

Project-2 Deployment of Online Ticketing and Event Management Applications and Infrastructure.

Submitted By – Arya Chaurasia

Date of Submission – 14/12/2025

Project Overview

BookMyShow is a popular online movie ticketing platform that allows users browse movies, book tickets, and manage their bookings. The company is transitioning from a monolithic architecture to a microservices-based architecture to improve scalability, deployment efficiency, and infrastructure management.

GitHub link - <https://github.com/Aryachaurasia23/Book-My-Show>

Steps to Deploy the Website successfully

Step-1- We will create an EC2 Instance BMS-Server (Ubuntu 24.04,t2.large) with EBS storage 28 Gib.

The screenshot shows the AWS EC2 Instances page. The left sidebar has sections for EC2, Instances, Images, and Elastic Block Store. The main area displays 'Instances (2) Info' with a table:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4
freelance	i-0989b11828972e8c9	Terminated	t2.micro	-	View alarms +	us-east-1a	-
BMS-Server	i-000a13c9efd6fe710	Running	t2.large	Initializing	View alarms +	us-east-1a	ec2-35-175-

Below the table, there's a section titled 'Select an instance' with a dropdown menu. The bottom of the screen shows the AWS navigation bar with CloudShell, Feedback, and Console Mobile App, along with various system icons like weather, language, and date/time.

Step-2-Opened below ports for the security group of the Bms-Server – 22, 80, 3000-10000, 443, 6443, 30000-32767.

Step-3- We installed Jenkins and JAVA and validated if they are installed or not.

```

ubuntu@ip-172-31-29-69:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
  Active: active (running) since Fri 2025-12-12 21:12:29 UTC; 52s ago
    Main PID: 3296 (java)
       Tasks: 50 (limit: 9498)
      Memory: 830.5M (peak: 848.0M)
        CPU: 15.684s
       CGroup: /system.slice/jenkins.service
           └─3296 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Dec 12 21:12:26 ip-172-31-29-69 jenkins[3296]: [LF]> This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Dec 12 21:12:26 ip-172-31-29-69 jenkins[3296]: [LF]>
Dec 12 21:12:26 ip-172-31-29-69 jenkins[3296]: [LF]> ****
Dec 12 21:12:29 ip-172-31-29-69 jenkins[3296]: 2025-12-12 21:12:29.964+0000 [id=30]      INFO      jenkins.InitReactorRunner$1#onAttained: Completed initialization
Dec 12 21:12:29 ip-172-31-29-69 jenkins[3296]: 2025-12-12 21:12:29.978+0000 [id=23]      INFO      hudson.lifecycle.Lifecycle$onReady: Jenkins is fully up and running
Dec 12 21:12:29 ip-172-31-29-69 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Dec 12 21:12:30 ip-172-31-29-69 jenkins[3296]: 2025-12-12 21:12:30.139+0000 [id=49]      INFO      h.m.DownloadService$Downloadable#load: Obtained the updated data
Dec 12 21:12:30 ip-172-31-29-69 jenkins[3296]: 2025-12-12 21:12:30.139+0000 [id=49]      INFO      hudson.util.Retriger$start: Performed the action check updates set
ubuntu@ip-172-31-29-69:~$ java --version
openjdk 17.0.17 2025-10-21
OpenJDK Runtime Environment (build 17.0.17+10-Ubuntu-124.04)
OpenJDK 64-Bit Server VM (build 17.0.17+10-Ubuntu-124.04, mixed mode, sharing)
ubuntu@ip-172-31-29-69:~$ i-000a13c9efd6fe710 (BMS-Server)
Public IPs: 35.175.196.177 Private IPs: 172.31.29.69

```

Step-4- we opened Jenkins with <Bms-server IP> : 8080

The screenshot shows the Jenkins dashboard. The URL in the address bar is 35.175.196.177:8080. The dashboard features a 'Welcome to Jenkins!' message, a 'Build Queue' section showing 'No builds in the queue.', and a 'Build Executor Status' section indicating '0/2'. There are buttons for 'Create a job' and 'Set up a distributed build'. The bottom of the screen shows a Windows taskbar with various icons and system status.

Step-5 Installed docker and giving permission to pull the images.

The terminal window shows the following commands and output:

```
ubuntu@ip-172-31-29-69:~$ docker --version
Docker version 29.1.3, build f52814d
ubuntu@ip-172-31-29-69:~$ docker pull hello-world
Using default tag: latest
permission denied while trying to connect to the docker API at unix:///var/run/docker.sock
ubuntu@ip-172-31-29-69:~$ sudo chmod 666 /var/run/docker.sock
ubuntu@ip-172-31-29-69:~$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
eaa52d2000f90: Download complete
Digest: sha256:d4aab6242e0cace87e2ec17a2ed3d779d18fbfd03042ea58f2995626396a274
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest
ubuntu@ip-172-31-29-69:~$ docker images
```

A table of Docker images is shown:

IMAGE	ID	DISK USAGE	CONTENT SIZE	EXTRA
hello-world:latest	d4aab6242e0	25.9kB	9.52kB	

The AWS CloudWatch interface for the BMS-Server instance (i-000a13c9ef6fe710) is also visible, showing public and private IP addresses.

Step-6- To create EKS Cluster, its not recommended to create using Root account so we created an IAM user – eks-arya-user

The screenshot shows the AWS IAM service in a web browser. The left sidebar has 'Identity and Access Management (IAM)' selected under 'Access management'. The main area displays a table titled 'Users (1)'. The table has one row for 'eks-arya-user', which is highlighted. The columns include 'User name', 'Path', 'Group', 'Last activity', 'MFA', 'Password age', 'Console last sign-in', and 'Access key last used'. The 'eks-arya-user' row shows a path of '/', group '0', last activity 'Now', and MFA status 'None'. The table header includes filters for 'Search', 'User name', 'Path', 'Group', 'Last activity', 'MFA', 'Password age', 'Console last sign-in', and 'Access key last used'. There are 'Create user' and 'Delete' buttons at the top right of the table.

Step-7- We attached these policies to the User.

The screenshot shows the 'Add permissions' step for the 'eks-arya-user'. The top navigation bar shows the URL 'us-east-1.console.aws.amazon.com/iam/home?region=us-east-1#/users/details/eks-arya-user/add-permissions'. The left sidebar shows 'Step 2' and 'Review'. The main area has a 'User details' section with the user name 'eks-arya-user'. Below it is a 'Permissions summary (6)' table. The table lists six AWS managed policies: 'AmazonEC2FullAccess', 'IAMFullAccess', 'AmazonEKS_CNI_Policy', 'AmazonEKSClusterPolicy', 'AmazonEKSWorkerNodePolicy', and 'AWSCloudFormationFullAccess'. The columns are 'Name', 'Type', and 'Used as'. At the bottom right of the table, there are 'Cancel', 'Previous', and 'Add permissions' buttons. The 'Add permissions' button is circled in red.

Step-8- Attached this inline policy for the same user.

The screenshot shows the AWS IAM Policy editor interface. On the left, a sidebar indicates "Step 1 Specify permissions" is selected. The main area is titled "Specify permissions" with a "Info" link. Below it, a note says "Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor." A "Policy editor" window is open, showing the following JSON code:

```
1 "Version": "2012-10-17",
2 "Statement": [
3     {
4         "Sid": "VisualEditor0",
5         "Effect": "Allow",
6         "Action": "eks:*",
7         "Resource": "*"
8     }
9 ]
10 ]
11 ]
```

To the right of the editor is a panel titled "Edit statement" with a "Select a statement" button and a note: "Select an existing statement in the policy or add a new statement." A "Add new statement" button is also present. The bottom of the screen shows the AWS navigation bar and a taskbar with various icons and weather information.

Step-9- We attached total 7 policies.

The screenshot shows the AWS IAM Policies page. The left sidebar lists "Identity and Access Management (IAM)" and "Access management" (with "Users" selected). A green banner at the top states "Policy eks-inline-policy created." Below it, a message says "Permissions are defined by policies attached to the user directly or through groups." A table displays the attached policies:

Policy name	Type	Attached via
AmazonEC2FullAccess	AWS managed	Directly
AmazonEKS_CNI_Policy	AWS managed	Directly
AmazonEKSClusterPolicy	AWS managed	Directly
AmazonEKWorkerNodePolicy	AWS managed	Directly
AWSCloudFormationFullAccess	AWS managed	Directly
eks-inline-policy	Customer inline	Inline
IAMFullAccess	AWS managed	Directly

At the bottom, a section titled "Permissions boundary (not set)" is shown. The bottom of the screen features the AWS navigation bar and a taskbar with various icons and weather information.

Step-10- Now we generated the access key for the User.

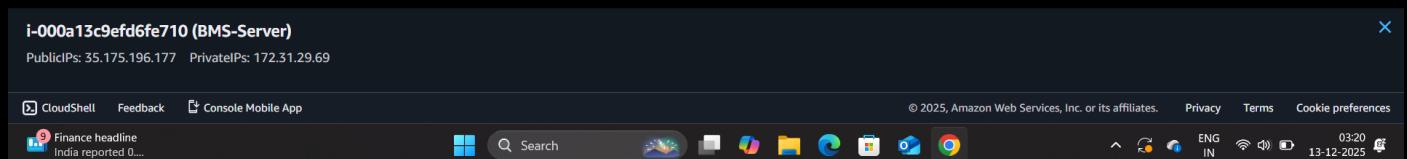
The screenshot shows the 'Create access key' page in the AWS IAM console. The user is at Step 3: 'Retrieve access keys'. A green banner at the top states: 'This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below this, the 'Access key' section displays the Access Key ID 'AKIASGDX3EU4572BNMH6' and the Secret Access Key, which is shown as a series of asterisks ('*****') with a 'Show' link. The 'Access key best practices' section lists several guidelines. At the bottom right, there are 'Download .csv file' and 'Done' buttons.

Step-11- Installed AWS CLI to interact with AWS account.

The screenshot shows a terminal session on an AWS Lambda instance. The user has run the command 'aws --version', which outputs: 'aws-cli/2.32.16 Python/3.13.11 Linux/6.14.0-1015-aws exe/x86_64.glibc.2.14'. The terminal window also displays the instance ID 'i-000a13c9efd6fe710' and its location '(BMS-Server)'. The AWS Lambda interface shows the instance's public and private IP addresses. The bottom of the screen shows the AWS CloudShell interface and standard Windows taskbar icons.

