aws sts get-caller-identity
{
  "UserId": "AIDAQL33ZQC6KT5MMGK4A",
  "Account": "025492553916",
  "Arn": "arn:aws:iam::025492553916:user/jenkins-user"
}
ubuntu@ip-172-31-77-132:~$

```

#### STEP 4: Install eksctl

```

curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_Linux_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

```

```

ubuntu@ip-172-31-77-132:~$ curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_Linux_amd64.tar.gz" | tar xz -C /tmp
ubuntu@ip-172-31-77-132:~$ sudo mv /tmp/eksctl /usr/local/bin
ubuntu@ip-172-31-77-132:~$ eksctl version
0.225.0
ubuntu@ip-172-31-77-132:~$

```

#### STEP 5: Install kubectl

```

curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

chmod +x kubectl

sudo mv kubectl /usr/local/bin/

kubectl version --client

```

```

ubuntu@ip-172-31-77-132:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload  Total  Spent    Left  Speed
100 55.8M  100 55.8M    0     0  140M      0 --:--:-- --:--:-- --:--:-- 140M
ubuntu@ip-172-31-77-132:~$ chmod +x kubectl
sudo mv kubectl /usr/local/bin/
ubuntu@ip-172-31-77-132:~$ kubectl version --client
Client Version: v1.35.4
Kustomize Version: v5.7.1
ubuntu@ip-172-31-77-132:~$

```

## STEP 6: Create EKS Cluster (optimised)

Run this:

```

eksctl create cluster \
--name bms-cluster \
--region us-east-1 \
--nodegroup-name bms-nodes \
--node-type t3.small \
--nodes 1 \
--nodes-min 1 \
--nodes-max 1

```

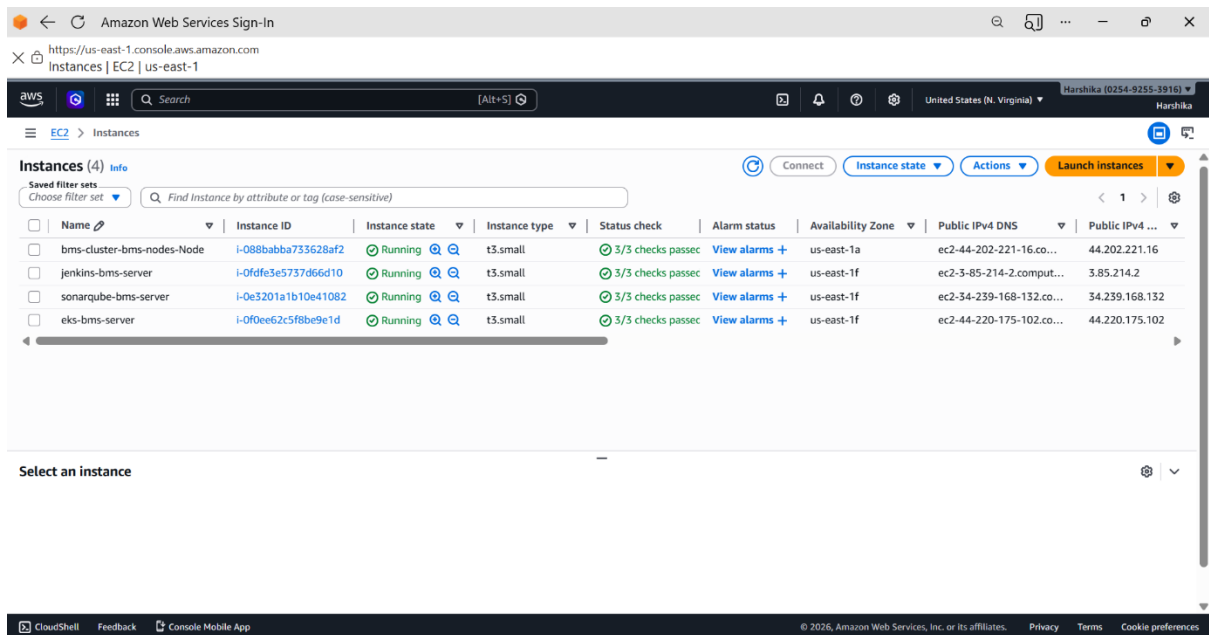
after completion

```

2026-04-19 15:53:27 [i] deploying stack "eksctl-bms-cluster-nodegroup-bms-nodes"
2026-04-19 15:53:27 [i] waiting for CloudFormation stack "eksctl-bms-cluster-nodegroup-bms-nodes"
2026-04-19 15:53:57 [i] waiting for CloudFormation stack "eksctl-bms-cluster-nodegroup-bms-nodes"
2026-04-19 15:54:34 [i] waiting for CloudFormation stack "eksctl-bms-cluster-nodegroup-bms-nodes"
2026-04-19 15:56:12 [i] waiting for CloudFormation stack "eksctl-bms-cluster-nodegroup-bms-nodes"
2026-04-19 15:56:12 [i] no tasks
2026-04-19 15:56:12 [i] created 0 nodegroup(s) in cluster "bms-cluster"
2026-04-19 15:56:12 [i] nodegroup "bms-nodes" has 1 node(s)
2026-04-19 15:56:12 [i] node "ip-192-168-5-81.ec2.internal" is ready
2026-04-19 15:56:12 [i] waiting for at least 1 node(s) to become ready in "bms-nodes"
2026-04-19 15:56:12 [i] nodegroup "bms-nodes" has 1 node(s)
2026-04-19 15:56:12 [i] node "ip-192-168-5-81.ec2.internal" is ready
2026-04-19 15:56:12 [i] created 1 managed nodegroup(s) in cluster "bms-cluster"
2026-04-19 15:56:12 [i] checking security group configuration for all nodegroups
2026-04-19 15:56:12 [i] all nodegroups have up-to-date cloudformation templates
ubuntu@ip-172-31-77-132:~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
ip-192-168-5-81.ec2.internal        Ready    <none>    2m20s  v1.34.6-eks-bbe087e
ubuntu@ip-172-31-77-132:~$ kubectl get pods -A
NAMESPACE   NAME                                READY    STATUS    RESTARTS   AGE
kube-system  aws-node-kg5s9                     2/2     Running   0           3m10s
kube-system  coredns-7d58d485c9-64hj5           1/1     Running   0           15m
kube-system  coredns-7d58d485c9-ts2r7           1/1     Running   0           15m
kube-system  kube-proxy-8xjt6                   1/1     Running   0           3m10s
kube-system  metrics-server-6f49c4bc6c-7djkx7   1/1     Running   0           14m
kube-system  metrics-server-6f49c4bc6c-n8dd8     1/1     Running   0           14m
ubuntu@ip-172-31-77-132:~$

```

## Check your EC2 Console for the node



## Step 7: Install VPC CNI (MOST IMPORTANT)

Run:

```
eksctl create addon \
--name vpc-cni \
--cluster bms-cluster \
--region us-east-1 \
--force
```

## Step 8: Install CoreDNS

```
eksctl create addon \
--name coredns \
--cluster bms-cluster \
--region us-east-1
```

## Step 9: Install kube-proxy

```
eksctl create addon \
--name kube-proxy \
--cluster bms-cluster \
--region us-east-1
```

## Step 10: (optional but good)

```
eksctl create addon \  
--name metrics-server \  
--cluster bms-cluster \  
--region us-east-1
```

Wait 2–3 minutes

Now check:

Eksctl get cluster

```
ubuntu@ip-172-31-77-132:~$ eksctl get cluster  
NAME          REGION    EKSTCL CREATED  
bms-cluster    us-east-1 True  
ubuntu@ip-172-31-77-132:~$ aws eks describe-cluster --name bms-cluster --region us-east-1 --query "cluster.status"  
"ACTIVE"  
ubuntu@ip-172-31-77-132:~$
```

kubectrl get nodes

```
2026-04-19 15:53:27 [i] deploying stack "eksctl-bms-cluster-nodegroup-bms-nodes"  
2026-04-19 15:53:27 [i] waiting for CloudFormation stack "eksctl-bms-cluster-nodegroup-bms-nodes"  
2026-04-19 15:53:57 [i] waiting for CloudFormation stack "eksctl-bms-cluster-nodegroup-bms-nodes"  
2026-04-19 15:54:34 [i] waiting for CloudFormation stack "eksctl-bms-cluster-nodegroup-bms-nodes"  
2026-04-19 15:56:12 [i] waiting for CloudFormation stack "eksctl-bms-cluster-nodegroup-bms-nodes"  
2026-04-19 15:56:12 [i] no tasks  
2026-04-19 15:56:12 [i] created 0 nodegroup(s) in cluster "bms-cluster"  
2026-04-19 15:56:12 [i] nodegroup "bms-nodes" has 1 node(s)  
2026-04-19 15:56:12 [i] node "ip-192-168-5-81.ec2.internal" is ready  
2026-04-19 15:56:12 [i] waiting for at least 1 node(s) to become ready in "bms-nodes"  
2026-04-19 15:56:12 [i] nodegroup "bms-nodes" has 1 node(s)  
2026-04-19 15:56:12 [i] node "ip-192-168-5-81.ec2.internal" is ready  
2026-04-19 15:56:12 [i] created 1 managed nodegroup(s) in cluster "bms-cluster"  
2026-04-19 15:56:12 [i] checking security group configuration for all nodegroups  
2026-04-19 15:56:12 [i] all nodegroups have up-to-date cloudformation templates  
ubuntu@ip-172-31-77-132:~$ kubectrl get nodes  
NAME                                STATUS    ROLES    AGE    VERSION  
ip-192-168-5-81.ec2.internal        Ready    <none>    2m20s    v1.34.6-eks-bbe087e  
ubuntu@ip-172-31-77-132:~$ kubectrl get pods -A  
NAMESPACE    NAME                                READY    STATUS    RESTARTS    AGE  
kube-system    aws-node-kg5s9                    2/2      Running    0            3m10s  
kube-system    coredns-7d58d485c9-64hj5         1/1      Running    0            15m  
kube-system    coredns-7d58d485c9-ts2r7         1/1      Running    0            15m  
kube-system    kube-proxy-8xjt6                 1/1      Running    0            3m10s  
kube-system    metrics-server-6f49c4bc6c-7dix7  1/1      Running    0            14m  
kube-system    metrics-server-6f49c4bc6c-n8dd8  1/1      Running    0            14m  
ubuntu@ip-172-31-77-132:~$
```

## 4. Log in to your Jenkins EC2 server

Then install Docker:

