PROJECT-6

DEPLOYMENT OF NETFLIX APPLICATION ON KUBERNETES (EKS)

-ANIL GOLLENA

Introduction

This document outlines the phased deployment of the Netflix application on Kubernetes using Amazon Elastic Kubernetes Service (EKS). The project follows a structured approach to ensure scalability, security, automation, and monitoring. Below is the detailed introduction for each phase of the deployment.

Phase 1: Deploy Netflix Application Locally on EC2 (t2.large)

- The first phase focuses on setting up the Netflix application in a controlled environment using AWS EC2 instances.
- **Objective**: Launch the Netflix application on an EC2 instance (t2.large) to validate its functionality in a local deployment environment.
- **Steps Involved**:
  - Provision an EC2 instance with sufficient resources (CPU, memory) to run the application.
  - Install necessary dependencies and runtime environments.
  - Deploy the application and verify its functionality locally.

Phase 2: Implementation of Security with SonarQube and Trivy

- Security is a critical aspect of this deployment.
- **Objective**: Ensure the code and deployment artifacts meet security standards.
- **Steps Involved**:
  - **SonarQube**: Analyze the application's source code for vulnerabilities and adherence to coding standards.
  - **Trivy**: Scan container images for known vulnerabilities before they are deployed.
  - Address identified issues to strengthen the overall security posture.

Phase 3: Automating the Deployment Using Jenkins Pipeline

- This phase introduces automation to improve deployment efficiency.
- **Objective**: Automate the build, test, and deployment processes of the Netflix application using Jenkins CI/CD pipelines.
- **Steps Involved**:
    - Set up Jenkins with a declarative pipeline.
    - Integrate SonarQube and Trivy scans into the pipeline for continuous security checks.
    - Automate deployment steps to ensure repeatability and minimize manual intervention.

Phase 4: Monitoring via Prometheus and Grafana

- Monitoring ensures the application's reliability and performance.
- **Objective**: Implement a robust monitoring solution using Prometheus and Grafana.
- **Steps Involved**:
    - Deploy Prometheus to collect metrics from the application and infrastructure.
    - Configure Grafana dashboards to visualize critical metrics and enable real-time monitoring.
    - Set up alerts for proactive issue detection and resolution.

Phase 5: Deploying on Kubernetes (EKS)

- The final phase involves deploying the application on a scalable Kubernetes cluster managed by EKS.
- **Objective**: Achieve high availability, scalability, and efficient resource management for the Netflix application.
- **Steps Involved**:
    - Create and configure an EKS cluster with worker nodes.
    - Deploy the Netflix application as Kubernetes deployments and services.
    - Use Kubernetes features like auto-scaling, load balancing, and rolling updates to enhance the deployment.
    - Integrate monitoring and alerting tools for end-to-end observability.

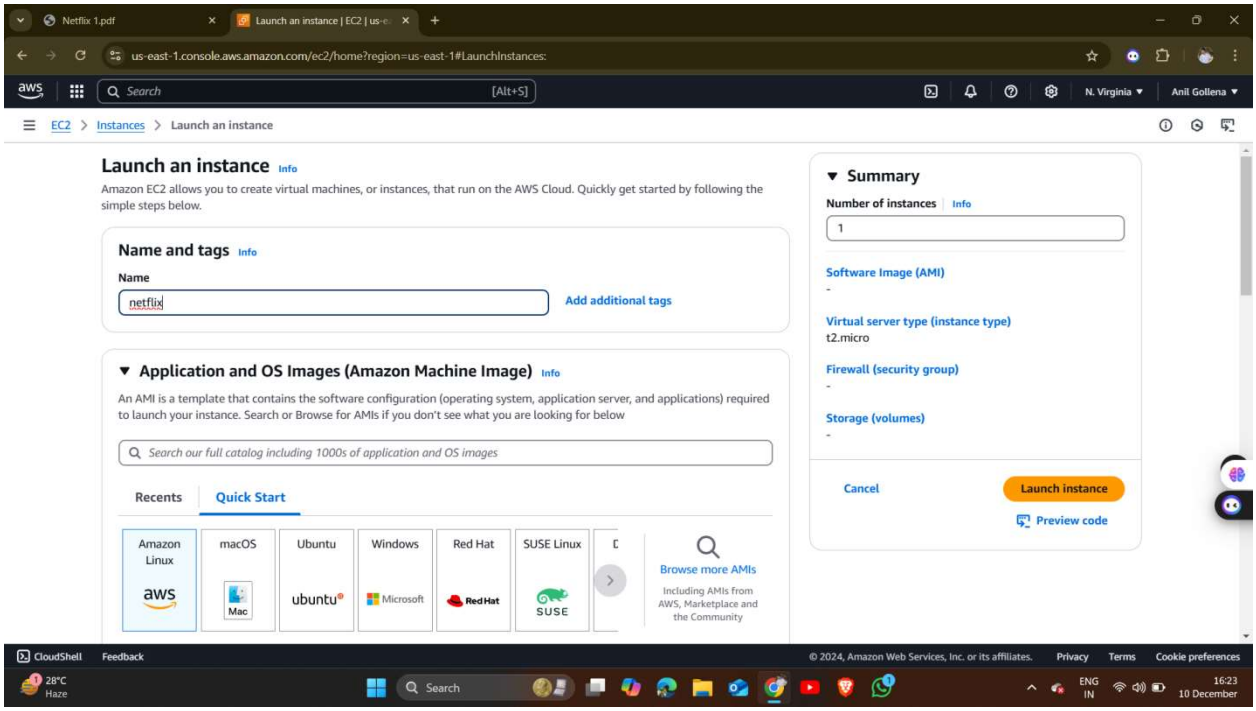
PHASE1: DEPLOYMENT OF NETFLIX APPLICATION ON EC2 (T2.LARGE).

Now go to aws management console and create a new instance with t2.large.

Choose ubuntu as operating system.

Increase the root volume 8GB→25GB.

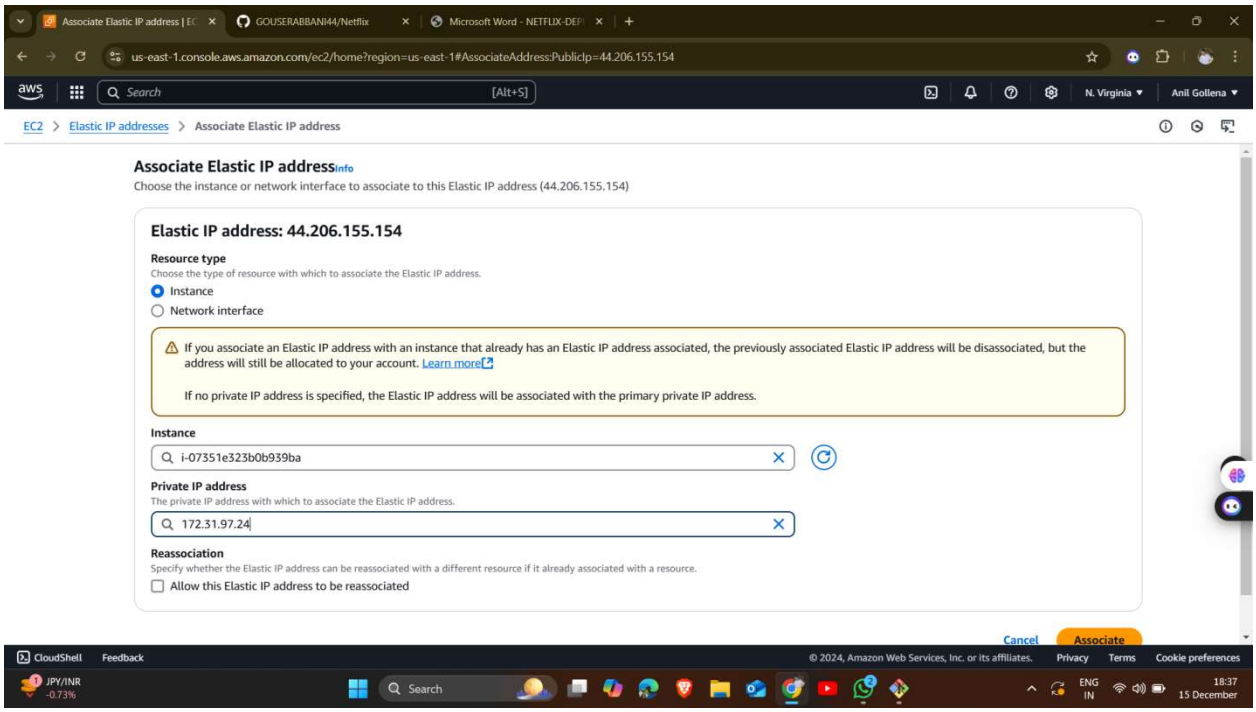
Launch the ec2 instance.

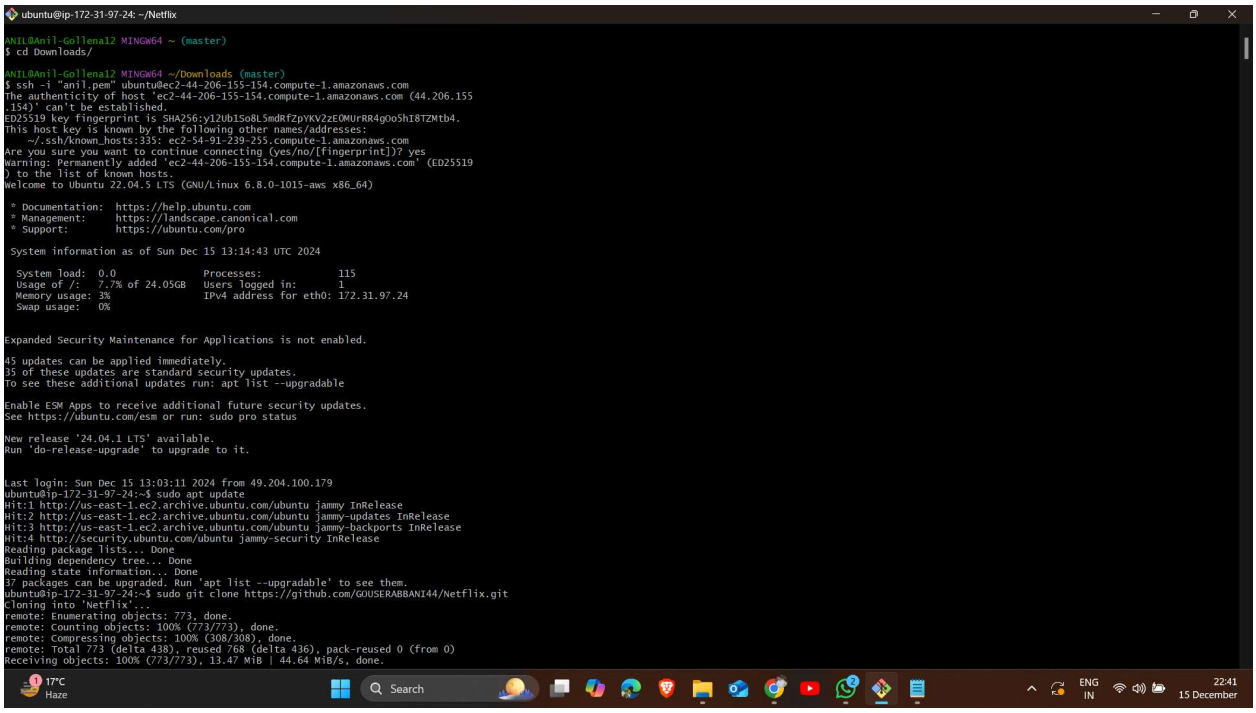


Now connect to the server

Install git and clone the repository

Now create elastic ip and attach it with instance.

