Deploy Netflix Clone on Kubernetes

Phase 1 →deploy Netflix locally on ec2 (t2.large)

Phase 2 → Implementation of security with SonarQube and trivy

Phase 3 →Now we automate the whole deployment using by Jenkins pipeline

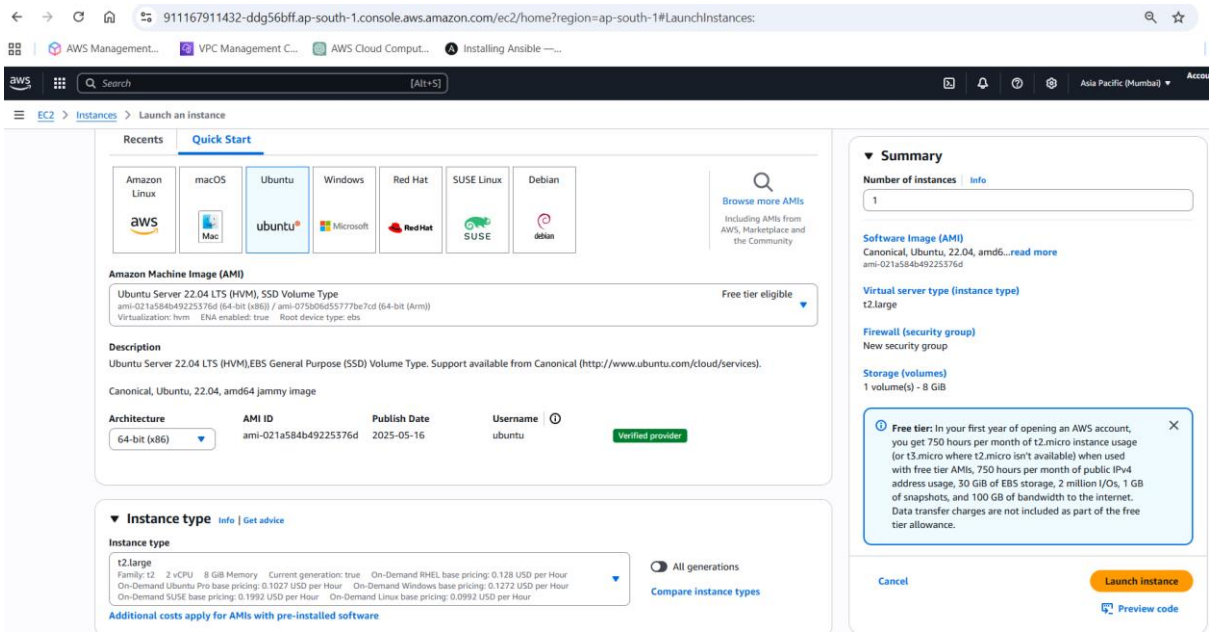
Phase 4 →Monitoring via Prometheus and Grafana

Phase 5 → Kubernetes →

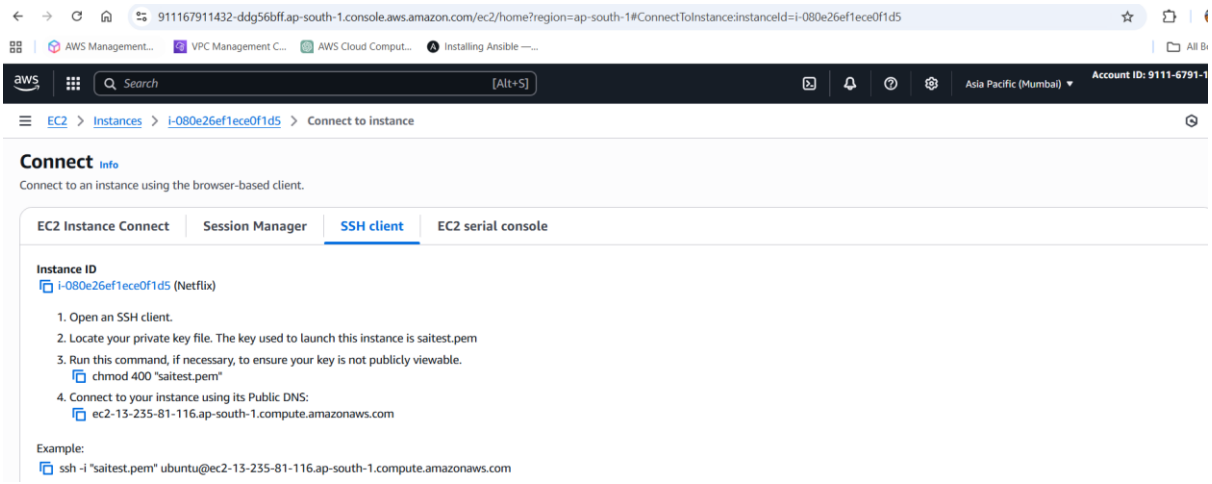
Phase 1 → deploy Netflix locally on ec2 (t2.large)

Step 1 →setup ec2

1. go to aws console and launch ubuntu 22.04 with t2.large and 35 gb of storage allocated with it do not forget to enable public Ip in vpc settings



2. let's connect to your ec2 via ssh using command "ssh -i "saitest.pem" ubuntu@ec2-13-235-81-116.ap-south-1.compute.amazonaws.com"



3. run the following commands

a. sudo su

b. apt update

clone the github repo by

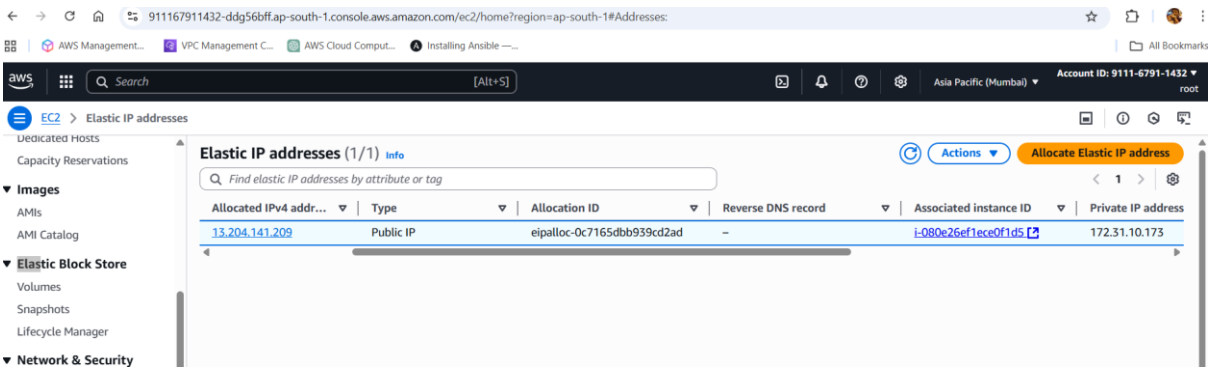
c. git clone https://github.com/sai241194/Deploy-Netflix-Clone-on-Kubernetes

Make sure to create an elastic ip address to associate with your instance

Go to **Elastic IPs → Actions → Associate Elastic IP address**.

Select your **instance ID** or the **private IP** of your EC2.

Save.



Step 2 → setup docker and build images

run the following commands to install docker→

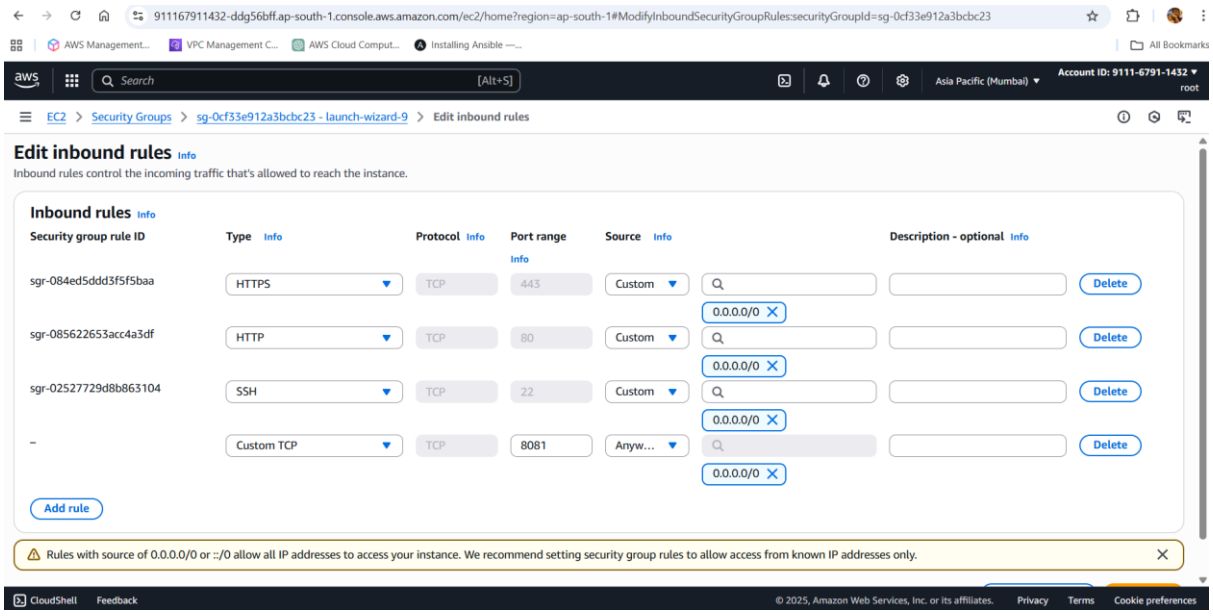
a. **apt-get install docker.io**
b. **usermod -aG docker $USER** # Replace with your username e.g 'ubuntu'
c. **newgrp docker**
d. **sudo chmod 777 /var/run/docker.sock** → #is used to grant full read, write, and execute permissions to all users for the Docker socket file, which allows unrestricted access to the Docker daemon and is a significant security risk.

run the following commands to build and run docker container →

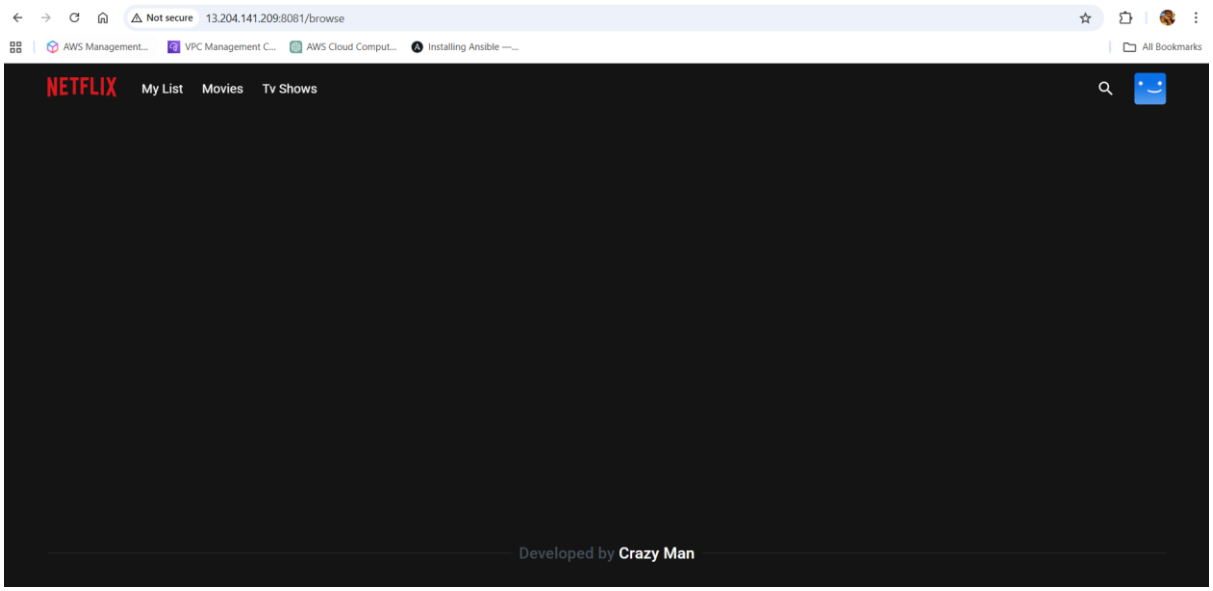
a. **docker build -t netflix .**

b. **docker run -d --name netflix -p 8081:80 netflix:latest** → this maps the container port to your ec2 port

c. **go to your ec2 →security groups →open the port 8081** by adding rule custom tcp = 8081



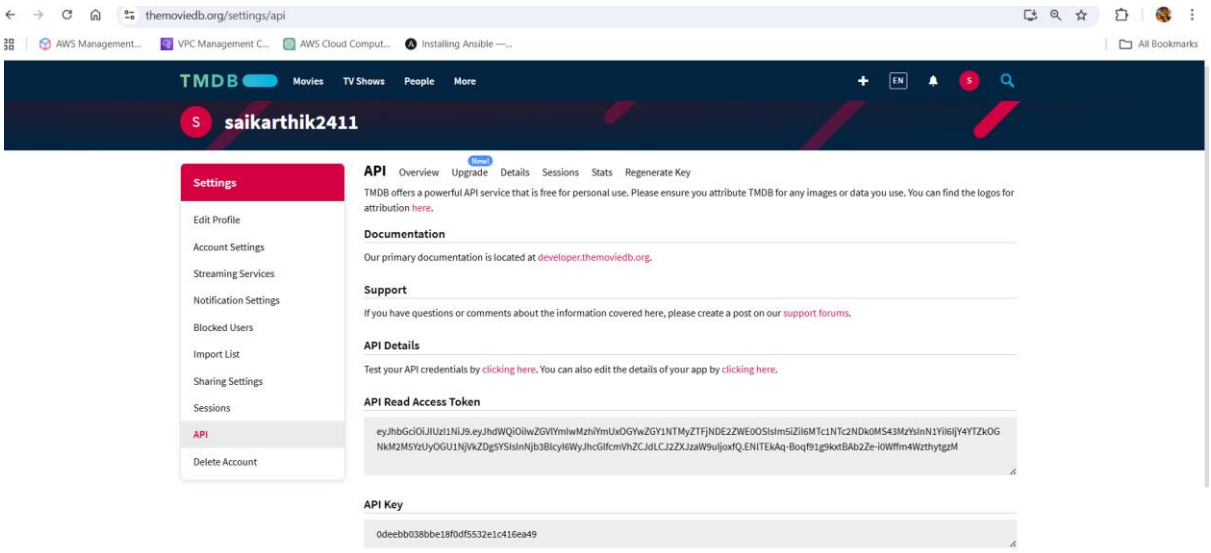
your application is running go to your ec2 copy public Ip and browse http://your_public_ip:8081 and it's open like this **(13.204.141.209:8081)**



Its shows a blank page of netflix because you do not have a API that's communicate with netflix database let's solve this

Step 3 → setup netflix API

- Open a web browser and navigate to TMDB (The Movie Database) website.
- Click on sign up
- Now enter your username and pass for sign in then go to your profile and select "Settings."
- Click on "API" from the left-side panel.
- Create a new API key by clicking "generate new API key " and accepting the terms and conditions.
- Provide the required basic details such as name and website URL of netflix and click "Submit."
- You will receive your TMDB API key.



Now delete the existing image by

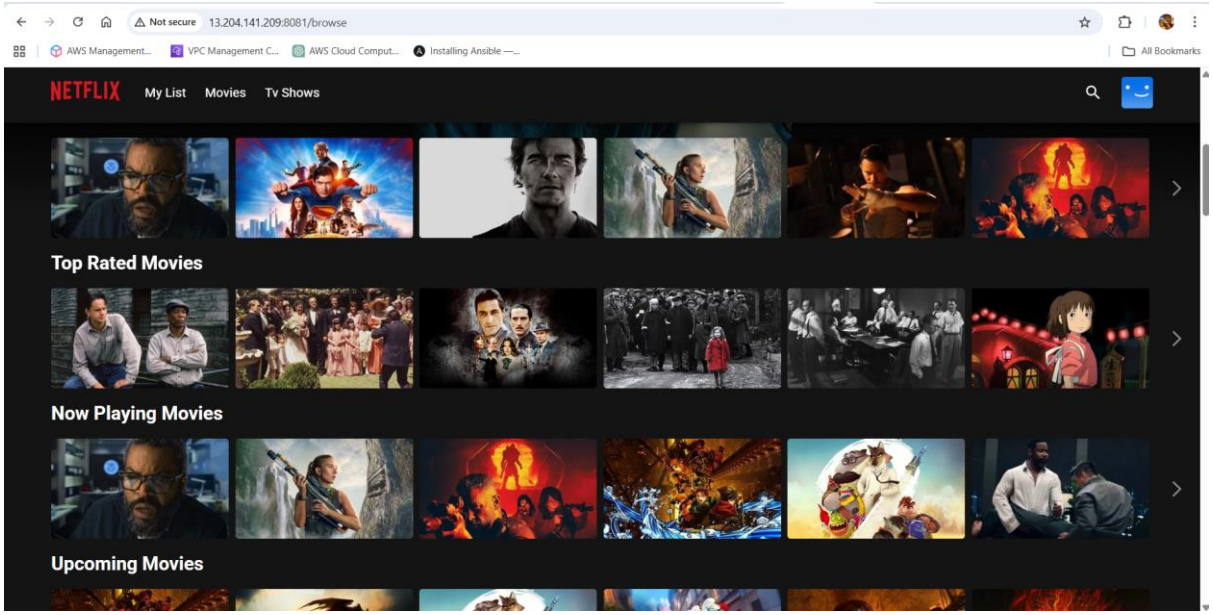
a. **docker stop <containerid>**
b. **docker rmi -f netflix**

Run the new container by following command

1. docker build -t netflix: latest --build-arg TMDB_V3_API_KEY=your_api_key .
2. docker run -d -p 8081:80 netflix

your container is created

now again browse the same URL you will see the whole netflix database is connected to your netflix clone app