```
sudo apt update
sudo apt install docker.io -y
sudo systemctl start docker
sudo systemctl enable docker
```

## Give Jenkins permission to use Docker

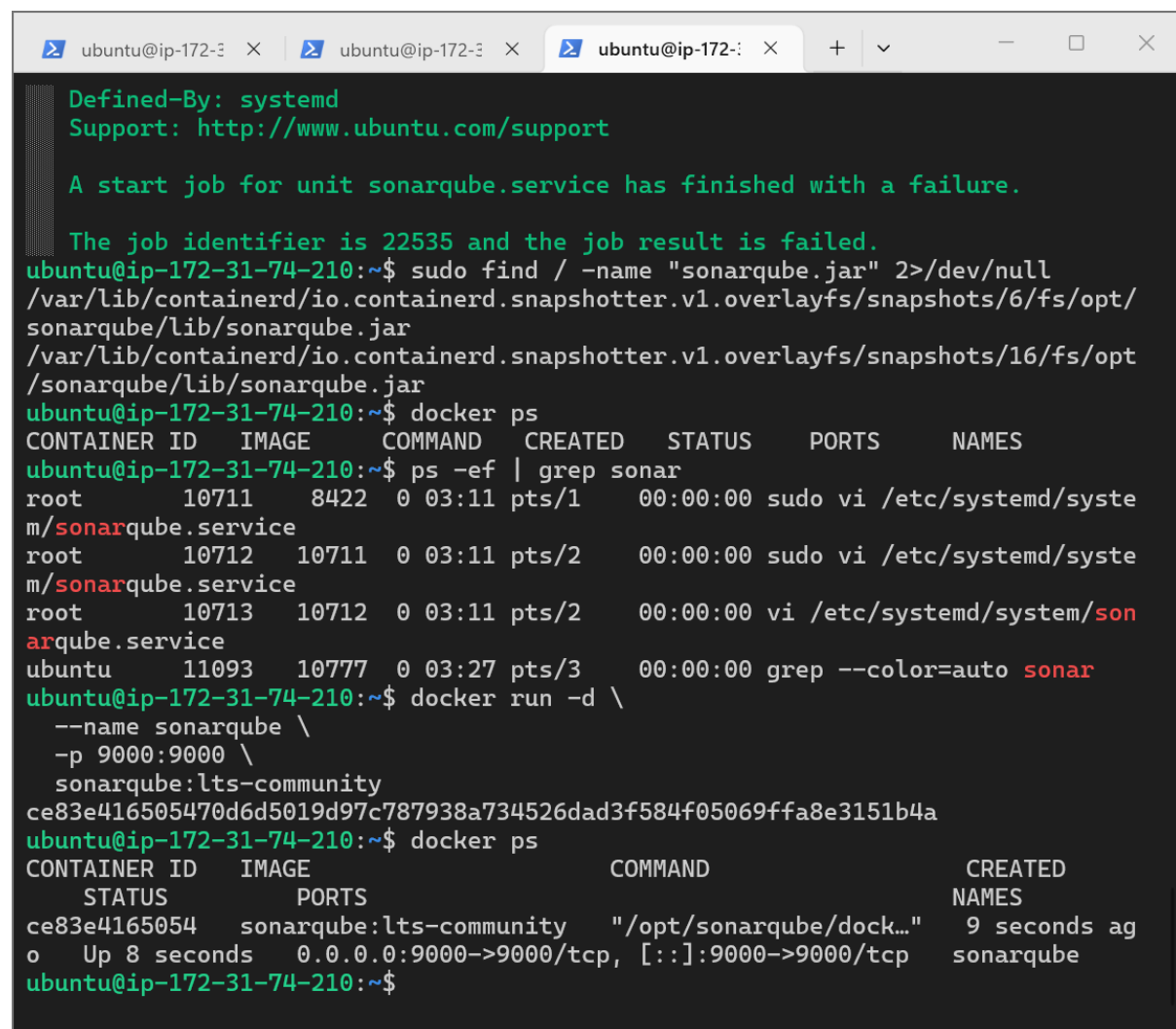
```
sudo usermod -aG docker jenkins
sudo usermod -aG docker ubuntu
```

Then restart:

```
sudo systemctl restart jenkins
```

## Verify

```
docker ps
```

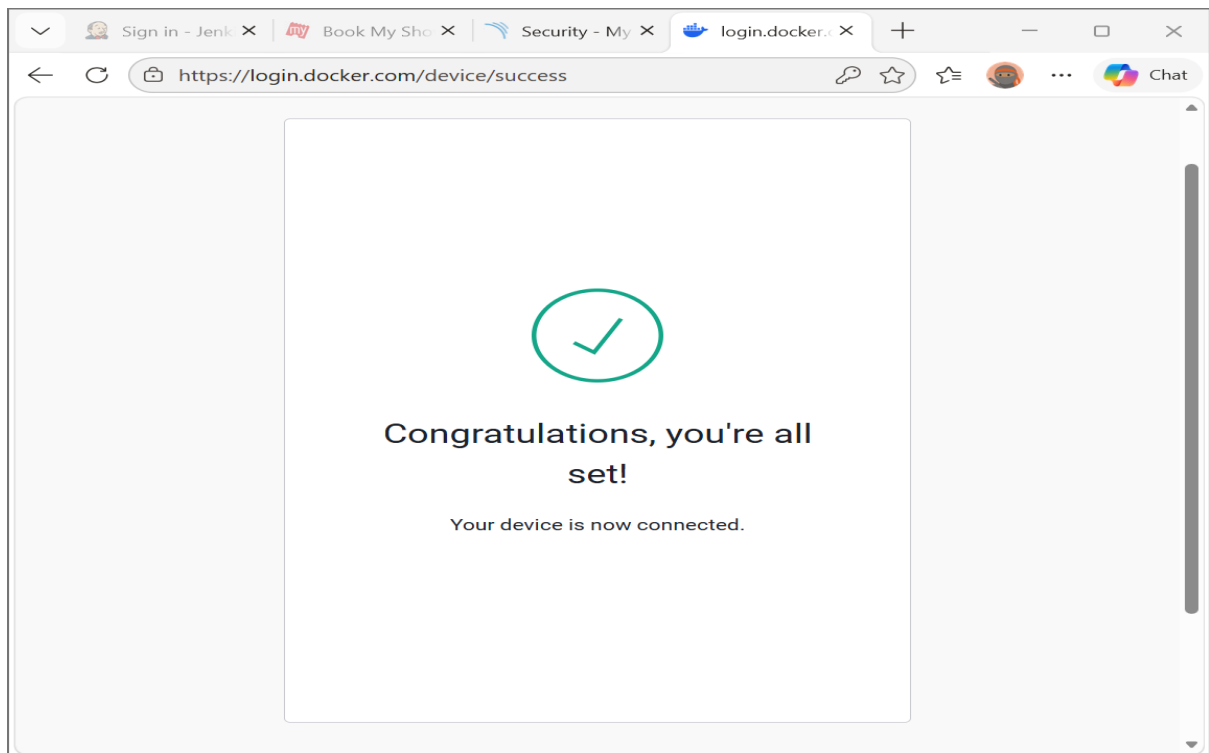
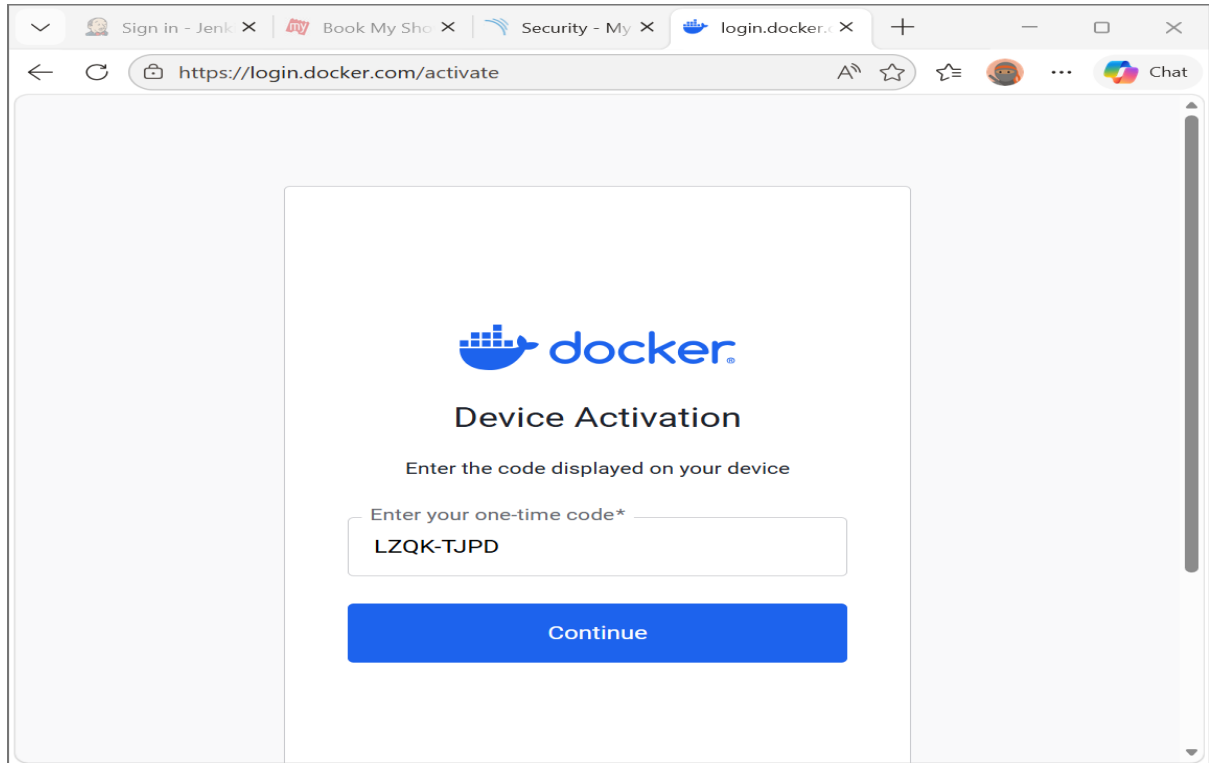
A terminal window with three tabs, all showing 'ubuntu@ip-172-31-74-210'. The terminal output shows the following sequence of commands and results:  
1. A green message: 'Defined-By: systemd' and 'Support: http://www.ubuntu.com/support'.  
2. A green message: 'A start job for unit sonarqube.service has finished with a failure.'  
3. A green message: 'The job identifier is 22535 and the job result is failed.'  
4. Command: 'sudo find / -name "sonarqube.jar" 2>/dev/null'. Output shows the file path: '/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/6/fs/opt/sonarqube/lib/sonarqube.jar'.  
5. Command: 'docker ps'. Output shows a table with columns: CONTAINER ID, IMAGE, COMMAND, CREATED, STATUS, PORTS, NAMES. It lists three containers related to 'sonarqube.service'.  
6. Command: 'ps -ef | grep sonar'. Output shows the same three containers.  
7. Command: 'docker run -d --name sonarqube -p 9000:9000 sonarqube:lts-community'. Output shows the container ID 'ce83e416505470d6d5019d97c787938a734526dad3f584f05069ffa8e3151b4a'.  
8. Command: 'docker ps'. Output shows a table with columns: CONTAINER ID, IMAGE, STATUS, PORTS, COMMAND, CREATED, NAMES. It shows the 'sonarqube' container is 'Up 8 seconds' and listening on '0.0.0.0:9000->9000/tcp'.

```
ubuntu@ip-172-31-74-210:~$ sudo find / -name "sonarqube.jar" 2>/dev/null
/var/lib/containerd/io.containerd.snapshotter.v1.overlayfs/snapshots/6/fs/opt/sonarqube/lib/sonarqube.jar
ubuntu@ip-172-31-74-210:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
ubuntu@ip-172-31-74-210:~$ ps -ef | grep sonar
root          10711         8422   0 03:11 pts/1    00:00:00 sudo vi /etc/systemd/system/sonarqube.service
root          10712        10711   0 03:11 pts/2    00:00:00 sudo vi /etc/systemd/system/sonarqube.service
root          10713        10712   0 03:11 pts/2    00:00:00 vi /etc/systemd/system/sonarqube.service
ubuntu       11093        10777   0 03:27 pts/3    00:00:00 grep --color=auto sonar
ubuntu@ip-172-31-74-210:~$ docker run -d \
--name sonarqube \
-p 9000:9000 \
sonarqube:lts-community
ce83e416505470d6d5019d97c787938a734526dad3f584f05069ffa8e3151b4a
ubuntu@ip-172-31-74-210:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS          NAMES
ce83e4165054   sonarqube:lts-community  "/opt/sonarqube/dock...  9 seconds ag
o   Up 8 seconds   0.0.0.0:9000->9000/tcp, [::]:9000->9000/tcp  sonarqube
ubuntu@ip-172-31-74-210:~$
```

## Step 1: Push your image to DockerHub

### Login

docker login



## Push image

docker push harshikaaa/bookmyshow:latest

```
ubuntu@ip-172-31-71-135: ~  
Your one-time device confirmation code is: LZQK-TJPD  
Press ENTER to open your browser or submit your device code here: https://login.docker.com/activate  
  
Waiting for authentication in the browser...  
  