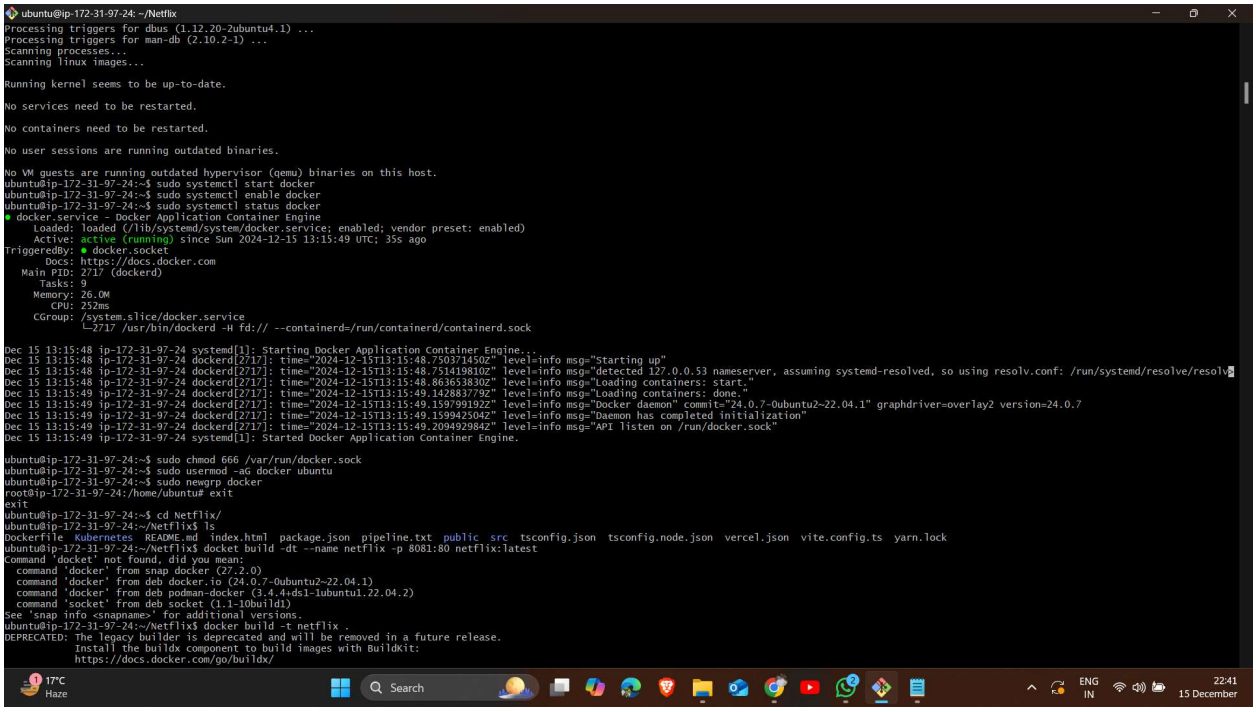

Associate Elastic IP address | EC
GOUSERABBANI44/Netflix
Microsoft Word - NETFLIX-DEP
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#AssociateAddress:PublicIp=44.206.155.154
Search
[Alt+S]
N. Virginia
Anil Gollena
EC2 > Elastic IP addresses > Associate Elastic IP address
Associate Elastic IP address Info
Choose the instance or network interface to associate to this Elastic IP address (44.206.155.154)
Elastic IP address: 44.206.155.154
Resource type
Choose the type of resource with which to associate the Elastic IP address.
Instance
Network interface
If you associate an Elastic IP address with an instance that already has an Elastic IP address associated, the previously associated Elastic IP address will be disassociated, but the address will still be allocated to your account. Learn more
If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address.
Instance
i-07351e323b0b939ba
Private IP address
The private IP address with which to associate the Elastic IP address.
172.31.97.24
Reassociation
Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.
Allow this Elastic IP address to be reassociated
Cancel
Associate
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
JPY/INR
-0.73%
Search
ENG
IN
18:37
15 December

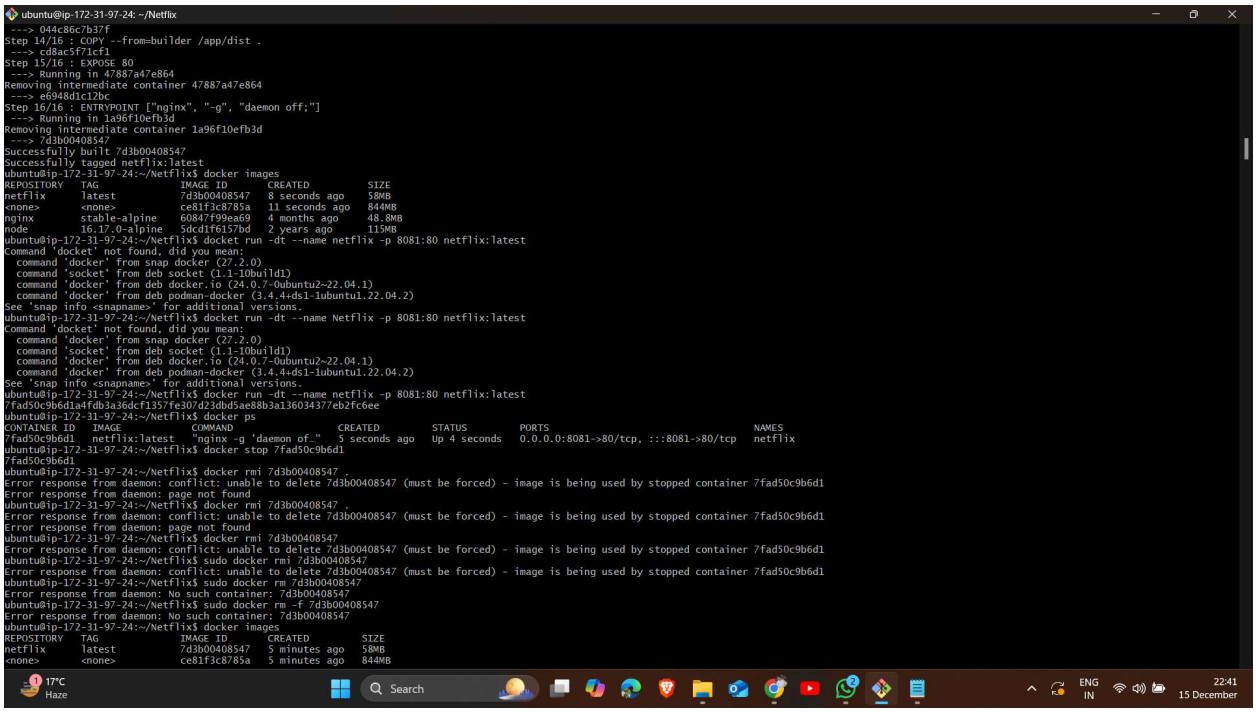

ubuntu@ip-172-31-97-24: ~/Netflix
ANIL@Anil-Gollena12 MINGW64 ~ (master)
$ cd Downloads/
ANIL@Anil-Gollena12 MINGW64 ~/Downloads (master)
$ ssh -i "anil.pem" ubuntu@ec2-44-206-155-154.compute-1.amazonaws.com
The authenticity of host 'ec2-44-206-155-154.compute-1.amazonaws.com (44.206.155
.154)' can't be established.
ED25519 key fingerprint is SHA256:y12Ub1So8L5mdRfZpYKV2zE0MUrRR4gOo5hI8TZMtb4.
This host key is known by the following other names/addresses:
~/.ssh/known_hosts:335: ec2-54-91-239-255.compute-1.amazonaws.com
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-206-155-154.compute-1.amazonaws.com' (ED25519
) to the list of known hosts.
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1015-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Sun Dec 15 13:14:43 UTC 2024
System load: 0.0 Processes: 115
Usage of /: 7.7% of 24.05GB Users logged in: 1
Memory usage: 3% IPv4 address for eth0: 172.31.97.24
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
45 updates can be applied immediately.
35 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
New release '24.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
Last login: Sun Dec 15 13:03:11 2024 from 49.204.100.179
ubuntu@ip-172-31-97-24:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
37 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-97-24:~$ sudo git clone https://github.com/GOUSERABBANI44/Netflix.git
Cloning into 'Netflix'...
remote: Enumerating objects: 773, done.
remote: Counting objects: 100% (773/773), done.
remote: Compressing objects: 100% (308/308), done.
remote: Total 773 (delta 438), reused 768 (delta 436), pack-reused 0 (from 0)
Receiving objects: 100% (773/773), 13.47 MiB | 44.64 MiB/s, done.
17°C
Haze
Search
ENG
IN
22:41
15 December

Now install docker and setup docker and images.

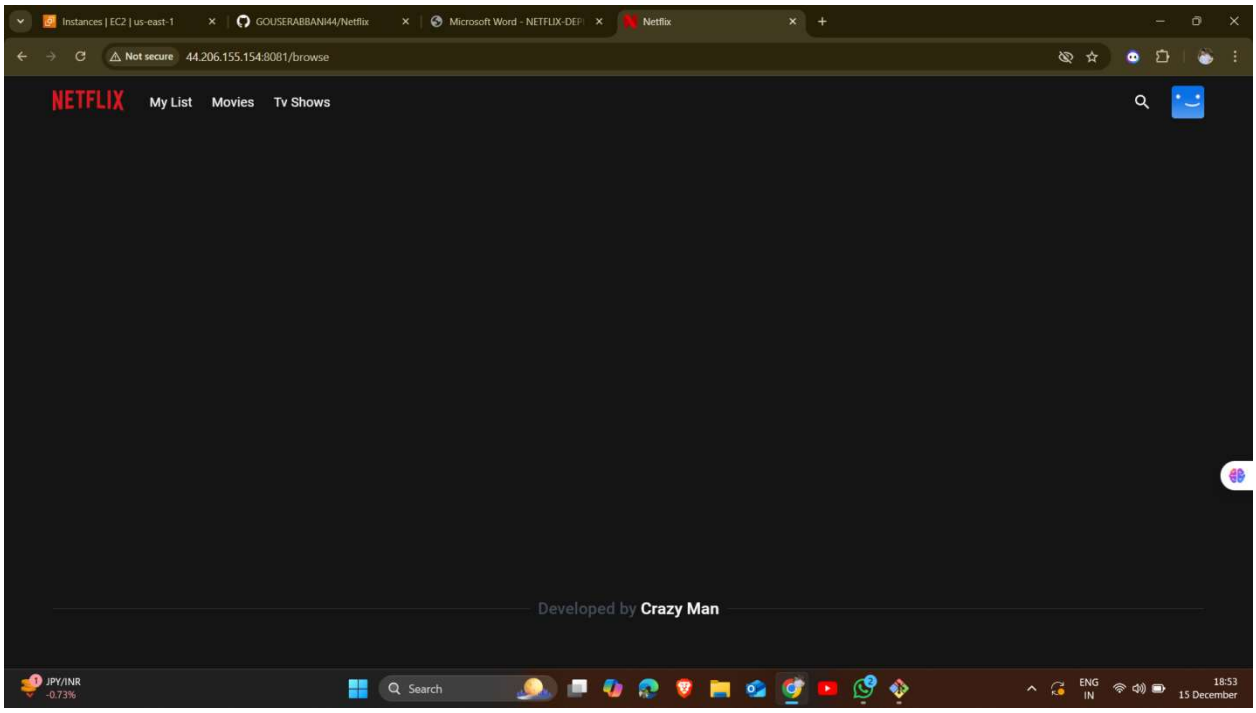
Now run the following command to set up docker.