Phase 2 → Implementation of security with SonarQube and trivy

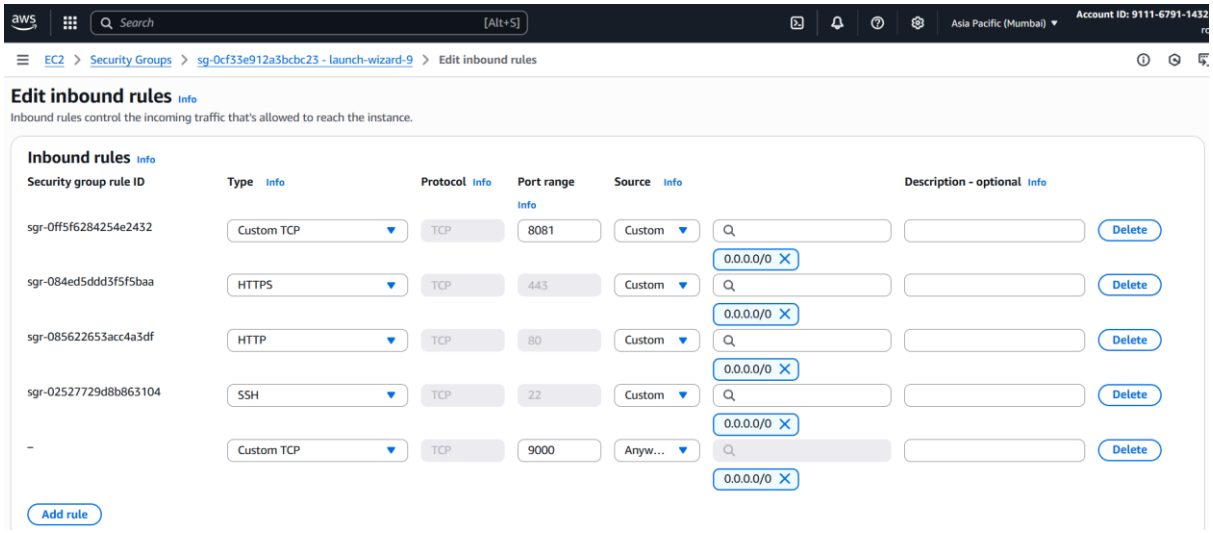
step 1 → setup SonarQube on your ec2

run the following commands to install and run the container of SonarQube on port no. 9000

a. **docker run -d --name sonar -p 9000:9000 sonarqube:lts-community**

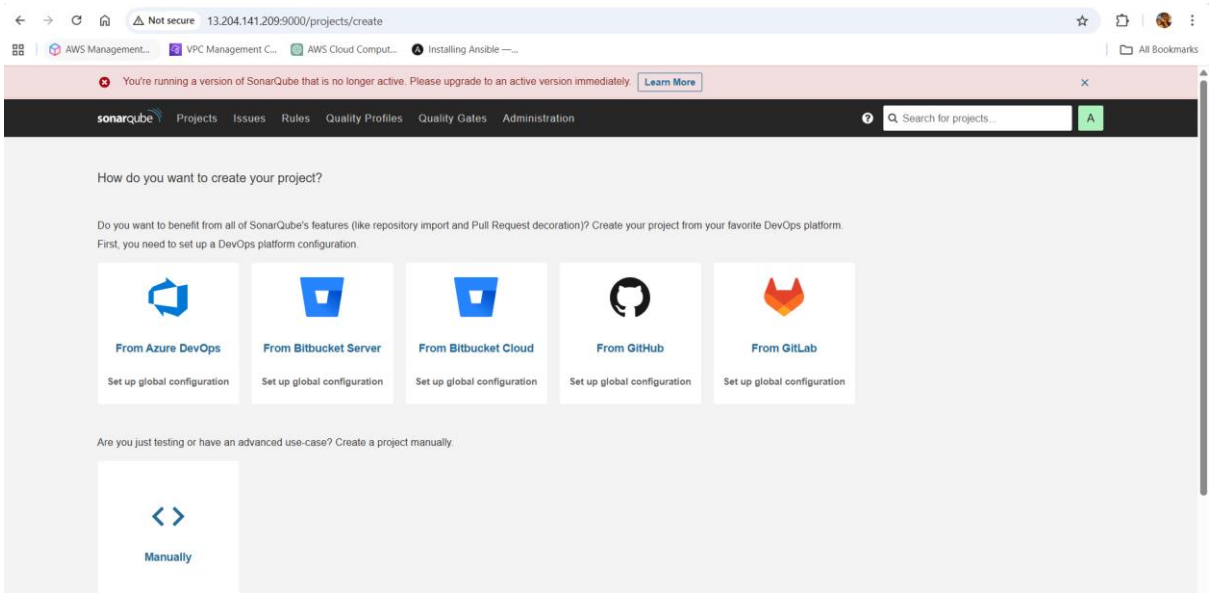
```
root@ip-172-31-10-173:~/Deploy-Netflix-Clone-on-Kubernetes# docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
a3be5d4ce401: Pull complete
467054790490: Pull complete
6898721bb2fa: Pull complete
e28927af8b0b: Pull complete
be0543ed4aff: Pull complete
ebcf7b760392: Pull complete
eba58e7bdf02: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:f98a29395d776ed67a3c94ed20a05be6a0df7da1bb4470355d88745e9aa70f26
Status: Downloaded newer image for sonarqube:lts-community
8622911bafab1e2c023b9a71b2d57323901c5ac4879bc393e381b4a6bd717d72
root@ip-172-31-10-173:~/Deploy-Netflix-Clone-on-Kubernetes#
```

go and open port 9000 for SonarQube to your ec2 security group



Now browse **http://your_public_ip:9000** you will see **SonarQube is running (http://13.204.141.209:9000)**

username =admin
password=admin



Step 2 → setup Trivy

run the following commands

a. sudo apt-get install wget apt-transport-https gnupg lsb-release -y

**b. wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | **

sudo gpg --dearmor -o /usr/share/keyrings/trivy.gpg

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

**https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | **

sudo tee /etc/apt/sources.list.d/trivy.list

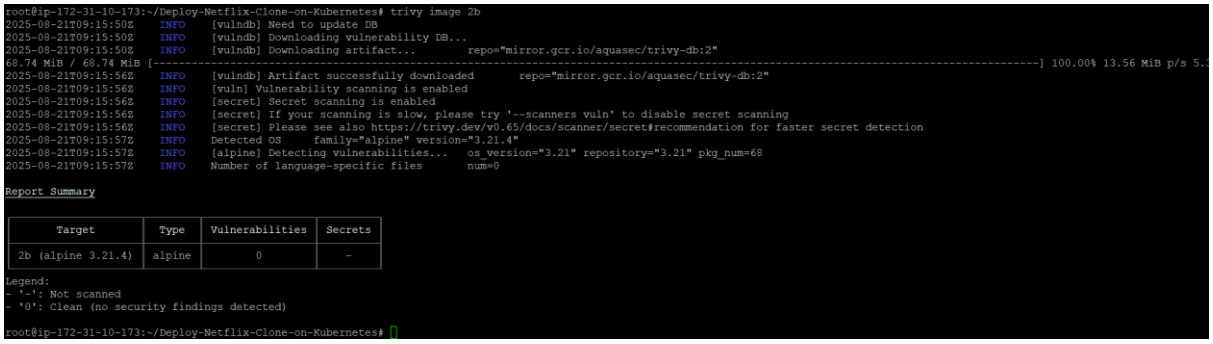
d. sudo apt-get update

e. sudo apt-get install trivy -y

now your trivy is ready to check and scan your image for any vulnerabilities

to check run the following commands

a. **trivy image <image id>**



Phase 3 →Now we automate the whole deployment using by jenkins pipeline

Step 1 →install and configure jenkins via terminal …jenkins also required java for run itself so we install java and then jenkins

commands to run on the terminal →

setup java

1. **sudo apt update**
2. **sudo apt install fontconfig openjdk-17-jre**

3. **java -version**

```
root@ip-172-31-10-173:~/Deploy-Netflix-Clone-on-Kubernetes# java -version
openjdk version "17.0.16" 2025-07-15
OpenJDK Runtime Environment (build 17.0.16+8-Ubuntu-0ubuntu122.04.1)
OpenJDK 64-Bit Server VM (build 17.0.16+8-Ubuntu-0ubuntu122.04.1, mixed mode, sharing)
root@ip-172-31-10-173:~/Deploy-Netflix-Clone-on-Kubernetes#
```

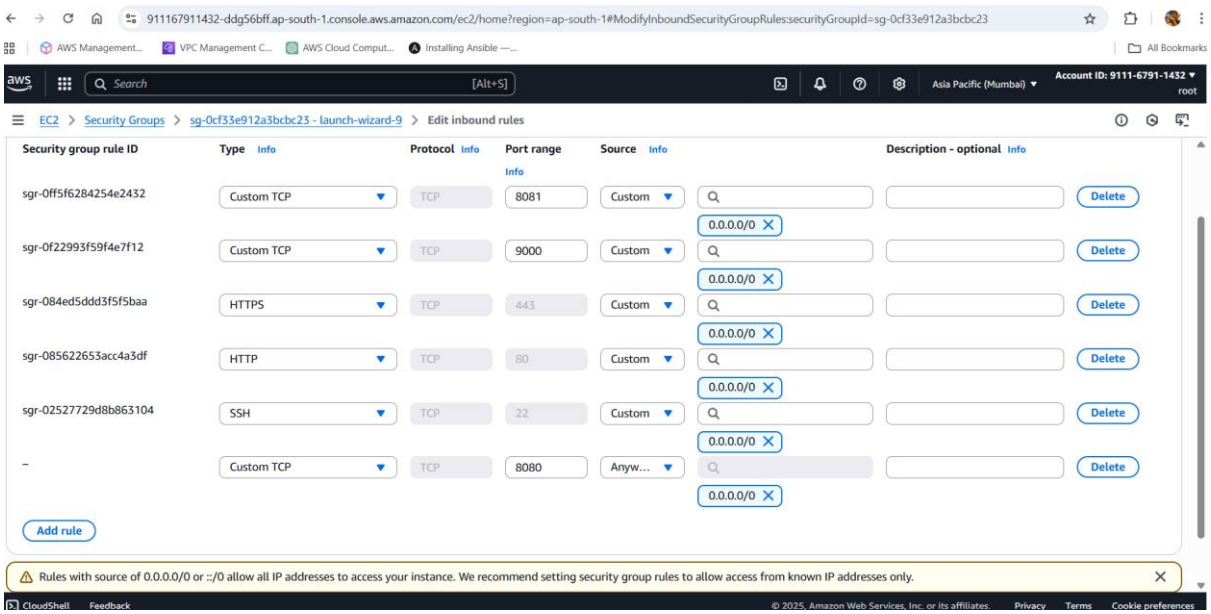
Setup jenkins

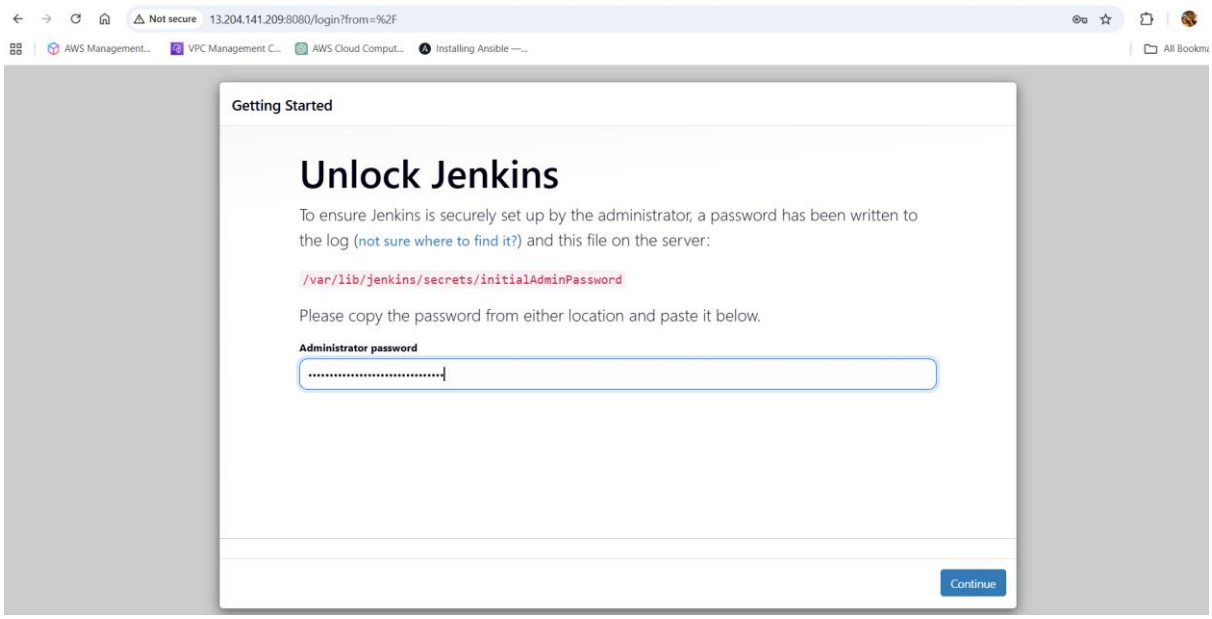
1. **sudo wget -O /usr/share/keyrings/jenkins-keyring.asc **
   https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
2. **echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] **
   **https://pkg.jenkins.io/debian-stable binary/ | sudo tee **
   /etc/apt/sources.list.d/jenkins.list > /dev/null
3. **sudo apt-get update**
4. **sudo apt-get install jenkins -y**
5. **sudo systemctl start jenkins**
6. **sudo systemctl enable jenkins**
7. **sudo systemctl status jenkins**
8. Access Jenkins in a web browser using the public IP of your EC2 instance http://publicIp:8080

   (http:// 13.203.185.228:8080/)

```
oot@ip-172-31-10-173:~/Deploy-Netflix-Clone-on-Kubernetes# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2025-08-21 09:35:05 UTC; 51s ago
   Main PID: 11427 (java)
      Tasks: 50 (limit: 9505)
     Memory: 943.0M
        CPU: 17.462s
     CGroup: /system.slice/jenkins.service
             └─11427 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

ug 21 09:34:59 ip-172-31-10-173 jenkins[11427]: 8fc9e843f6a1464c8b54ba5ec76dedd5
ug 21 09:34:59 ip-172-31-10-173 jenkins[11427]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
ug 21 09:34:59 ip-172-31-10-173 jenkins[11427]: ************************************************************
ug 21 09:34:59 ip-172-31-10-173 jenkins[11427]: ************************************************************
ug 21 09:34:59 ip-172-31-10-173 jenkins[11427]: ************************************************************
ug 21 09:35:05 ip-172-31-10-173 jenkins[11427]: 2025-08-21 09:35:05.127+0000 [id=31]       INFO       jenkins.InitReactorRunner$1#onAttained: Completed initialization
ug 21 09:35:05 ip-172-31-10-173 jenkins[11427]: 2025-08-21 09:35:05.143+0000 [id=23]       INFO       hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and running
ug 21 09:35:05 ip-172-31-10-173 systemd[1]: Started Jenkins Continuous Integration Server.
ug 21 09:35:06 ip-172-31-10-173 jenkins[11427]: 2025-08-21 09:35:06.759+0000 [id=49]       INFO       h.m.DownloadService$Downloadable#load: Obtained the updated data file for hud
ug 21 09:35:06 ip-172-31-10-173 jenkins[11427]: 2025-08-21 09:35:06.759+0000 [id=49]       INFO       hudson.util.Retrier#start: Performed the action check updates server successfu
ines 1-20/20 (END)
```

Make sure to add Port 8080 in security group





Login as admin user and go to plugins and install below

Install some suggested plugins for pipeline to run without errors

1 Eclipse Temurin Installer (Install without restart)

2 SonarQube Scanner (Install without restart)

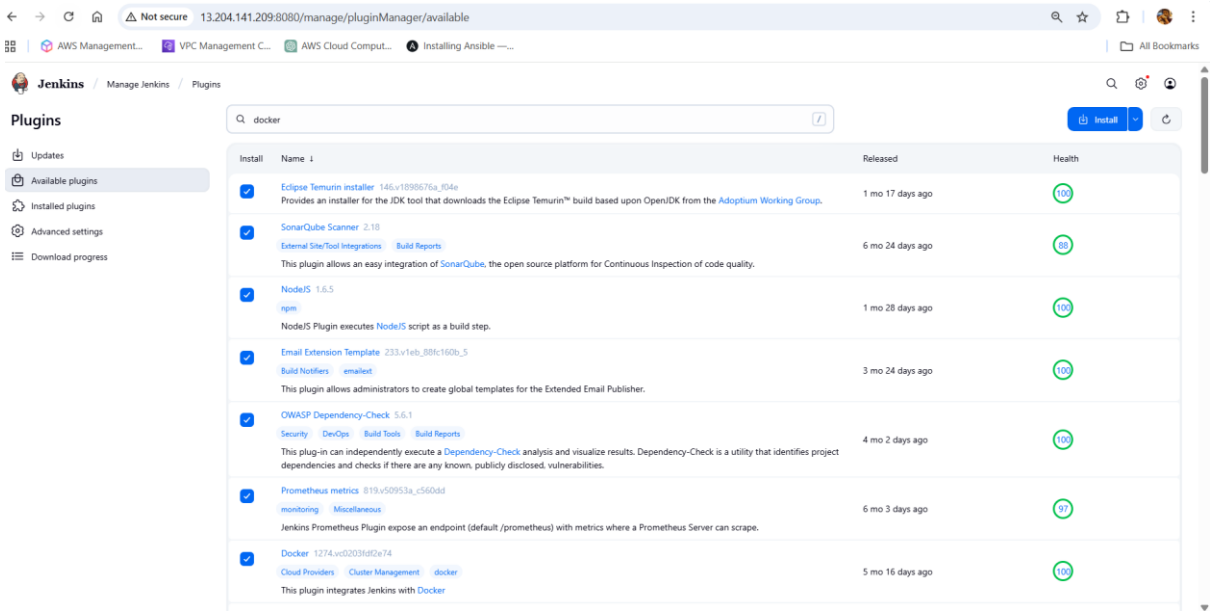
3 Nodejs Plugin (Install Without restart)