Step-12- Configuring AWS and installing KubeCTL to interact with K8S.

```
ubuntu@ip-172-31-29-69:~$ aws configure
AWS Access Key ID [None]: AKIASGDX3EU4S72BNMH6
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
ubuntu@ip-172-31-29-69:~$ vi kubectl.sh
ubuntu@ip-172-31-29-69:~$ cat kubectl.sh
curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin
kubectl version --short --client
ubuntu@ip-172-31-29-69:~$
```



Step-13- Installed plugins in Jenkins – Eclipse Temurian Installer, NodeJS, Docker, Docker commons, Docker pipeline, Docker API, docker-build-step, Pipeline stage view, K8S, K8S CLI, K8S Client API, K8S Credentials, Config file provider, Prometheus metrics.

The screenshot shows the Jenkins plugin manager interface. On the left, there's a sidebar with links for Updates, Available plugins, Installed plugins, Advanced settings, and Download progress. The main area lists installed plugins with status indicators (green checkmark for success, orange warning for Prometheus metrics). A message at the bottom indicates that the Prometheus plugin needs a restart for updates to take effect.

Plugin	Status
Docker API	Success
Dev Tools Symbols API	Success
Javadoc	Success
JSch dependency	Success
Maven Integration	Success
docker-build-step	Success
Kubernetes Client API	Success
Kubernetes Credentials	Success
Kubernetes	Success
Kubernetes CLI	Success
Kubernetes Credentials Provider	Success
Config File Provider	Success
Prometheus metrics	Warning: prometheus plugin doesn't support dynamic loading. Jenkins needs to be restarted for the update to take effect.
Loading plugin extensions	Running

Step-14- Managed tools for jdk

The screenshot shows the Jenkins interface for managing tools. A new JDK tool is being configured with the name "jdk17". The "Install automatically" checkbox is checked, and the "Install from adoptium.net" section is expanded, showing a dropdown menu set to "jdk-17.0.8.1+1". There are buttons for "+ Add Installer" and "+ Add JDK". At the bottom are "Save" and "Apply" buttons.

Step-15 Managed tool for NodeJS

The screenshot shows the Jenkins interface for managing tools. A new NodeJS tool is being configured with the name "node23". The "Install automatically" checkbox is checked, and the "Install from nodejs.org" section is expanded, showing a dropdown menu set to "NodeJS 23.7.0". A note states: "For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail" with an unchecked "Force 32bit architecture" checkbox. A section for "Global npm packages to install" is present with a note: "Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax 'packageName@version'". At the bottom are "Save" and "Apply" buttons.

Step-16- Managed tool for docker and clicked Save.

The screenshot shows the Jenkins 'Manage Jenkins' interface with the 'Tools' configuration page selected. A new Docker tool is being configured with the name 'docker'. The 'Install automatically' checkbox is checked. Under the 'Download from docker.com' section, the 'Docker version' is set to 'latest'. A green success message at the bottom indicates the configuration was saved successfully.

Step-17- We gave credentials for docker to Jenkins

The screenshot shows the Jenkins 'Manage Jenkins' interface with the 'Credentials' configuration page selected. A new global credential for Docker is being updated. The 'Scope' is set to 'Global'. The 'Username' is 'arya023' and the 'ID' is 'docker'. The 'Description' is 'docker-creds'. The 'Save' button is visible at the bottom.

Step-18- Here we created the EKS Cluster and it had taken sometime.

```
2025-12-12 22:32:46 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:33:46 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:34:46 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:35:46 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:36:46 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:37:46 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:38:46 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:39:46 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:40:47 [i] waiting for CloudFormation stack "eksctl-bms-eks-cluster"
2025-12-12 22:40:48 [i] recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl cannot configure the requested permission
s; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identity associations; after addon creation is completed, add all recommended policies t
o the config file, under 'addon.PodIdentityAssociations', and run 'eksctl update addon'
2025-12-12 22:40:48 [i] creating addon: vpc-cni
2025-12-12 22:40:48 [i] successfully created addon: vpc-cni
2025-12-12 22:40:48 [i] creating addon: kube-proxy
2025-12-12 22:40:49 [i] successfully created addon: kube-proxy
2025-12-12 22:40:49 [i] creating addon: coredns
2025-12-12 22:40:49 [i] successfully created addon: coredns
2025-12-12 22:42:49 [i] waiting for the control plane to become ready
2025-12-12 22:42:50 [v] saved kubeconfig as "/home/ubuntu/.kube/config"
2025-12-12 22:42:50 [i] no tasks
2025-12-12 22:42:50 [v] all EKS cluster resources for "bms-eks" have been created
2025-12-12 22:42:50 [i] creating addon: metrics-server
2025-12-12 22:42:51 [i] successfully created addon: metrics-server
2025-12-12 22:42:52 [i] kubectl command should work with "/home/ubuntu/.kube/config", try 'kubectl get nodes'
2025-12-12 22:42:52 [v] EKS cluster "bms-eks" in "us-east-1" region is ready
ubuntu@ip-172-31-29-69:~$
```

Step-19- After sometime, we verified that our cluster was made.

Step-20- Here we created nodes for our EKS Cluster. Note – in ssh-public-key, give only key name not .pem

```
ubuntu@ip-172-31-29-69:~$ eksctl create nodegroup --cluster=bms-eks \
    --region=us-east-1 \
    --name=node2 \
    --node-type=t3.medium \
    --nodes=3 \
    --nodes-min=2 \
    --nodes-max=4 \
    --node-volume-size=20 \
    --ssh-access \
    --ssh-public-key=mynewkey \
    --managed \
    --asg-access \
    --external-dns-access \
    --full-ecr-access \
    --appmesh-access \
    --alb-ingress-access
```

Step-21- It took 7-10 minutes and after that our nodes were created.

The screenshot shows the AWS EKS console interface. The left sidebar navigation includes 'Amazon Elastic Kubernetes Service' (selected), 'Clusters' (selected), 'Settings' (Dashboard settings, Console settings), 'Amazon EKS Anywhere' (Enterprise Subscriptions), and 'Related services' (Amazon ECR, AWS Batch). The main content area displays the 'Clusters' section for the 'bms-eks' cluster.

Node groups (1)

Node groups implement basic compute scaling through EC2 Auto Scaling groups.

Group name	Desired size	AMI release version	Launch template	Status
node2	3	1.30.14-20251209	eksctl-bms-eks-nodegroup-node2 (1)	Active

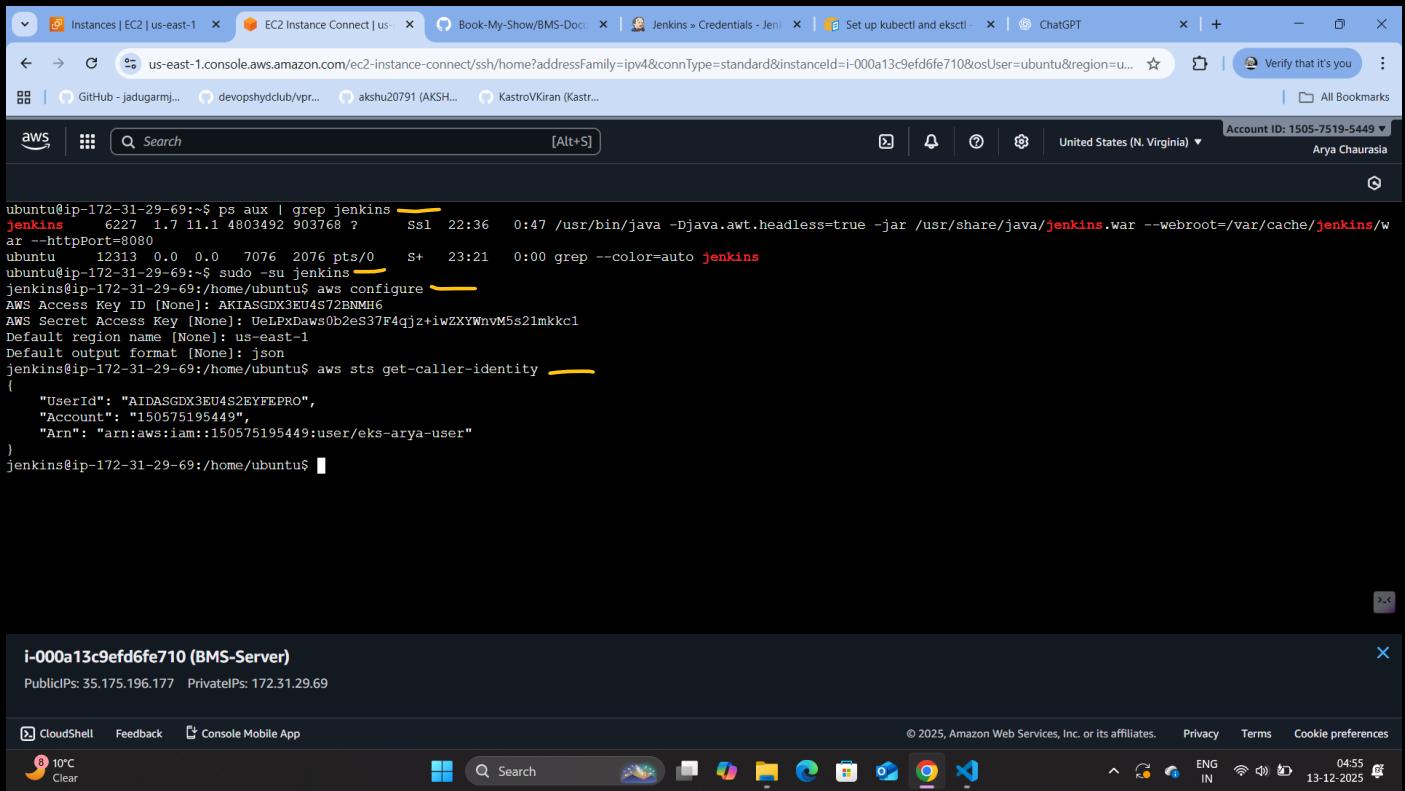
Fargate profiles (0)

No Fargate profiles

This cluster does not have any Fargate profiles.