- sudo apt-get install docker.io
- sudo chmod 666 /var/run/docker.sock
- sudo usermod -aG docker ubuntu
- sudo newgrp docker
- exit
- newgrp docker
- sudo systemctl start docker sudo
- systemctl enable docker
- sudo systemctl status docker

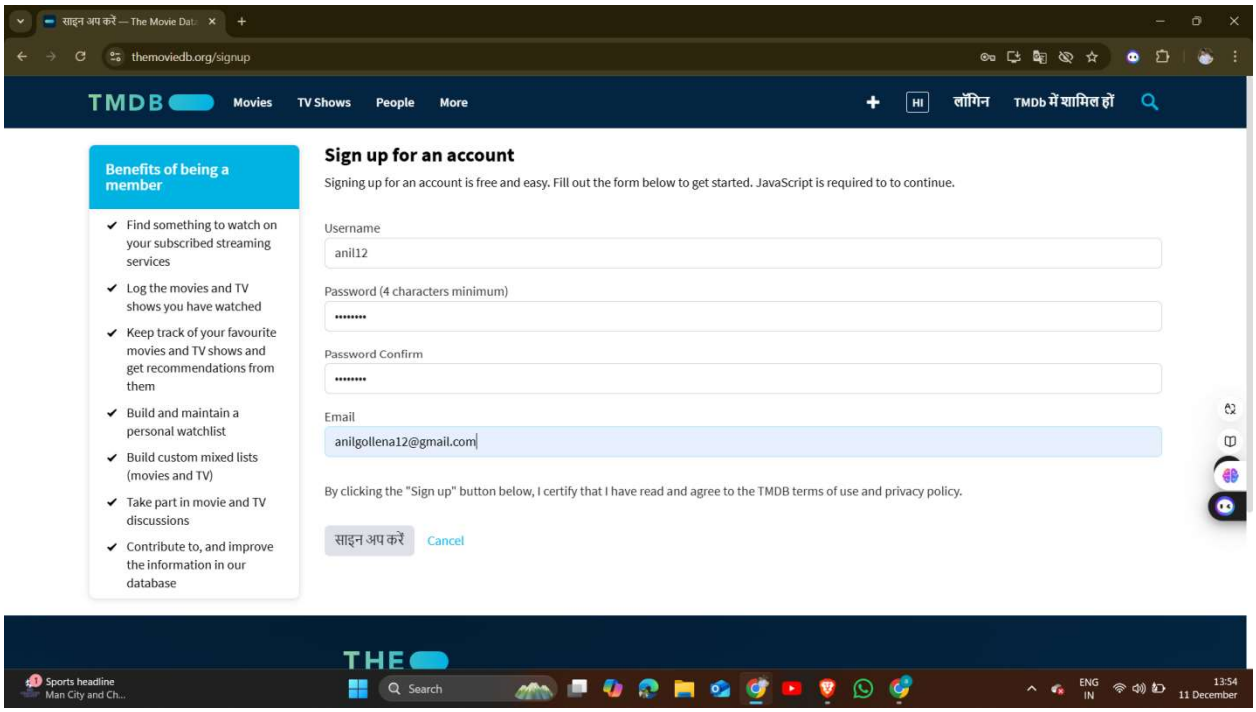




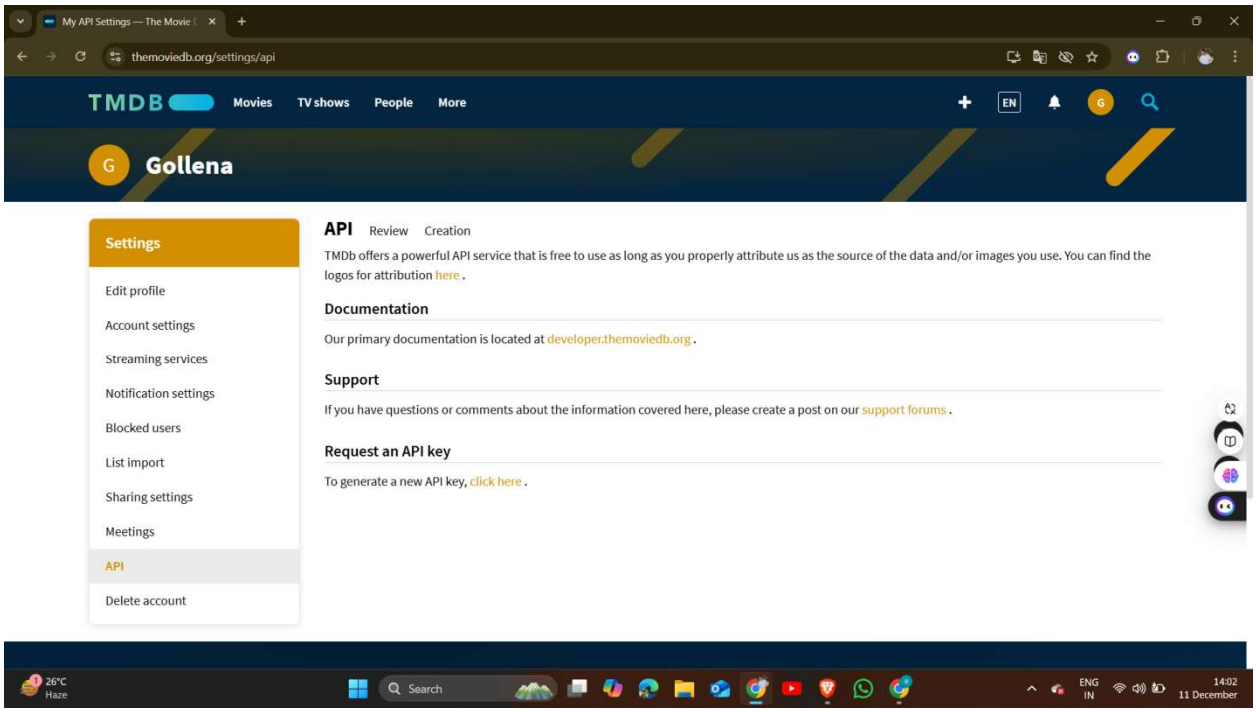
Now copy the public of ec2 instance, here Netflix page is deployed.



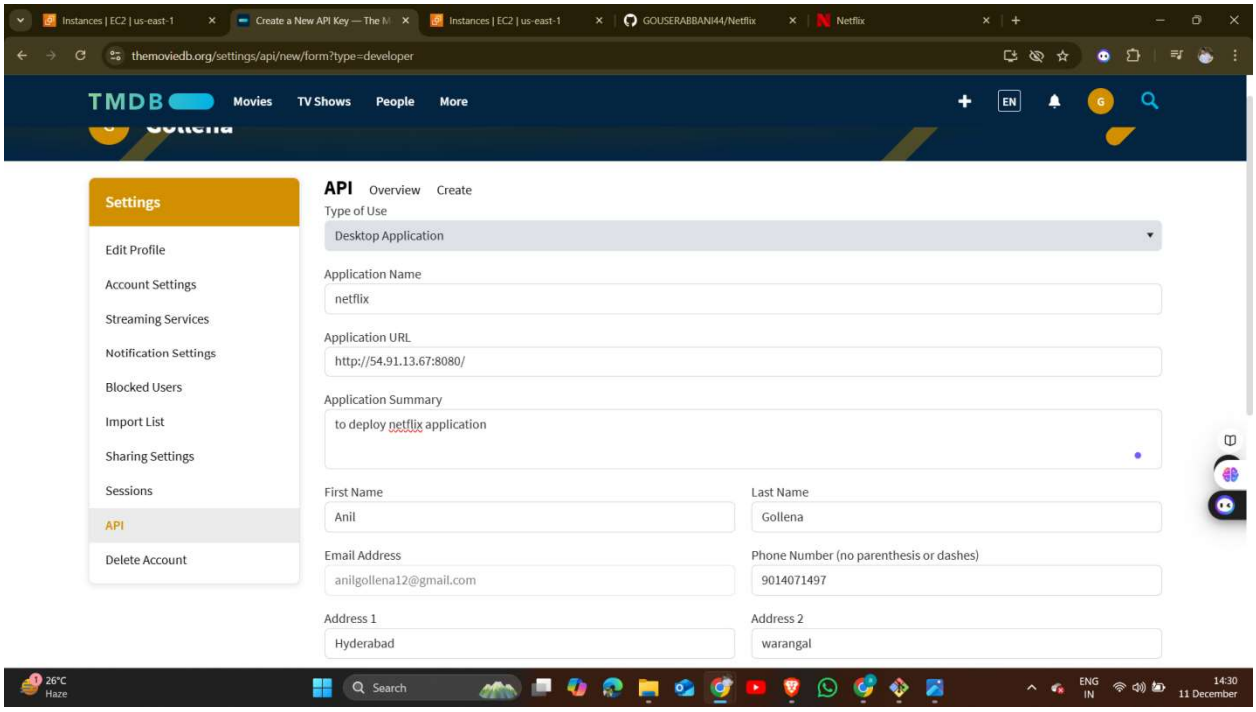
Now create TMDB movie database account in the browser here is the steps to create.