4 Email Extension Plugin

5 OWASP

6 Prometheus metrics →to monitor Jenkins on Grafana dashboard

7 Download all the docker related plugins

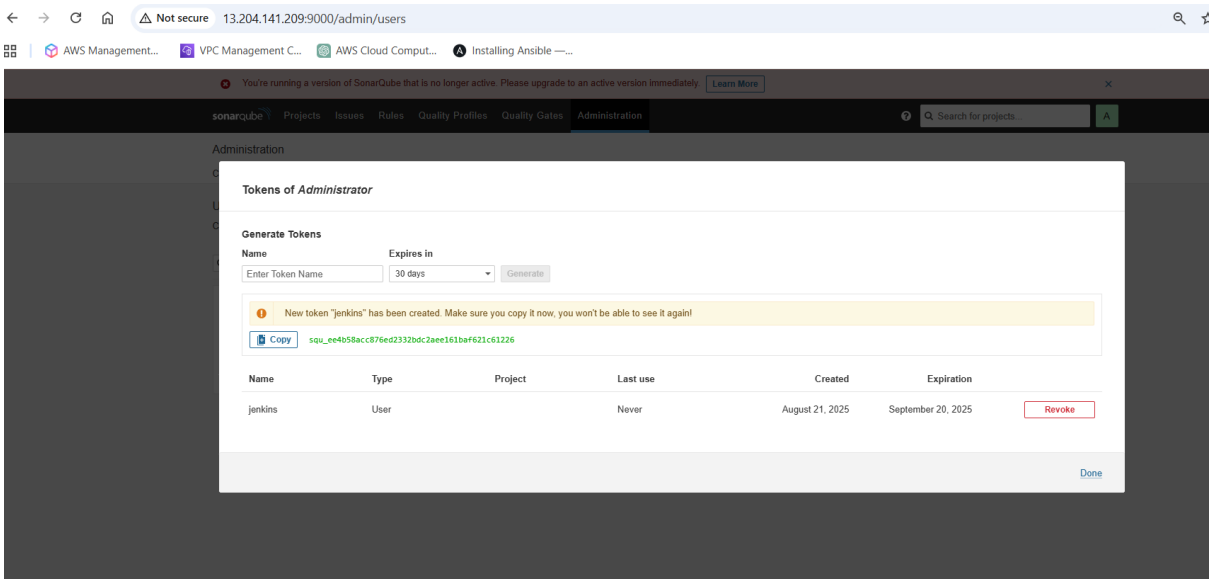


Step 2 → add credentials of SonarQube and Docker

1st we generate a token for SonarQube to use in Jenkins credentials as secret text

setup SonarQube credentials

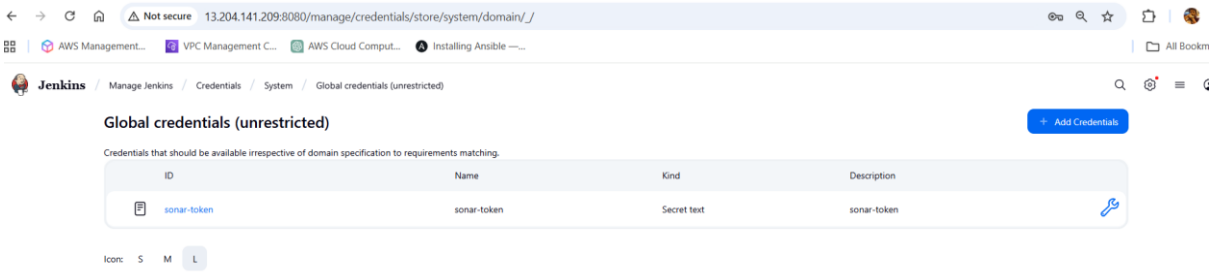
1. go to http://publicip:9000
2. now enter your username and password
3. click on Administration→ security →users →token →generate token



4. copy the token and go to your Jenkins →manage Jenkins →credentials →global →add credentials

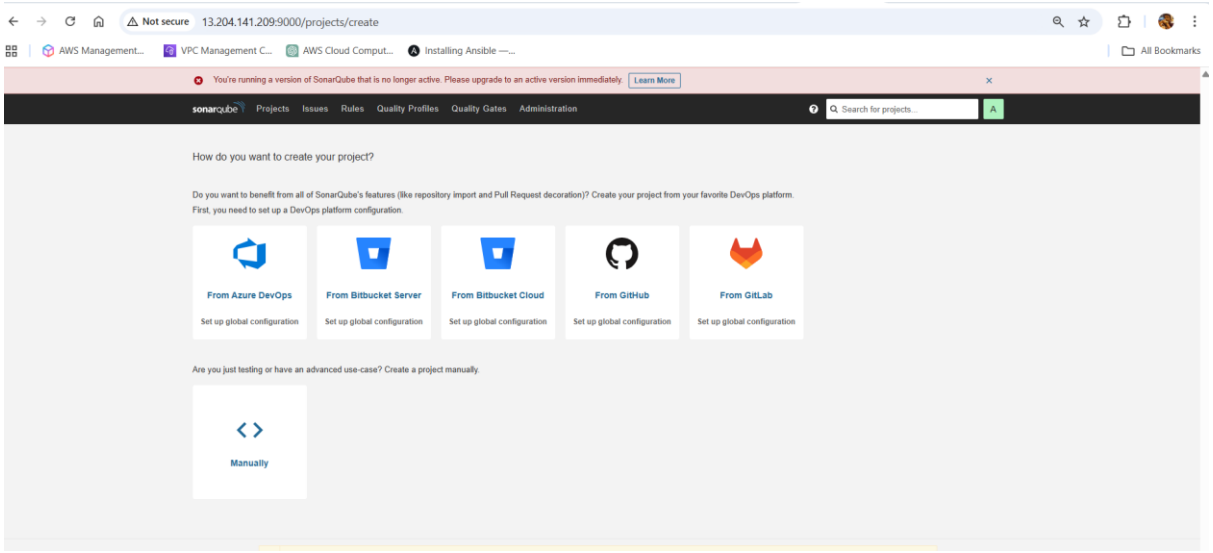
5. select secret text from dropdown

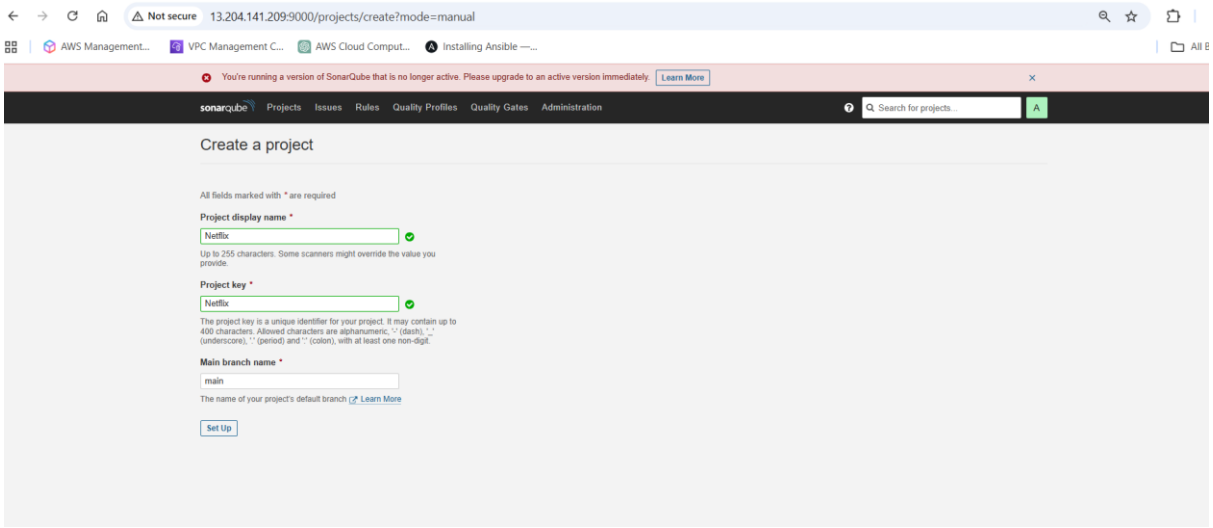
6. secret text ==your token, id =sonar-token →click on create

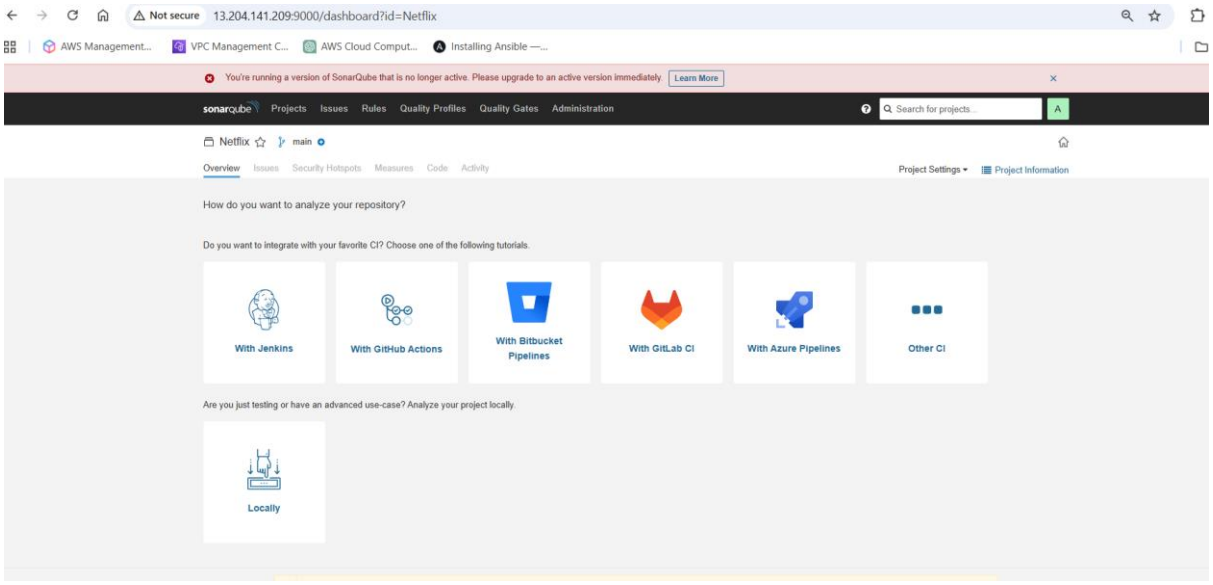


setup projects in SonarQube for Jenkins

1. go to your SonarQube server
2. click on projects-Manually
3. in the name field type Netflix
4. click on set up

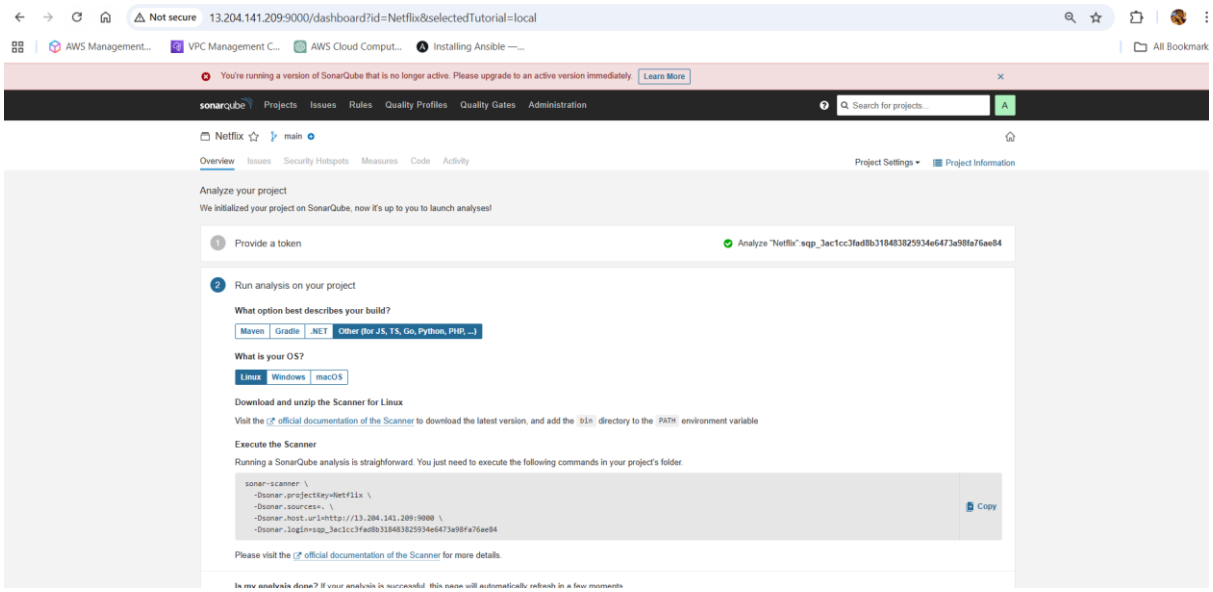

Not secure
13.204.141.209:9000/projects/create
AWS Management...
VPC Management C...
AWS Cloud Comput...
Installing Ansible —...
All Bookmarks
You're running a version of SonarQube that is no longer active. Please upgrade to an active version immediately.
Learn More
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
A
How do you want to create your project?
Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.
First, you need to set up a DevOps platform configuration.
From Azure DevOps
Set up global configuration
From Bitbucket Server
Set up global configuration
From Bitbucket Cloud
Set up global configuration
From GitHub
Set up global configuration
From GitLab
Set up global configuration
Are you just testing or have an advanced use-case? Create a project manually.
Manually


Not secure
13.204.141.209:9000/projects/create?mode=manual
AWS Management...
VPC Management C...
AWS Cloud Comput...
Installing Ansible —...
You're running a version of SonarQube that is no longer active. Please upgrade to an active version immediately.
Learn More
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
A
Create a project
All fields marked with * are required
Project display name *
Netflix
Up to 255 characters. Some scanners might override the value you provide.
Project key *
Netflix
The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.
Main branch name *
main
The name of your project's default branch
Learn More
Set Up



click on this option locally

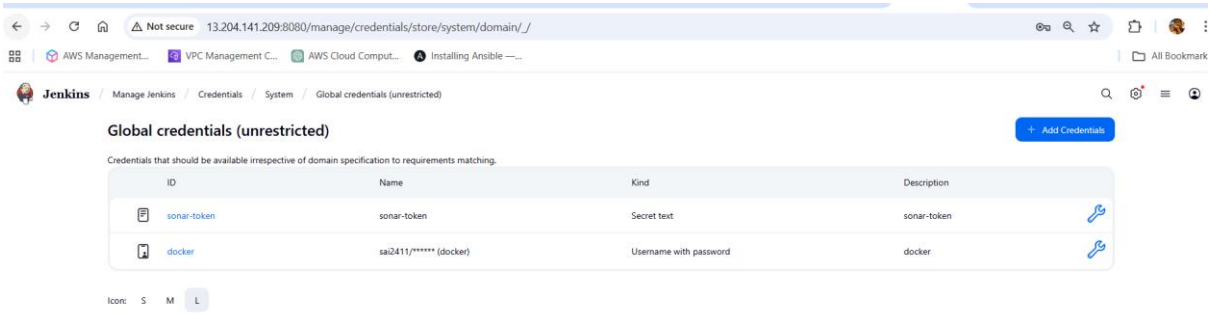
Then click on generate-continue



select the os and the following commands used in Jenkins pipeline code that set SonarQube to watch over Jenkins

Setup docker credentials

1. go to your Jenkins →manage Jenkins →credentials →global →add credentials
2. provide your username and password of your docker hub
3. id==docker

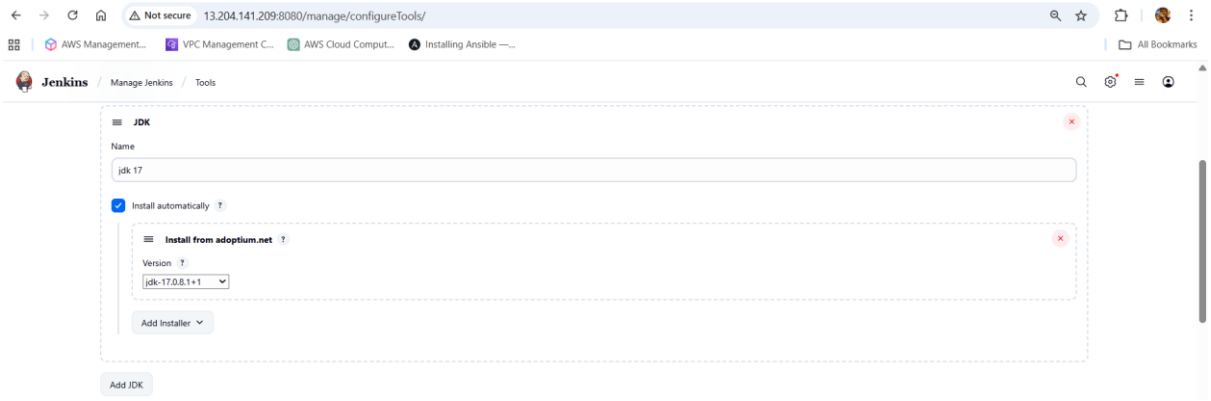


Step 3→Now we are going to setup tools for Jenkins

go to manage Jenkins → tools

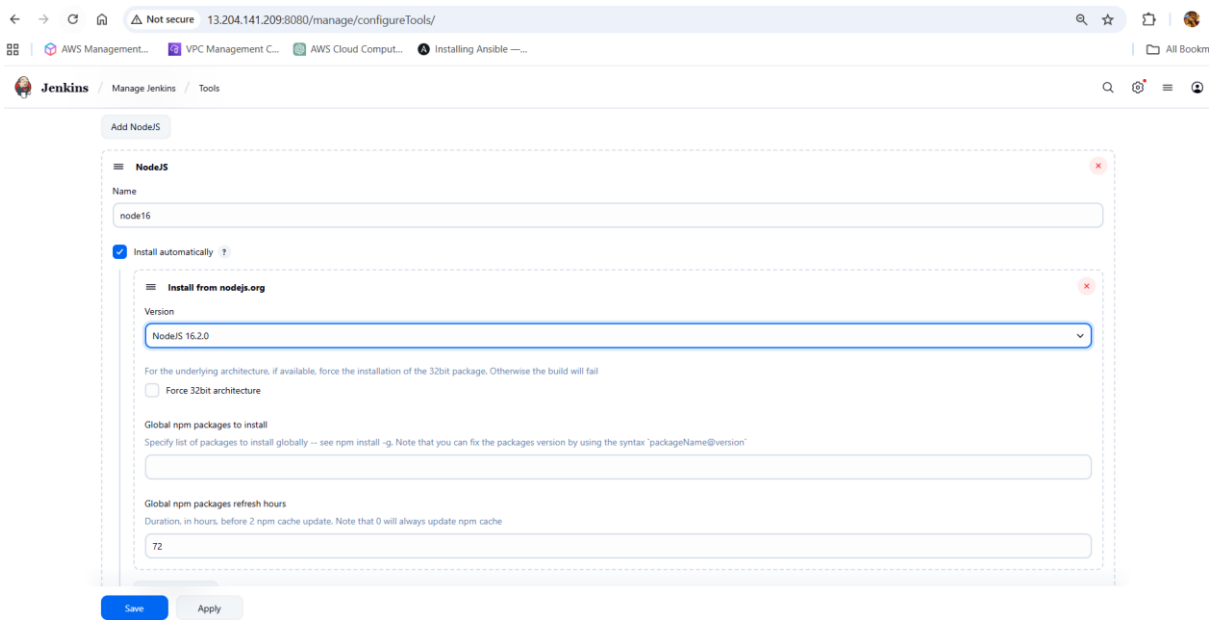
a. add jdk

1. click on add jdk and select installer adoptium.net
2. choose jdk 17.0.8.1+1version and in name section enter jdk 17