WARNING! Your credentials are stored unencrypted in '/home/ubuntu/.docker/config.json'.  
Configure a credential helper to remove this warning. See  
https://docs.docker.com/go/credential-store/  
  
Login Succeeded  
ubuntu@ip-172-31-71-135:~$  
ubuntu@ip-172-31-71-135:~$  
ubuntu@ip-172-31-71-135:~$ docker push harshikaaa/bookmyshow:latest  
The push refers to repository [docker.io/harshikaaa/bookmyshow]  
3697be50c98b: Layer already exists  
ac1eee27316a: Layer already exists  
a0697b23cfal: Layer already exists  
b0e8f5eb32df: Layer already exists  
6d65868elf20: Already exists  
cb284be93d41: Layer already exists  
3e6b9d1a9511: Layer already exists  
37927ed901b1: Layer already exists  
79b2f47ad444: Layer already exists  
c6b30c3f1696: Layer already exists  
461077a72fb7: Layer already exists  
4ab82a03a340: Layer already exists  
e23f099911d6: Layer already exists  
cda7f44f2bdd: Layer already exists  
latest: digest: sha256:b09f2da459d08826eec356e903db5e194dd7f4fcf5b36add10df2ea13184c52c size: 856  
ubuntu@ip-172-31-71-135:~$
```

## Now deploy to EKS (final step)

Go to your EKS server, mine- (ip-172-31-77-132)

### Step 1: Create deployment.yaml (vi)

vi deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: bms-app

spec:

replicas: 1

selector:

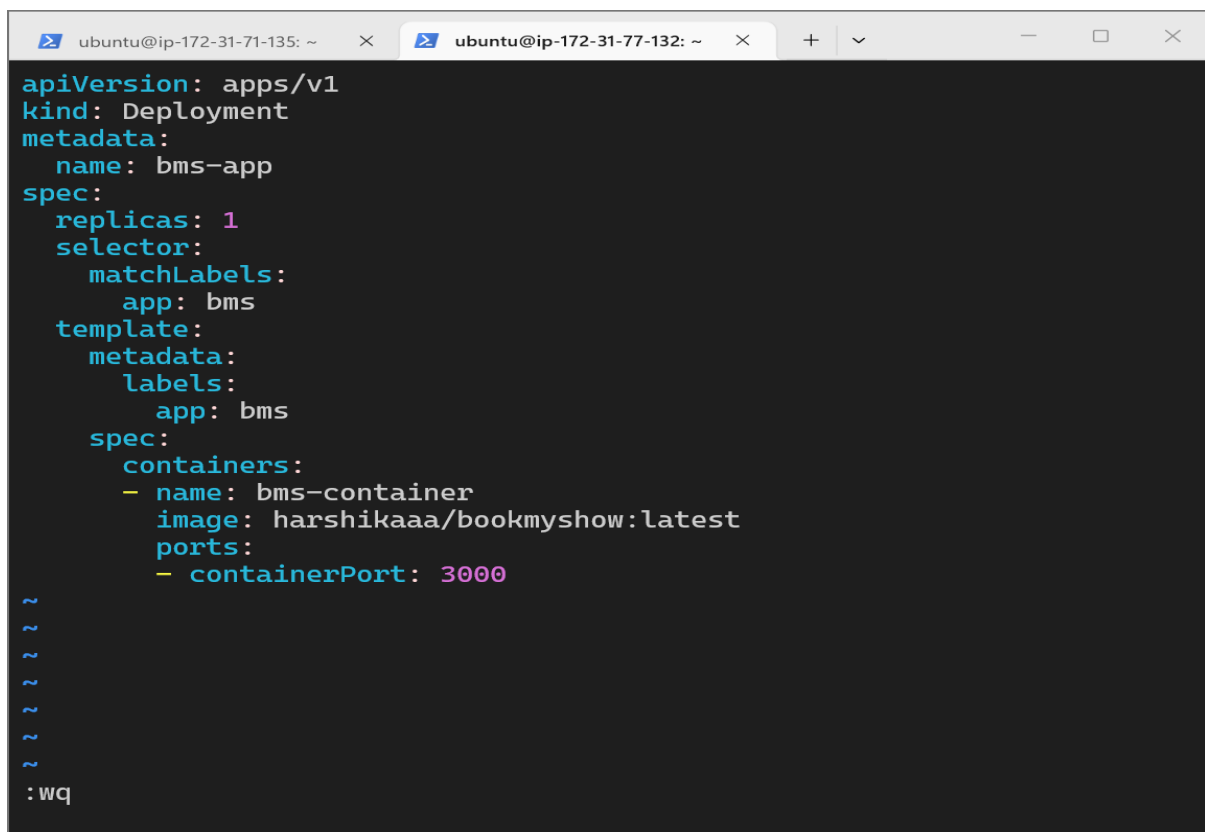
matchLabels:

app: bms

template:

metadata:

```
labels:
  app: bms
spec:
  containers:
  - name: bms-container
    image: harshikaaa/bookmyshow:latest
    ports:
    - containerPort: 3000
```



The screenshot shows a terminal window with two tabs. The active tab is titled 'ubuntu@ip-172-31-77-132: ~'. The terminal displays a Kubernetes Deployment manifest for 'bms-app'. The manifest includes fields for 'apiVersion', 'kind', 'metadata', 'spec', 'replicas', 'selector', 'matchLabels', 'template', 'metadata', 'labels', 'spec', 'containers', 'name', 'image', and 'ports'. The 'containers' section lists a single container named 'bms-container' with the image 'harshikaaa/bookmyshow:latest' and a 'containerPort' of '3000'. The terminal also shows a vertical line of tilde characters on the left and a prompt ':wq' at the bottom.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: bms-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bms
  template:
    metadata:
      labels:
        app: bms
    spec:
      containers:
      - name: bms-container
        image: harshikaaa/bookmyshow:latest
        ports:
        - containerPort: 3000
~
~
~
~
~
~
~
~
~
~
:wq
```

## Step 2: Create service.yaml

vi service.yaml

Press i and paste:

```
apiVersion: v1
kind: Service
metadata:
  name: bms-service
spec:
```

```
type: LoadBalancer
selector:
  app: bms
ports:
  - protocol: TCP
    port: 80
    targetPort: 3000
```

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

### Step 3: Apply configs

```
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
```

## Step 4: Check pods

```
kubectl get pods
```

```
ubuntu@ip-172-31-77-132:~$ vi deployment.yaml
19L, 329B written
ubuntu@ip-172-31-77-132:~$ vi service.yaml
[New] 12L, 178B written
ubuntu@ip-172-31-77-132:~$ kubectl apply -f deployment.yaml
kubectl apply -f service.yaml
deployment.apps/bms-app created
service/bms-service created
ubuntu@ip-172-31-77-132:~$ kubectl get pods
NAME                                READY   STATUS             RESTARTS   AG
E
bms-app-6f86c9d967-gql56           0/1     ContainerCreating   0           10
s
ubuntu@ip-172-31-77-132:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
bms-app-6f86c9d967-gql56           1/1     Running   0           2m41s
ubuntu@ip-172-31-77-132:~$
```

## Step 5: Get public URL

kubectl get svc

```
ubuntu@ip-172-31-77-132:~$ kubectl get svc
NAME      TYPE          CLUSTER-IP      EXTERNAL-IP
bms-service LoadBalancer  10.100.247.173   ae99c65dd23334066a26ab997e574b43-971348613.us-east-1.elb.amazonaws.com
kubernetes ClusterIP      10.100.0.1       <none>
```

## What you'll see

| NAME        | TYPE         | CLUSTER-IP | EXTERNAL-IP | PORT(S)      |
|-------------|--------------|------------|-------------|--------------|
| bms-service | LoadBalancer | 10.x.x.x   | <pending>   | 80:xxxxx/TCP |

Wait 1–3 minutes until:

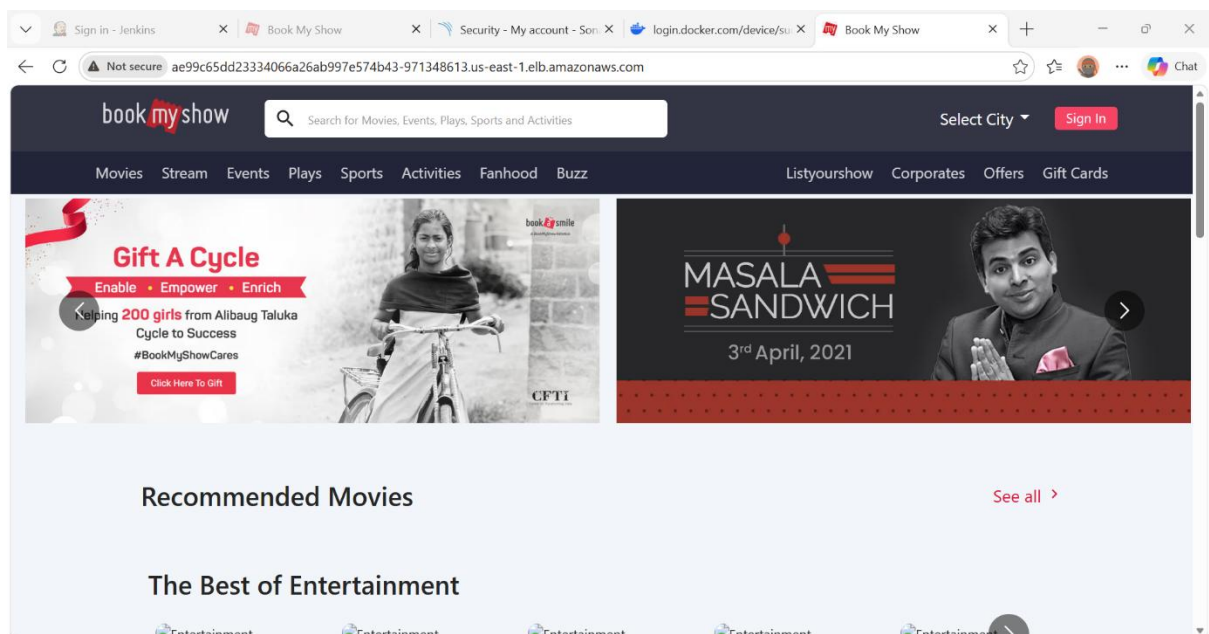
EXTERNAL-IP → something.elb.amazonaws.com

## Final step

Open in browser:

http://<EXTERNAL-IP>

http://ae99c65dd23334066a26ab997e574b43-971348613.us-east-1.elb.amazonaws.com



## 5. Configure DNS in Hostinger

### Mine LoadBalancer URL

Use this (your EKS endpoint):

ae99c65dd23334066a26ab997e574b43-971348613.us-east-1.elb.amazonaws.com

### Step 1: Open Hostinger DNS

1. Log in to **Hostinger**
2. Go to:
  - **Domains**
  - Click: [harshika-devops.online](#)
3. Open:  
**DNS / Nameservers / Manage DNS**