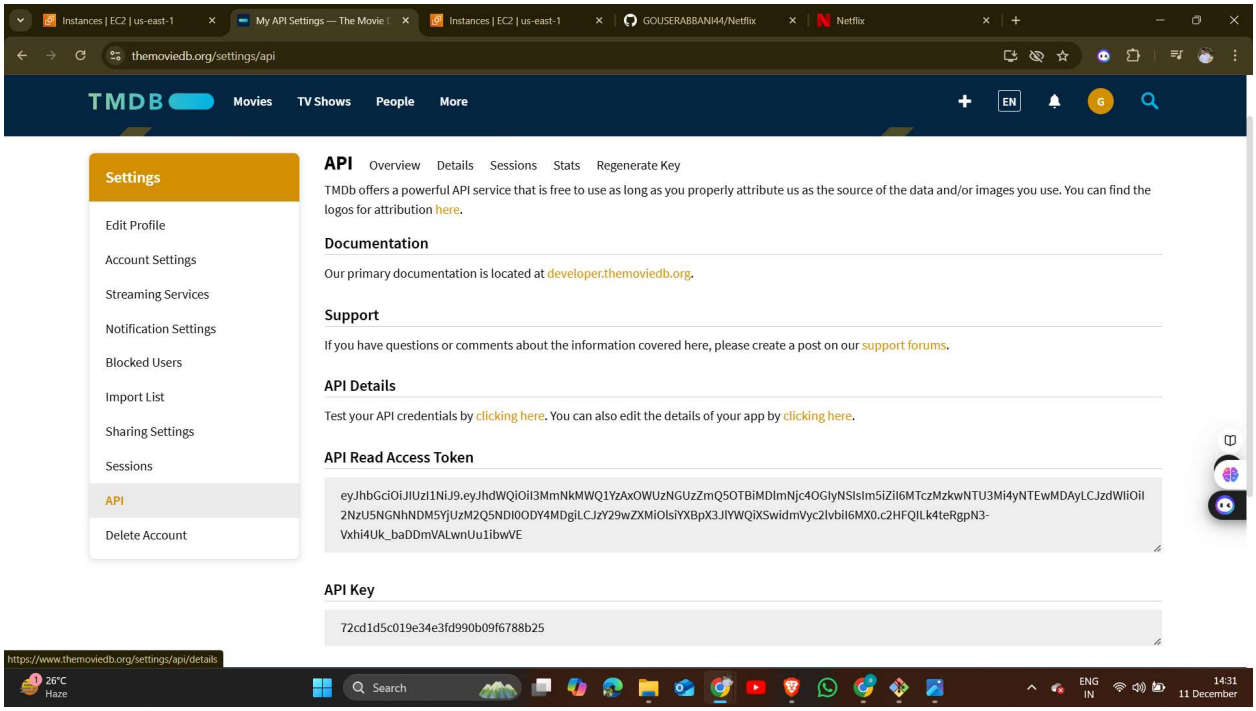
Now under settings→API→create a API



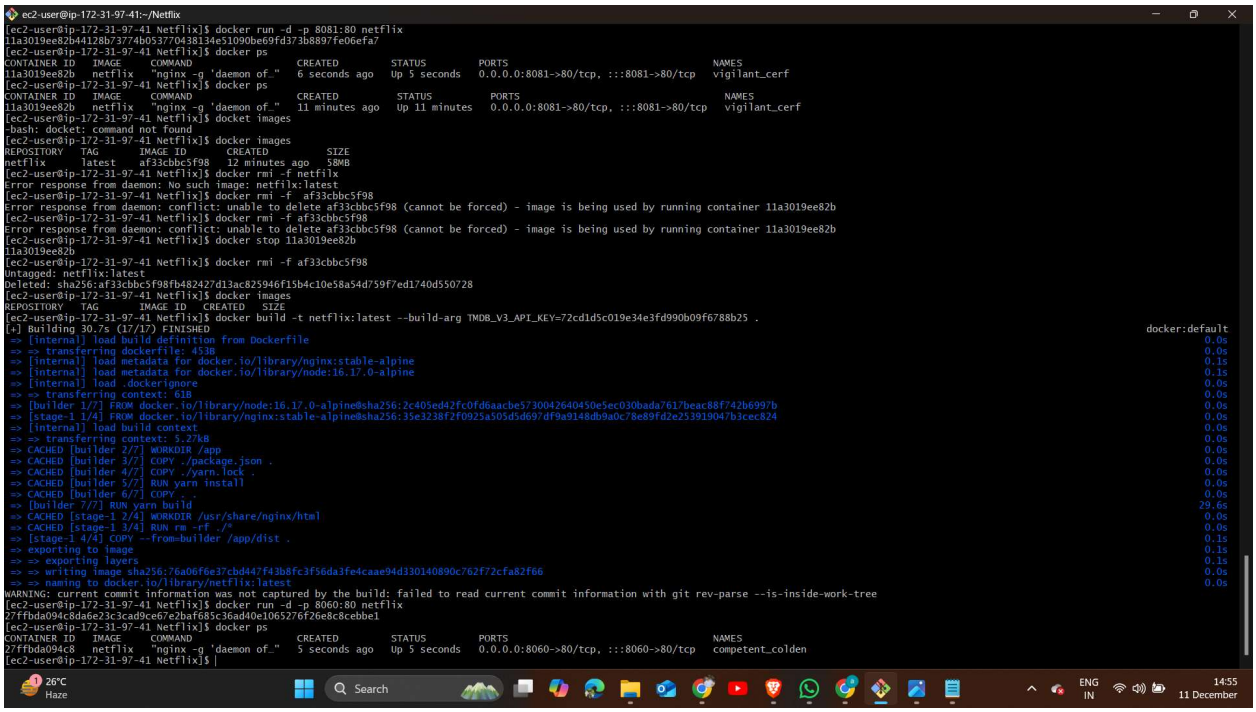
Choose desktop application.



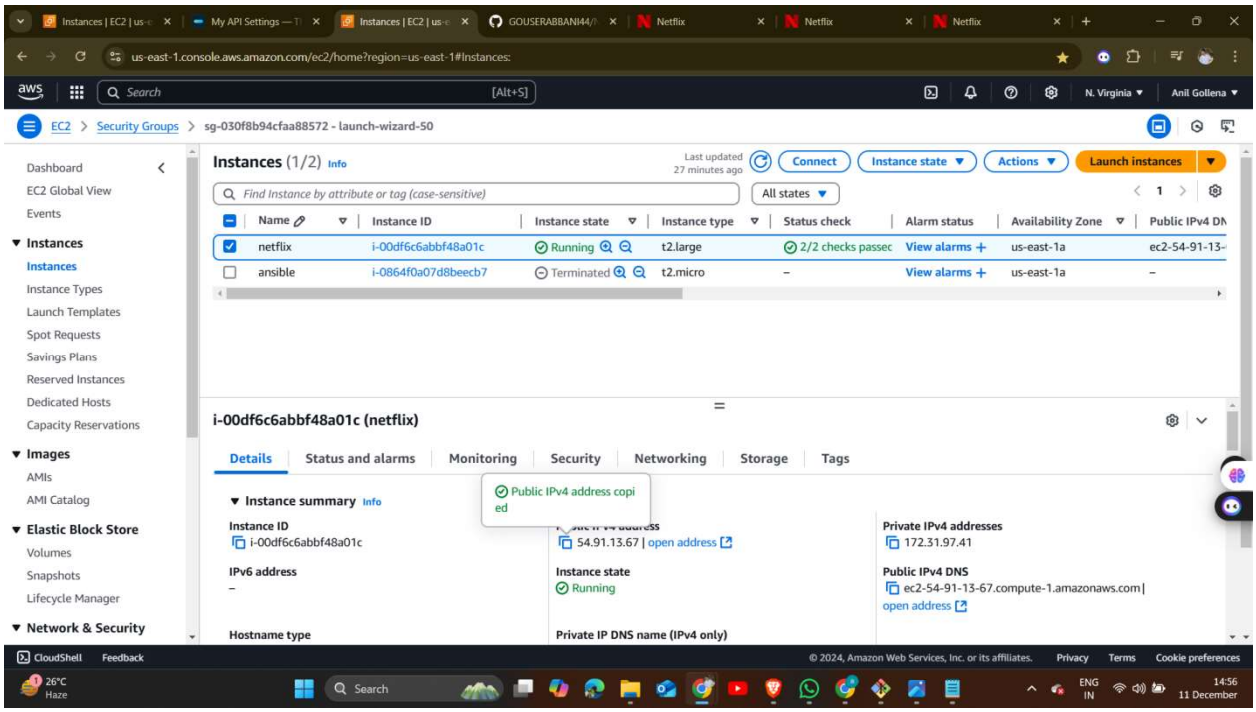
Here a new API is created copy that key

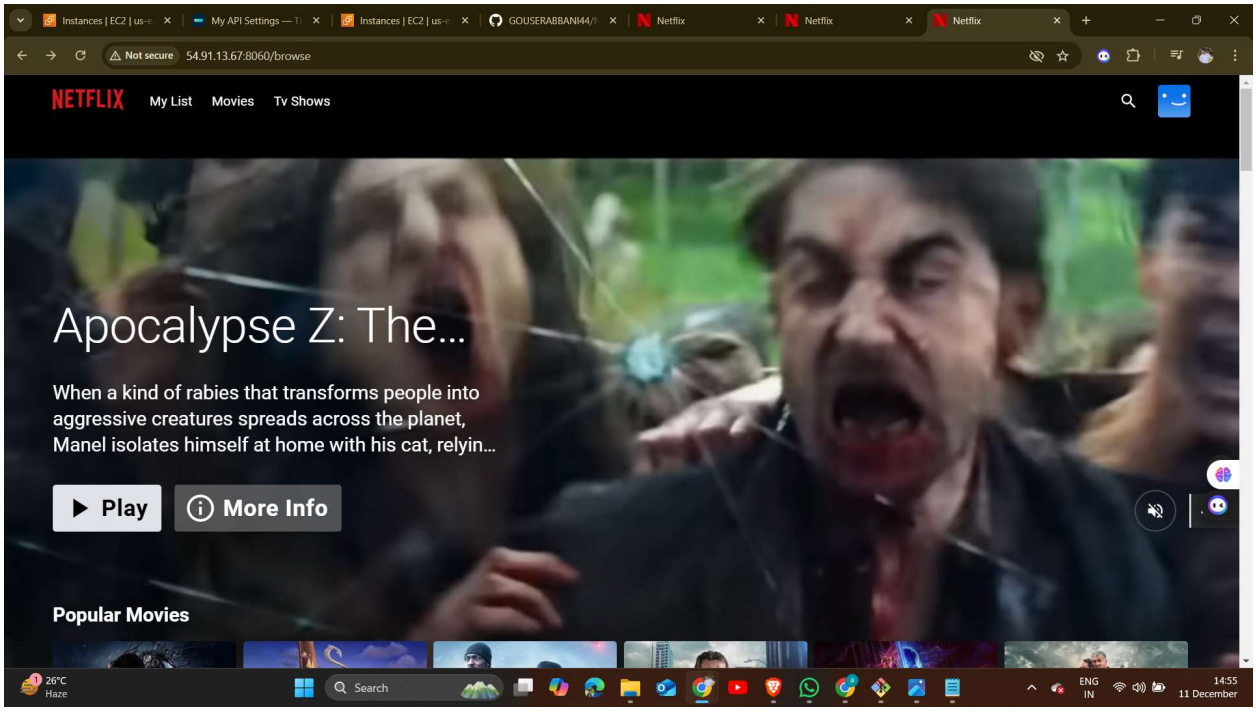


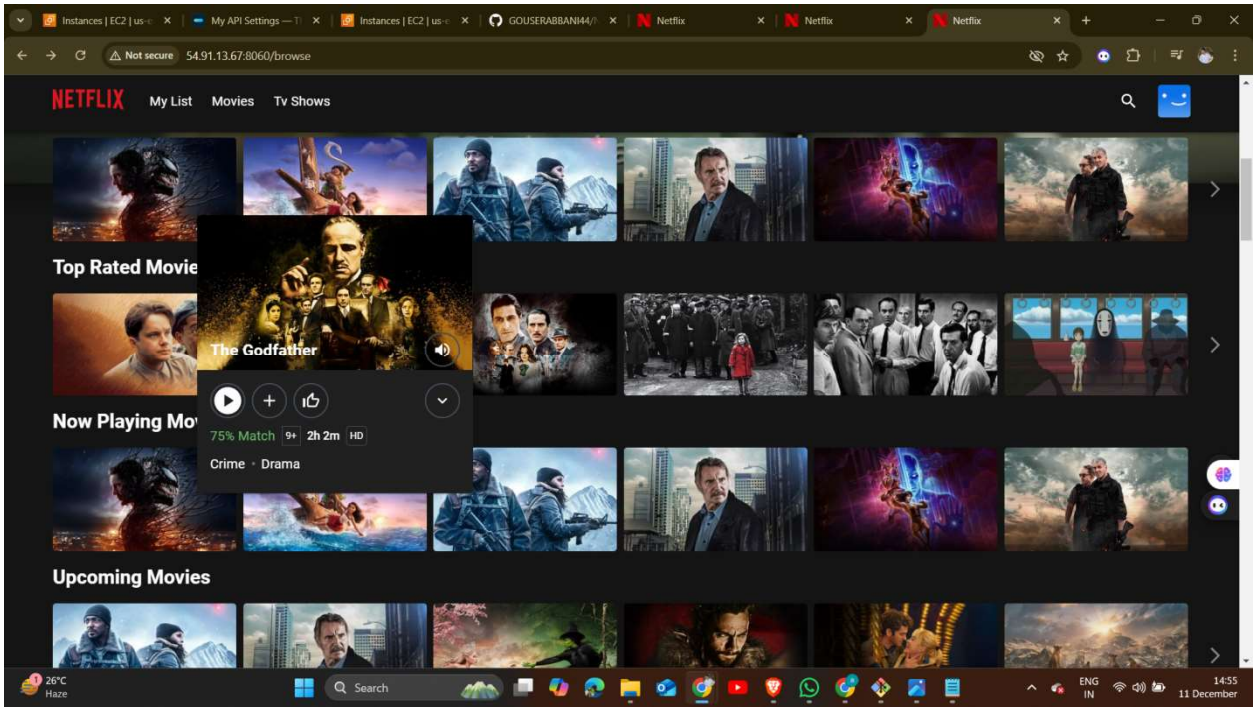
After creation of API now navigate to command shell and delete container and images and run the new container with the new API key



Now copy the public IP along with the port number in browser.