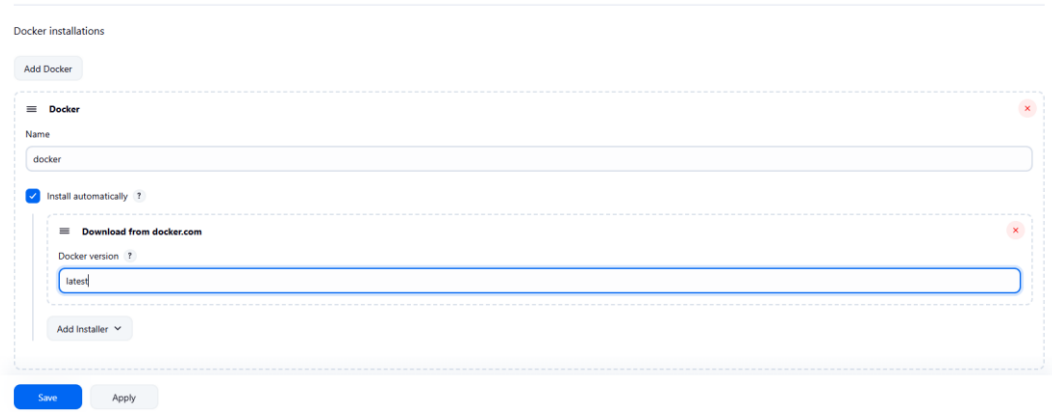
b. add node js

1. click on add NodeJS
2. enter node16 in name section
3. choose version NodeJS 16.2.0



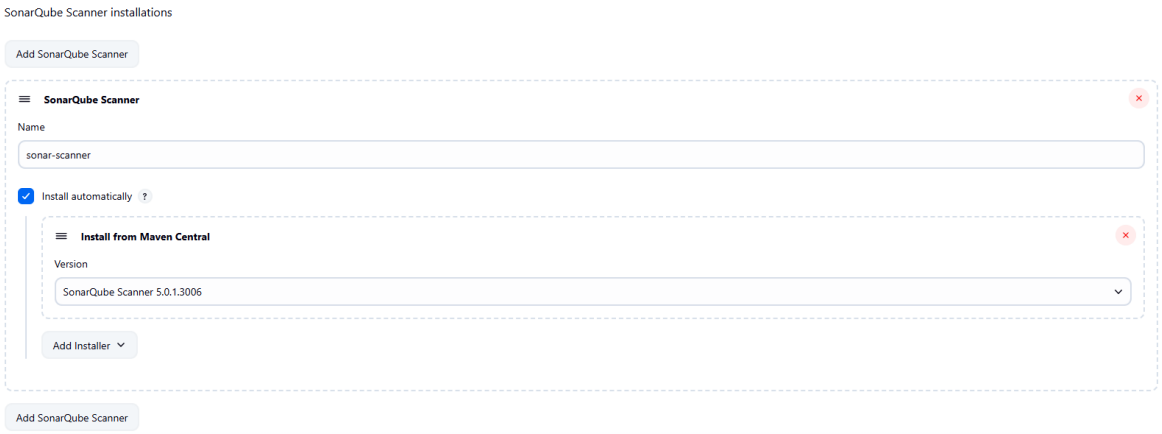
c. add docker →

1. click on add docker
2. name==docker
3. add installer ==download from docker.com



d. add SonarQube →

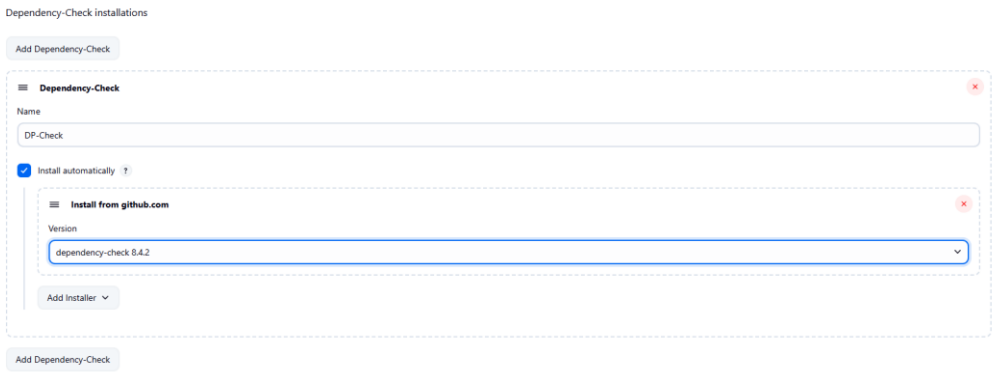
1. add sonar scanner
2. name ==sonar-scanner



e. add OWASP dependency check

Adding the Dependency-Check plugin in the “Tools” section of Jenkins allows you to perform automated security checks on the dependencies used by your application

1. add dependency check
2. name == DP-Check
3. from add installer select install from github.com



Click on apply and save at last

Step 4 →Configure global setting for SonarQube

1. go to manage Jenkins →Configure global setting →add SonarQube servers
2. n**ame ==sonar-server**
3. **server URL==http://public_ip:9000**

4. **server authentication token == sonar-token**

Step 5 →let's run the Pipeline →

1. **go to new item →select pipeline** →in the name section type **netflix**
2. scroll down to the pipeline script and copy paste the following code

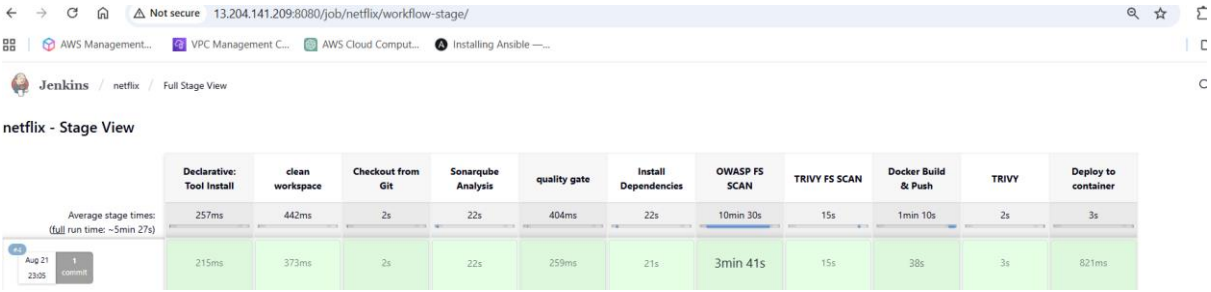
```
pipeline{
    agent any
    tools{
        jdk 'jdk 17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanWs()
            }
```

```
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/sai241194/Deploy-Netflix-Clone-on-Kubernetes.git'
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Netflix \
                    -Dsonar.projectKey=Netflix '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh "npm install"
```

```
        }
    }
    stage('OWASP FS SCAN') {
        steps {
            dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-Check'
            dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
        }
    }
    stage('TRIVY FS SCAN') {
        steps {
            sh "trivy fs . > trivyfs.txt"
        }
    }
    stage("Docker Build & Push"){
        steps{
            script{
                withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){
                    sh "docker build --build-arg TMDB_V3_API_KEY=<yourapikey> -t netflix ."
                    sh "docker tag netflix sai2411/netflix:latest "
                    sh "docker push sai2411/netflix:latest "
                }
            }
        }
    }
```

```
        stage("TRIVY"){
            steps{
                sh "trivy image sai2411/netflix:latest > trivyimage.txt"
            }
        }
        stage('Deploy to container'){
            steps{
                sh 'docker run -d --name netflix1 -p 8082:80 sai2411/netflix:latest'
            }
        }
    }
}
```

3. Your Pipeline is started to build the following docker image and send it to dockerhub with all security checks

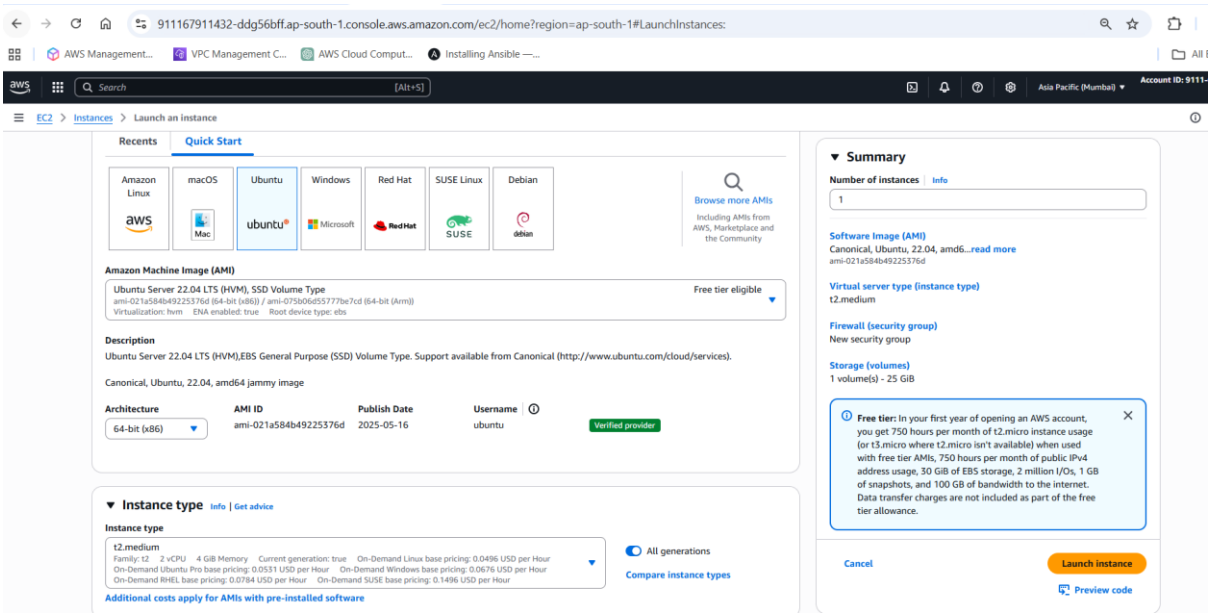


Phase 4 →Monitoring via Prometheus and grafana

Step 1 →Setup another server or EC2 for monitoring

1. go to ec2 console and launch an instance having a base image of ubuntu and with t2.medium specs because Minimum Requirements to Install Prometheus:

- 2 CPU cores.
- 4 GB of memory.
- 25 GB of free disk space



Step 2 →Installing Prometheus:

1. **First, create a dedicated Linux user for Prometheus and download Prometheus**

   a.)sudo useradd --system --no-create-home --shell /bin/false Prometheus

   b.)wget https://github.com/prometheus/prometheus/releases/download/v2.47.1/prometheus-2.47.1.linux-amd64.tar.gz

2. **Extract Prometheus files, move them, and create directories:**

   a.tar -xvf prometheus-2.47.1.linux-amd64.tar.gz

   b.cd prometheus-2.47.1.linux-amd64/

c. sudo mkdir -p /data /etc/prometheus

d. sudo mv prometheus promtool /usr/local/bin/

e. sudo mv consoles/ console_libraries/ /etc/prometheus/

f. sudo mv prometheus.yml /etc/prometheus/prometheus.yml

3. Set ownership for directories:

a. sudo chown -R prometheus:prometheus /etc/prometheus/ /data/

4. Create a systemd unit configuration file for Prometheus:

a. sudo nano /etc/systemd/system/prometheus.service

Add the following code to the prometheus.service 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
```

b. press →**ctrl+o** #for save and then **ctrl+x** #for exit from the file

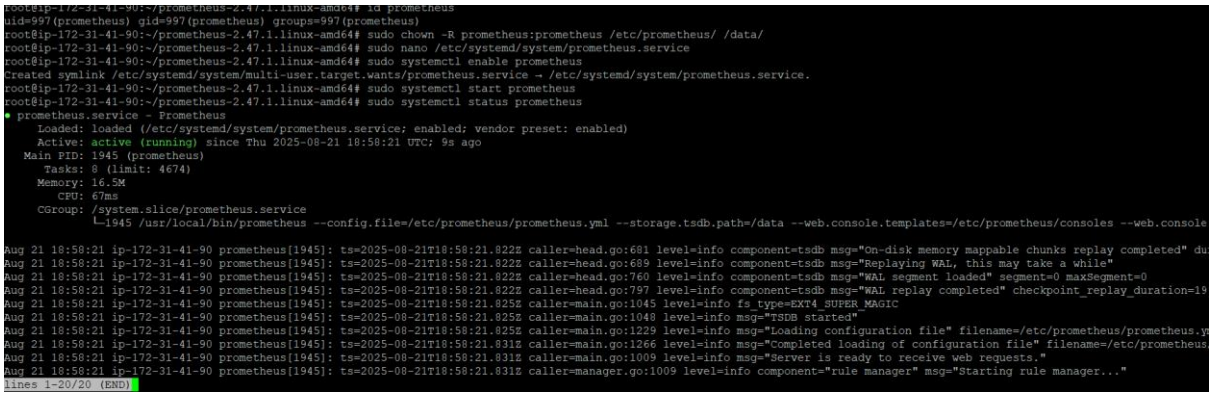
Here’s a explanation of the key parts in this above file:

- User and Group specify the Linux user and group under which Prometheus will run.
- ExecStart is where you specify the Prometheus binary path, the location of the configuration file (prometheus.yml), the storage directory, and other settings.
- web.listen-address configures Prometheus to listen on all network interfaces on port 9090.
- web.enable-lifecycle allows for management of Prometheus through API calls

5. Enable and start Prometheus:

a. sudo systemctl enable prometheus
b. sudo systemctl start prometheus

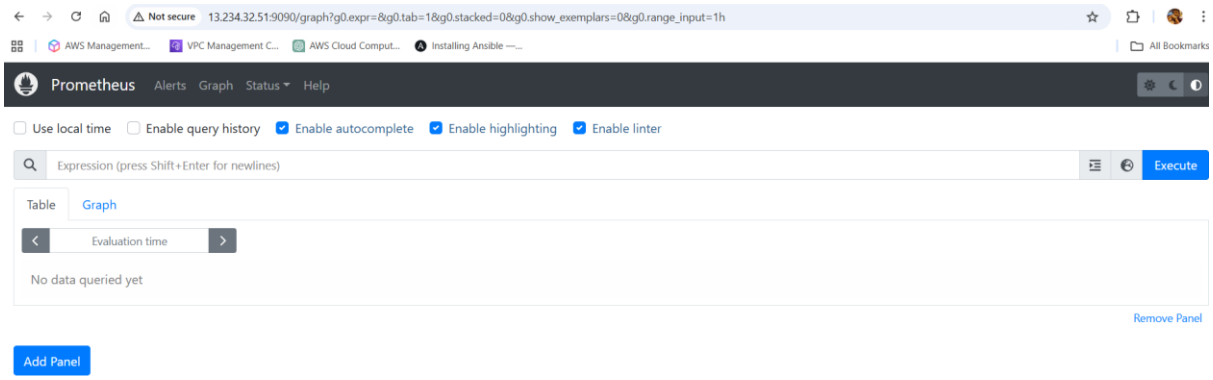
c. sudo systemctl status Prometheus



Now go to your security group of your ec2 to enable port 9090 in which Prometheus will run

go to → http://public_ip:9090 to see the webpage of Prometheus

(http://13.234.32.51:9090/)



Step 3 → Installing Node Exporter:

Run the following commands for installation

1. **Create a system user for Node Exporter and download Node Exporter:**
   a. **sudo useradd --system --no-create-home --shell /bin/false node_exporter**
   b. wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz

2. **Extract Node Exporter files, move the binary, and clean up:**

   a. tar -xvf node_exporter-1.6.1.linux-amd64.tar.gz

   b. sudo mv node_exporter-1.6.1.linux-amd64/node_exporter /usr/local/bin/

   c. rm -rf node_exporter*

3. **Create a systemd unit configuration file for Node Exporter:**

   a. sudo nano /etc/systemd/system/node_exporter.service

   add the following code to the node_exporter.service file:

   provide more detailed information about what might be going wrong. For example:

[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

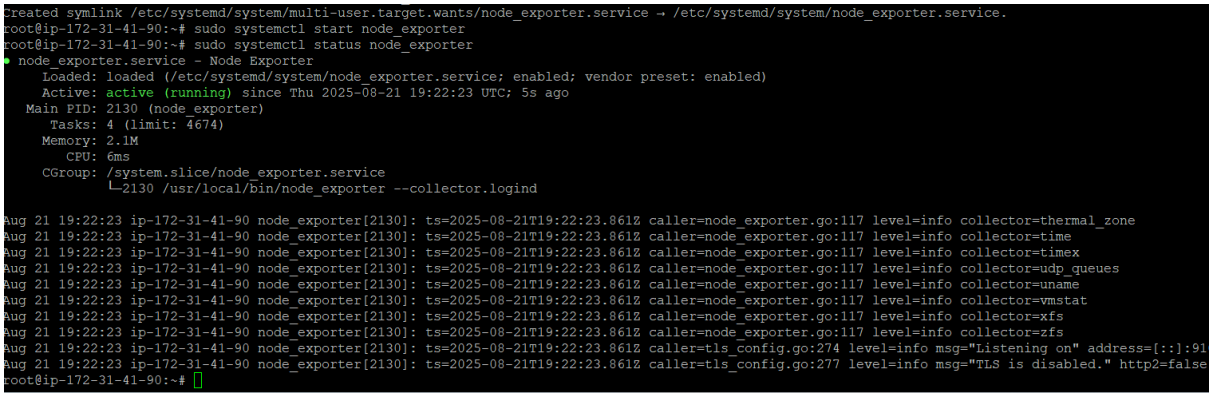
b.) press → ctrl+o then ctrl+x

4. Enable and start Node Exporter:

a. sudo systemctl enable node_exporter
b. sudo systemctl start node_exporter

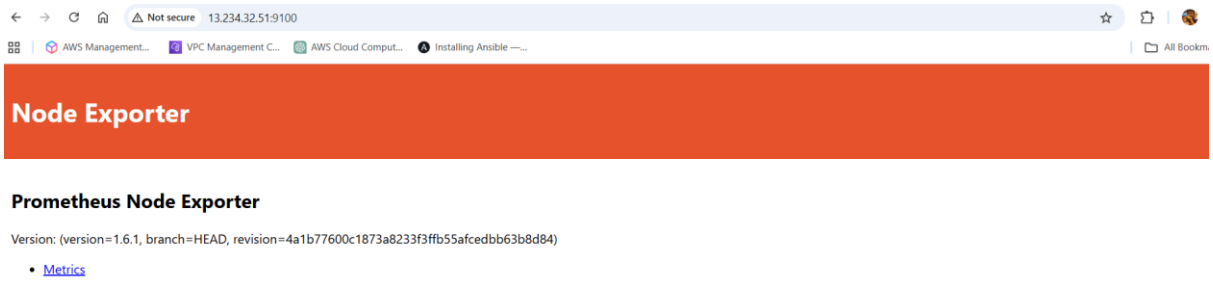
c. sudo systemctl status node_exporter

node exporter service is now running



Enable 9100 port no in security groups.

go to → http://public_ip:9100 to see the webpage of Node Exporter (http://13.234.32.51:9100/)



Step 4→Configure Prometheus Plugin Integration:

1. go to your EC2 and run →**cd /etc/prometheus**
2. you have to edit the prometheus.yml file to moniter anything

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']

  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['13.204.141.209:8080']
```