### Step 2: Add DNS Record

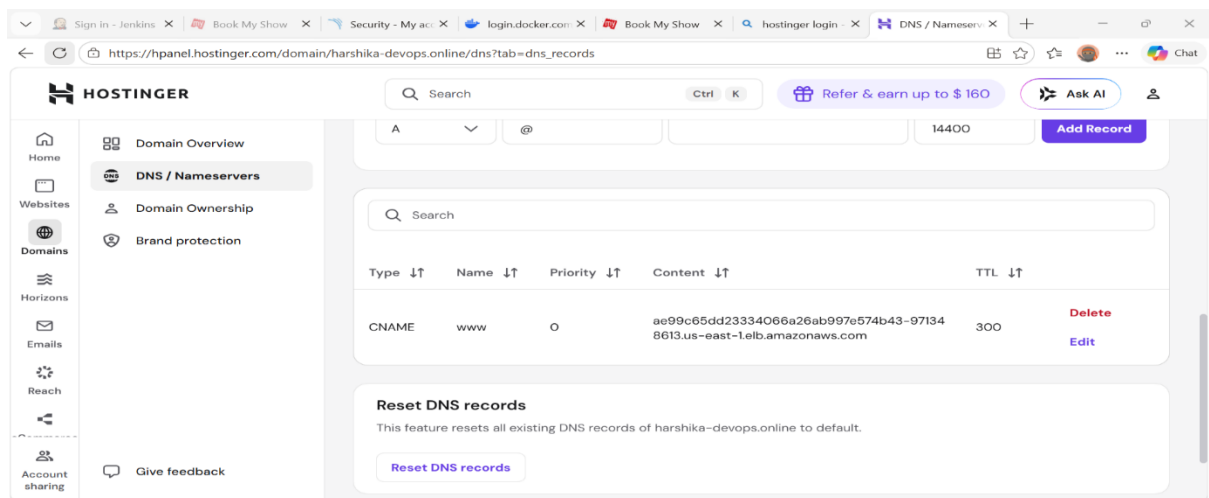
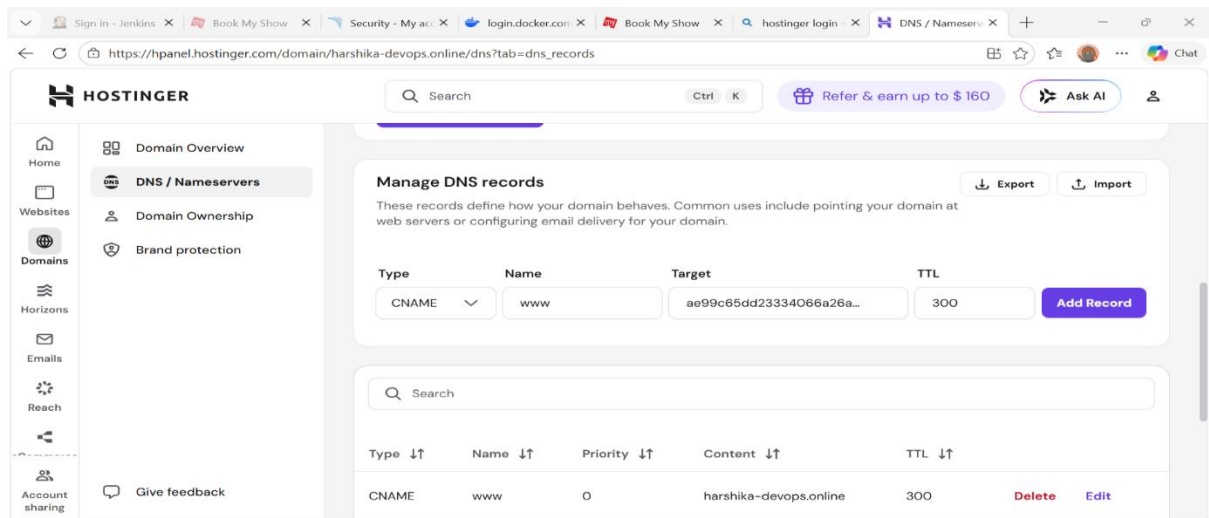
#### Add a CNAME record

| Field | Value |
|-------|-------|
|-------|-------|

|      |       |
|------|-------|
| Type | CNAME |
|------|-------|

|      |     |
|------|-----|
| Name | www |
|------|-----|

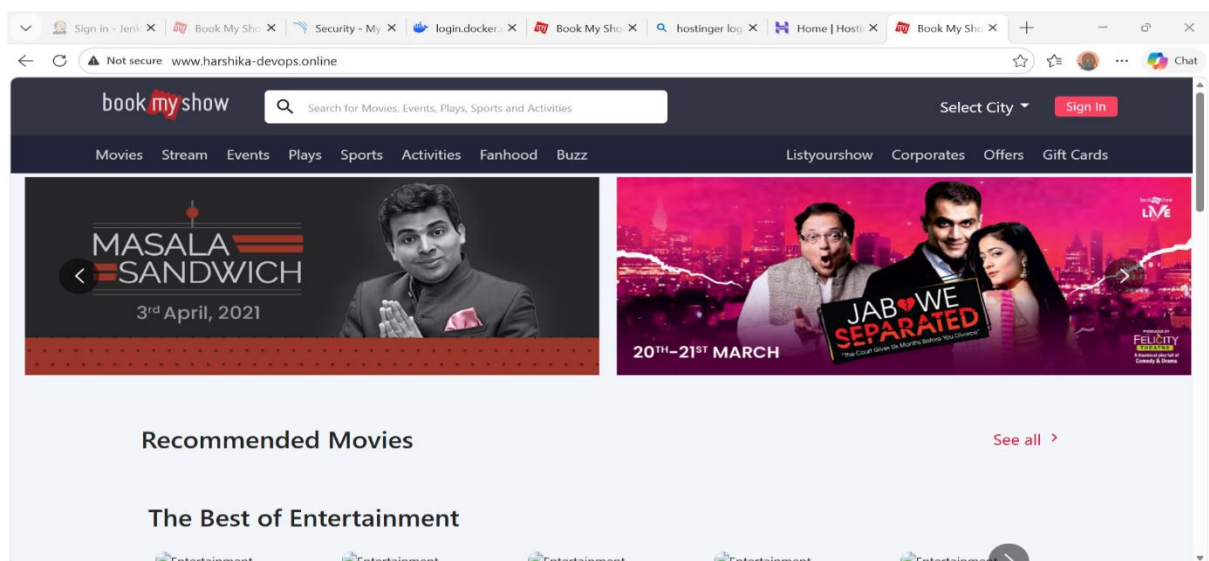
|        |                                                                        |
|--------|------------------------------------------------------------------------|
| Target | ae99c65dd23334066a26ab997e574b43-971348613.us-east-1.elb.amazonaws.com |
|--------|------------------------------------------------------------------------|



## Step 6: Test

Open:

<http://www.harshika-devops.online>



## 6. Enable webhook trigger in Jenkins

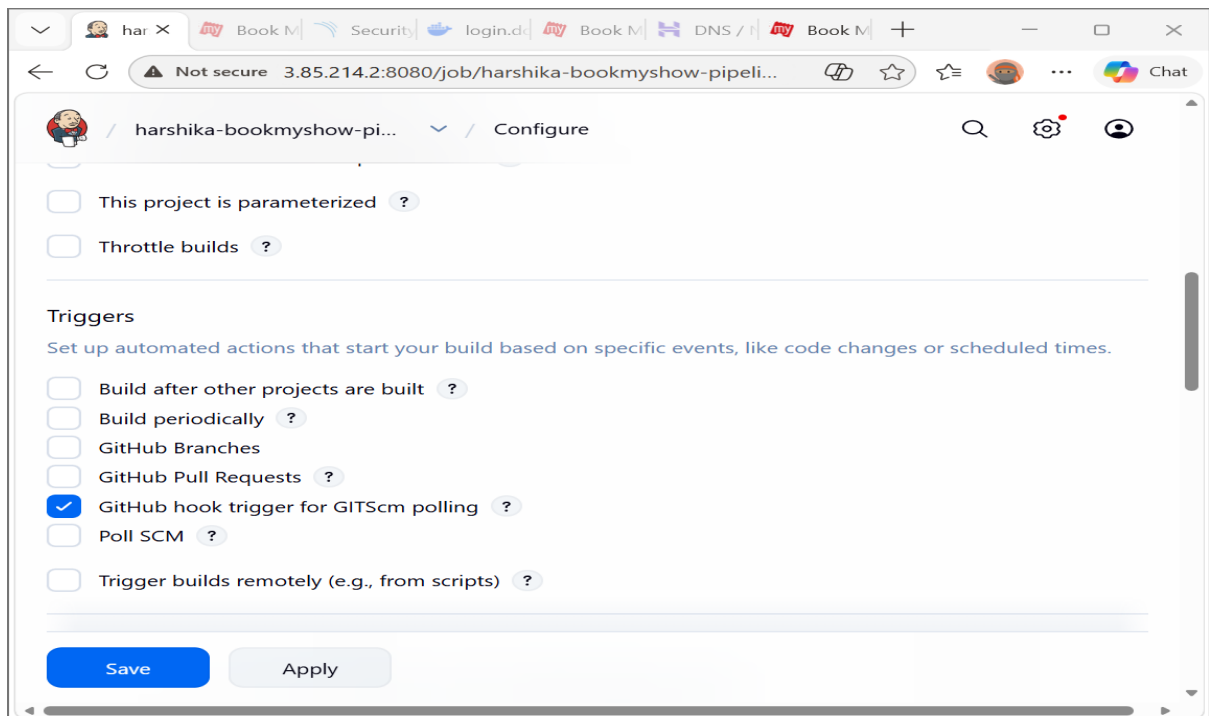
In Jenkins:

1. Open your job → **Harshika-bookmyshow-pipeline**
2. Click → **Configure**
3. Scroll to → **Build Triggers**

Enable:

GitHub hook trigger for GITScm polling

Click **Save**



## 7. Add a webhook in GitHub

Go to your repo in GitHub:

**harshika369/Book-My-Show**

Then:

1. Click → **Settings**
2. Click → **Webhooks**

### 3. Click → **Add webhook**

**Fill these fields EXACTLY**

**Payload URL:**

http://<your-jenkins-ip>:8080/github-webhook/

**Content type:**

application/json

**Events:**

Select:

Just the push event

GitHub - Add webhook

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

**Payload URL \***

http://3.85.214.2:8080/github-webhook/

**Content type \***

application/json

**Secret**

**SSL verification**

By default, we verify SSL certificates when delivering payloads.

☐ Enable SSL verification ☒ Disable (not recommended)

**Which events would you like to trigger this webhook?**

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ **Active**

We will deliver event details when this hook is triggered.