Not secure 54.91.13.67:8060/browse
NETFLIX
My List
Movies
Tv Shows
Apocalypse Z: The...
When a kind of rabies that transforms people into aggressive creatures spreads across the planet, Manel isolates himself at home with his cat, relyin...
Play
More Info
Popular Movies


Not secure 54.91.13.67:8060/browse
NETFLIX
My List
Movies
Tv Shows
Top Rated Movie
The Godfather
75% Match
9+
2h 2m
HD
Crime • Drama
Now Playing Mo
Upcoming Movies

PHSAE2: IMPLEMENTATION OF SECURITY WITH SONARQUBE AND TRIVY.

Overview of Tools:

- **SonarQube**: A code quality and security analysis tool that helps identify vulnerabilities, bugs, and code smells in source code during development.
- **Trivy**: An open-source vulnerability scanner for containers, file systems, and dependencies.

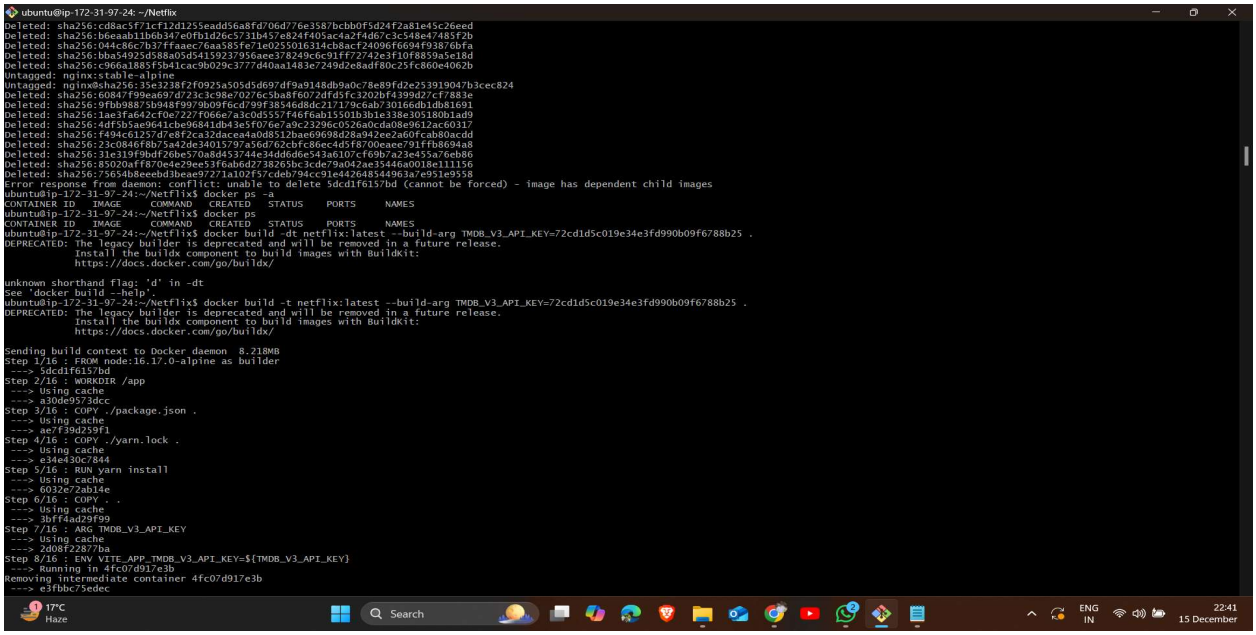
Use Case:

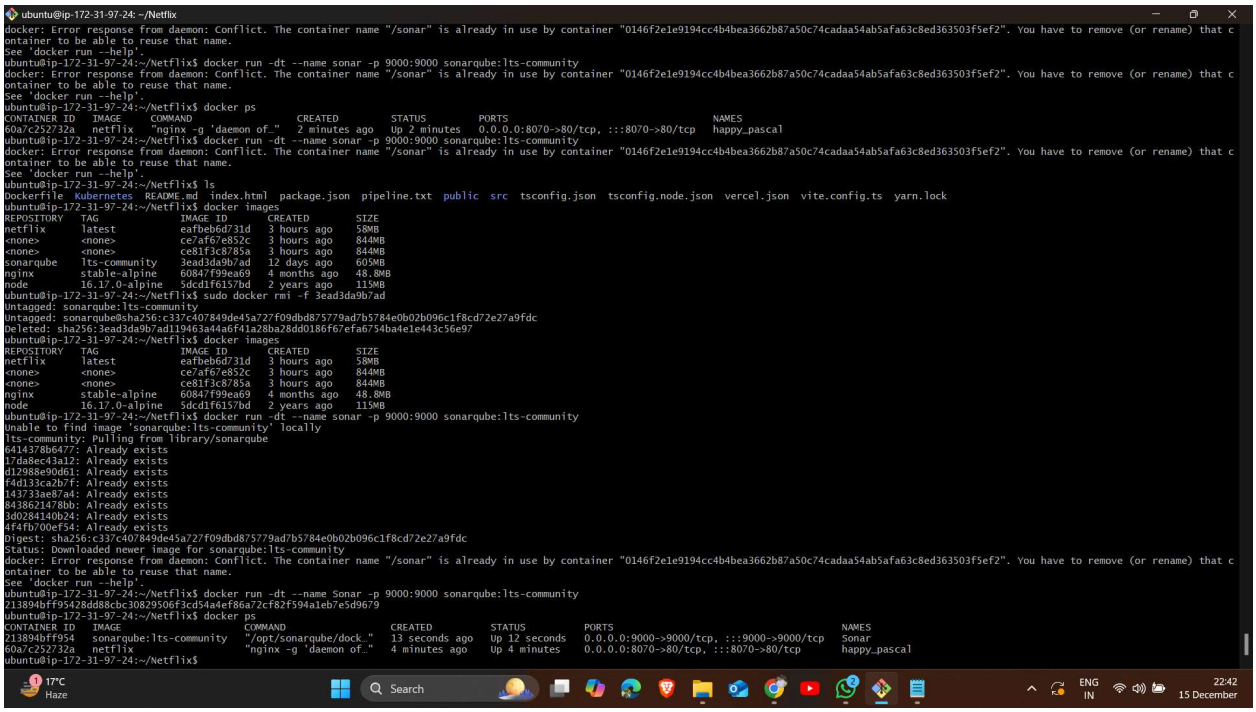
- **SonarQube**: Ensures the codebase adheres to secure coding standards and identifies issues like SQL injection, cross-site scripting (XSS), and other vulnerabilities.
- **Trivy**: Focuses on scanning container images, repositories, and dependencies for known security vulnerabilities.

here is the step by step process that I implementing security with sonarqube and trivy.

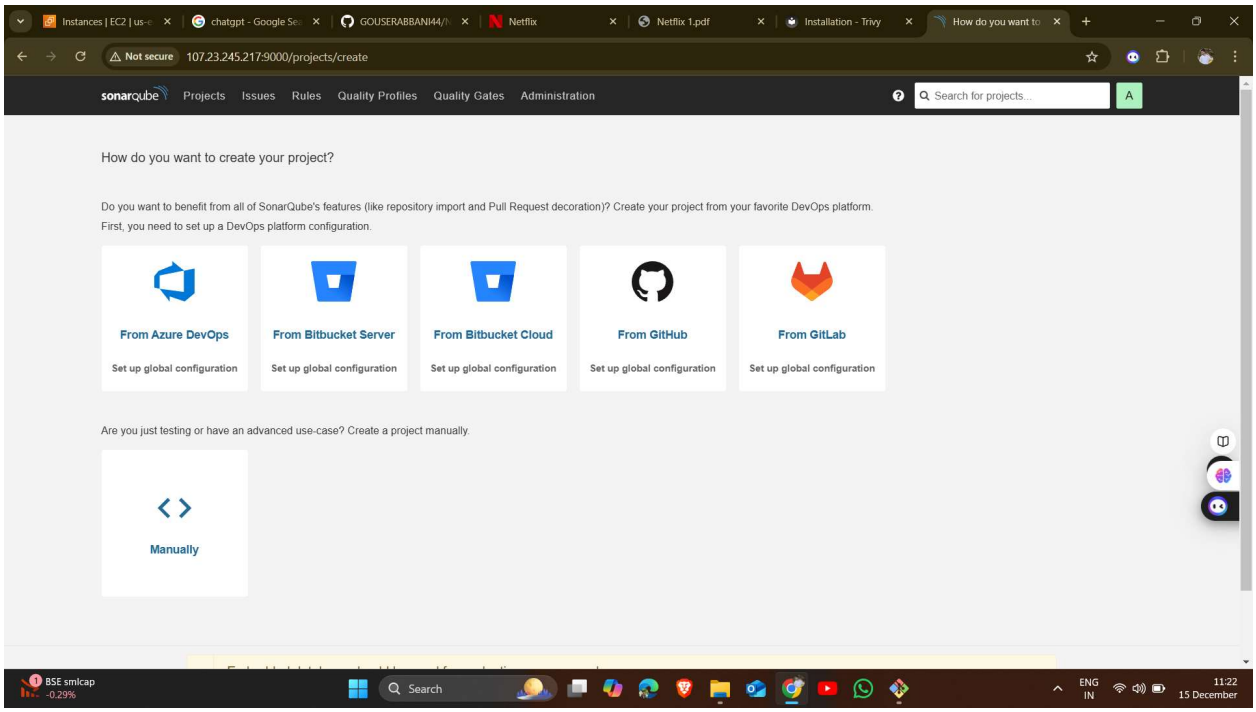
Now run the docker image with the SonarQube with the command