[Add Fargate profile](#)

Step-22-To Know Jenkins is running on which user , if its Jenkins then switch to jenkins user ad verify the credentials.



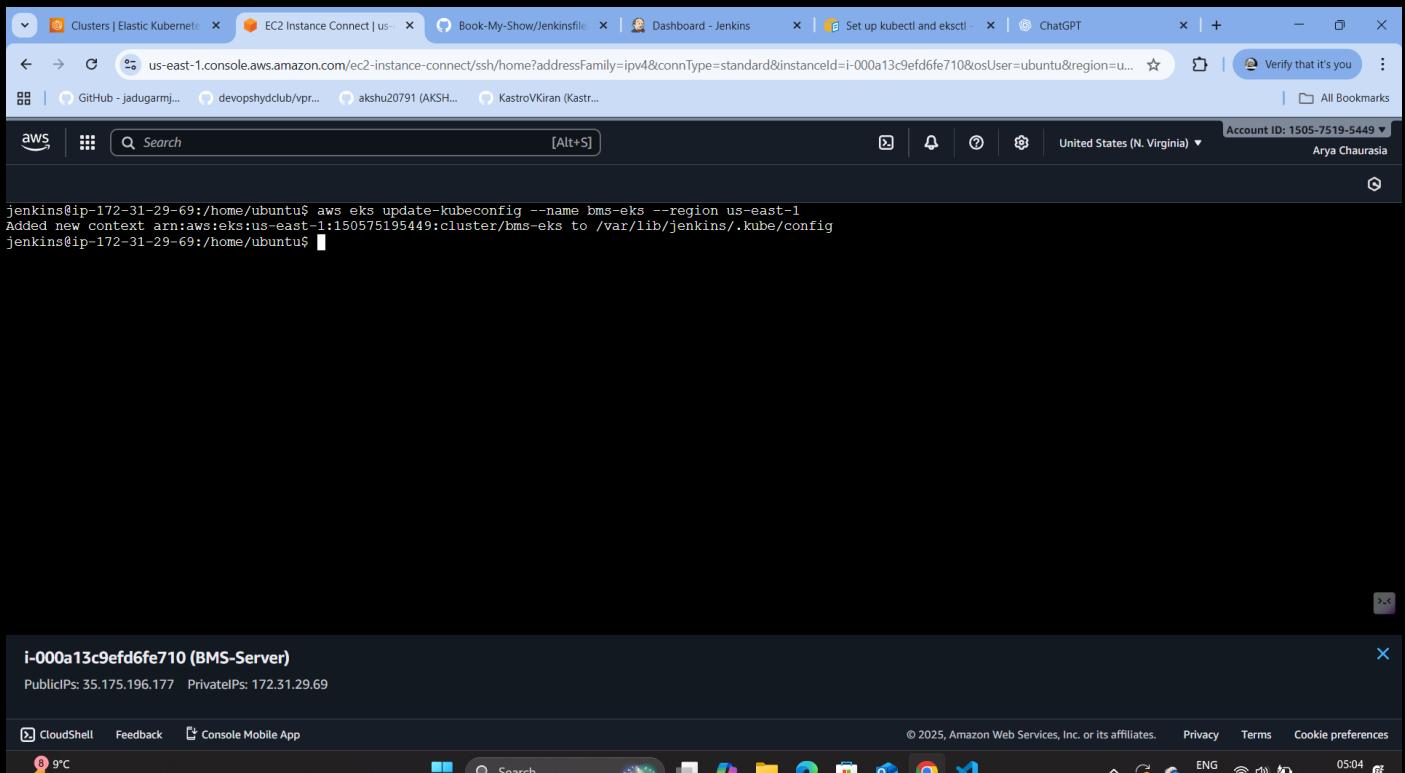
```
ubuntu@ip-172-31-29-69:~$ ps aux | grep jenkins
jenkins 6227 1.7 11.1 4803492 903768 ? Ssl 22:36 0:47 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
ubuntu 12313 0.0 0.0 7076 2076 pts/0 S+ 23:21 0:00 grep --color=auto jenkins
ubuntu@ip-172-31-29-69:~$ sudo -su jenkins
jenkins@ip-172-31-29-69:~/home/ubuntu$ aws configure
AWS Access Key ID [None]: AKIASGDX3EU4S72BMH6
AWS Secret Access Key [None]: UeLPxDaws0b2es37F4qjz+iwZXYWnvMs21mkkc1
Default region name [None]: us-east-1
Default output format [None]: json
jenkins@ip-172-31-29-69:~/home/ubuntu$ aws sts get-caller-identity
{
    "UserId": "AIDASGDX3EU4S2EYFEPRO",
    "Account": "150575195449",
    "Arn": "arn:aws:iam::150575195449:user/eks-arya-user"
}
jenkins@ip-172-31-29-69:~/home/ubuntu$
```

i-000a13c9efd6fe710 (BMS-Server)
PublicIPs: 35.175.196.177 PrivateIPs: 172.31.29.69

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

10°C Clear 10:45 13-12-2025

Step-23-Here we update the kubeconfig file



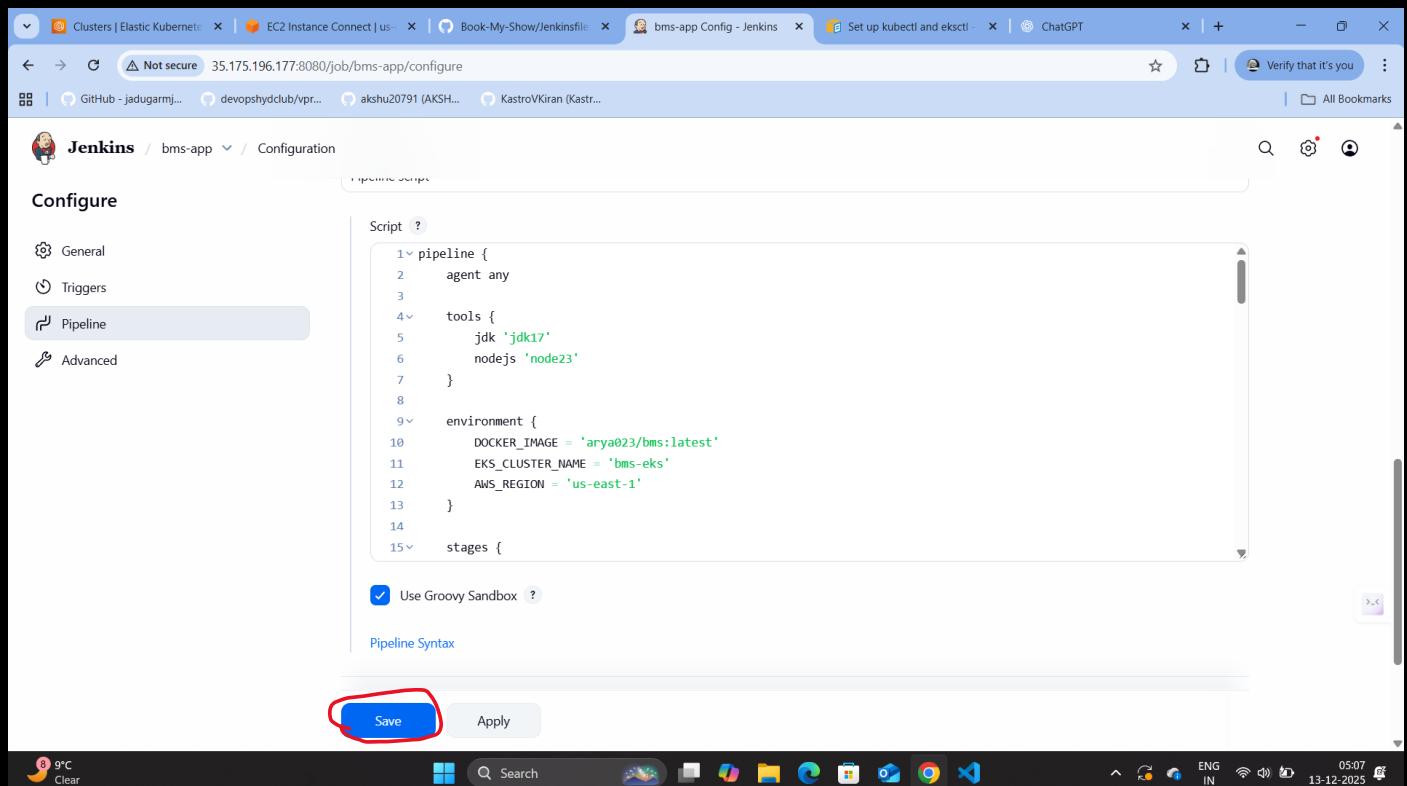
```
jenkins@ip-172-31-29-69:~/home/ubuntu$ aws eks update-kubeconfig --name bms-eks --region us-east-1
Added new context arn:aws:eks:us-east-1:150575195449:cluster/bms-eks to /var/lib/jenkins/.kube/config
jenkins@ip-172-31-29-69:~/home/ubuntu$
```

i-000a13c9efd6fe710 (BMS-Server)
PublicIPs: 35.175.196.177 PrivateIPs: 172.31.29.69

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

9°C Clear 05:04 13-12-2025

Step-24- Now we will create a job named – bms-app and in that we will run the pipeline.