**Add webhook**

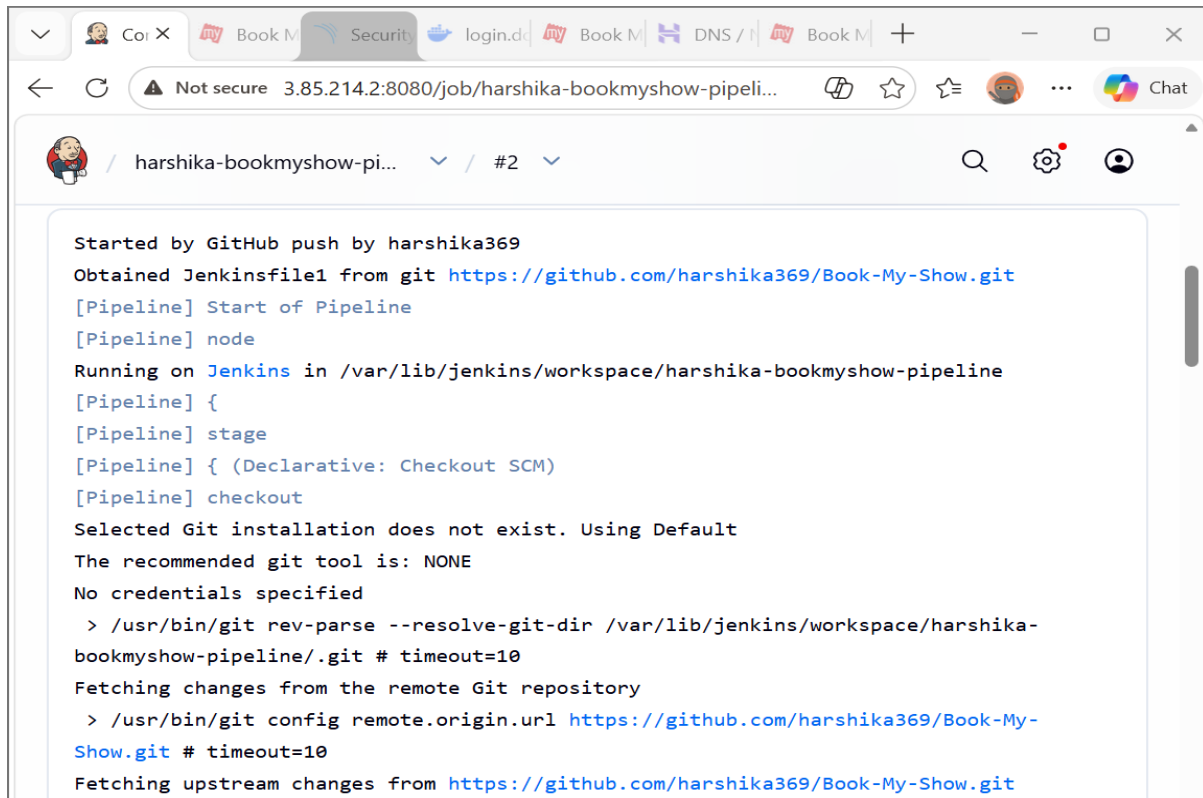
Click:

Add webhook

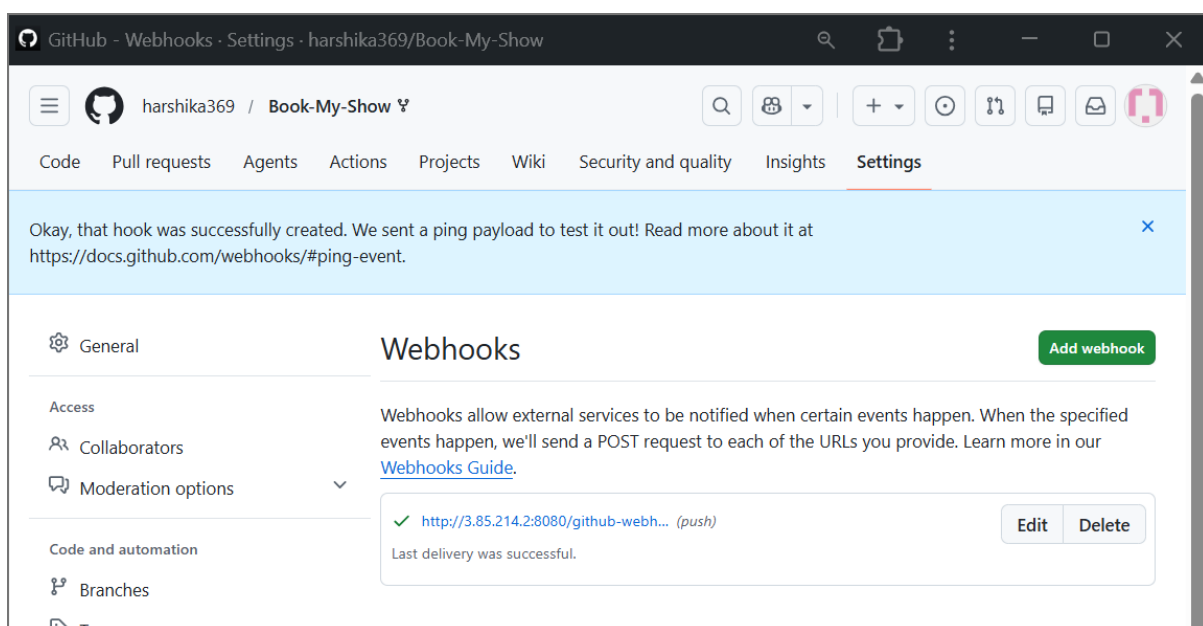
Check the Webhook:

Trigger push

Go to GitHub: your repo → edit any file → commit again



```
Started by GitHub push by harshika369
Obtained Jenkinsfile1 from git https://github.com/harshika369/Book-My-Show.git
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/harshika-bookmyshow-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> /usr/bin/git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/harshika-bookmyshow-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> /usr/bin/git config remote.origin.url https://github.com/harshika369/Book-My-Show.git # timeout=10
Fetching upstream changes from https://github.com/harshika369/Book-My-Show.git
```



GitHub - Webhooks · Settings · harshika369/Book-My-Show

harshika369 / Book-My-Show

Code Pull requests Agents Actions Projects Wiki Security and quality Insights Settings

Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at <https://docs.github.com/webhooks/#ping-event>.

### Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <http://3.85.214.2:8080/github-webh...> (push) Edit Delete

Last delivery was successful.

**In Jenkins-Server:**

## **9. Edit Dockerfile using vi**

### **Open Dockerfile**

```
cd Book-My-Show/bookmyshow-app  
vi Dockerfile
```

Paste this **final working Dockerfile**:

```
FROM node:18
```

```
WORKDIR /app
```

```
COPY package*.json ./
```

```
RUN npm install --legacy-peer-deps
```

```
COPY . .
```

```
RUN npm run build || true
```

```
EXPOSE 3000
```

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

### **Save and exit**

Press:Esc

Then type: :wq

and hit **Enter**



## 11. Configure Job in Jenkins

### In General

Check: GitHub project

Add: <https://github.com/harshika369/Book-My-Show>

### In Build Triggers

Check: GitHub hook trigger for GITScm polling

This is VERY important for auto trigger

### In Pipeline Section

**Definition:** Pipeline script from SCM

**SCM:** Git

**Repository URL:** <https://github.com/harshika369/Book-My-Show.git>

**Branch:** \*/main

**Script Path:** Jenkinsfile2

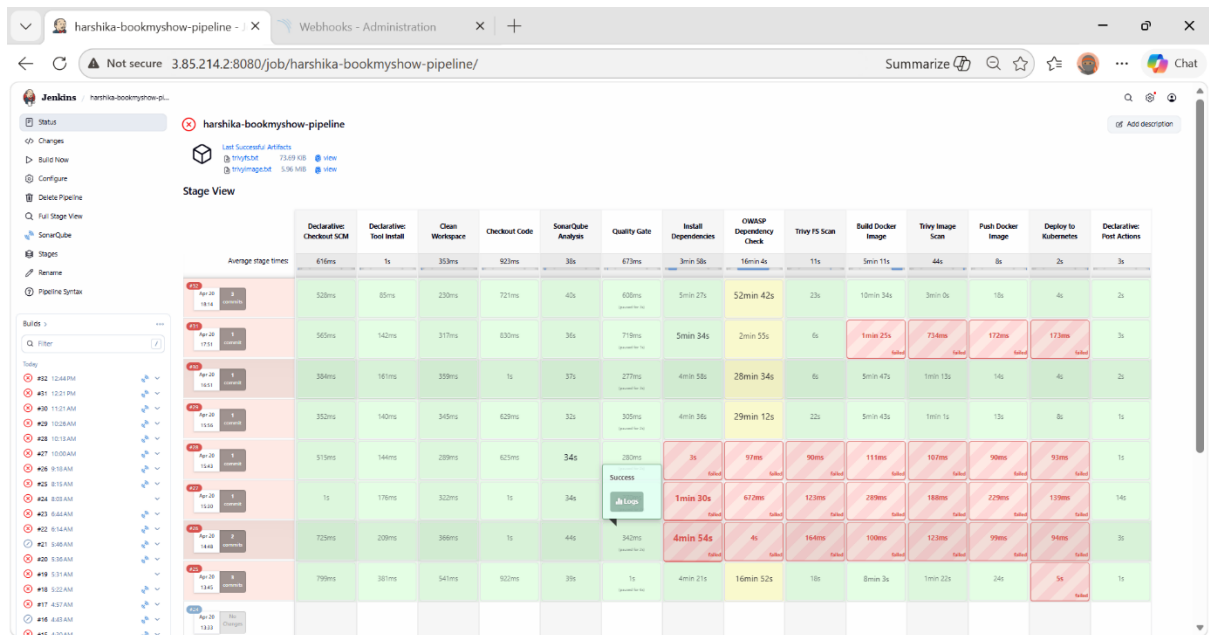
## 12. Replace the final script in JenkinsFile2 -without OWASP Stage

### Justification to Disable OWASP

“OWASP Dependency Check was integrated into the Jenkins pipeline for security scanning. However, during execution, it significantly increased build time (30–45 minutes) due to large NVD database downloads (~300k+ CVEs) and network retries.

Since the primary objective of this project is to demonstrate a complete CI/CD pipeline (build, scan, Docker, deploy to Kubernetes), OWASP was configured as a non-blocking stage / optionally skipped, to ensure pipeline reliability and timely execution.

Lightweight security scanning using Trivy is still included, which validates container and filesystem vulnerabilities efficiently.”



## File- JenkinsFile2

The image shows the GitHub repository 'Book-My-Show' with the 'Jenkinsfile2' file selected. The file content is as follows:

```

1 pipeline {
2   agent any
3
4   environment {
5     DOCKER_IMAGE = 'harshikaaa/bookmyshow:latest'
6     DOCKER_CREDENTIALS = 'docker-creds'
7   }
8
9   tools {
10    nodejs 'node18'
11  }
12
13  stages {
14    stage('Clean Workspace') {
15      steps {
16        cleanWs()
17      }
18    }
19
20    stage('Checkout Code') {
21      steps {
22        git branch: 'main', url: 'https://github.com/harshika369/Book-My-Show.git'
23      }
24    }
25  }
26
27  }

```

pipeline {

agent any

environment {

DOCKER\_IMAGE = 'harshikaaa/bookmyshow:latest'

```
    DOCKER_CREDENTIALS = 'docker-creds'
  }

  tools {
    nodejs 'node18'
  }

  stages {

    stage('Clean Workspace') {
      steps {
        cleanWs()
      }
    }

    stage('Checkout Code') {
      steps {
        git branch: 'main', url: 'https://github.com/harshika369/Book-My-Show.git'
      }
    }

    stage('SonarQube Analysis') {
      steps {
        dir('bookmyshow-app') {
```

```

script {
    def scannerHome = tool 'sonar-scanner'
    withSonarQubeEnv('sonar-server') {
        withCredentials([string(credentialsId: 'sonar-token', variable:
'SONAR_TOKEN'))] {
            sh """
                ${scannerHome}/bin/sonar-scanner \
                -Dsonar.projectName=BMS \
                -Dsonar.projectKey=bms-app \
                -Dsonar.sources=. \
                -Dsonar.host.url=$SONAR_HOST_URL \
                -Dsonar.token=$SONAR_TOKEN
            """
        }
    }
}

stage('Quality Gate') {
    steps {
        waitForQualityGate abortPipeline: true
    }
}