Docker run –dt --name sonar –p 9000:9000 sonarqube:lts-community

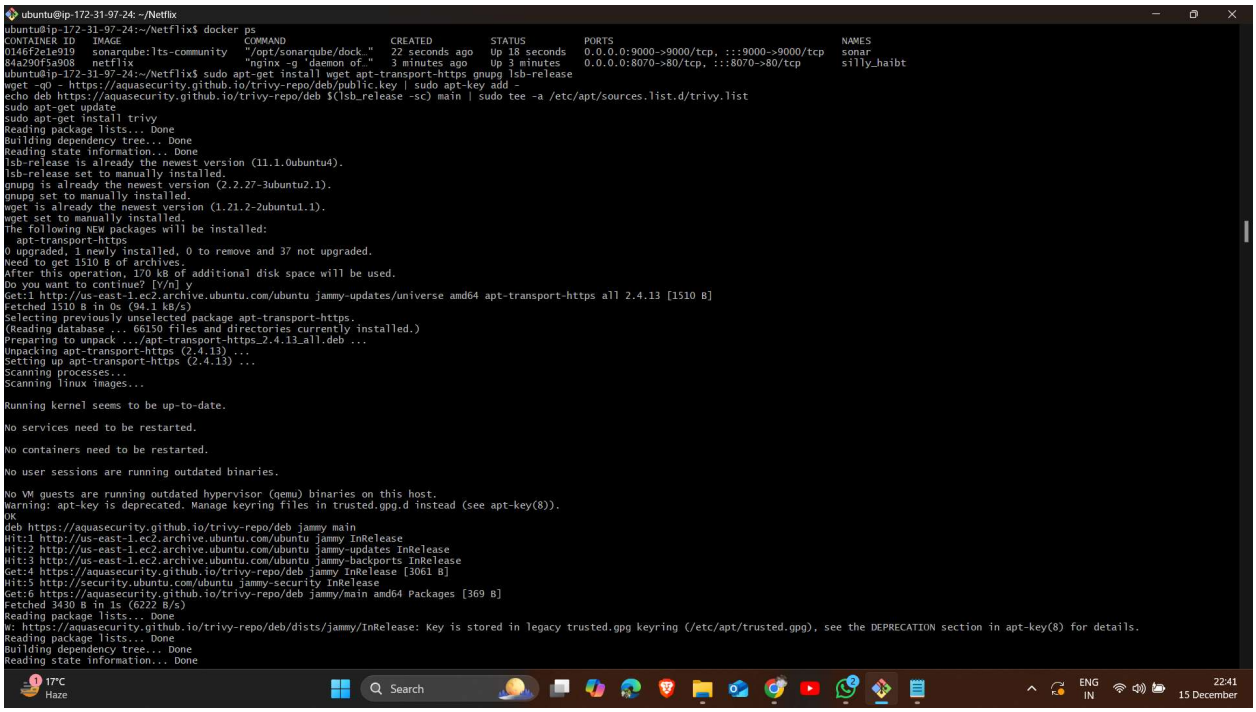




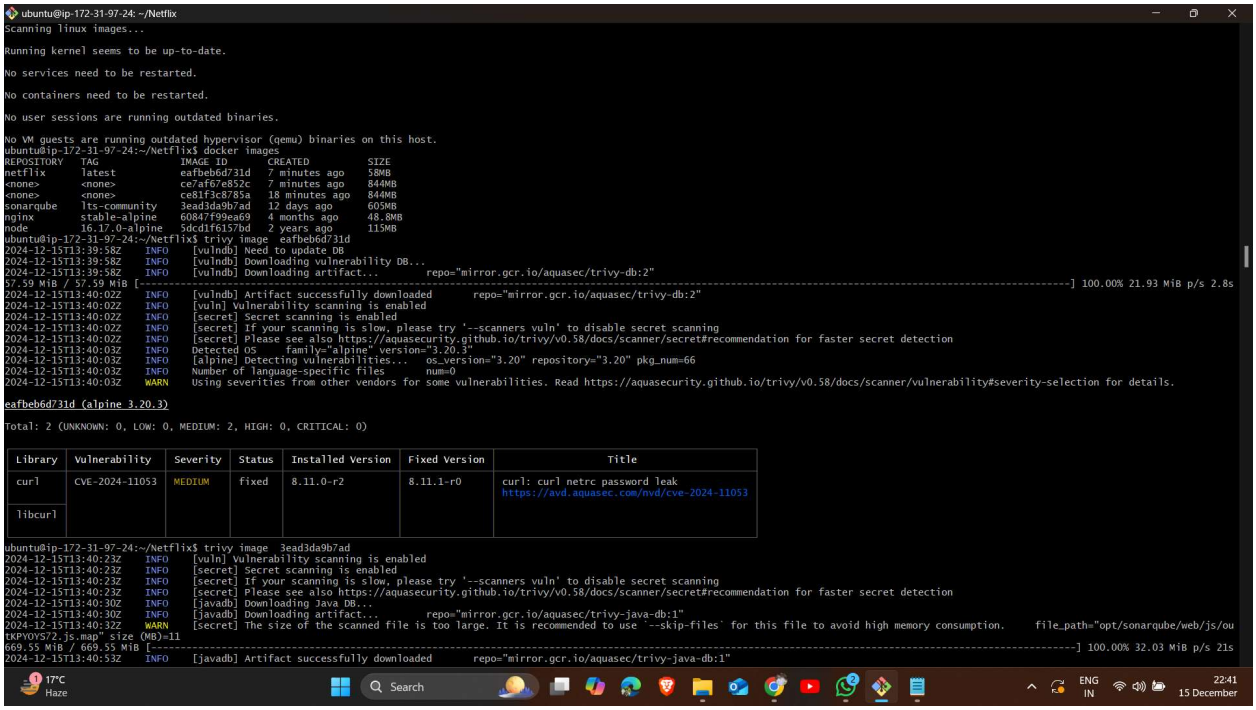
Now access the sonarqube application by copying the IP along with the port number 9000

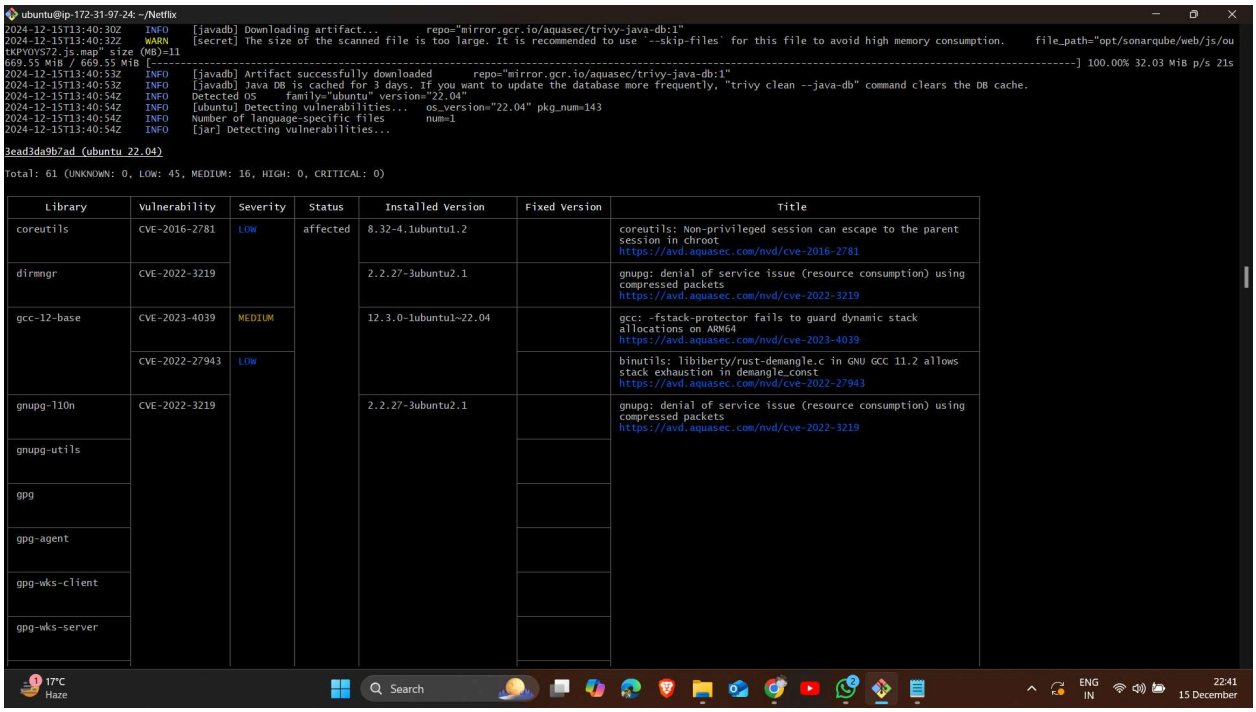


Now install trivy for that get address link from google and copy the address in the command shell.



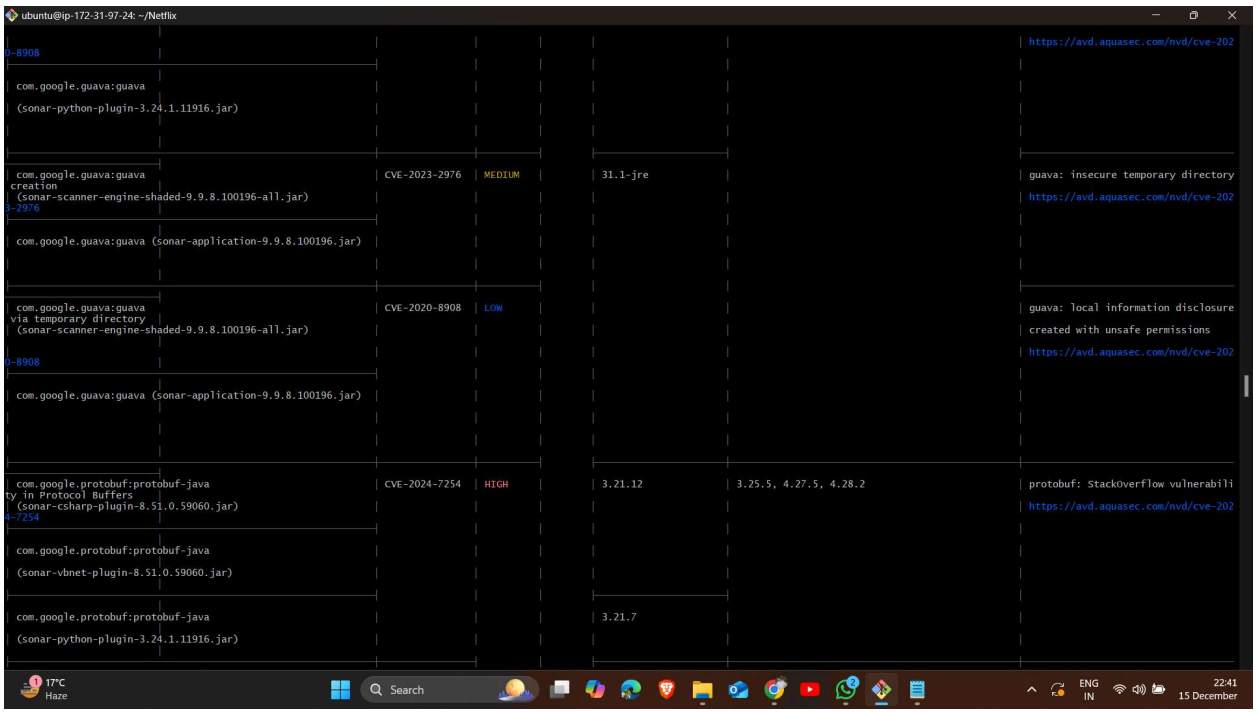
Now check the images through trivy.