add the above code with proper indentation like this →

root@ip-172-31-41-90: /etc/prometheus

```
# my global config
global:
  scrape_interval: 15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

# 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: ['localhost:9100']

  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['13.204.141.209:8080']

~
~
~
~
~
```

a. Check the validity of the configuration file →

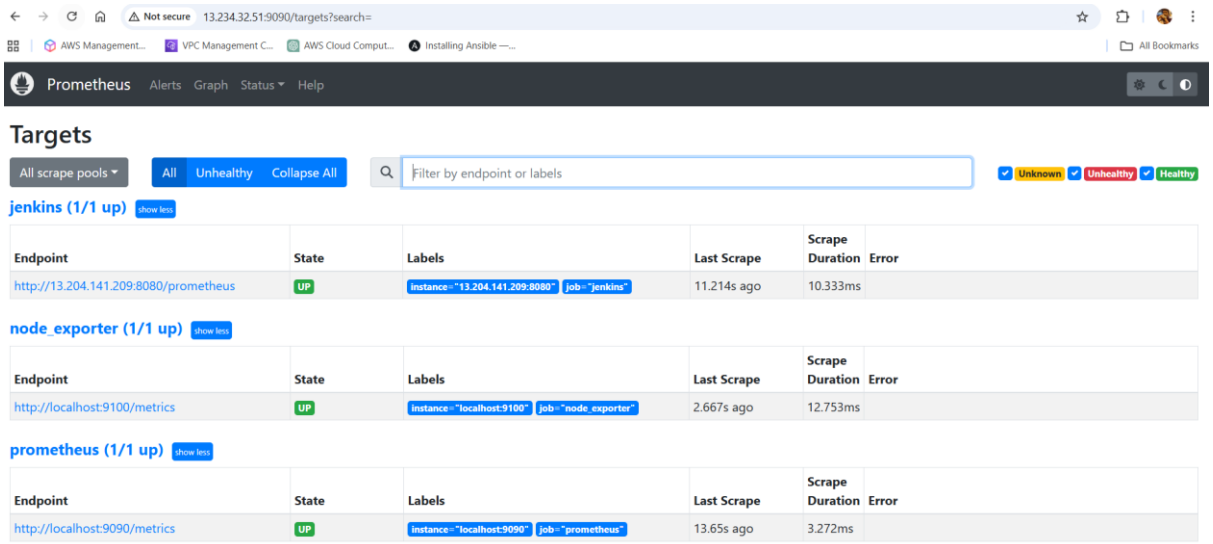
promtool check config /etc/prometheus/prometheus.yml

```
root@ip-172-31-41-90:/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

root@ip-172-31-41-90:/etc/prometheus#
```

b. Reload the Prometheus configuration without restarting →

curl -X POST http://localhost:9090/-/reload

go to your Prometheus tab again and click on status and select targets you will there is three targets present as we enter in yaml file for monitoring



Step 5 →Setup Grafana →

Install Dependencies:

a. sudo apt-get update
b. sudo apt-get install -y apt-transport-https software-properties-common

Add the GPG Key for Grafana:

a. wget -q -O- https://packages.grafana.com/gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/grafana.gpg

Add the repository for Grafana stable releases:

a. echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee /etc/apt/sources.list.d/grafana.list

Update the package list , install and start Grafana:

a. sudo apt-get update
b. sudo apt-get -y install grafana

c. sudo systemctl enable grafana-server

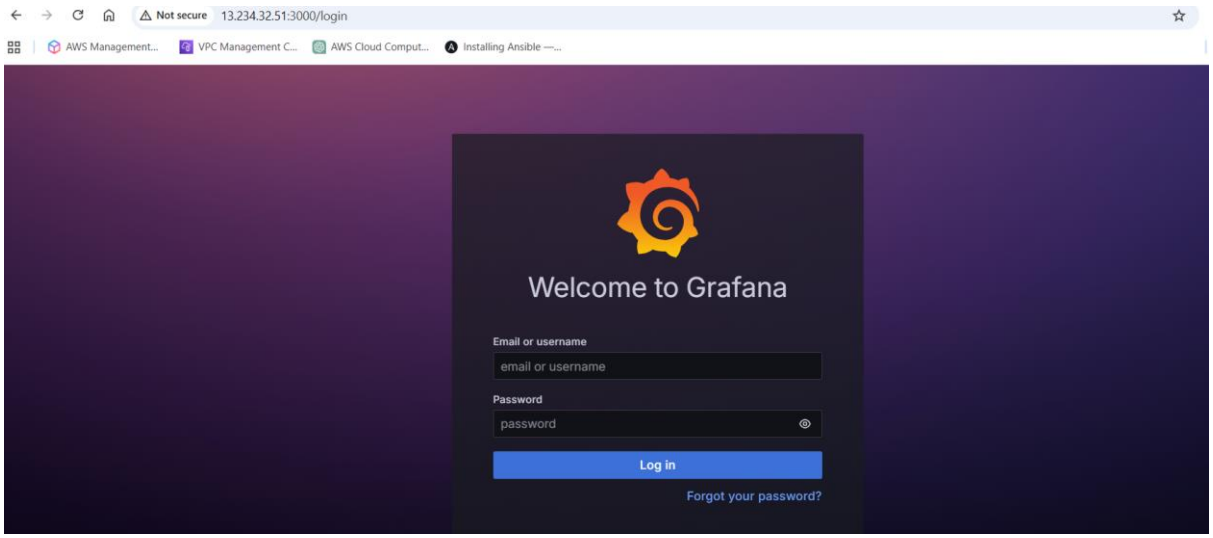
d. sudo systemctl start grafana-server

e. sudo systemctl status grafana-server

```
oot@ip-172-31-41-90:~# sudo systemctl start grafana-server
oot@ip-172-31-41-90:~# 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 Thu 2025-08-21 19:59:16 UTC; 6s ago
       Docs: http://docs.grafana.org
   Main PID: 3725 (grafana)
      Tasks: 7 (limit: 4674)
     Memory: 43.1M
        CPU: 1.982s
     CGroup: /system.slice/grafana-server.service
             └─3725 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:default.paths

ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.807126303Z level=info msg="Migration successfully executed" id="Update uid column
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.815293805Z level=info msg="Executing migration" id="Add unique index migration_uid
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.816368459Z level=info msg="Migration successfully executed" id="Add unique index m
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.8247119682 level=info msg="Executing migration" id="add migration run uid column"
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.8308527Z level=info msg="Migration successfully executed" id="add migration run ui
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.8380402552 level=info msg="Executing migration" id="Update uid column values for m
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.838168513Z level=info msg="Migration successfully executed" id="Update uid column
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.8455149032 level=info msg="Executing migration" id="Add unique index migration_run
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.846384326Z level=info msg="Migration successfully executed" id="Add unique index m
ug 21 19:59:22 ip-172-31-41-90 grafana[3725]: logger=migrator t=2025-08-21T19:59:22.856252867Z level=info msg="Executing migration" id="Rename table cloud_migration t
ines 1-21/21 (END)
```

Now go to your ec2 security group and open port no. 3000 in which grafana runs

Go and browse http://public_ip:3000 to access your grafana web interface



username = admin, password =admin

Step 6 →Add Prometheus Data Source:

To visualize metrics, you need to add a data source.

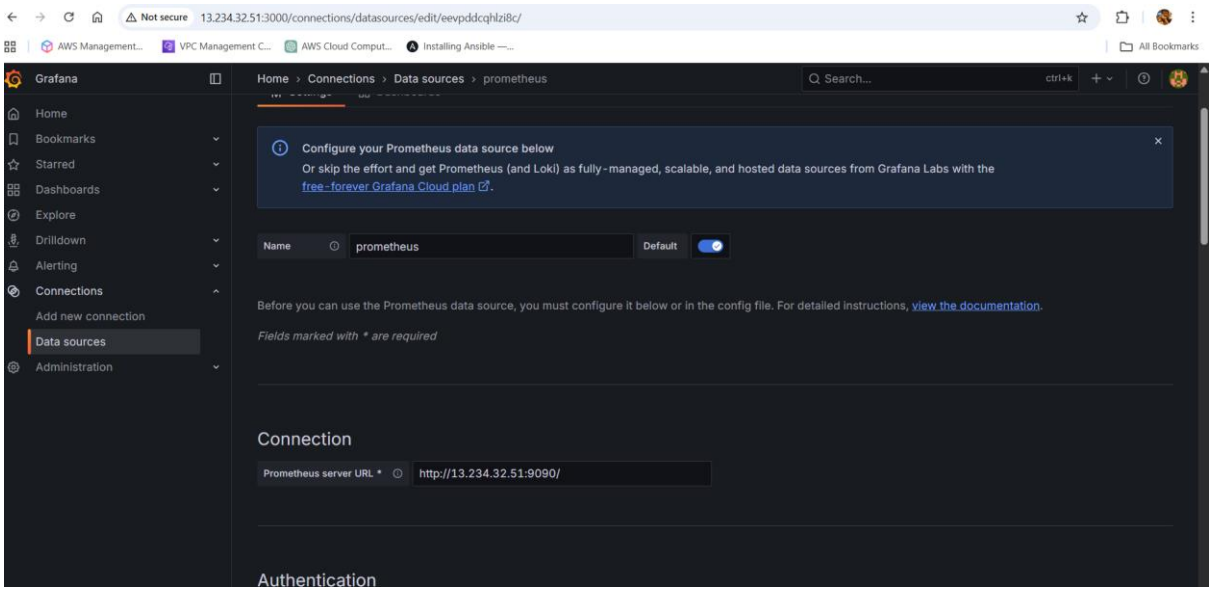
Follow these steps:

- Click on the gear icon (⚙) in the left sidebar to open the “Configuration” menu.
- Select “Data Sources.”
- Click on the “Add data source” button.

- Choose “Prometheus” as the data source type.

- In the “HTTP” section: Set the “URL” to http://localhost:9090 (assuming Prometheus is running on the same server).

- Click the “Save & Test” button to ensure the data source is working.



Step 7 → Import a Dashboard

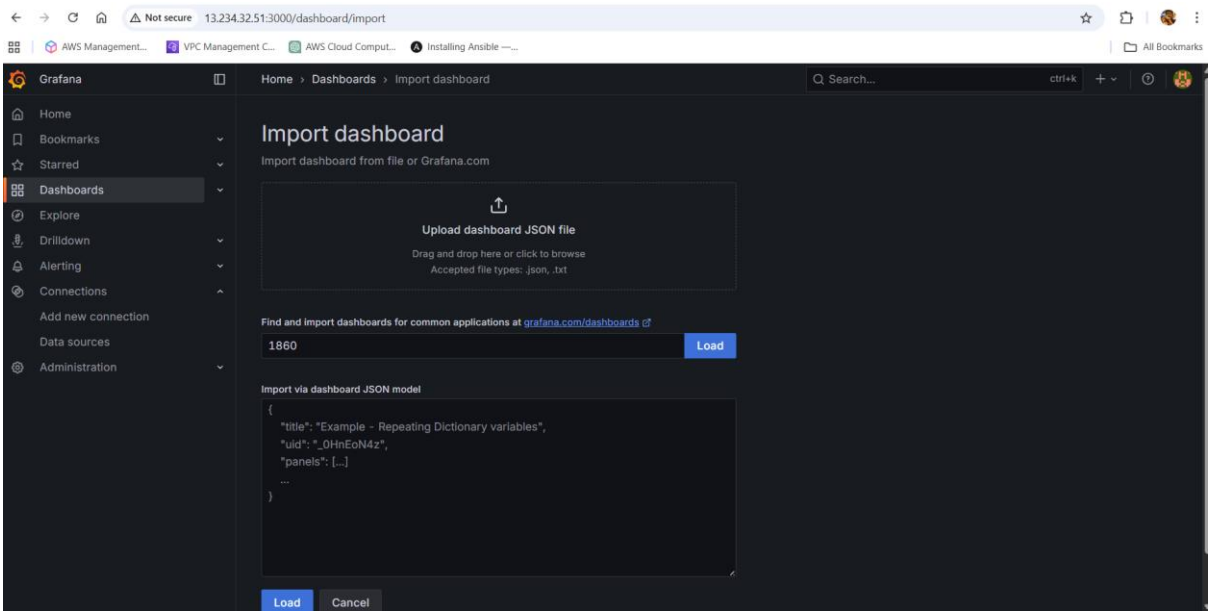
- Click on the “+” (plus) icon in the left sidebar to open the “Create” menu.

- Select “Dashboard.”

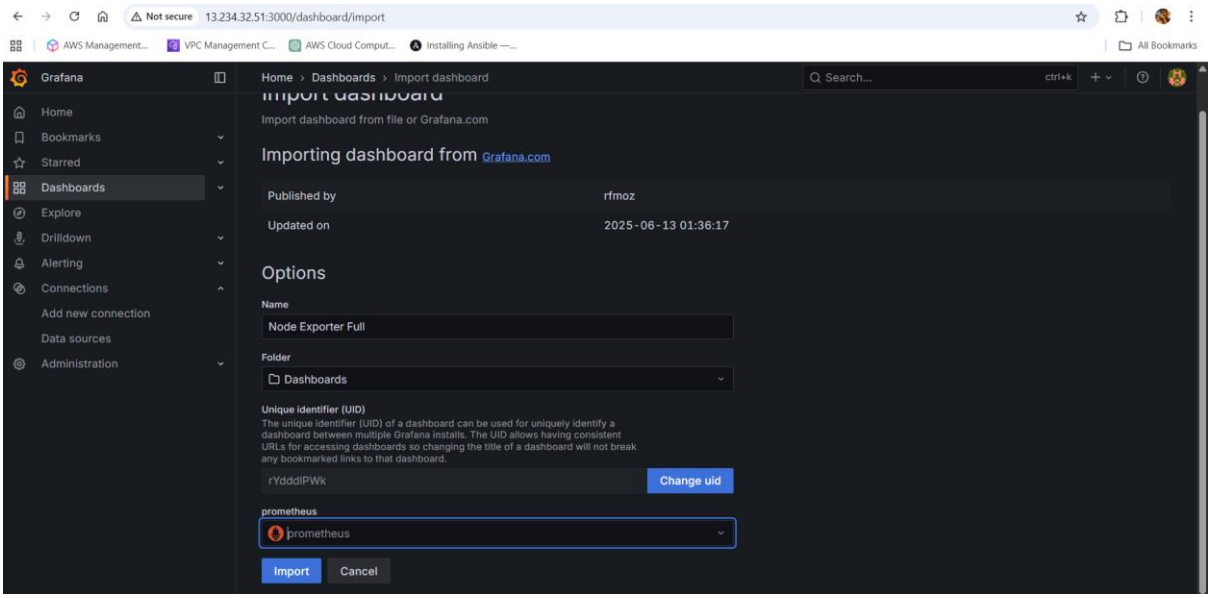
- Click on the “Import” dashboard option.