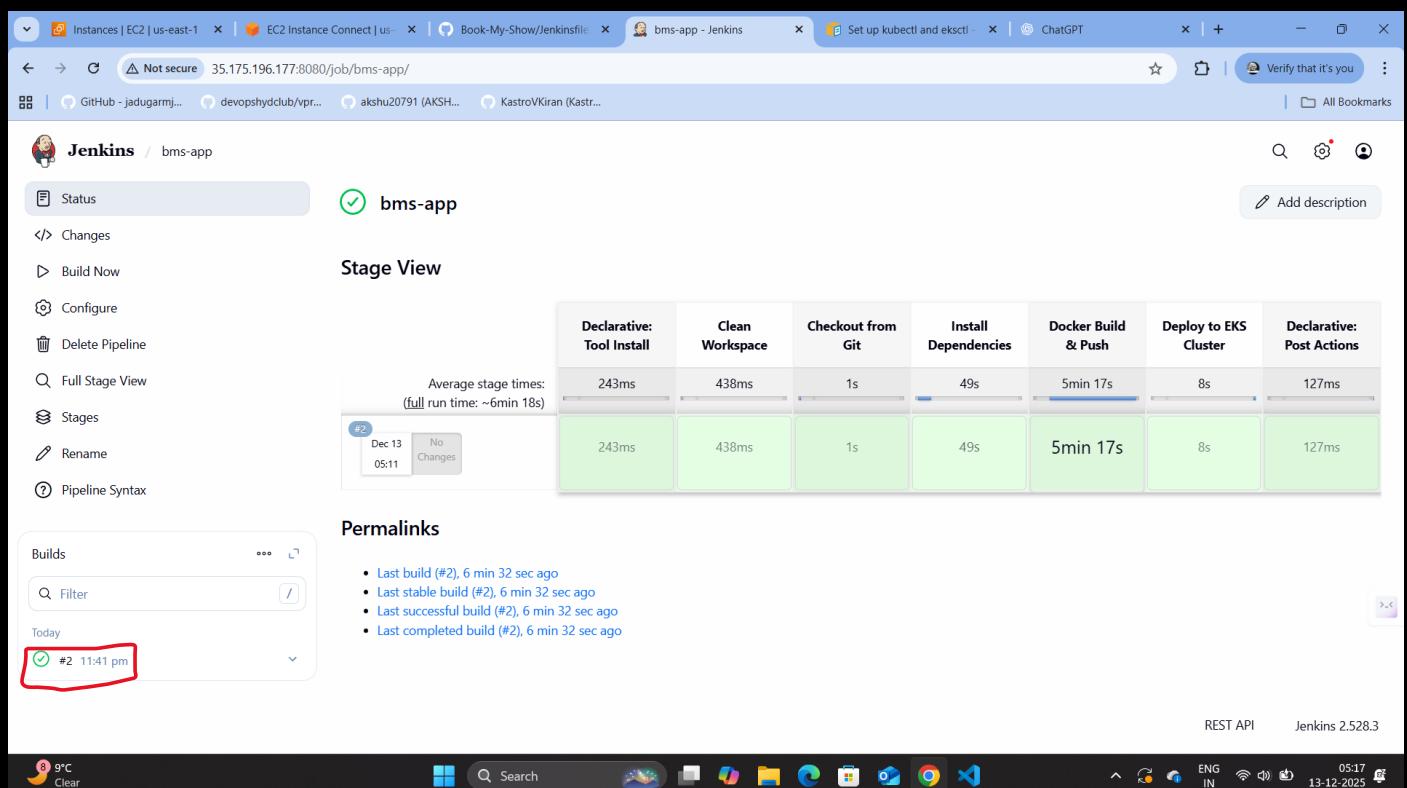


The screenshot shows the Jenkins Pipeline configuration page for a job named 'bms-app'. The left sidebar has tabs: General, Triggers, Pipeline (which is selected), and Advanced. The main area contains a Groovy script editor with the following code:

```
1 pipeline {
2     agent any
3
4     tools {
5         jdk 'jdk17'
6         nodejs 'node23'
7     }
8
9     environment {
10        DOCKER_IMAGE = 'arya023/bms:latest'
11        EKS_CLUSTER_NAME = 'bms-eks'
12        AWS_REGION = 'us-east-1'
13    }
14
15    stages {
```

Below the script, there is a checkbox labeled 'Use Groovy Sandbox' with a question mark icon. At the bottom of the editor are two buttons: 'Save' (highlighted with a red circle) and 'Apply'.

Step-25- The build is successful for the bms-app



The screenshot shows the Jenkins job details for 'bms-app'. The left sidebar includes links for Status, Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Stages, Rename, and Pipeline Syntax. The main area displays a 'Stage View' table with the following data:

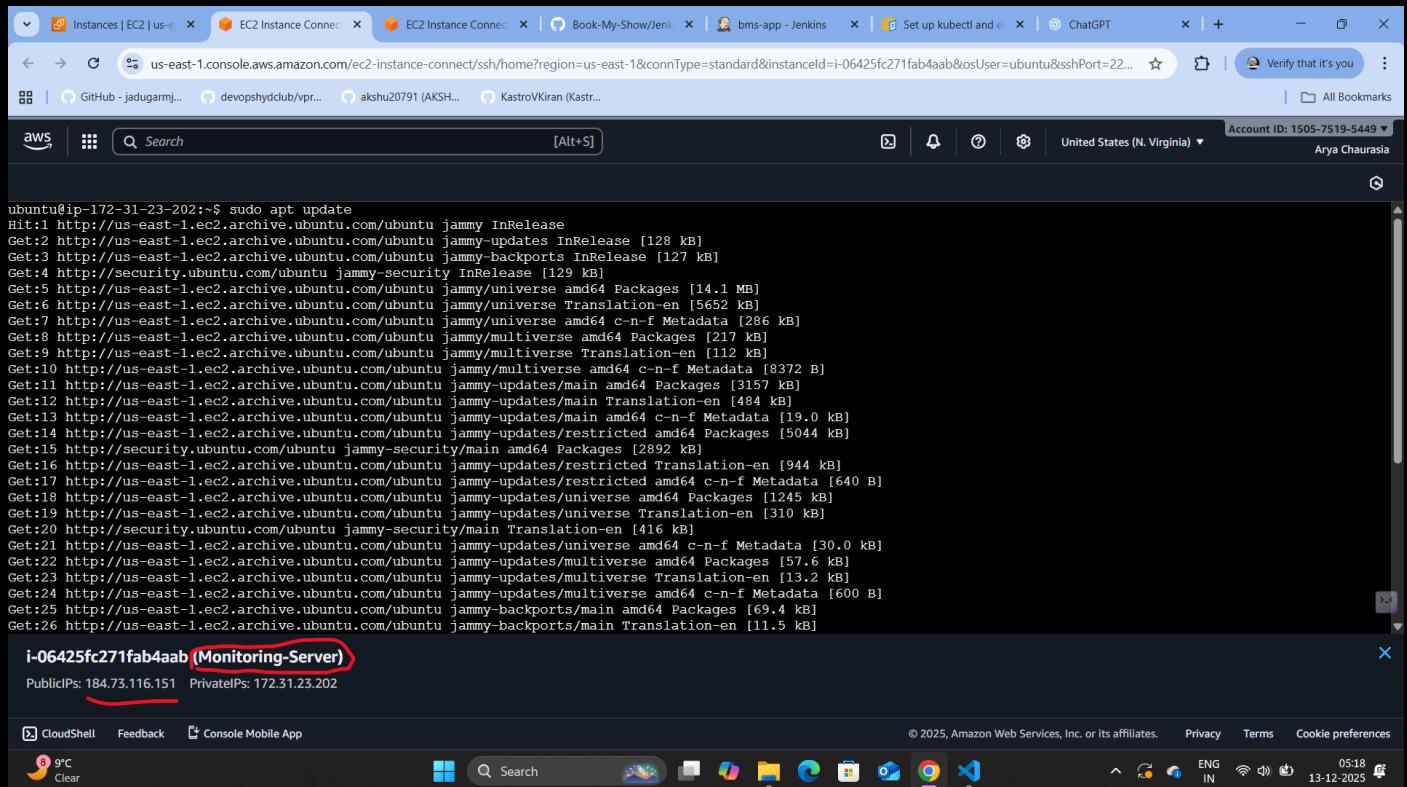
Declarative: Tool Install	Clean Workspace	Checkout from Git	Install Dependencies	Docker Build & Push	Deploy to EKS Cluster	Declarative: Post Actions
243ms	438ms	1s	49s	5min 17s	8s	127ms
243ms	438ms	1s	49s	5min 17s	8s	127ms

A message at the top of the table says 'Average stage times: (full run time: ~6min 18s)'. Below the table, a 'Permalinks' section lists recent builds:

- Last build (#2), 6 min 32 sec ago
- Last stable build (#2), 6 min 32 sec ago
- Last successful build (#2), 6 min 32 sec ago
- Last completed build (#2), 6 min 32 sec ago

At the bottom left, a 'Builds' section shows a dropdown menu with the option '#2 11:41 pm' highlighted with a red box. The bottom right corner shows the Jenkins version 'Jenkins 2.528.3'.

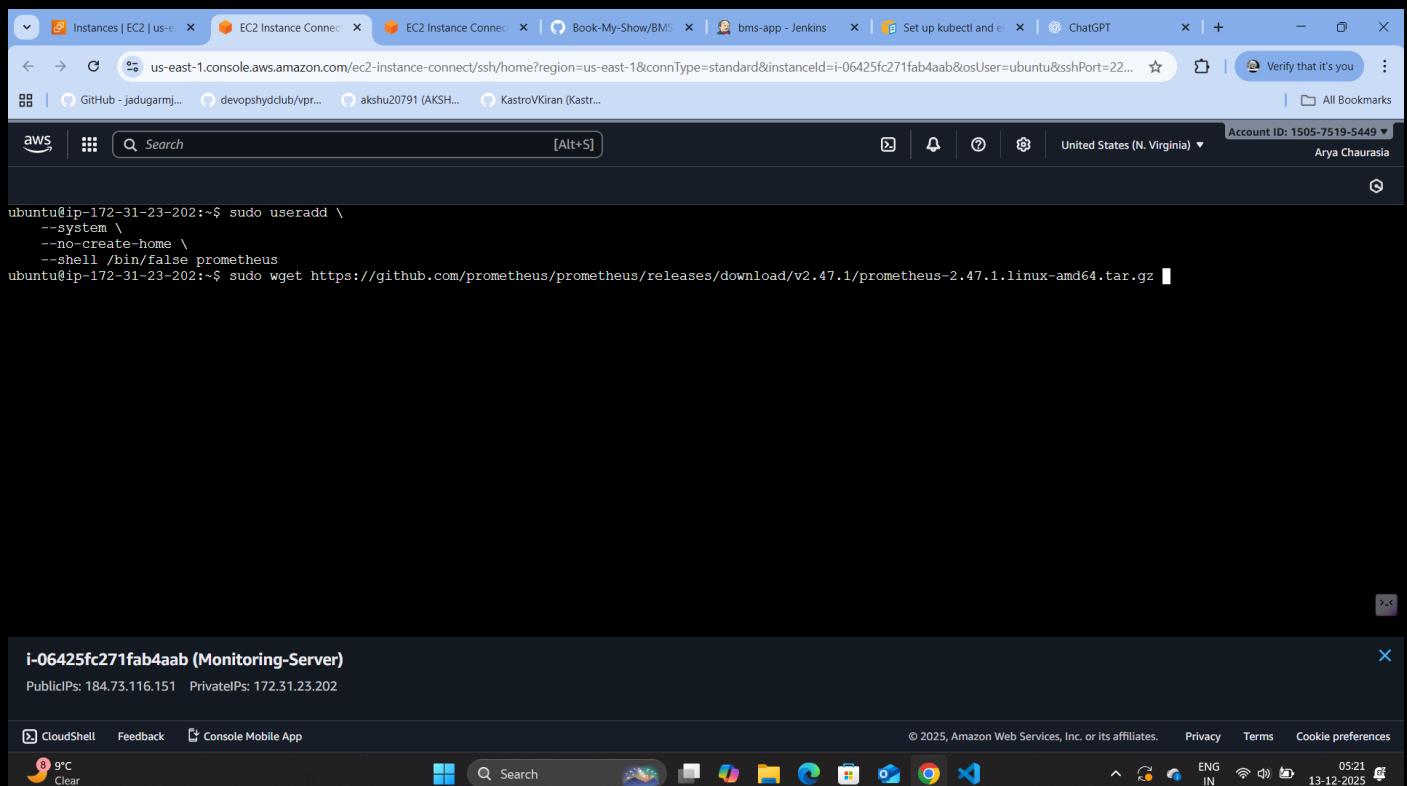
Step-26- Now we will make another instance for Prometheus which is Monitoring server.



```
ubuntu@ip-172-31-23-202:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [3157 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [494 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [19.0 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [5044 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2892 kB]
Get:16 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [944 kB]
Get:17 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [640 B]
Get:18 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1245 kB]
Get:19 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [310 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [416 kB]
Get:21 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [30.0 kB]
Get:22 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [57.6 kB]
Get:23 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [13.2 kB]
Get:24 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [600 B]
Get:25 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [69.4 kB]
Get:26 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [11.5 kB]

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202
```