ubuntu@ip-172-31-97-24: ~/Netflix
2024-12-15T13:40:30Z INFO [javadb] Downloading artifact... repo="mirror.gcr.io/aquasec/trivy-java-db:1"
2024-12-15T13:40:32Z WARN [secret] The size of the scanned file is too large. It is recommended to use `--skip-files` for this file to avoid high memory consumption. file_path="opt/sonarqube/web/js/ou
tKPYOYS72.js.map" size (MB)=11
669.55 MiB / 669.55 MiB [-----] 100.00% 32.03 MiB p/s 21s
2024-12-15T13:40:53Z INFO [javadb] Artifact successfully downloaded repo="mirror.gcr.io/aquasec/trivy-java-db:1"
2024-12-15T13:40:53Z INFO [javadb] Java DB is cached for 3 days. If you want to update the database more frequently, "trivy clean --java-db" command clears the DB cache.
2024-12-15T13:40:54Z INFO Detected OS family="ubuntu" version="22.04"
2024-12-15T13:40:54Z INFO [ubuntu] Detecting vulnerabilities... os_version="22.04" pkg_num=143
2024-12-15T13:40:54Z INFO Number of language-specific files num=1
2024-12-15T13:40:54Z INFO [jar] Detecting vulnerabilities...
3ead3da9b7ad (ubuntu 22.04)
Total: 61 (UNKNOWN: 0, LOW: 45, MEDIUM: 16, HIGH: 0, CRITICAL: 0)
Library | Vulnerability | Severity | Status | Installed Version | Fixed Version | Title
coreutils | CVE-2016-2781 | LOW | affected | 8.32-4.1ubuntu1.2 | | coreutils: Non-privileged session can escape to the parent session in chroot https://avd.aquasec.com/nvd/cve-2016-2781
dirmngr | CVE-2022-3219 | | | 2.2.27-3ubuntu2.1 | | gnupg: denial of service issue (resource consumption) using compressed packets https://avd.aquasec.com/nvd/cve-2022-3219
gcc-12-base | CVE-2023-4039 | MEDIUM | | 12.3.0-1ubuntu1~22.04 | | gcc: -fstack-protector fails to guard dynamic stack allocations on ARM64 https://avd.aquasec.com/nvd/cve-2023-4039
CVE-2022-27943 | LOW | | | | binutils: libiberty/rust-demangle.c in GNU GCC 11.2 allows stack exhaustion in demangle_const https://avd.aquasec.com/nvd/cve-2022-27943
gnupg-l10n | CVE-2022-3219 | | | 2.2.27-3ubuntu2.1 | | gnupg: denial of service issue (resource consumption) using compressed packets https://avd.aquasec.com/nvd/cve-2022-3219
gnupg-utils
gpg
gpg-agent
gpg-wks-client
gpg-wks-server
17°C Haze
Search
ENG IN
22:41 15 December


ubuntu@ip-172-31-97-24: ~/Netflix
gpgv
libc-bin | CVE-2016-20013 | | | 2.35-0ubuntu3.8 | | sha256crypt and sha512crypt through 0.6 allow attackers to cause a denial of... https://avd.aquasec.com/nvd/cve-2016-20013
libc6
libgcc-s1 | CVE-2023-4039 | MEDIUM | | 12.3.0-1ubuntu1~22.04 | | gcc: -fstack-protector fails to guard dynamic stack allocations on ARM64 https://avd.aquasec.com/nvd/cve-2023-4039
CVE-2022-27943 | LOW | | | | binutils: libiberty/rust-demangle.c in GNU GCC 11.2 allows stack exhaustion in demangle_const https://avd.aquasec.com/nvd/cve-2022-27943
libgcrypt20 | CVE-2024-2236 | | | 1.9.4-3ubuntu3 | | libgcrypt: vulnerable to Marvin Attack https://avd.aquasec.com/nvd/cve-2024-2236
libgssapi-krb5-2 | CVE-2024-26462 | MEDIUM | | 1.19.2-2ubuntu0.4 | | krb5: Memory leak at /krb5/src/kdc/ndr.c https://avd.aquasec.com/nvd/cve-2024-26462
CVE-2024-26458 | LOW | | | | krb5: Memory leak at /krb5/src/lib/rpc/pmap_rmt.c https://avd.aquasec.com/nvd/cve-2024-26458
CVE-2024-26461 | | | | | krb5: Memory leak at /krb5/src/lib/gssapi/krb5/k5sealv3.c https://avd.aquasec.com/nvd/cve-2024-26461
libk5crypto3 | CVE-2024-26462 | MEDIUM | | | | krb5: Memory leak at /krb5/src/kdc/ndr.c https://avd.aquasec.com/nvd/cve-2024-26462
CVE-2024-26458 | LOW | | | | krb5: Memory leak at /krb5/src/lib/rpc/pmap_rmt.c https://avd.aquasec.com/nvd/cve-2024-26458
CVE-2024-26461 | | | | | krb5: Memory leak at /krb5/src/lib/gssapi/krb5/k5sealv3.c https://avd.aquasec.com/nvd/cve-2024-26461
libkrb5-3 | CVE-2024-26462 | MEDIUM | | | | krb5: Memory leak at /krb5/src/kdc/ndr.c https://avd.aquasec.com/nvd/cve-2024-26462
CVE-2024-26458 | LOW | | | | krb5: Memory leak at /krb5/src/lib/rpc/pmap_rmt.c https://avd.aquasec.com/nvd/cve-2024-26458
CVE-2024-26461 | | | | | krb5: Memory leak at /krb5/src/lib/gssapi/krb5/k5sealv3.c https://avd.aquasec.com/nvd/cve-2024-26461
libkrb5support0 | CVE-2024-26462 | MEDIUM | | | | krb5: Memory leak at /krb5/src/kdc/ndr.c https://avd.aquasec.com/nvd/cve-2024-26462
CVE-2024-26458 | LOW | | | | krb5: Memory leak at /krb5/src/lib/rpc/pmap_rmt.c https://avd.aquasec.com/nvd/cve-2024-26458
17°C Haze
Search
ENG IN
22:41 15 December


ubuntu@ip-172-31-97-24: ~/Netflix
login | CVE-2023-29383 | | 1:4.8.1-2ubuntu2.2 | | shadow: Improper input validation in shadow-utils package utility chfn https://avd.aquasec.com/nvd/cve-2023-29383
ncurses-base | CVE-2023-45918 | | 6.3-2ubuntu0.1 | | ncurses: NULL pointer dereference in tgetstr in tinfo/lib_termcap.c https://avd.aquasec.com/nvd/cve-2023-45918
CVE-2023-50495 | | | | ncurses: segmentation fault via _nc_wrap_entry() https://avd.aquasec.com/nvd/cve-2023-50495
ncurses-bin | CVE-2023-45918 | | | | ncurses: NULL pointer dereference in tgetstr in tinfo/lib_termcap.c https://avd.aquasec.com/nvd/cve-2023-45918
CVE-2023-50495 | | | | ncurses: segmentation fault via _nc_wrap_entry() https://avd.aquasec.com/nvd/cve-2023-50495
openssl | CVE-2024-41996 | | 3.0.2-0ubuntu1.18 | | openssl: remote attackers (from the client side) to trigger unnecessarily expensive server-side... https://avd.aquasec.com/nvd/cve-2024-41996
passwd | CVE-2023-29383 | | 1:4.8.1-2ubuntu2.2 | | shadow: Improper input validation in shadow-utils package utility chfn https://avd.aquasec.com/nvd/cve-2023-29383
wget | CVE-2021-31879 | MEDIUM | 1.21.2-2ubuntu1.1 | | wget: authorization header disclosure on redirect https://avd.aquasec.com/nvd/cve-2021-31879
2024-12-15T13:40:54Z INFO Table result includes only package filenames. Use '--format json' option to get the full path to the package file.
Java (jar)
Total: 37 (UNKNOWN: 0, LOW: 4, MEDIUM: 14, HIGH: 18, CRITICAL: 1)
Library | Vulnerability | Severity | Status | Installed Version | Fixed Version | Title
ch.qos.logback:logback-classic (sonar-scanner-engine-shaded-9.9.8.100196-all.jar) | CVE-2023-6378 | HIGH | fixed | 1.2.10 | 1.3.12, 1.4.12, 1.2.13 | logback: serialization vulnerability in logback receiver https://avd.aquasec.com/nvd/cve-2023-6378
ch.qos.logback:logback-classic (sonar-application-9.9.8.100196.jar)
ch.qos.logback:logback-core (sonar-scanner-engine-shaded-9.9.8.100196-all.jar)
17°C Haze
Search
ENG IN
22:41 15 December


ubuntu@ip-172-31-97-24: ~/Netflix
https://avd.aquasec.com/nvd/cve-2020-8908
com.google.guava:guava (sonar-python-plugin-3.24.1.11916.jar)
com.google.guava:guava | CVE-2023-2976 | MEDIUM | 31.1-jre | guava: insecure temporary directory creation (sonar-scanner-engine-shaded-9.9.8.100196-all.jar) https://avd.aquasec.com/nvd/cve-2023-2976
com.google.guava:guava (sonar-application-9.9.8.100196.jar)
com.google.guava:guava | CVE-2020-8908 | LOW | guava: local information disclosure via temporary directory (sonar-scanner-engine-shaded-9.9.8.100196-all.jar) created with unsafe permissions https://avd.aquasec.com/nvd/cve-2020-8908
com.google.guava:guava (sonar-application-9.9.8.100196.jar)
com.google.protobuf:protobuf-java | CVE-2024-7254 | HIGH | 3.21.12 | 3.25.5, 4.27.5, 4.28.2 | protobuf: StackOverflow vulnerability in Protocol Buffers (sonar-csharp-plugin-8.51.0.59060.jar) https://avd.aquasec.com/nvd/cve-2024-7254
com.google.protobuf:protobuf-java (sonar-vbnet-plugin-8.51.0.59060.jar)
com.google.protobuf:protobuf-java | 3.21.7 (sonar-python-plugin-3.24.1.11916.jar)
17°C Haze
Search
ENG IN
22:41 15 December

Here the trivy check is passed for both the images.

This I implemented security using sonarqube and Trivy.

PHASE 3: AUTOMATE THE WHOLE DEPLOYMENT USING JENKINS CI/CD PIPELINE.

About Jenkins:

- **Jenkins** is an open-source automation server widely used for **continuous integration (CI)** and **continuous delivery/deployment (CD)**.
- It automates repetitive tasks like building, testing, and deploying software, enhancing productivity and reducing manual effort.
- Jenkins supports a vast ecosystem of **plugins** for integrating with tools, frameworks, and platforms.