```

```
stage('Install Dependencies') {  
  steps {  
    dir('bookmyshow-app') {  
      sh '''  
        echo "Node version:"  
        node -v  
        npm -v  
  
        npm config set registry https://registry.npmjs.org/  
        npm config set fetch-retries 5  
        npm config set fetch-retry-mintimeout 20000  
        npm config set fetch-retry-maxtimeout 120000  
        npm config set fetch-timeout 600000  
  
        rm -rf node_modules package-lock.json  
        npm cache clean --force  
  
        npm install --legacy-peer-deps || npm install --legacy-peer-deps  
        '''  
      }  
    }  
  }  
  
  stage('Trivy FS Scan') {
```

```
steps {  
    sh 'trivy fs bookmyshow-app || true'  
}  
}
```

```
stage('Build Docker Image') {  
    steps {  
        sh 'docker build -t $DOCKER_IMAGE -f bookmyshow-app/Dockerfile  
bookmyshow-app'  
    }  
}
```

```
stage('Trivy Image Scan') {  
    steps {  
        sh 'trivy image $DOCKER_IMAGE || true'  
    }  
}
```

```
stage('Push Docker Image') {  
    steps {  
        script {  
            docker.withRegistry("", DOCKER_CREDENTIALS) {  
                sh 'docker push $DOCKER_IMAGE'  
            }  
        }  
    }  
}
```

```
}  
  
}  
  
stage('Deploy to Kubernetes') {  
    steps {  
        sh '''  
        kubectl apply -f deployment.yml  
        kubectl apply -f service.yml  
        '''  
    }  
}  
}  
  
post {  
    success {  
        emailx (   
            subject: "SUCCESS: Build Passed",  
            body: "Pipeline executed successfully 🚀",  
            to: "harshi[REDACTED]@gmail.com"  
        )  
    }  
  
    failure {  
        emailx (   
            subject: "FAILURE: Build Failed",
```

```

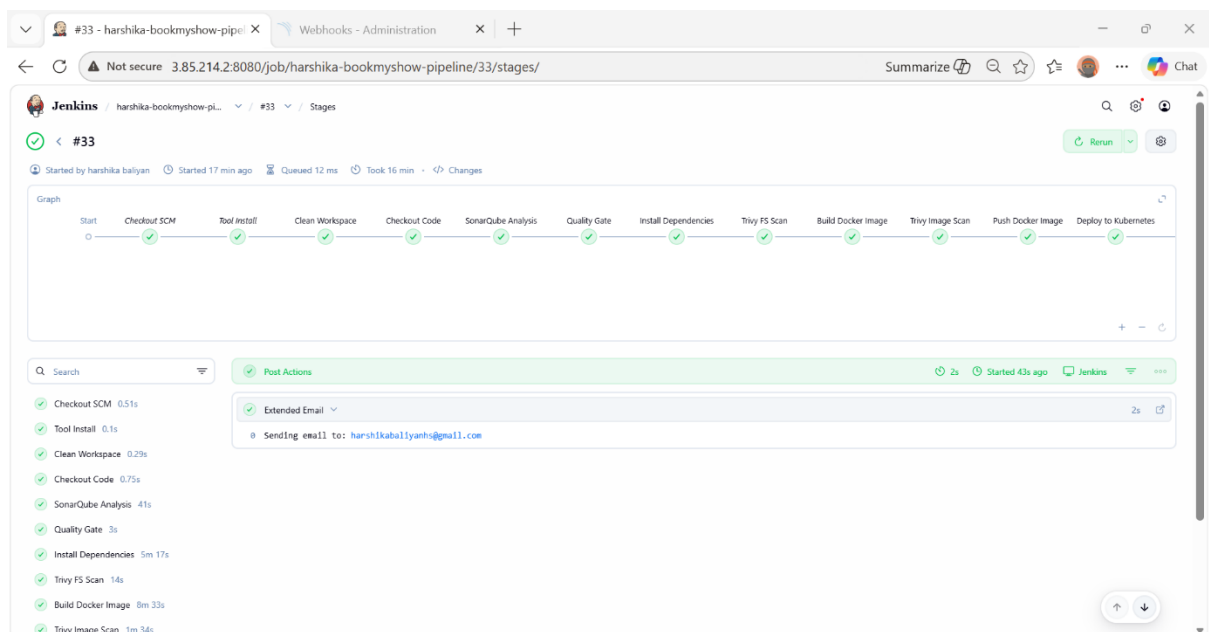
body: "Pipeline failed ❌. Please check Jenkins logs.",
to: "harshika[REDACTED]@gmail.com"
)
}
}
}
}

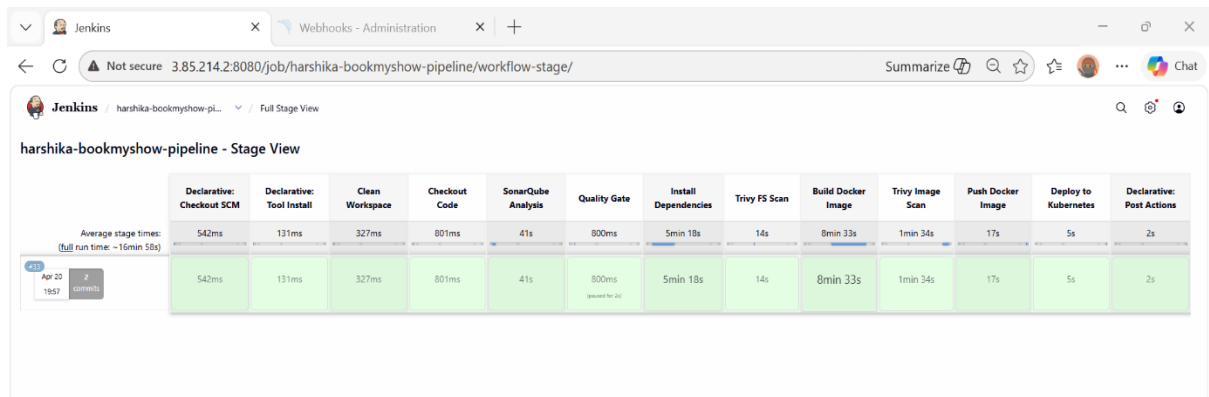
```

```

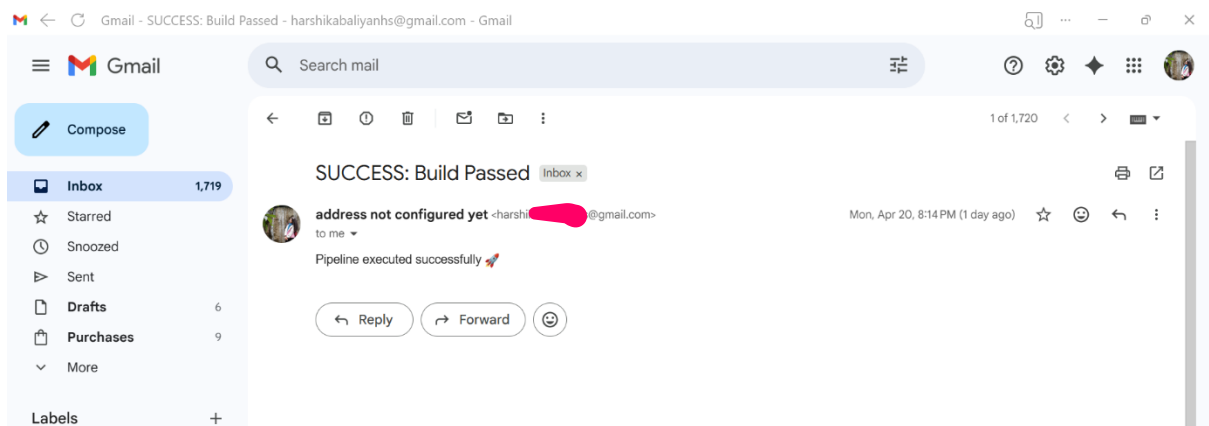
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy to Kubernetes)
[Pipeline] tool
[Pipeline] envVarsForTool
[Pipeline] withEnv
[Pipeline] {
[Pipeline] sh
+ kubectl apply -f deployment.yml
deployment.apps/bms-app unchanged
+ kubectl apply -f service.yml
service/bms-service unchanged
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] emailText
Sending email to: harshikabaliyanhs@gmail.com
[Pipeline]
[Pipeline] // stage
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // withEnv
[Pipeline]
[Pipeline] // node
[Pipeline] end of Pipeline
Finished: SUCCESS

```





## Notification of Successful Build:



Now

kubectl get nodes

kubectl get pods

```
ubuntu@ip-172-31-71-135:~$ kubectl get pods
NAME                                READY  STATUS   RESTARTS  AGE
bms-app-658cc79876-n6mhj            1/1    Running   0          14s
bms-app-c7756889d-f2h8z             1/1    Running   0          14s
bms-app-df6d7784d-tvvrX             0/1    ContainerCreating  0          13s
ubuntu@ip-172-31-71-135:~$
```

## Your App URL

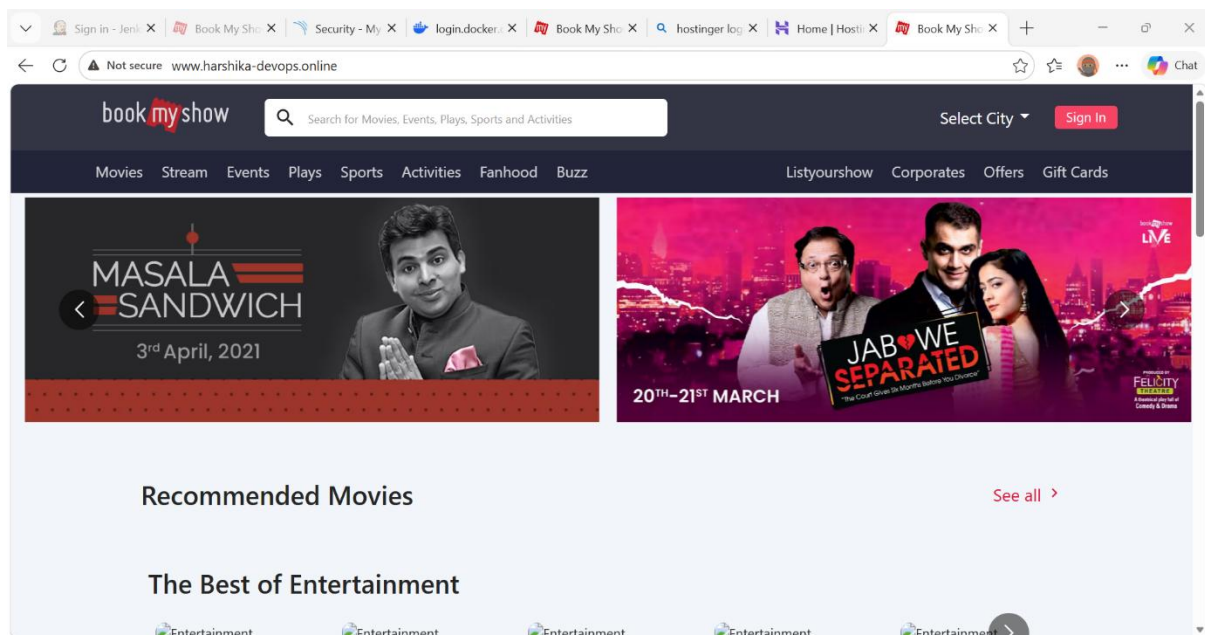
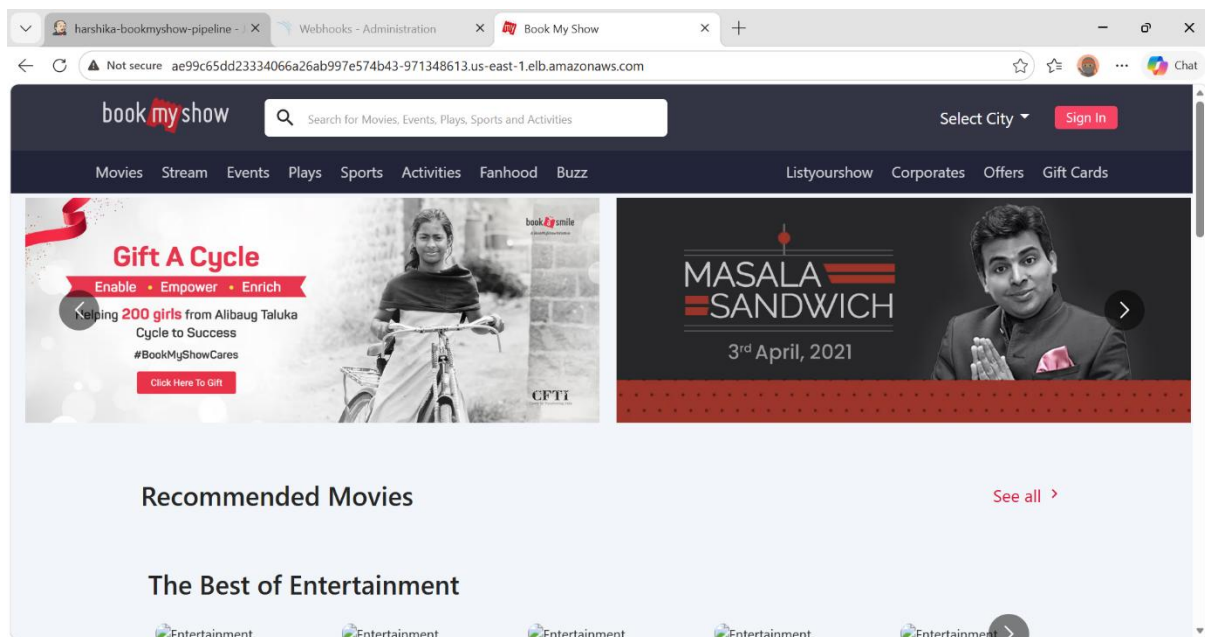
bms-service LoadBalancer