Step-27- With this command we created the Prometheus user.



```
ubuntu@ip-172-31-23-202:~$ sudo useradd \
--system \
--no-create-home \
--shell /bin/false prometheus
ubuntu@ip-172-31-23-202:~$ sudo wget https://github.com/prometheus/prometheus/releases/download/v2.47.1/prometheus-2.47.1.linux-amd64.tar.gz
```

Step-28- Downloading the Prometheus.

```
ubuntu@ip-172-31-23-202:~$ tar -xvf prometheus-2.47.1.linux-amd64.tar.gz
prometheus-2.47.1.linux-amd64/
prometheus-2.47.1.linux-amd64/LICENSE
prometheus-2.47.1.linux-amd64/NOTICE
prometheus-2.47.1.linux-amd64/prometheus.yml
prometheus-2.47.1.linux-amd64/consoles/
prometheus-2.47.1.linux-amd64/consoles/prometheus.html
prometheus-2.47.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.47.1.linux-amd64/consoles/node-cpu.html
prometheus-2.47.1.linux-amd64/consoles/index.html.example
prometheus-2.47.1.linux-amd64/consoles/node.html
prometheus-2.47.1.linux-amd64/consoles/node-disk.html
prometheus-2.47.1.linux-amd64/consoles/node-overview.html
prometheus-2.47.1.linux-amd64/promtool
prometheus-2.47.1.linux-amd64/console_libraries/
prometheus-2.47.1.linux-amd64/console_libraries/prom.lib
prometheus-2.47.1.linux-amd64/console_libraries/menu.lib
prometheus-2.47.1.linux-amd64/prometheus
ubuntu@ip-172-31-23-202:~$ ls
prometheus-2.47.1.linux-amd64_prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-23-202:~$ sudo mkdir -p /data/etc/prometheus
ubuntu@ip-172-31-23-202:~$ cd prometheus-2.47.1.linux-amd64/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ pwd
/home/ubuntu/prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo mv prometheus promtool /usr/local/bin/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo mv consoles/ console_libraries/ /etc/prometheus/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo chown -R prometheus:prometheus /etc/prometheus/ /data/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$
```

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 9°C Clear ENG IN 05:25 13-12-2025

Step-29-Move the prometheus binary and a promtool to the /usr/local/bin/.

```
prometheus-2.47.1.linux-amd64/LICENSE
prometheus-2.47.1.linux-amd64/NOTICE
prometheus-2.47.1.linux-amd64/prometheus.yml
prometheus-2.47.1.linux-amd64/consoles/
prometheus-2.47.1.linux-amd64/consoles/prometheus.html
prometheus-2.47.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.47.1.linux-amd64/consoles/node-cpu.html
prometheus-2.47.1.linux-amd64/consoles/index.html.example
prometheus-2.47.1.linux-amd64/consoles/node.html
prometheus-2.47.1.linux-amd64/consoles/node-disk.html
prometheus-2.47.1.linux-amd64/consoles/node-overview.html
prometheus-2.47.1.linux-amd64/promtool
prometheus-2.47.1.linux-amd64/console_libraries/
prometheus-2.47.1.linux-amd64/console_libraries/prom.lib
prometheus-2.47.1.linux-amd64/console_libraries/menu.lib
prometheus-2.47.1.linux-amd64/prometheus
ubuntu@ip-172-31-23-202:~$ ls
prometheus-2.47.1.linux-amd64_prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-23-202:~$ sudo mkdir -p /data/etc/prometheus
ubuntu@ip-172-31-23-202:~$ cd prometheus-2.47.1.linux-amd64/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ pwd
/home/ubuntu/prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo mv prometheus promtool /usr/local/bin/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo mv consoles/ console_libraries/ /etc/prometheus/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo mv prometheus.yml /etc/prometheus/prometheus.yml
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo chown -R prometheus:prometheus /etc/prometheus/ /data/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$
```

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 9°C Clear ENG IN 05:27 13-12-2025

Step-30- Set the correct ownership for the /etc/prometheus/ and data directory

```
prometheus-2.47.1.linux-amd64/console_libraries/prom.lib
prometheus-2.47.1.linux-amd64/console_libraries/menu.lib
prometheus-2.47.1.linux-amd64/prometheus
ubuntu@ip-172-31-23-202:~$ ls
prometheus-2.47.1.linux-amd64 prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-23-202:~$ sudo mkdir -p /data /etc/prometheus
ubuntu@ip-172-31-23-202:~$ cd prometheus-2.47.1.linux-amd64/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ pwd
/home/ubuntu/prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo mv promtool /usr/local/bin/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo mv consoles/ console_libraries/ /etc/prometheus/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo mv prometheus.yml /etc/prometheus/prometheus.yml
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ sudo chown -R prometheus:prometheus /etc/prometheus/ /data/
ubuntu@ip-172-31-23-202:~/prometheus-2.47.1.linux-amd64$ cd
ubuntu@ip-172-31-23-202:~$ ls
prometheus-2.47.1.linux-amd64 prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-23-202:~$ rm -rf prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-23-202:~$ ls
prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-23-202:~$ prometheus --version
prometheus, version 2.47.1 (branch: HEAD, revision: c4d1a8beff37cc004f1dc4ab9d2e73193f51aaeb)
  build user:          root@4829330363be
  build date:         20231004-10:31:16
  go version:        go1.21.1
  platform:          linux/amd64
  tags:              netgo,builtinassets,stringlabels
ubuntu@ip-172-31-23-202:~$ █
```

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

9°C Clear ENG IN 05:21 13-12-2025

Step-31- We are going to use systemd which is a system and service manager for linux OS, for that we need to create a systemd unit configuration file.

```
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target
StartLimitIntervalSec=500
StartLimitBurst=5
[Service]
User=prometheus
Group=prometheus
Type=simple
Restart=on-failure
RestartSec=5s
ExecStart=/usr/local/bin/prometheus \
--config.file=/etc/prometheus/prometheus.yml \
--storage.tsdb.path=/data \
--web.console.templates=/etc/prometheus/consoles \
--web.console.libraries=/etc/prometheus/console_libraries \
--web.listen-address=0.0.0.0:9090 \
--web.enable-lifecycle
[Install]
WantedBy=multi-user.target
~
~
~
~
:wq█
```

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

News for you Ozempic is final... ENG IN 05:21 13-12-2025

Step-32- Now we will check the status of the prometheus

The screenshot shows a terminal session on an AWS EC2 instance. The user has run several commands to configure and start the Prometheus service:

```
ubuntu@ip-172-31-23-202:~$ sudo vi /etc/systemd/system/prometheus.service
ubuntu@ip-172-31-23-202:~$ sudo systemctl enable prometheus
Created symlink /etc/systemd/system/multi-user.target.wants/prometheus.service → /etc/systemd/system/prometheus.service.
ubuntu@ip-172-31-23-202:~$ sudo systemctl start prometheus
ubuntu@ip-172-31-23-202:~$ sudo systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-12-13 00:04:12 UTC; 28s ago
     Main PID: 2146 (prometheus)
       Tasks: 8 (limit: 4670)
      Memory: 18.5M
        CPU: 82ms
       CGroup: /system.slice/prometheus.service
           └─2146 /usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/data --web.console.templates=/etc/prometheus/consoles

Dec 13 00:04:12 ip-172-31-23-202 prometheus[2146]: ts=2025-12-13T00:04:12.699Z caller=head.go:681 level=info component=tsdb msg="On-disk memory mappable chunks replayed"
Dec 13 00:04:12 ip-172-31-23-202 prometheus[2146]: ts=2025-12-13T00:04:12.699Z caller=head.go:689 level=info component=tsdb msg="Replaying WAL, this may take a while"
Dec 13 00:04:12 ip-172-31-23-202 prometheus[2146]: ts=2025-12-13T00:04:12.701Z caller=head.go:760 level=info component=tsdb msg="WAL segment loaded" segment=0 maxSegment=1
Dec 13 00:04:12 ip-172-31-23-202 prometheus[2146]: ts=2025-12-13T00:04:12.701Z caller=head.go:797 level=info component=tsdb msg="WAL replay completed" checkpoint_replayed=true
Dec 13 00:04:12 ip-172-31-23-202 prometheus[2146]: ts=2025-12-13T00:04:12.706Z caller=main.go:1045 level=info fs_type=EXT4_SUPER_MAGIC
Dec 13 00:04:12 ip-172-31-23-202 prometheus[2146]: ts=2025-12-13T00:04:12.707Z caller=main.go:1048 level=info msg="TSDB started"
Dec 13 00:04:12 ip-172-31-23-202 prometheus[2146]: ts=2025-12-13T00:04:12.707Z caller=main.go:1229 level=info msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
```

Below the terminal, a section titled "i-06425fc271fab4aab (Monitoring-Server)" lists Public IPs (184.73.116.151) and Private IPs (172.31.23.202).