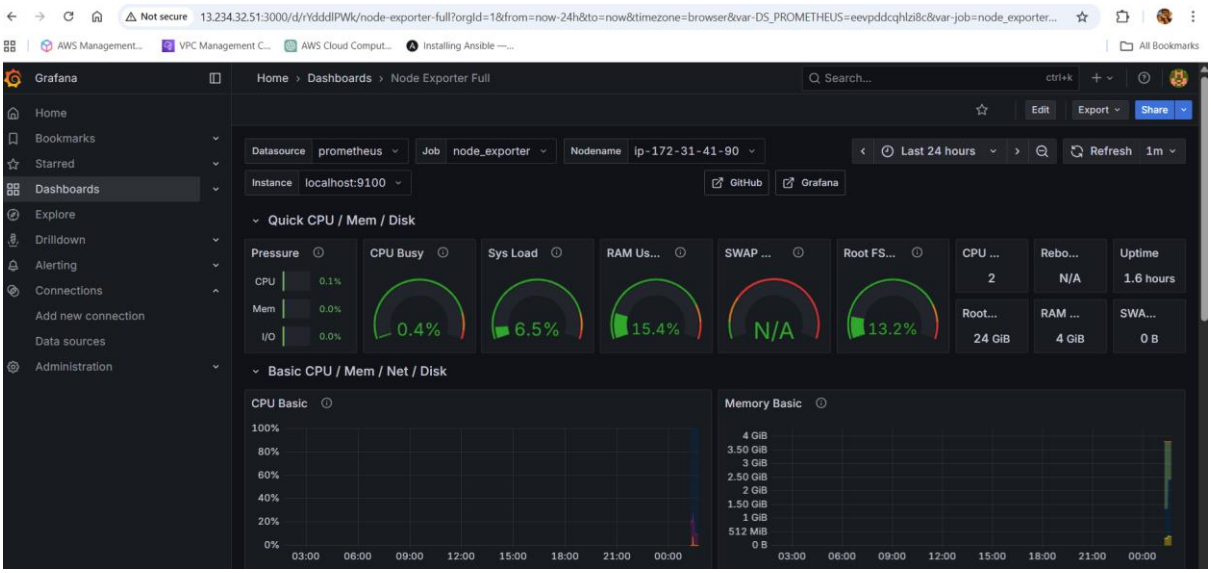
- Enter the dashboard code you want to import (e.g., code 1860).

- Click the “Load” button.



- Select the data source you added (Prometheus) from the dropdown.
- Click on the “Import” button.



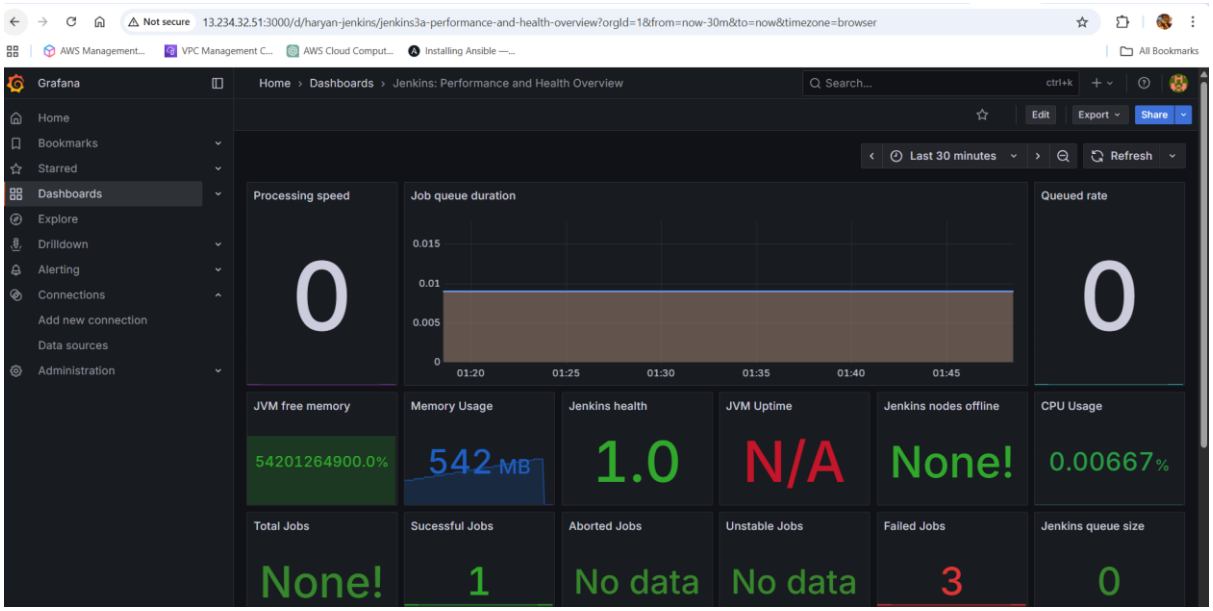


Step 7 →Configure global setting for Prometheus

go to manage Jenkins →system →search for Prometheus — apply →save

Step 8→ import a dashboard for Jenkins

- Click on the “+” (plus) icon in the left sidebar to open the “Create” menu.
- Select “Dashboard.”
- Click on the “Import” dashboard option.
- Enter the dashboard code you want to import (e.g., code 9964).
- Click the “Load” button.
- Select the data source you added (Prometheus) from the dropdown.

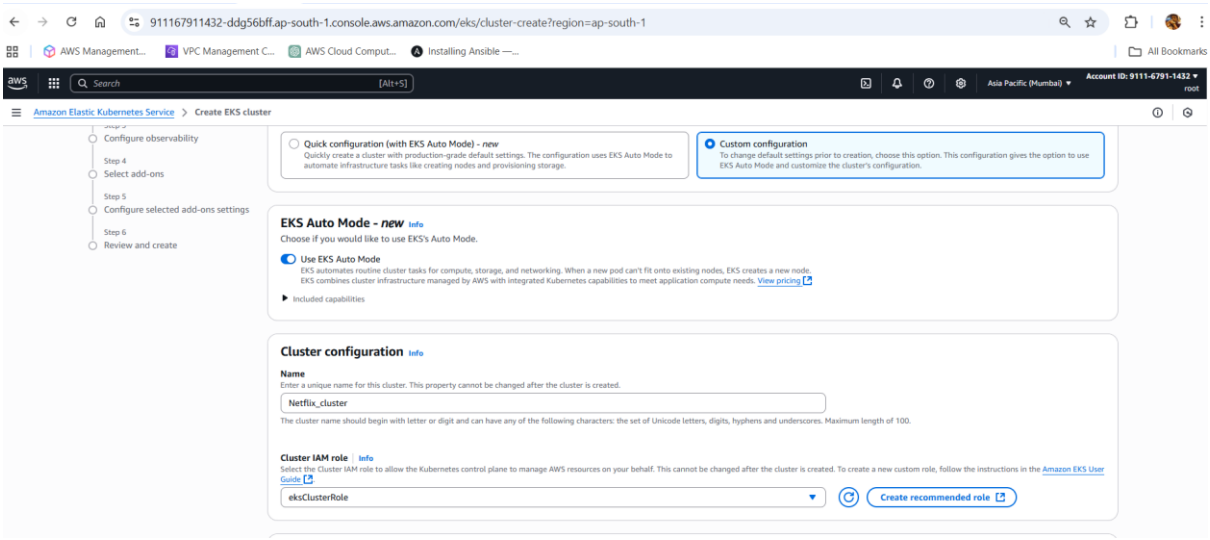


Phase 5: Kubernetes →

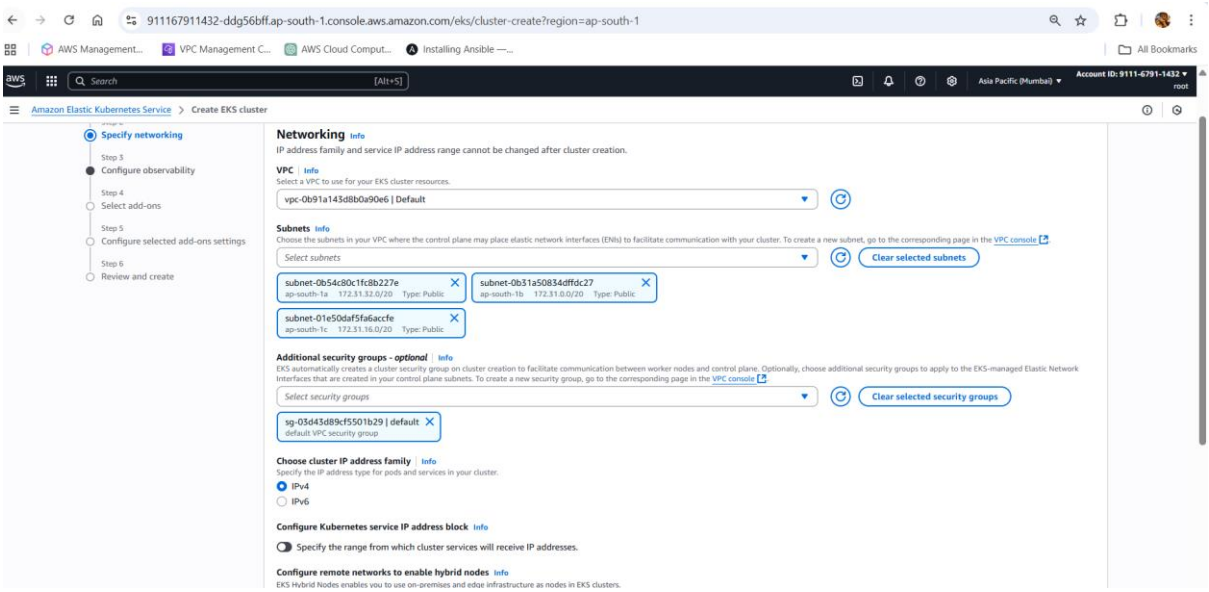
Step 1 →Setup EKS(Elastic Kubernetes Service) on aws

1. go to console and search for EKS
2. click on add a cluster
3. add a name to your cluster and choose a service role if you don't have then follow below documentation to create it

https://docs.aws.amazon.com/eks/latest/userguide/cluster-iam-role.html?source=post_page-----9ae6091b88bd

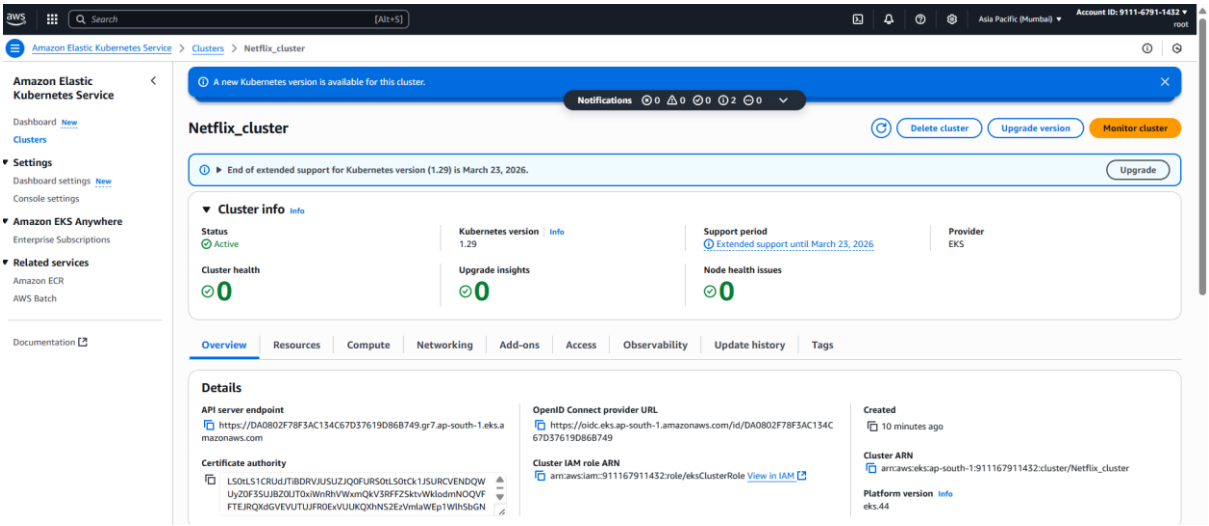


4. click on next and choose the default vpc and in subnet option make sure to remove all the private subnet and remain everything as it is