EXTERNAL-IP: ae99c65dd23334066a26ab997e574b43-971348613.us-east-1.elb.amazonaws.com

PORT: 80

## Open browser

<http://ae99c65dd23334066a26ab997e574b43-971348613.us-east-1.elb.amazonaws.com>



## Phase 4: Monitoring Setup using Prometheus & Grafana on Kubernetes

### Step 0: Prerequisites

Make sure:

kubectl get nodes

```
ubuntu@ip-172-31-71-135:~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE      VERSION
ip-192-168-35-177.ec2.internal     Ready    <none>   6m26s    v1.34.6-eks-bbe087e
ubuntu@ip-172-31-71-135:~$ kubectl get pods -n kube-system
NAME                                READY    STATUS    RESTARTS   AGE
aws-node-smdzq                      2/2      Running   0           6m36s
coredns-7d58d485c9-2wrl6           1/1      Running   0           3h35m
coredns-7d58d485c9-r9m95           1/1      Running   0           3h19m
kube-proxy-pb7gn                    1/1      Running   0           6m36s
metrics-server-6f49c4bc6c-5jw5z    1/1      Running   0           20h
metrics-server-6f49c4bc6c-9j8mh    1/1      Running   0           20h
ubuntu@ip-172-31-71-135:~$
```

### Step 1: Create Namespace

kubectl create namespace monitoring

```
ubuntu@ip-172-31-71-135:~$ kubectl create namespace monitoring
namespace/monitoring created
```

### Step 2: Create Prometheus Config

cat <<EOF > prometheus-config.yaml

apiVersion: v1

kind: ConfigMap

metadata:

name: prometheus-config

namespace: monitoring

data:

prometheus.yml: |

global:

scrape\_interval: 15s

scrape\_configs:

- job\_name: 'prometheus'

```

static_configs:
  - targets: ['localhost:9090']

- job_name: 'node-exporter'
  static_configs:
    - targets: ['node-exporter.monitoring.svc.cluster.local:9100']
EOF

```

EOF

Apply:

kubectl apply -f prometheus-config.yaml

```

ubuntu@ip-172-31-71-135:~$ cat <<EOF > prometheus.yml
global:
  scrape_interval: 30s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
EOF
ubuntu@ip-172-31-71-135:~$ kubectl create configmap prometheus-config \
--from-file=prometheus.yml \
-n monitoring
configmap/prometheus-config created
ubuntu@ip-172-31-71-135:~$ cat <<EOF > prometheus-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
  namespace: monitoring
spec:

```

### Step 3: Deploy Prometheus

cat <<EOF > prometheus-deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

name: prometheus

namespace: monitoring

spec:

replicas: 1

selector:

matchLabels:

app: prometheus

template:

```
metadata:
  labels:
    app: prometheus
spec:
  containers:
  - name: prometheus
    image: prom/prometheus
    args:
      - "--config.file=/etc/prometheus/prometheus.yml"
    ports:
      - containerPort: 9090
  resources:
    requests:
      memory: "100Mi"
      cpu: "100m"
    limits:
      memory: "200Mi"
      cpu: "200m"
  volumeMounts:
  - name: config
    mountPath: /etc/prometheus
  volumes:
  - name: config
    configMap:
      name: prometheus-config
```

EOF

Apply:

kubectl apply -f prometheus-deployment.yaml

```
ubuntu@ip-172-31-71-135:~$ cat <<EOF > prometheus-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
  namespace: monitoring
spec:
```

## Step 4: Expose Prometheus

```
kubectl expose deployment prometheus \
  --type=LoadBalancer \
  --port=9090 \
  -n monitoring
```

```
ubuntu@ip-172-31-71-135:~$ kubectl expose deployment prometheus \
  --type=LoadBalancer \
  --port=9090 \
  --name=prometheus-lb \
  -n monitoring
service/prometheus-lb exposed
ubuntu@ip-172-31-71-135:~$ kubectl get svc -n monitoring
NAME                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
prometheus          NodePort            10.100.183.191  <none>           9090:30930/TCP   28m
prometheus-lb       LoadBalancer       10.100.118.179  aaf45a575143046a08bcd0f0f49802
4d-808355246.us-east-1.elb.amazonaws.com  9090:32389/TCP   13s
ubuntu@ip-172-31-71-135:~$
```

## Step 5: Deploy Node Exporter

```
cat <<EOF > node-exporter.yaml
```

```
apiVersion: apps/v1
```

```
kind: DaemonSet
```

```
metadata:
```

```
  name: node-exporter
```

```
  namespace: monitoring
```

```
spec:
```

```
  selector:
```

```
    matchLabels:
```

```
      app: node-exporter
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: node-exporter
```

```
    spec:
```

```
      containers:
```

```
        - name: node-exporter
```

```
          image: prom/node-exporter:v1.8.1
```

```
ports:
  - containerPort: 9100
resources:
  requests:
    memory: "30Mi"
    cpu: "20m"
  limits:
    memory: "50Mi"
    cpu: "50m"
```

EOF

Apply:

```
kubectl apply -f node-exporter.yaml
```

### **Step 6: Node Exporter Service**

```
kubectl expose daemonset node-exporter \
  --type=ClusterIP \
  --port=9100 \
  -n monitoring
```

### **Step 7: Deploy Grafana**

```
cat <<EOF > grafana.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grafana
  namespace: monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      app: grafana
  template:
    metadata:
      labels:
```

```
  app: grafana
spec:
  containers:
  - name: grafana
    image: grafana/grafana:10.4.2
    ports:
    - containerPort: 3000
  resources:
    requests:
      memory: "80Mi"
      cpu: "50m"
    limits:
      memory: "150Mi"
      cpu: "100m"
```

---

```
apiVersion: v1
kind: Service
metadata:
  name: grafana
  namespace: monitoring
spec:
  type: LoadBalancer
  selector:
    app: grafana
  ports:
  - port: 3000
    targetPort: 3000
```

EOF

Apply:

```
kubectrl apply -f grafana.yaml
```

```
ubuntu@ip-172-31-71-135: ~  
ports:  
- containerPort: 3000  
resources:  
  requests:  
    memory: "80Mi"  
    cpu: "50m"  
  limits:  
    memory: "150Mi"  
    cpu: "100m"  
---  
apiVersion: v1  
kind: Service  
metadata:  
  name: grafana  
  namespace: monitoring  
spec:  
  type: LoadBalancer  
  selector:  
    app: grafana  
  ports:  
    - port: 3000  
      targetPort: 3000  
EOF  
ubuntu@ip-172-31-71-135:~$ kubectl apply -f grafana.yaml  
deployment.apps/grafana created  
service/grafana created  
ubuntu@ip-172-31-71-135:~$ kubectl get pods -n monitoring  
NAME                                READY   STATUS    RESTARTS   AGE  
grafana-9597d5b7d-ggdkb             0/1     Pending   0           8s  
prometheus-6d77d9fb8b-mxjbz        1/1     Running   0          19m  
ubuntu@ip-172-31-71-135:~$ kubectl taint nodes --all node.kubernetes.io/memory-pressure-  
kubectl delete pod -n monitoring -l app=grafana  
node/ip-192-168-35-177.ec2.internal untainted  
pod "grafana-9597d5b7d-ggdkb" deleted from monitoring namespace  
ubuntu@ip-172-31-71-135:~$ kubectl get svc -n monitoring  
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP  
PORT(S)                            AGE  
grafana                             LoadBalancer       10.100.106.7     a5d3c6779c34744f6b1a2b6fa4aff0c9-1221203744.us-east-1.elb.amazonaws.com  
3000:30928/TCP                     43s  
prometheus                          NodePort            10.100.183.191   <none>  
9090:30930/TCP                     62m  
prometheus-lb                      LoadBalancer       10.100.118.179   aaf45a575143046a08bcd0f0f498024d-808355246.us-east-1.elb.amazonaws.com  
9090:32389/TCP                     34m  
ubuntu@ip-172-31-71-135:~$ kubectl get pods -n monitoring  
NAME                                READY   STATUS    RESTARTS   AGE  
grafana-9597d5b7d-5pgfv             1/1     Running   0           52s  
prometheus-6d77d9fb8b-mxjbz        1/1     Running   0          21m  
ubuntu@ip-172-31-71-135:~$
```

## Step 8: Verify Pods

`kubectl get pods -n monitoring`

Expected:

grafana      Running

prometheus   Running

node-exporter Running

## Step 9: Get External URLs

`kubectl get svc -n monitoring`

Use:

- Prometheus → `http://<EXTERNAL-IP>:9090`
- Grafana → `http://<EXTERNAL-IP>:3000`

```
ubuntu@ip-172-31-71-135:~$ kubectl get svc -n monitoring
NAME                TYPE          CLUSTER-IP      EXTERNAL-IP
PORT(S)            AGE
grafana             LoadBalancer  10.100.106.7     a5d3c6779c34744f6b1a2b6fa4aff0c9-1221203744.us-east-1.elb.amazonaws.com
3000:30928/TCP      43s
prometheus          NodePort      10.100.183.191   <none>
9090:30930/TCP      62m
prometheus-lb       LoadBalancer  10.100.118.179   aaf45a575143046a08bcd0f0f498024d-808355246.us-east-1.elb.amazonaws.com
9090:32389/TCP      34m
ubuntu@ip-172-31-71-135:~$ kubectl get pods -n monitoring
NAME                                READY   STATUS    RESTARTS   AGE
grafana-9597d5b7d-5pgfv            1/1     Running   0           52s
prometheus-6d77d9fb8b-mxjbz       1/1     Running   0           21m
ubuntu@ip-172-31-71-135:~$
```

## Step 10: Verify Prometheus

Open:

<http://<PROMETHEUS-IP>:9090/targets>

Check:

- prometheus → UP
- node-exporter → UP

Run query:

up

The screenshot shows the Prometheus web interface at the URL `aaf45a575143046a08bcd0f0f498024d-808355246.us-east-1.elb.amazonaws.com:9090/targets`. The page title is 'Prometheus' and the navigation bar includes 'Query', 'Alerts', and 'Status > Target health'. Below the navigation bar, there are filters for 'Select scrape pool', 'Filter by target health', and 'Filter by endpoint or labels'. The main content area displays two target groups:

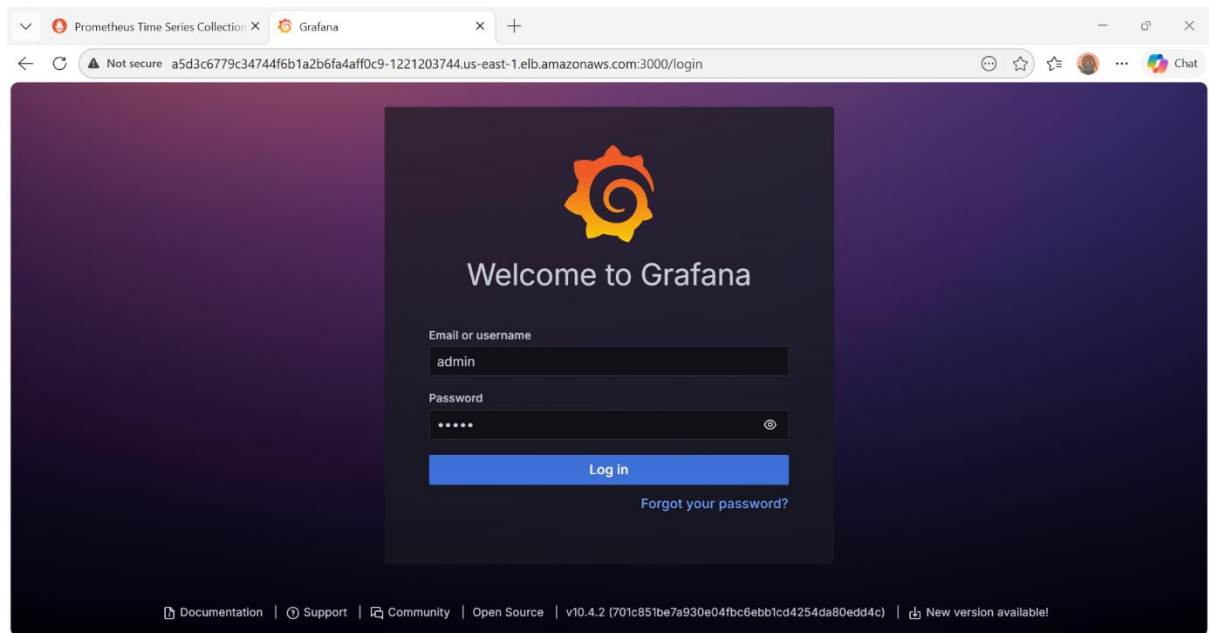
- node-exporter**: 1 / 1 up. The table shows one target with endpoint `http://node-exporter.monitoring.svc.cluster.local:9100/metrics`, labels `instance="node-exporter.monitoring.svc.cluster.local:9100"` and `job="node-exporter"`, last scrape at 11.91s ago, and a state of 'UP'.
- prometheus**: 1 / 1 up. The table shows one target with endpoint `http://localhost:9090/metrics`, labels `instance="localhost:9090"` and `job="prometheus"`, last scrape at 5.397s ago, and a state of 'UP'.

## Step 11: Configure Grafana

Login:

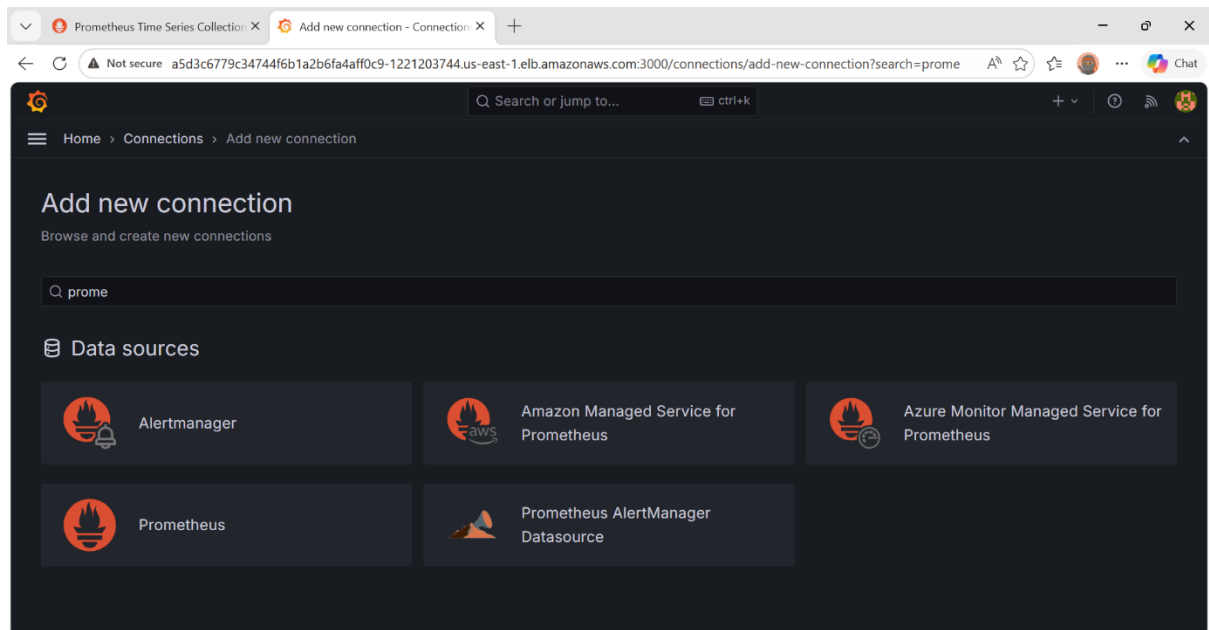
username: admin

password: admin



## Add Data Source

- Type: Prometheus

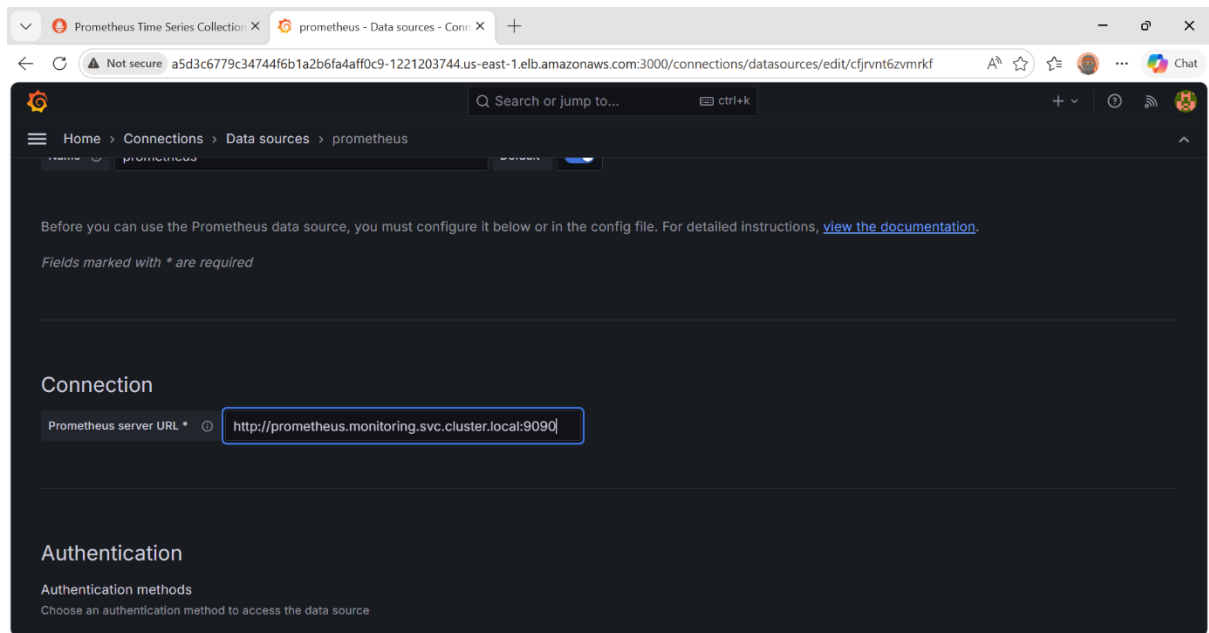


URL:

http://prometheus.monitoring.svc.cluster.local:9090

Click:

Save & Test



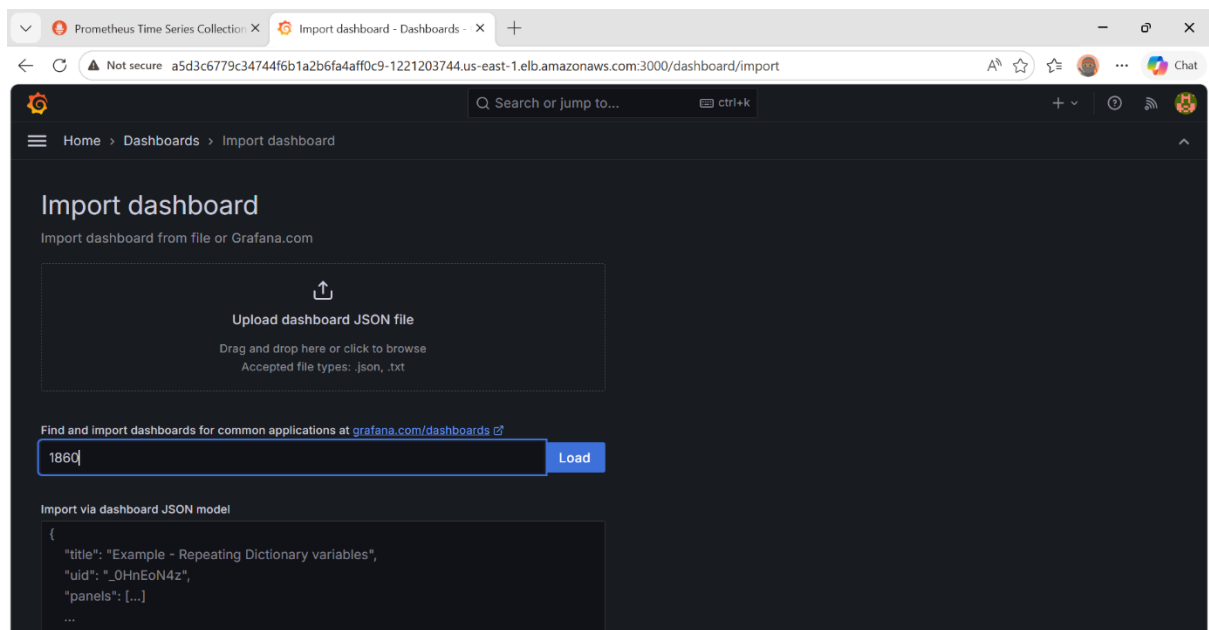
## Step 12: Import Dashboard

Go to:

- Dashboards → Import

Enter ID:

1860



Importing dashboard from [Grafana.com](#)

Published by rfmoz

Updated on 2026-04-11 13:45:40

### Options

Name

Node Exporter Full

Folder

Dashboards

Unique identifier (UID)

The unique identifier (UID) of a dashboard can be used to uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

rYddIPWk [Change uid](#)

[Import](#) [Cancel](#)

Select:

- Data Source → Prometheus

Other

Custom query parameters  [ⓘ](#)  Example: max\_source\_resolution=5m&timeout

HTTP method  [ⓘ](#)  POST

Exemplars

[+ Add](#)

✓ Successfully queried the Prometheus API.

Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#).

[Delete](#) [Save & test](#)

## Step 13: Final Output

Dashboard shows:

- CPU usage
- Memory usage
- Disk usage
- Network stats