Step-33-<monitoring-server IP> : 9090 to run the prometheus

The screenshot shows the Prometheus Time Series Collector interface at the URL <http://184.73.116.151:9090/graph>. The page displays a search bar with the query `g0.expr=&g0.tab=1&g0.stackable=0&g0.show_exemplars=0&g0.range_input=1h`. Below the search bar, there are tabs for "Table" and "Graph". A message "No data queried yet" is displayed. At the bottom left, there is a "Add Panel" button.

Step-34- Now we will download the node exporter and extract the files.

The screenshot shows a browser window with multiple tabs open. The active tab is a CloudShell session titled "aws" where a file is being downloaded:

```
node_exporter-1.6.1.linux-amd64.tar.gz 100%[=====] 9.89M ---KB/s in 0.09s
```

Below the progress bar, the terminal output shows the download and extraction process:

```
2025-12-13 00:09:02 (115 MB/s) - 'node_exporter-1.6.1.linux-amd64.tar.gz' saved [10368103/10368103]

ubuntu@ip-172-31-23-202:~$ ls
node_exporter-1.6.1.linux-amd64.tar.gz  prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-23-202:~$ tar -xvf node_exporter-1.6.1.linux-amd64.tar.gz
node_exporter-1.6.1.linux-amd64/
node_exporter-1.6.1.linux-amd64/NOTICE
node_exporter-1.6.1.linux-amd64/node_exporter
node_exporter-1.6.1.linux-amd64/LICENSE
ubuntu@ip-172-31-23-202:~$ ls
node_exporter-1.6.1.linux-amd64  node_exporter-1.6.1.linux-amd64.tar.gz  prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-23-202:~$ sudo mv node_exporter-1.6.1.linux-amd64/node_exporter /usr/local/bin/
ubuntu@ip-172-31-23-202:~$ rm -rf node_exporter
ubuntu@ip-172-31-23-202:~$ ls
prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-23-202:~$ node exporter --version
node_exporter, version 1.6.1 (branch: HEAD, revision: 4a1b77600c1873a8233f3ffb55afcedbb63b8d84)
  build user: root@586879db1le5
  build date: 20230717-12:10:52
  go version: go1.20.6
  platform: linux/amd64
  tags: netgo osusergo static_build
ubuntu@ip-172-31-23-202:~$
```

At the bottom of the terminal window, there is a message about the monitoring server:

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202

Step-35-Adding the following content to the node_exporter.service file

The screenshot shows a browser window with multiple tabs open. The active tab is a CloudShell session titled "aws" where a configuration file is being edited:

```
[Description]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
StartLimitIntervalSec=500
StartLimitBurst=5

[Service]
User=node_exporter
Group=node_exporter
Type=simple
Restart=on-failure
RestartSec=5s
ExecStart=/usr/local/bin/node_exporter --collector.logind

[Install]
WantedBy=multi-user.target

-- INSERT --
:wq
```

At the bottom of the terminal window, there is a message about the monitoring server:

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202

Step-36-Now we will enable and start the node exporter.

```
ubuntu@ip-172-31-23-202:~$ sudo vi /etc/systemd/system/node_exporter.service
ubuntu@ip-172-31-23-202:~$ sudo systemctl enable node_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node_exporter.service → /etc/systemd/system/node_exporter.service.
ubuntu@ip-172-31-23-202:~$ sudo systemctl start node_exporter
ubuntu@ip-172-31-23-202:~$ sudo systemctl status node_exporter
● node_exporter.service - Node Exporter
   Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-12-13 00:13:19 UTC; 27s ago
     Main PID: 2233 (node_exporter)
       Tasks: 4 (limit: 4670)
      Memory: 2.6M
        CPU: 7ms
       CGroup: /system.slice/node_exporter.service
           └─ 2233 /usr/local/bin/node_exporter --collector.logind

Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=node_exporter.go:117 level=info collector=thermal_zone
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=node_exporter.go:117 level=info collector=time
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=node_exporter.go:117 level=info collector=timex
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=node_exporter.go:117 level=info collector=udp_queues
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=node_exporter.go:117 level=info collector=uname
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=node_exporter.go:117 level=info collector=vmstat
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=node_exporter.go:117 level=info collector=xfs
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=node_exporter.go:117 level=info collector=zfs
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.150Z caller=tls_config.go:274 level=info msg="Listening on" address=[::]:9100
Dec 13 00:13:19 ip-172-31-23-202 node_exporter[2233]: ts=2025-12-13T00:13:19.151Z caller=tls_config.go:277 level=info msg="TLS is disabled." http2=false address=[::]:9100
13
```

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 10°C Clear ENG IN 05:43 13-12-2025

Step-37- To scrape metrics from Node Exporter and Jenkins, we need to modify the prometheus.yml file

```
ubuntu@ip-172-31-23-202:~$ ls
prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-23-202:~$ cd /etc/prometheus/
ubuntu@ip-172-31-23-202:/etc/prometheus$ sudo vi prometheus.yml
ubuntu@ip-172-31-23-202:/etc/prometheus$ promtool check config /etc/prometheus/prometheus.yml
Checking /etc/prometheus/prometheus.yml
SUCCESS: /etc/prometheus/prometheus.yml is valid prometheus config file syntax
ubuntu@ip-172-31-23-202:/etc/prometheus$ curl -X POST http://localhost:9090/-/reload
```

i-06425fc271fab4aab (Monitoring-Server)
PublicIPs: 184.73.116.151 PrivateIPs: 172.31.23.202

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences Sunrise soon 7:03 am 06:09 ENG IN 13-12-2025

Step-38- Here we see that our node_exporter is not running so we will go to the security group of the monitoring server.

The screenshot shows the Prometheus Targets page. At the top, there are tabs for All, Unhealthy, and Collapse All. A search bar is present, along with filters for Unknown, Unhealthy, and Healthy. Below this, a table lists targets:

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	12.156s ago	4.304ms	

In the list above the table, 'node_exporter (0/1 up)' is shown as Unhealthy, indicated by a red X icon. Other targets listed are Jenkins (1/1 up) and prometheus (1/1 up).

Step-39- We will enable the port 9100 for node_exporter.

The screenshot shows the AWS CloudShell interface. The user is navigating through the AWS Management Console to edit inbound rules for a security group. The 'Edit inbound rules' section is visible, showing three existing rules and one new rule being added. The new rule is highlighted with a red circle around the 'Save rules' button.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0c801063ec0633536	Custom TCP	TCP	9090	Custom	
sgr-0ea3b2e8ea988cc7a	SSH	TCP	22	Custom	
-	Custom TCP	TCP	9100	Anywh...	

Add rule

Save rules (highlighted with a red circle)

Step-40- Now refresh the page and we will see that the node_exporter is running.

The screenshot shows a dark-themed web browser window with several tabs at the top. The active tab is titled "Prometheus Time Series Collection". Below the tabs, the URL bar shows "Not secure 184.73.116.151:9090/targets?search=". The main content area is titled "Targets" and displays a list of monitored endpoints. The list includes:

- Jenkins (1/1 up)
- node_exporter (1/1 up) [show more]
- prometheus (1/1 up) [show more]

A green checkmark is drawn next to the "node_exporter (1/1 up)" entry. At the bottom of the browser window, there is a taskbar with various icons and system status information.

Step-41- Now we will install Grafana and add the gpg key for Grafana and you should see OK when the command is executed.

The screenshot shows a terminal window within an AWS CloudShell session. The session ID is i-06425fc271fab4aab (Monitoring-Server). The user has run the following command:

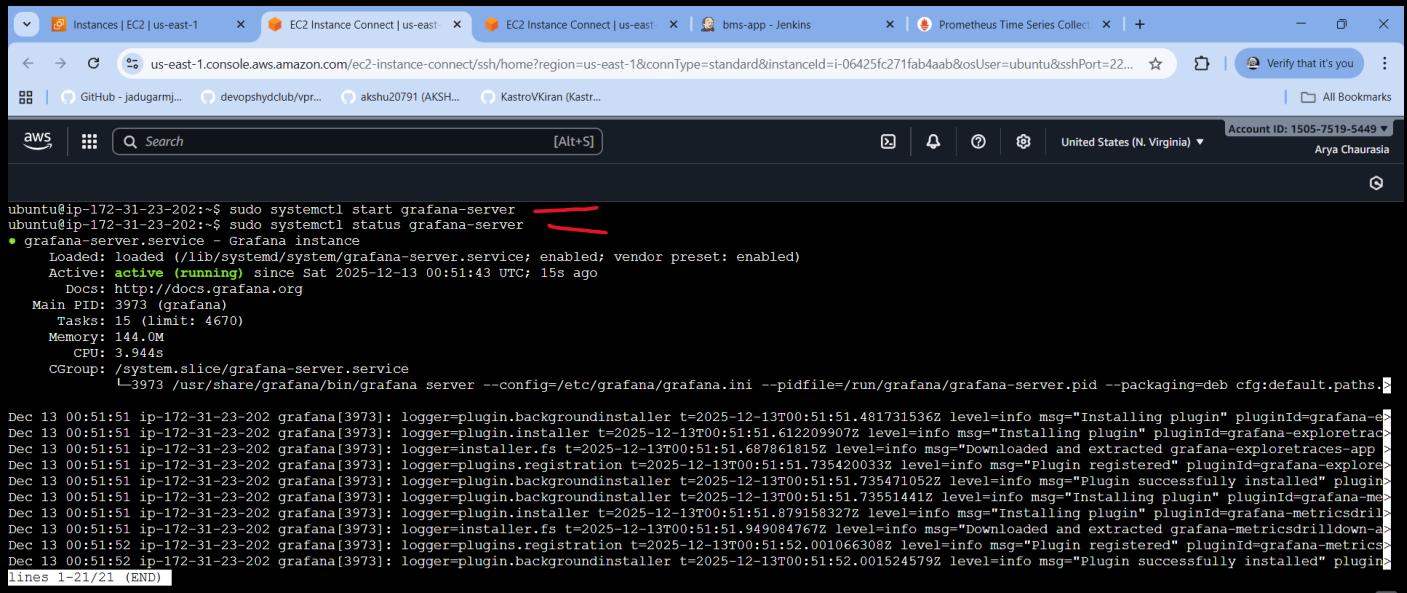
```
ubuntu@ip-172-31-23-202:~$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
```