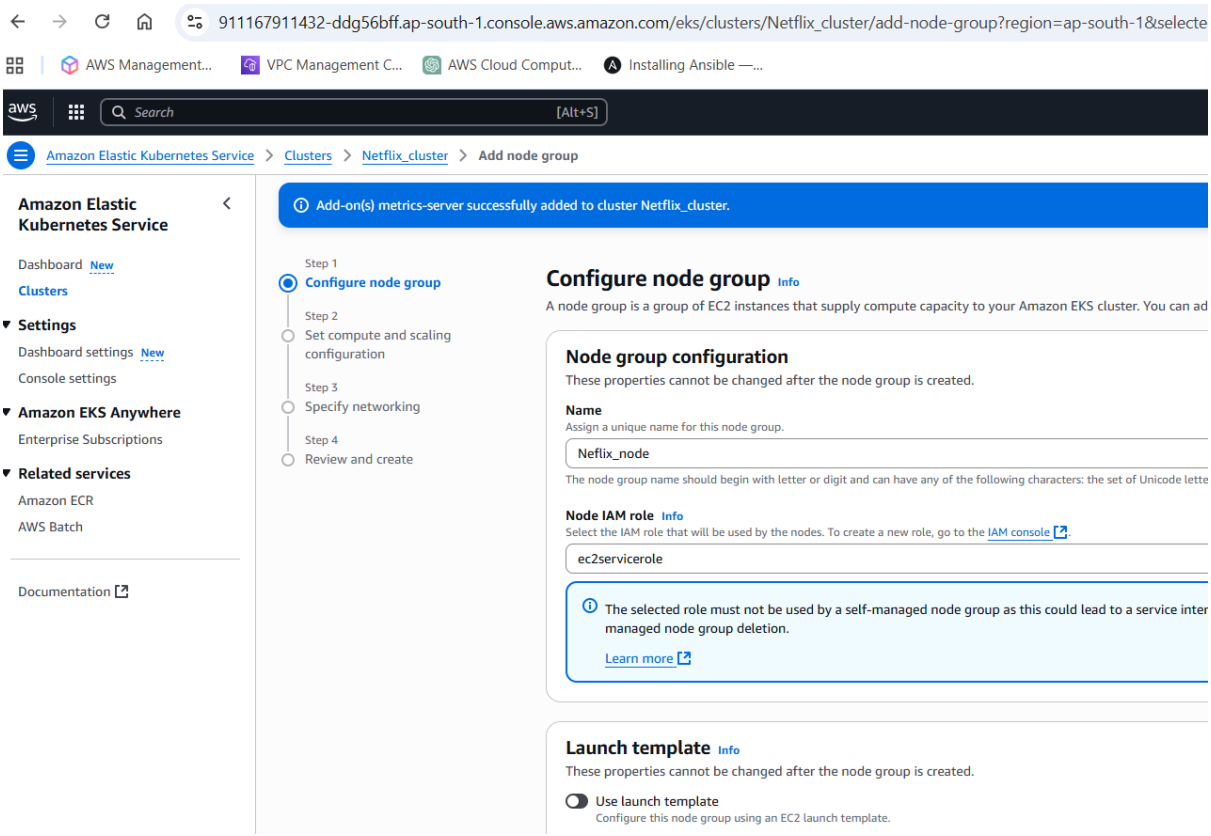


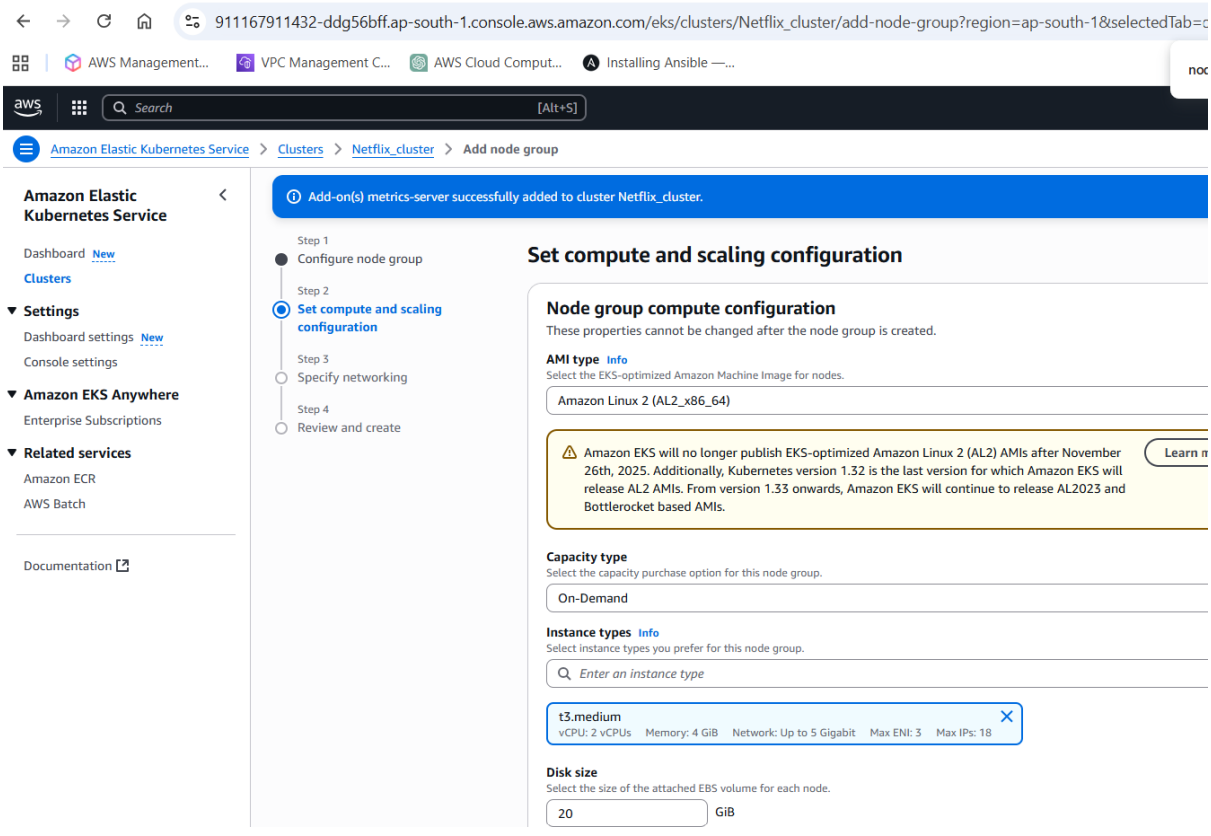
5. click on next →next →next→create

6. when you cluster is ready then **go to compute option and add a node group**

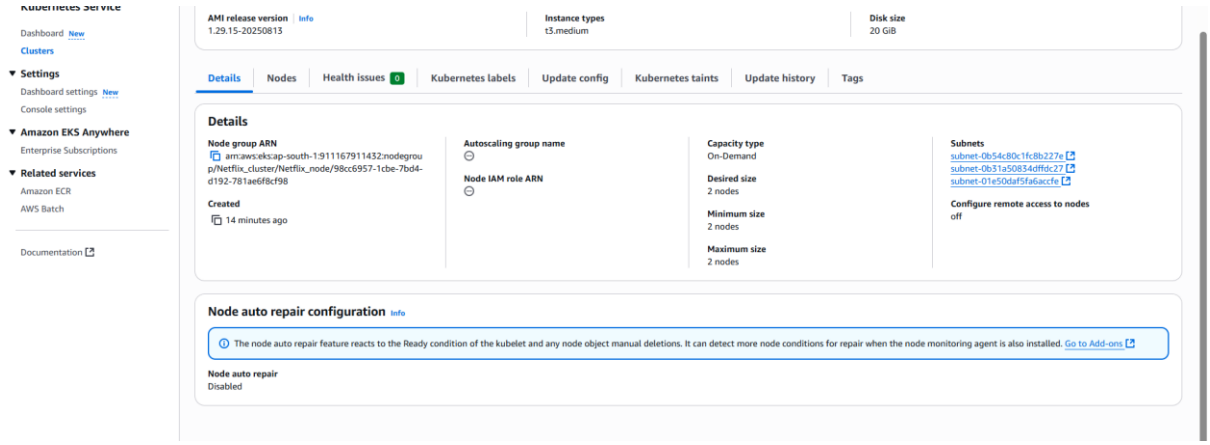


7. choose the below options


911167911432-ddg56bff.ap-south-1.console.aws.amazon.com/eks/clusters/Netflix_cluster/add-node-group?region=ap-south-1&selecte
AWS Management...
VPC Management C...
AWS Cloud Comput...
Installing Ansible —...
Search
[Alt+S]
Amazon Elastic Kubernetes Service > Clusters > Netflix_cluster > Add node group
Amazon Elastic Kubernetes Service
Dashboard New
Clusters
Settings
Dashboard settings New
Console settings
Amazon EKS Anywhere
Enterprise Subscriptions
Related services
Amazon ECR
AWS Batch
Documentation
Add-on(s) metrics-server successfully added to cluster Netflix_cluster.
Step 1
Configure node group
Step 2
Set compute and scaling configuration
Step 3
Specify networking
Step 4
Review and create
Configure node group Info
A node group is a group of EC2 instances that supply compute capacity to your Amazon EKS cluster. You can ad
Node group configuration
These properties cannot be changed after the node group is created.
Name
Assign a unique name for this node group.
Neflix_node
The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode lette
Node IAM role Info
Select the IAM role that will be used by the nodes. To create a new role, go to the IAM console.
ec2servicerole
The selected role must not be used by a self-managed node group as this could lead to a service inter
managed node group deletion.
Learn more
Launch template Info
These properties cannot be changed after the node group is created.
Use launch template
Configure this node group using an EC2 launch template.


911167911432-ddg56bff.ap-south-1.console.aws.amazon.com/eks/clusters/Netflix_cluster/add-node-group?region=ap-south-1&selectedTab=c
AWS Management...
VPC Management C...
AWS Cloud Comput...
Installing Ansible —...
Search
[Alt+S]
Amazon Elastic Kubernetes Service > Clusters > Netflix_cluster > Add node group
Amazon Elastic Kubernetes Service
Dashboard New
Clusters
Settings
Dashboard settings New
Console settings
Amazon EKS Anywhere
Enterprise Subscriptions
Related services
Amazon ECR
AWS Batch
Documentation
Add-on(s) metrics-server successfully added to cluster Netflix_cluster.
Step 1
Configure node group
Step 2
Set compute and scaling configuration
Step 3
Specify networking
Step 4
Review and create
Set compute and scaling configuration
Node group compute configuration
These properties cannot be changed after the node group is created.
AMI type Info
Select the EKS-optimized Amazon Machine Image for nodes.
Amazon Linux 2 (AL2_x86_64)
Amazon EKS will no longer publish EKS-optimized Amazon Linux 2 (AL2) AMIs after November 26th, 2025. Additionally, Kubernetes version 1.32 is the last version for which Amazon EKS will release AL2 AMIs. From version 1.33 onwards, Amazon EKS will continue to release AL2023 and Bottlerocket based AMIs.
Capacity type
Select the capacity purchase option for this node group.
On-Demand
Instance types Info
Select instance types you prefer for this node group.
Enter an instance type
t3.medium
vCPU: 2 vCPUs Memory: 4 GiB Network: Up to 5 Gigabit Max ENI: 3 Max IPs: 18
Disk size
Select the size of the attached EBS volume for each node.
20
GiB

Remain everything as it is →click on next-next and click on create



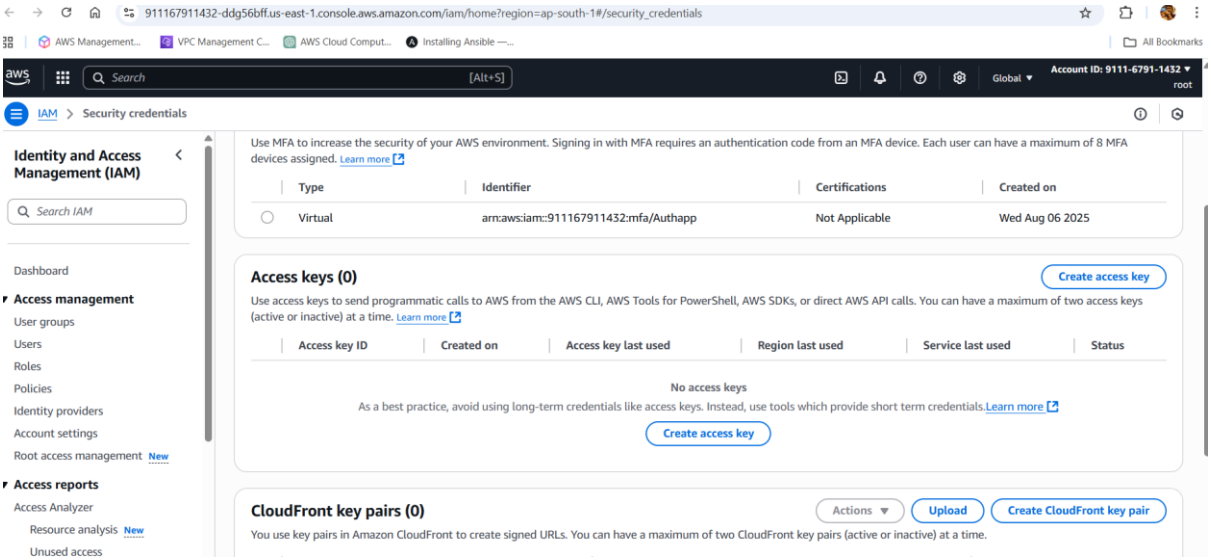
Step 2 →Installing aws -cli to control the cluster from our terminal

Now go to your terminal and run

1. **pip install awscli**
2. **aws configure**
3. **enter a access key and then secret access key**

How to create access key

1. go to your console click on your username and then select Security Credentials option
2. scroll down to access keys and click on create



Copy and paste access and secret key to terminal.

```
root@ip-172-31-41-90:~# aws configure
AWS Access Key ID [****************6EFG]: AKIA5IJOW6YEP7JR6EFG
AWS Secret Access Key [****************e/XR]: UY6Q3nyYDOMS5bbK+H6VPinuc4n3ZMFuM2
8ge/XR
Default region name [ap-south-1]: ap-south-1
Default output format [json]: json
root@ip-172-31-41-90:~#
```

Run the following command

1. **aws eks update-kubeconfig --name YourClusterName --region ap-south-1**

   This command is used to update the Kubernetes configuration (kubeconfig) file for an Amazon Elastic Kubernetes Service (EKS) cluster named "Netflix_cluster" in the "ap-south" region, allowing you to interact with the cluster using kubectl

```
root@ip-172-31-41-90:~# aws eks update-kubeconfig --name Netflix_cluster --region ap-south-1
Added new context arn:aws:eks:ap-south-1:911167911432:cluster/Netflix_cluster to /root/.kube/config
root@ip-172-31-41-90:~#
```

Step 3→Install helm →

 # Add Helm GPG key

curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null

 # Install transport HTTPS

sudo apt-get install apt-transport-https --yes

 # Add Helm repository

echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/helm.gpg] https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list

 # Update package list and install Helm

sudo apt-get update

sudo apt-get install helm -y

Install Kubectl

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

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin/

kubectl version --client

Step →4 Install Node Exporter using Helm

 # 1. Add the Prometheus Community Helm repo

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

 # 2. Update repo

helm repo update

 # 3. Create a namespace for node-exporter

kubectl create namespace prometheus-node-exporter

 # 4. Install Node Exporter

helm install prometheus-node-exporter prometheus-community/prometheus-node-exporter --namespace prometheus-node-exporter

1. **kubectl get pods -n prometheus-node-exporter → run this to verify**

```
root@ip-172-31-41-90:~# kubectl get pods -n prometheus-node-exporter
NAME                                 READY   STATUS    RESTARTS   AGE
prometheus-node-exporter-s2x9z   1/1     Running   0          76s
prometheus-node-exporter-vwznb   1/1     Running   0          76s
root@ip-172-31-41-90:~#
```

Add a Job to Scrape Metrics on nodeip:9100/metrics in prometheus.yml:

```
- job_name: 'Netflix'
  metrics_path: '/metrics'
  static_configs:
    - targets: ['65.0.170.249:9100']
```

Don’t forget to reload or restart Prometheus to apply these changes to your configuration Deploy Application with ArgoCD

Step 5→Install Argo cd

1. Add the Argo CD Helm repository:

- helm repo add argo-cd https://argoproj.github.io/argo-helm

2. Update your Helm repositories:

- helm repo update

3. Create a namespace for Argo CD (optional but recommended):

- kubectl create namespace argocd

4. Install Argo CD using Helm:

- helm install argocd argo-cd/argo-cd -n argocd

5. kubectl get all -n argocd

```
root@ip-172-31-41-90:/# kubectl get all -n argocd
NAME                                                READY   STATUS      RESTARTS   AGE
pod/argocd-application-controller-0                 1/1     Running     0          29s
pod/argocd-applicationset-controller-59f7fc666-2wdzm   1/1     Running     0          29s
pod/argocd-dex-server-8b99cf66f-nbcrw               1/1     Running     0          29s
pod/argocd-notifications-controller-b7bf48757-bjt8t   1/1     Running     0          29s
pod/argocd-redis-84897889d6-2zmfg                   1/1     Running     0          29s
pod/argocd-redis-secret-init-6hztz                  0/1     Completed   0          45s
pod/argocd-repo-server-77654bf79c-rjwtb             1/1     Running     0          29s
pod/argocd-server-66bc5bd49b-mjpf4                  1/1     Running     0          29s

NAME                                       TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)             AGE
service/argocd-applicationset-controller   ClusterIP   100.70.147.163   <none>        7000/TCP            29s
service/argocd-dex-server                  ClusterIP   100.68.65.30     <none>        5556/TCP,5557/TCP   29s
service/argocd-redis                       ClusterIP   100.68.14.153    <none>        6379/TCP            29s
service/argocd-repo-server                 ClusterIP   100.68.14.26     <none>        8081/TCP            29s
service/argocd-server                      ClusterIP   100.70.240.192   <none>        80/TCP,443/TCP      29s

NAME                                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/argocd-applicationset-controller   1/1     1            1           29s
deployment.apps/argocd-dex-server                  1/1     1            1           29s
deployment.apps/argocd-notifications-controller    1/1     1            1           29s
deployment.apps/argocd-redis                       1/1     1            1           29s
deployment.apps/argocd-repo-server                 1/1     1            1           29s
deployment.apps/argocd-server                      1/1     1            1           29s

NAME                                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/argocd-applicationset-controller-59f7fc666    1         1         1       29s
replicaset.apps/argocd-dex-server-8b99cf66f                   1         1         1       29s
replicaset.apps/argocd-notifications-controller-b7bf48757     1         1         1       29s
replicaset.apps/argocd-redis-84897889d6                       1         1         1       29s
replicaset.apps/argocd-repo-server-77654bf79c                 1         1         1       29s
replicaset.apps/argocd-server-66bc5bd49b                      1         1         1       29s

NAME                                             READY   AGE
statefulset.apps/argocd-application-controller   1/1     29s

NAME                              STATUS     COMPLETIONS   DURATION   AGE
job.batch/argocd-redis-secret-init   Complete   1/1           15s        45s
root@ip-172-31-41-90:/#
```

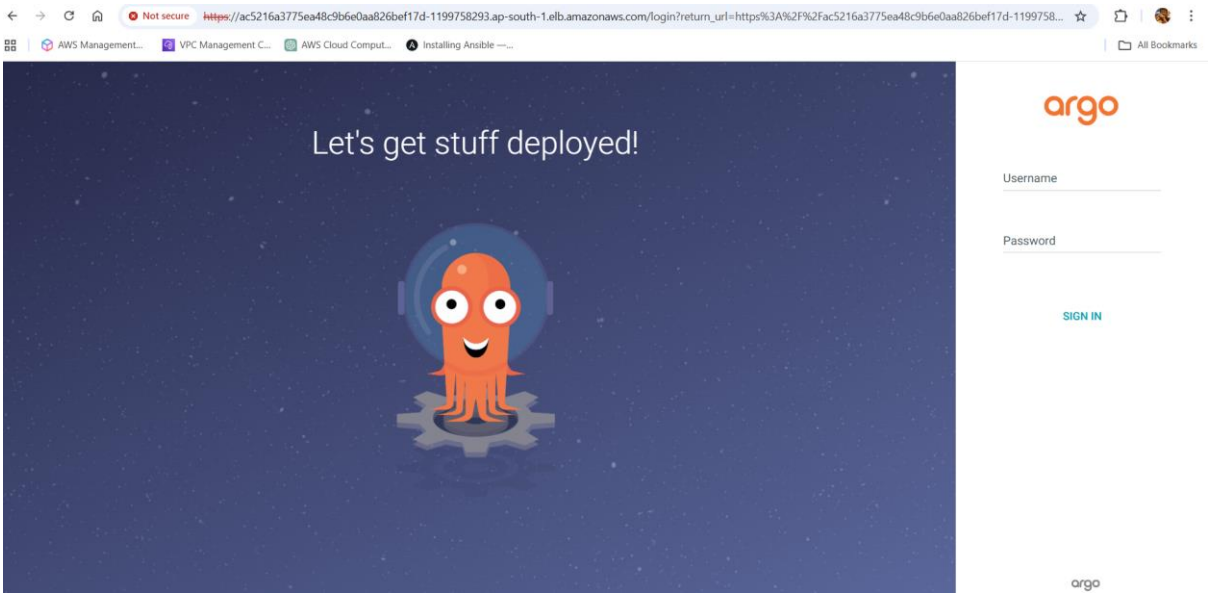
Expose argocd-server

By default, argocd-server is not publicly exposed. For the purpose of this workshop, we will use a Load Balancer to make it usable:

1. kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
2. export ARGOCD_SERVER=$(kubectl get svc argocd-server -n argocd -o json | jq -r '.status.loadBalancer.ingress[0].hostname')
3. echo $ARGOCD_SERVER

```
root@ip-172-31-41-90:/# export ARGOCD_SERVER=$(kubectl get svc argocd-server -n argocd -o json | jq -r '.status.loadBalancer.ingress[0].hostname')
echo $ARGOCD_SERVER
ac5216a3775ea48c9b6e0aa826bef17d-1199758293.ap-south-1.elb.amazonaws.com
root@ip-172-31-41-90:/#
```

you have given a url copy and paste it on your chrome browser

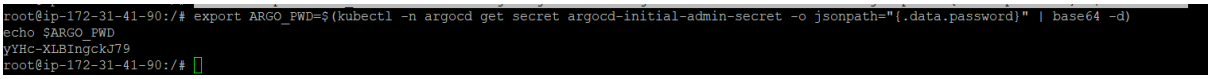


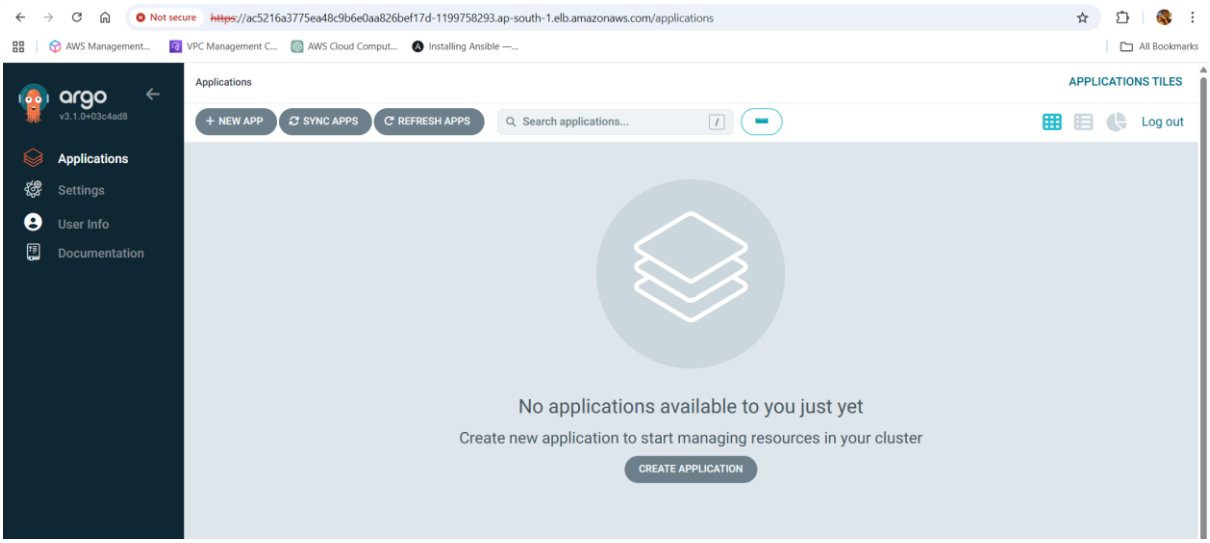
For Password →

1. export ARGO_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d`
2. echo $ARGO_PWD
3. output == password for argocd