The output of the command shows a warning about the deprecation of apt-key and a message indicating success:

```
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).  
OK
```

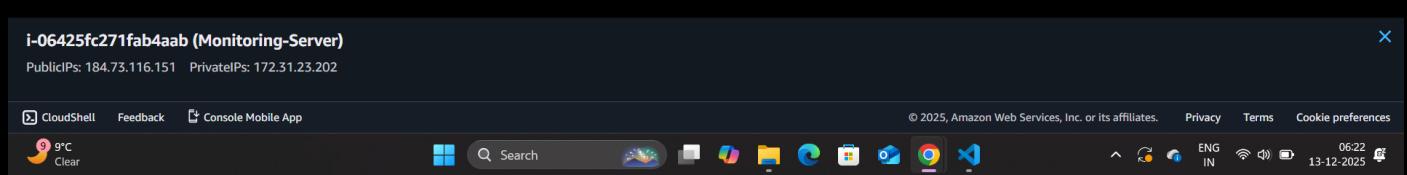
At the bottom of the terminal window, there is a footer with links for CloudShell, Feedback, and Console Mobile App, along with copyright and legal information.

Step-42- Enable and start the Grafan Service and also we will check the Grafana status.

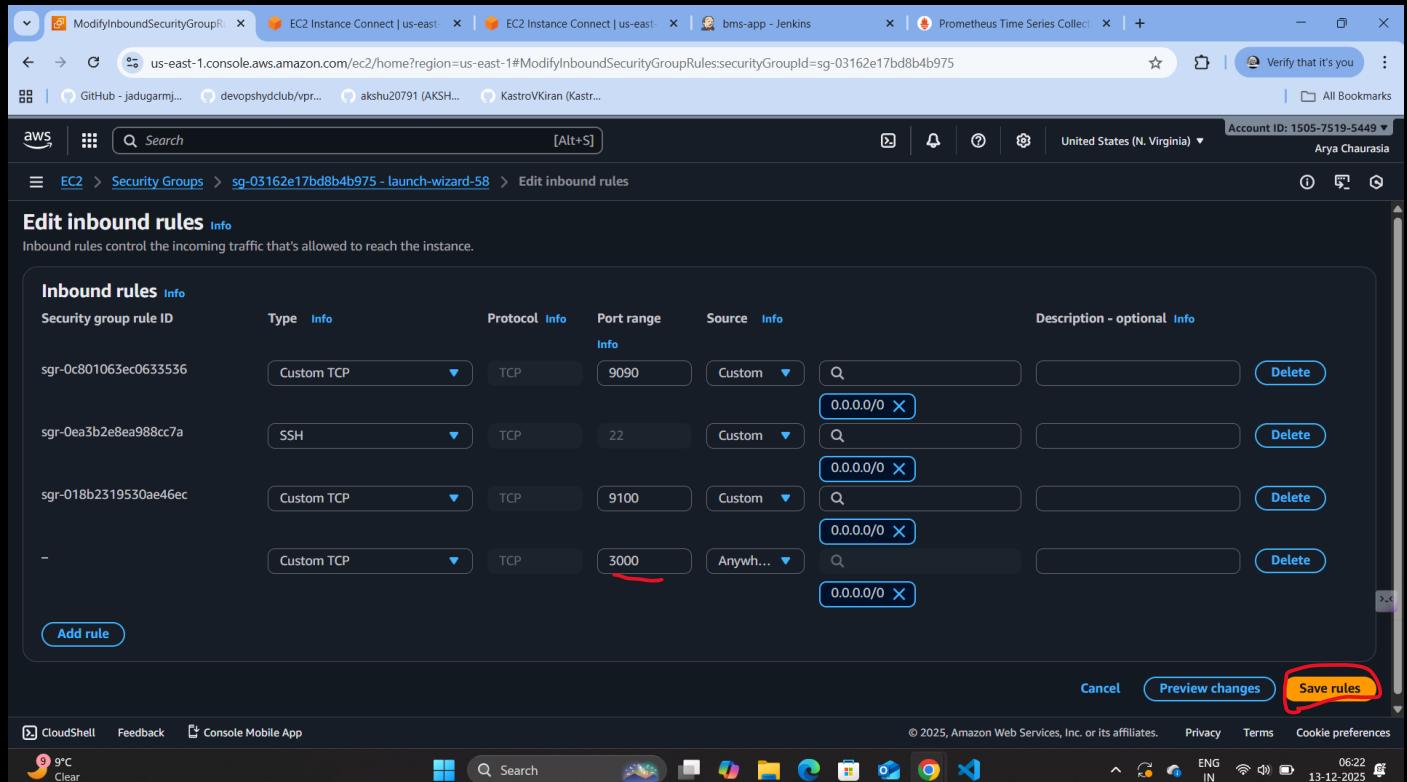


```
ubuntu@ip-172-31-23-202:~$ sudo systemctl start grafana-server
ubuntu@ip-172-31-23-202:~$ sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
   Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2025-12-13 00:51:14 UTC; 15s ago
     Docs: http://docs.grafana.org
 Main PID: 3973 (grafana)
   Tasks: 15 (limit: 4670)
    Memory: 144.0M
      CPU: 3.944s
     CGroup: /system.slice/grafana-server.service
             └─3973 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg=default.paths

Dec 13 00:51:51 ip-172-31-23-202 grafana[3973]: logger=plugin.backgroundinstaller t=2025-12-13T00:51:51.481731536Z level=info msg="Installing plugin" pluginId=grafana-exploretraces-app
Dec 13 00:51:51 ip-172-31-23-202 grafana[3973]: logger=plugin.installer t=2025-12-13T00:51:51.612209907Z level=info msg="Installing plugin" pluginId=grafana-exploretraces-app
Dec 13 00:51:51 ip-172-31-23-202 grafana[3973]: logger=installer.fs t=2025-12-13T00:51:51.687861815Z level=info msg="Downloaded and extracted grafana-exploretraces-app"
Dec 13 00:51:51 ip-172-31-23-202 grafana[3973]: logger=plugins.registration t=2025-12-13T00:51:51.735420033Z level=info msg="Plugin registered" pluginId=grafana-exploretraces-app
Dec 13 00:51:51 ip-172-31-23-202 grafana[3973]: logger=plugin.backgroundinstaller t=2025-12-13T00:51:51.735471052Z level=info msg="Plugin successfully installed" pluginId=grafana-exploretraces-app
Dec 13 00:51:51 ip-172-31-23-202 grafana[3973]: logger=plugin.backgroundinstaller t=2025-12-13T00:51:51.735514412Z level=info msg="Installing plugin" pluginId=grafana-metricsdrilldown
Dec 13 00:51:51 ip-172-31-23-202 grafana[3973]: logger=plugin.installer t=2025-12-13T00:51:51.879158327Z level=info msg="Installing plugin" pluginId=grafana-metricsdrilldown
Dec 13 00:51:51 ip-172-31-23-202 grafana[3973]: logger=installer.fs t=2025-12-13T00:51:51.949084767Z level=info msg="Downloaded and extracted grafana-metricsdrilldown"
Dec 13 00:51:52 ip-172-31-23-202 grafana[3973]: logger=plugins.registration t=2025-12-13T00:51:52.001066308Z level=info msg="Plugin registered" pluginId=grafana-metricsdrilldown
Dec 13 00:51:52 ip-172-31-23-202 grafana[3973]: logger=plugin.backgroundinstaller t=2025-12-13T00:51:52.001524579Z level=info msg="Plugin successfully installed" pluginId=grafana-metricsdrilldown
lines 1-21/21 (END)
```



Step-43-We enable the port 3000 for Grafana.



The screenshot shows the 'Edit inbound rules' section of the AWS EC2 Security Groups configuration. It lists four existing rules and one new rule being added:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0c801063ec0633536	Custom TCP	TCP	9090	Custom	
sgr-0ea3b2e8ea988cc7a	SSH	TCP	22	Custom	
sgr-018b2319530ae46ec	Custom TCP	TCP	9100	Custom	
-	Custom TCP	TCP	3000	Anywh...	

A red box highlights the '3000' port entry in the 'Port range' field of the new rule. At the bottom right, a red box highlights the 'Save rules' button.

Step-44- <Monitoring Server IP>3000 to open the Grafana.

The screenshot shows the Grafana home page. The URL in the address bar is `184.73.116.151:3000/?orgId=1&from=now-6h&to=now&timezone=browser`. The left sidebar has a 'Basic' section with instructions for setting up Grafana. Three main panels are visible: 'TUTORIAL DATA SOURCE AND DASHBOARDS' (Grafana fundamentals), 'DATA SOURCES' (Add your first data source), and 'DASHBOARDS' (Create your first dashboard). Below these are sections for Dashboards, Starred dashboards, and Recently viewed dashboards. A blue banner at the bottom right reads 'CAN data analysis with Grafana Assistant'. The system tray at the bottom shows weather (9°C), battery status, and system icons.

Step-45-On the right site upper corner + sign is there, click there on import dashboards, give name and IP address of the monitoring server.

The screenshot shows the 'Data sources' configuration page for Prometheus. The URL in the address bar is `184.73.116.151:3000/connections/datasources/edit/ef6xau4764n40d/`. The left sidebar has a 'Data sources' section selected. The main area shows a 'Connection' section with a 'Prometheus server URL' input field containing `http://184.73.116.151:9090`. A red box highlights the '+' button in the top right corner of the main content area. The system tray at the bottom shows weather (9°C), battery status, and system icons.

Step-46-Click save to get the dashboard for the prometheus

The screenshot shows the Grafana interface with the URL `184.73.116.151:3000/connections/datasources/edit/ef6xau4764n40d/`. The left sidebar is open, showing 'Connections' selected under 'Data sources'. The main area displays configuration for a Prometheus connection, including fields for 'Custom query parameters', 'HTTP method' (set to POST), 'Series limit' (set to 40000), and 'Use series endpoint' (unchecked). Below these are sections for 'Exemplars' and a success message: 'Successfully queried the Prometheus API.' A red circle highlights the 'Save & test' button at the bottom of the configuration area. The status bar at the bottom right shows the date and time as 13-12-2025.

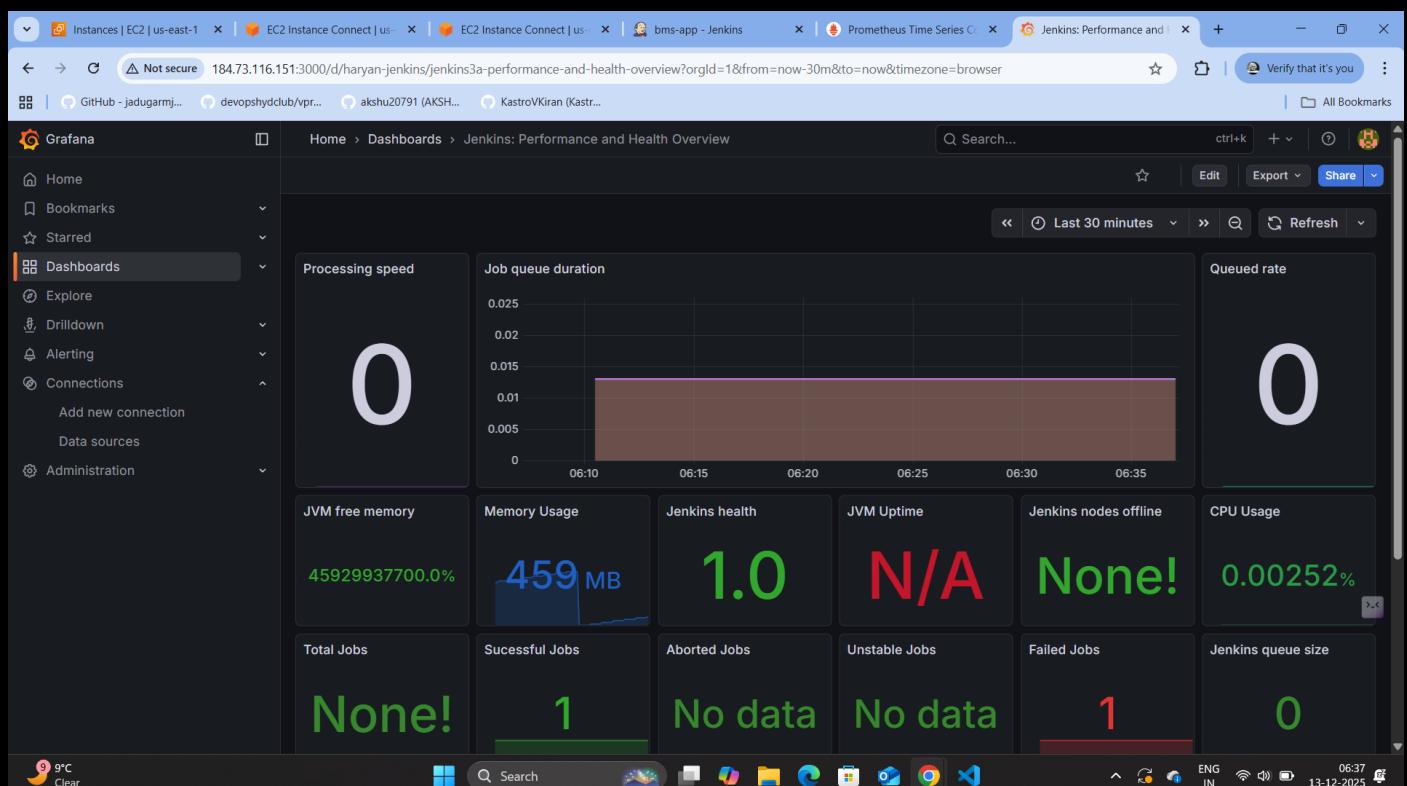
Step-47- Here is our dashboard

The screenshot shows the Grafana interface with the URL `184.73.116.151:3000/d/rYdddlPWk/node-exporter-full?orgId=1&from=now-24h&to=now&timezone=browser&var_ds_prometheus=ef6xau4764n40d&var-job=no...`. The left sidebar is open, showing 'Dashboards' selected. The main area displays a dashboard titled 'Node Exporter Full' with various metrics. At the top, there are dropdowns for 'Datasource' (prometheus-1), 'Job' (node_exporter), and 'Nodename' (ip-172-31-23-202). Below these are sections for 'Quick CPU / Mem / Disk' and 'Basic CPU / Mem / Net / Disk'. A red circle highlights the 'Save & test' button at the bottom of the dashboard configuration area. The status bar at the bottom right shows the date and time as 13-12-2025.

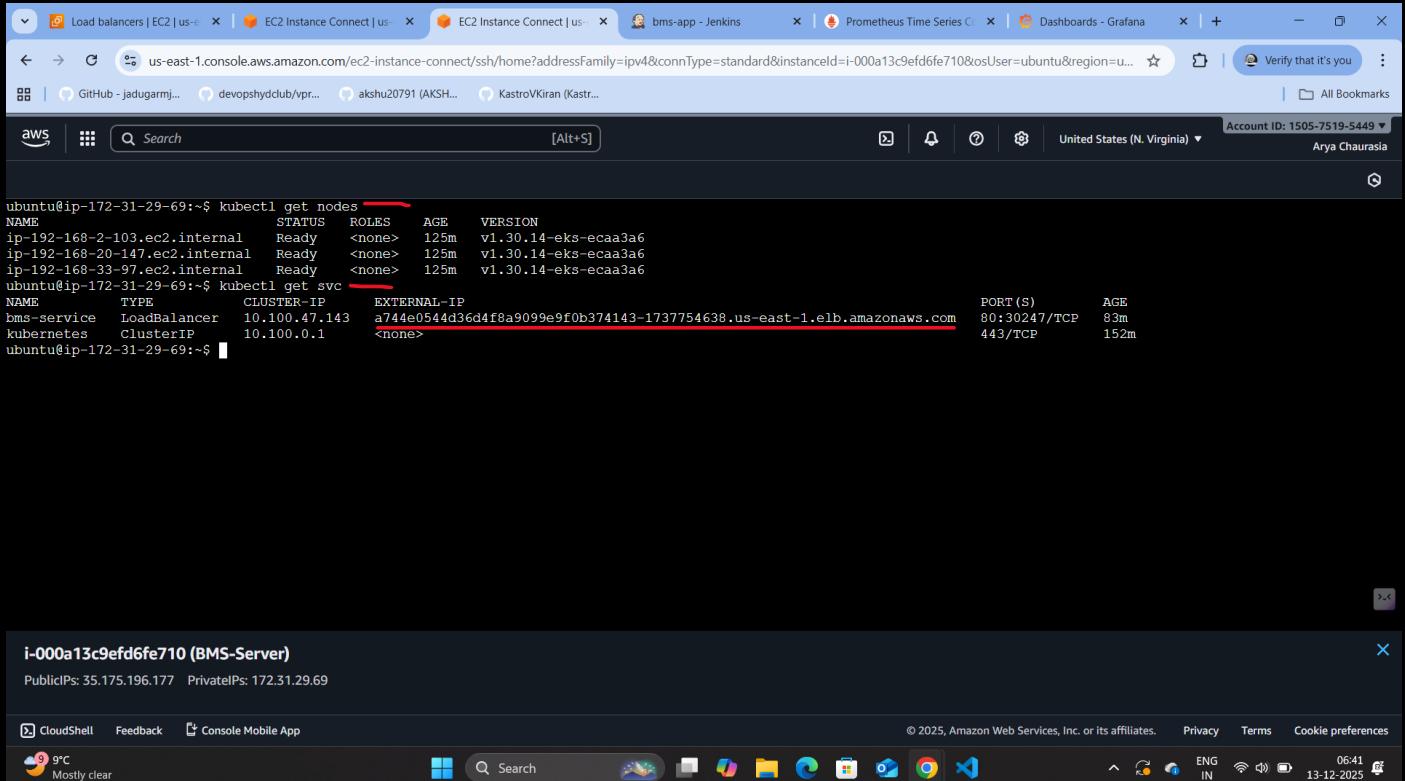
Step-48- 9964 is dashboard number for Jenkins , load it

The screenshot shows the Grafana interface with the URL `184.73.116.151:3000/dashboard/import`. On the left, there's a sidebar with 'Dashboards' selected. In the main area, there's a section for importing dashboards from file or [Grafana.com](grafana.com/dashboards), with a text input field containing '9964'. Below it is a JSON model editor with some code. At the bottom are 'Load' and 'Cancel' buttons.

Step-49- Here we can see the performance of the Jenkins.



Step-50- It shows the worker nodes inside the K8S Cluster, kubectl get svc will give you the URL for your application, copy it and paste it to the browser.



```
ubuntu@ip-172-31-29-69:~$ kubectl get nodes
NAME           STATUS    ROLES   AGE     VERSION
ip-192-168-2-103.ec2.internal   Ready    <none>  125m   v1.30.14-eks-ecaa3a6
ip-192-168-20-147.ec2.internal Ready    <none>  125m   v1.30.14-eks-ecaa3a6
ip-192-168-33-97.ec2.internal Ready    <none>  125m   v1.30.14-eks-ecaa3a6
ubuntu@ip-172-31-29-69:~$ kubectl get svc
NAME          TYPE      CLUSTER-IP   EXTERNAL-IP
bms-service   LoadBalancer 10.100.47.143  a744e0544d36d4f8a9099e9f0b374143-1737754638.us-east-1.elb.amazonaws.com
kubernetes    ClusterIP  10.100.0.1    <none>
ubuntu@ip-172-31-29-69:~$
```

i-000a13c9efd6fe710 (BMS-Server)
Public IPs: 35.175.196.177 Private IPs: 172.31.29.69

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

9°C Mostly clear 06:41 13-12-2025

Step-51- Here we seen the deployment of our application.