After running this, you can log in to Argo CD UI using:

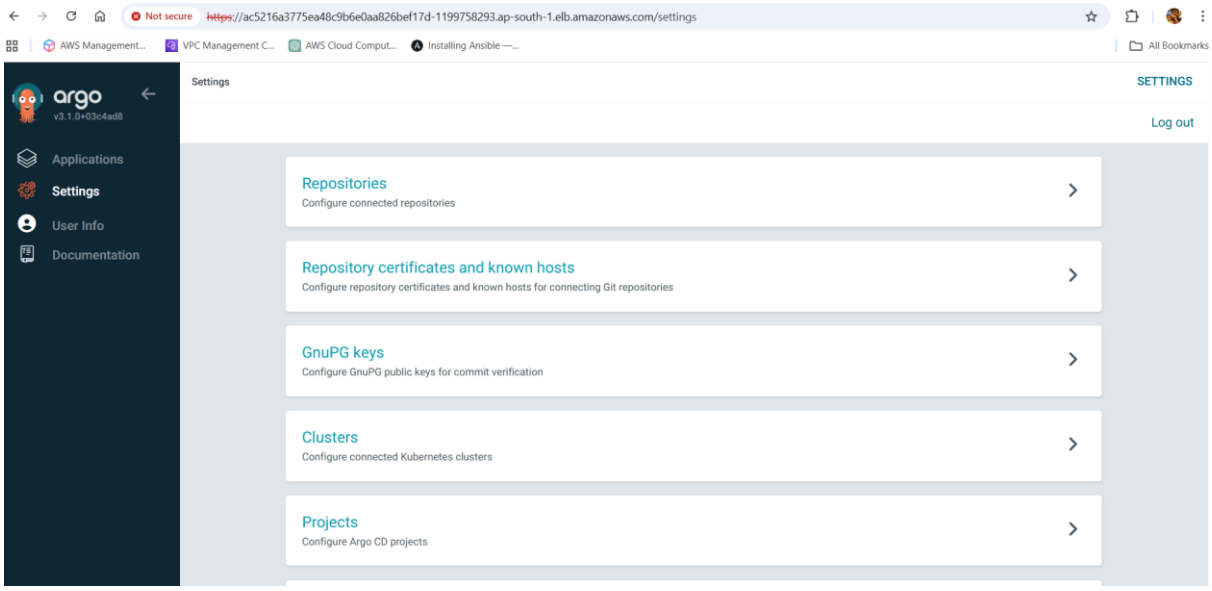
- Username: admin
- Password: the value of $ARGO_PWD



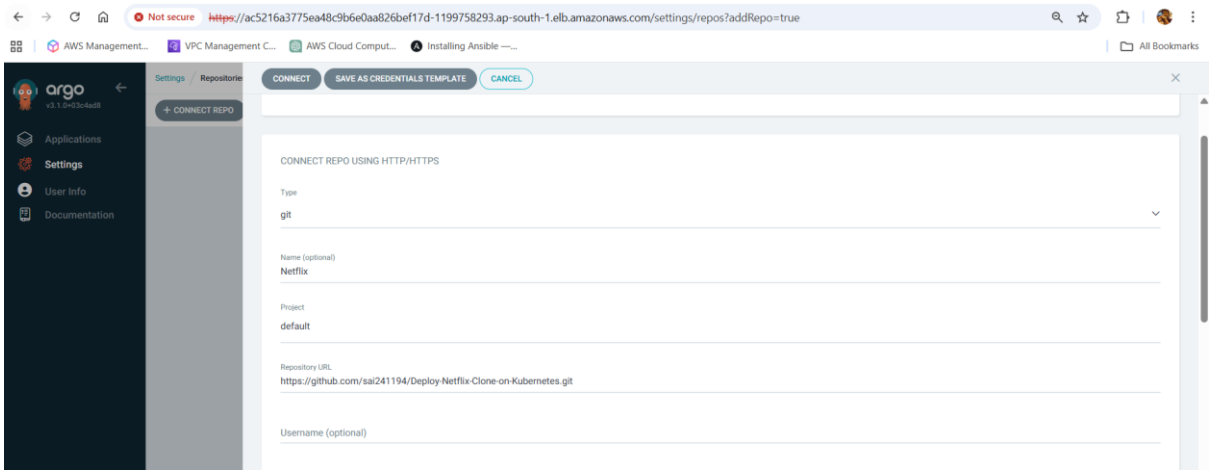


Step 5 →Deploy Application with ArgoCD

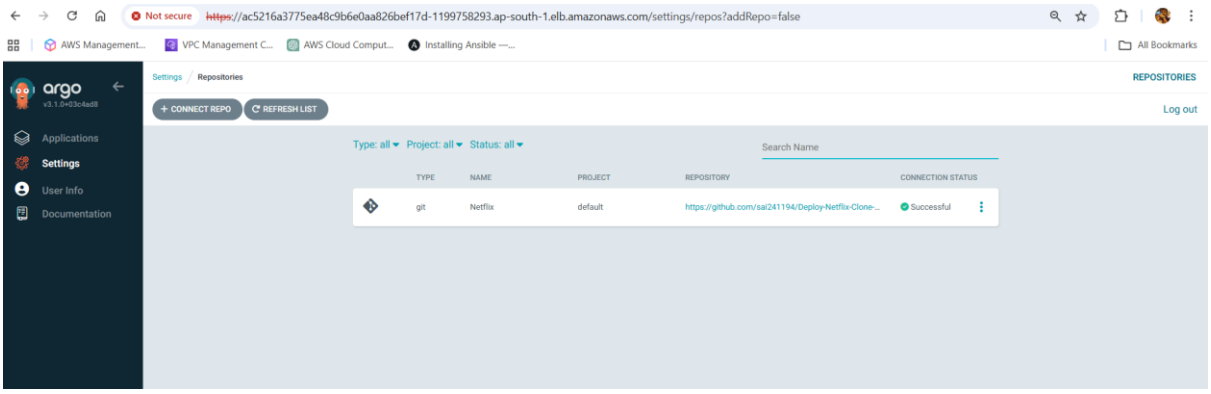
Sign into argo cd using above steps →



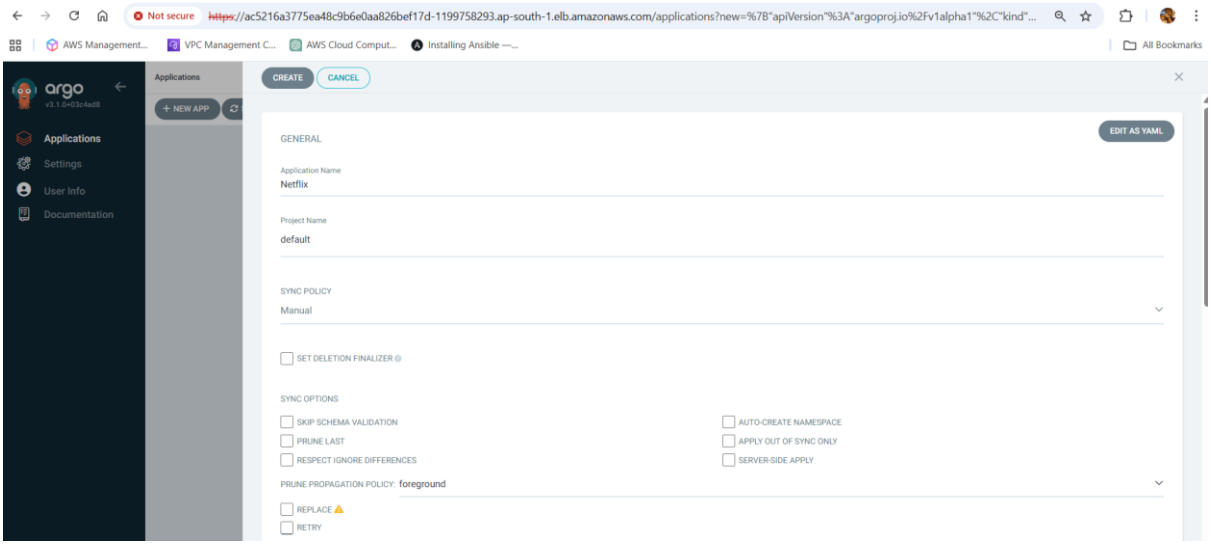
1. **go to setting →repositories →connect repo and choose HTTP/HTTPS method and give below details**

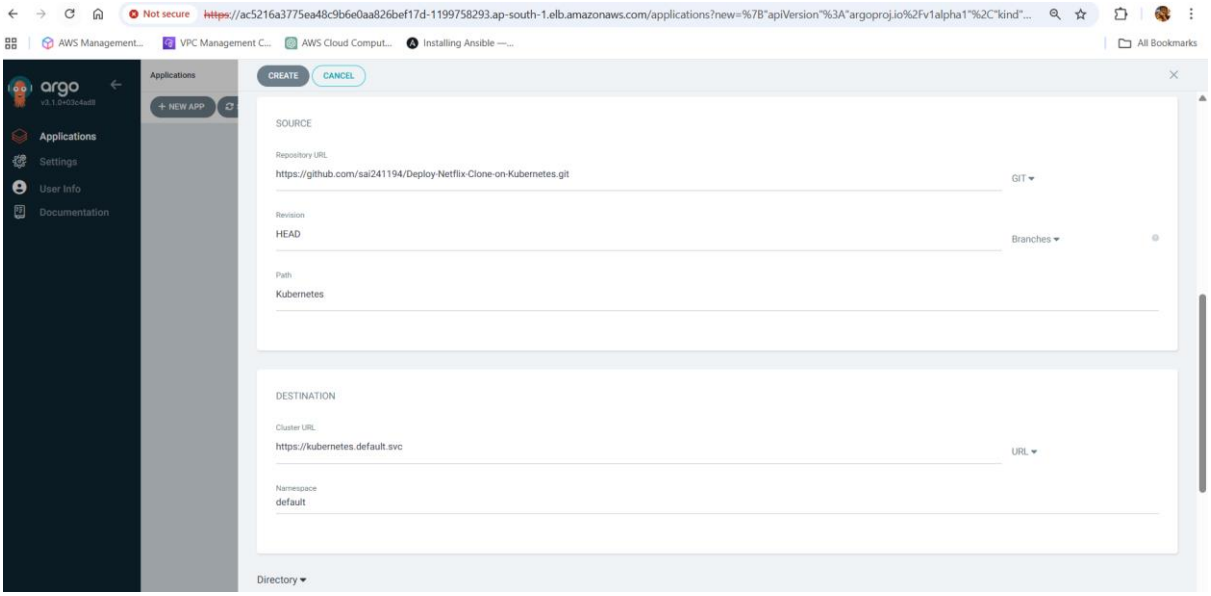


2. **click on connect output →successful**



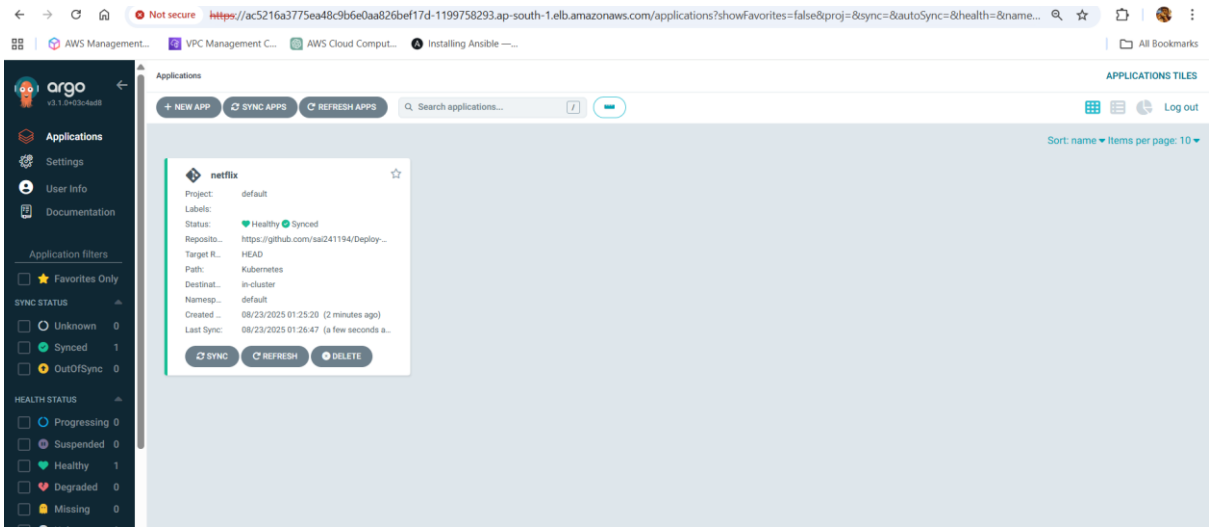
3. Now go to application →newapp



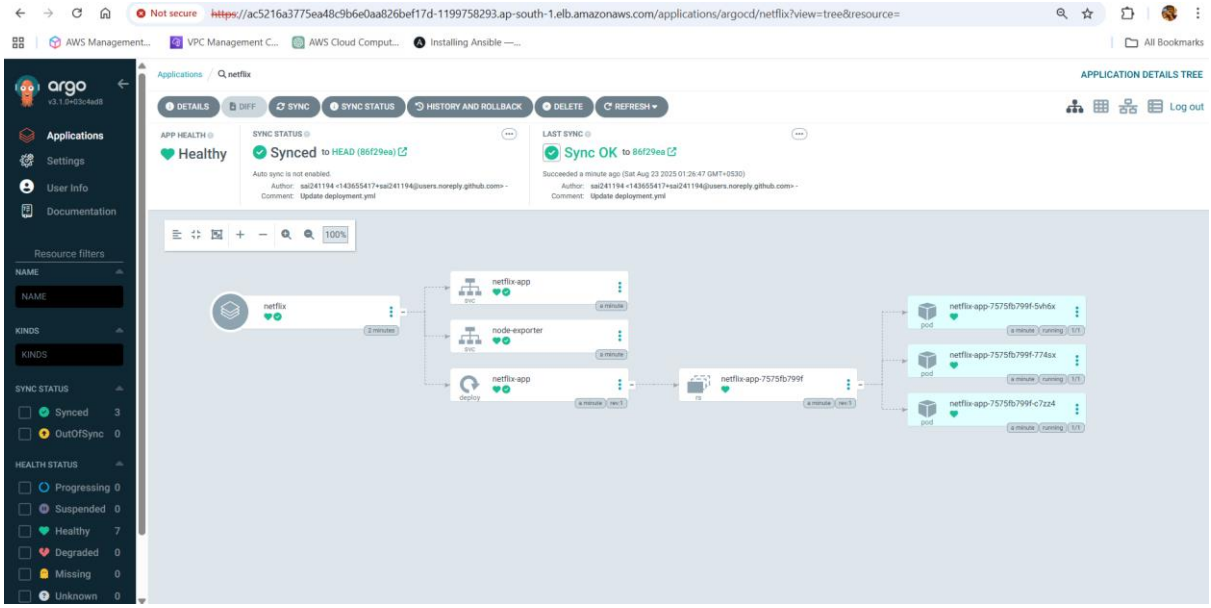


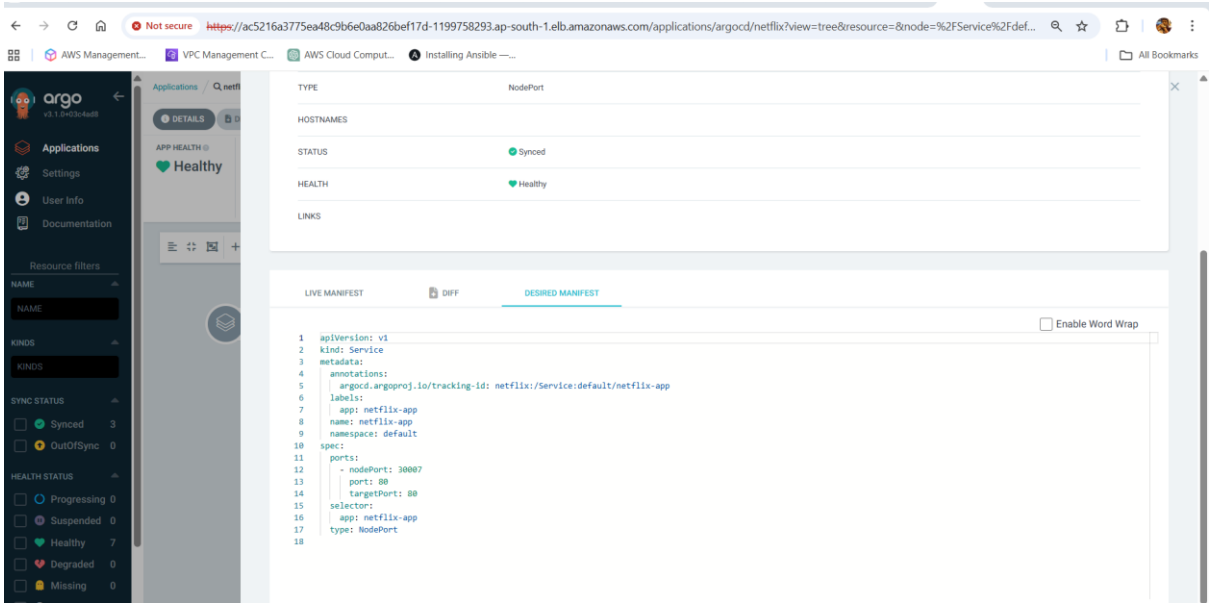
4. select the above options and click on create

5. Now click on sync →force →sync →ok



click on netflix app





6.go to your cluster and select the node created by node group and expose port 30007

7.now browse the application by Publicip:30007