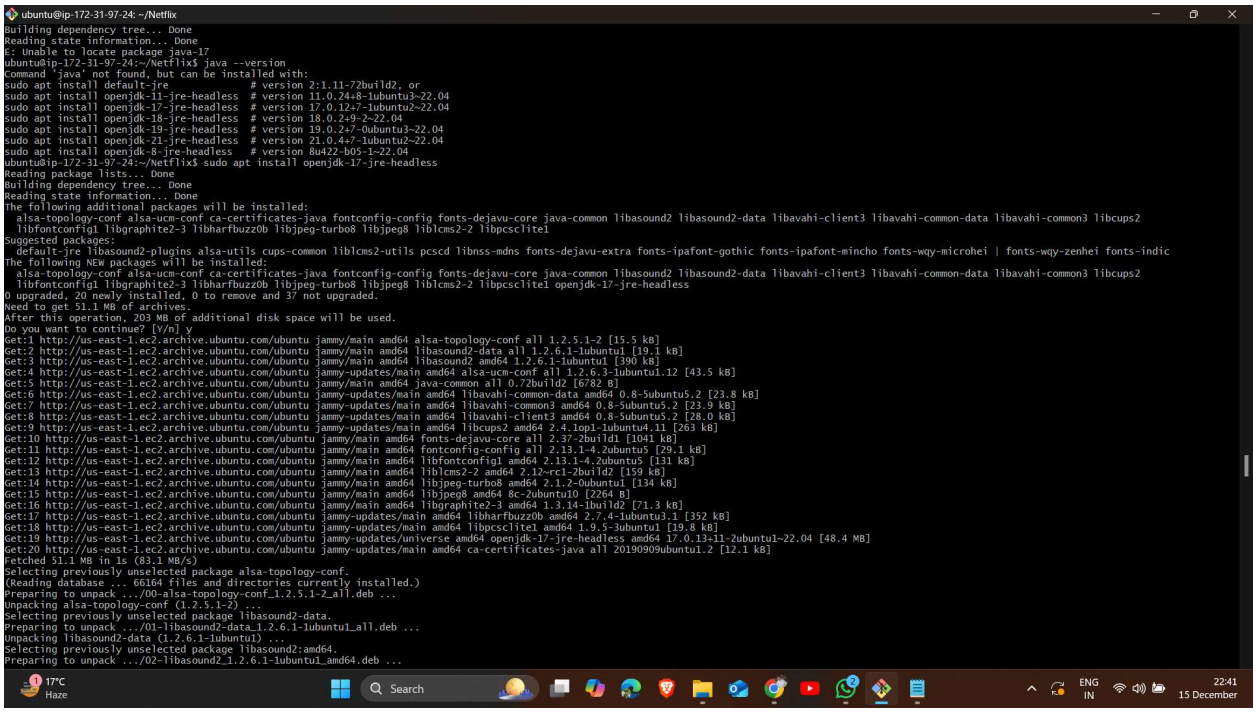
Jenkins CI/CD Pipeline:

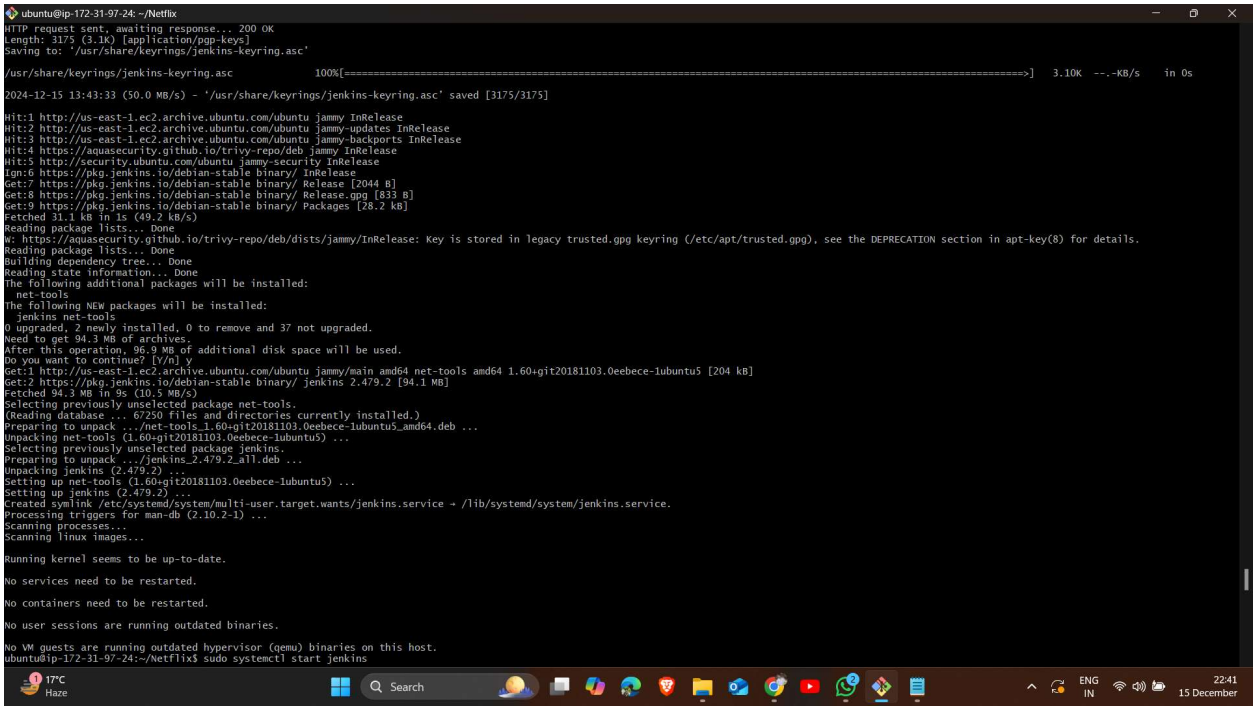
- A **Jenkins Pipeline** is a suite of plugins that enables defining and automating the steps involved in CI/CD as **code** (Pipeline-as-Code).
- It is written using a **Groovy-based Domain-Specific Language (DSL)**.
- Supports **Declarative** and **Scripted** syntax for creating pipelines.

Advantages of jenkins pipelines:

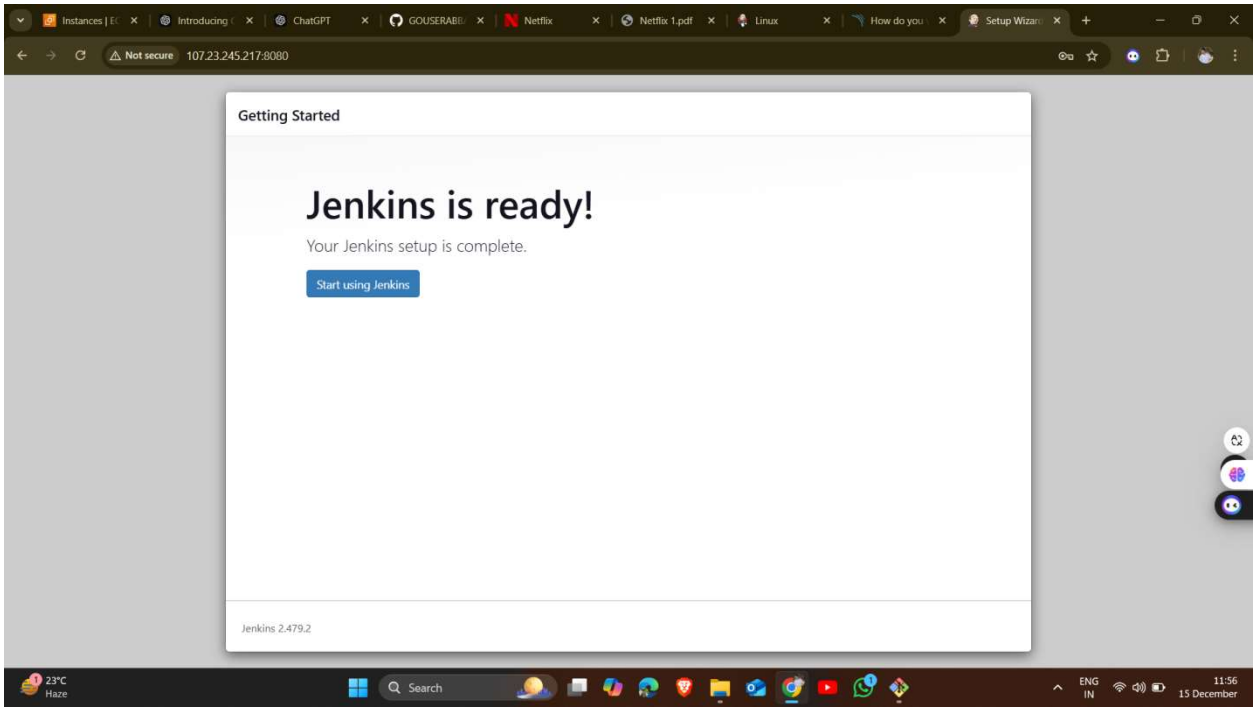
- Facilitates **DevOps** practices by automating CI/CD processes.
- Ensures faster delivery and higher software quality.
- Easily customizable for various workflows.

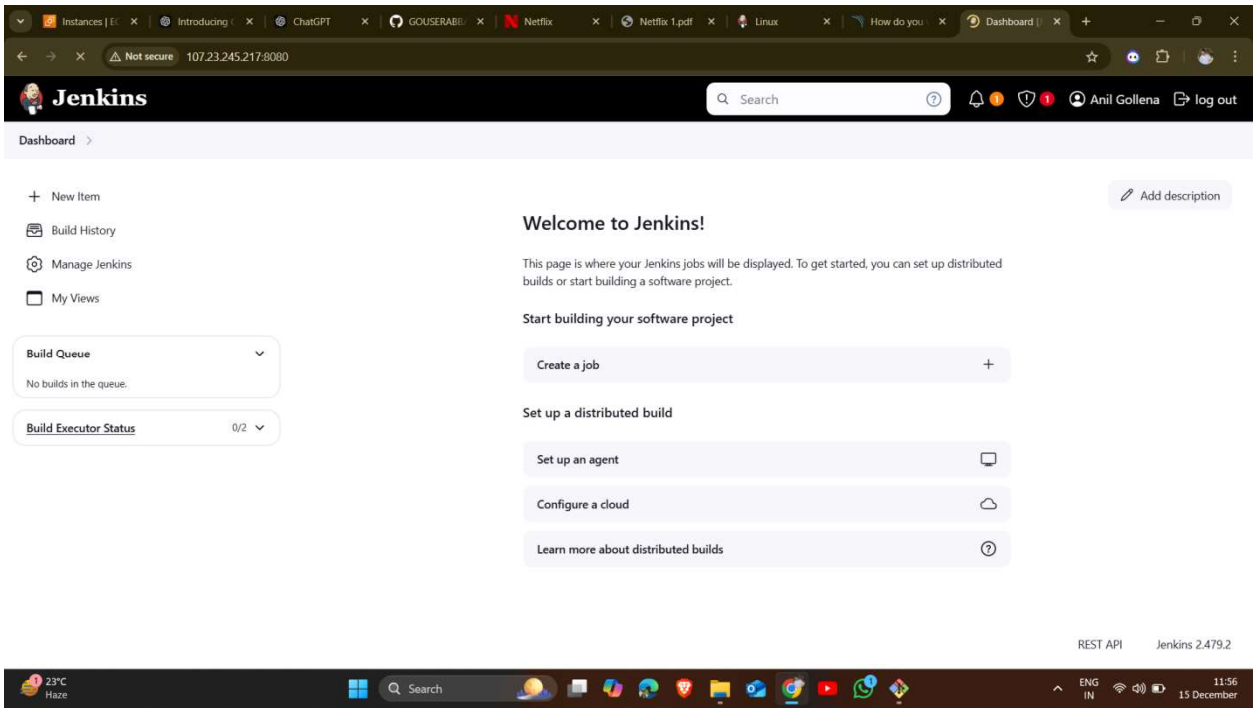
HERE, install Jenkins and set up before setting up install java dependency 17th version.





Now copy the public ip of Jenkins, access the Jenkins.

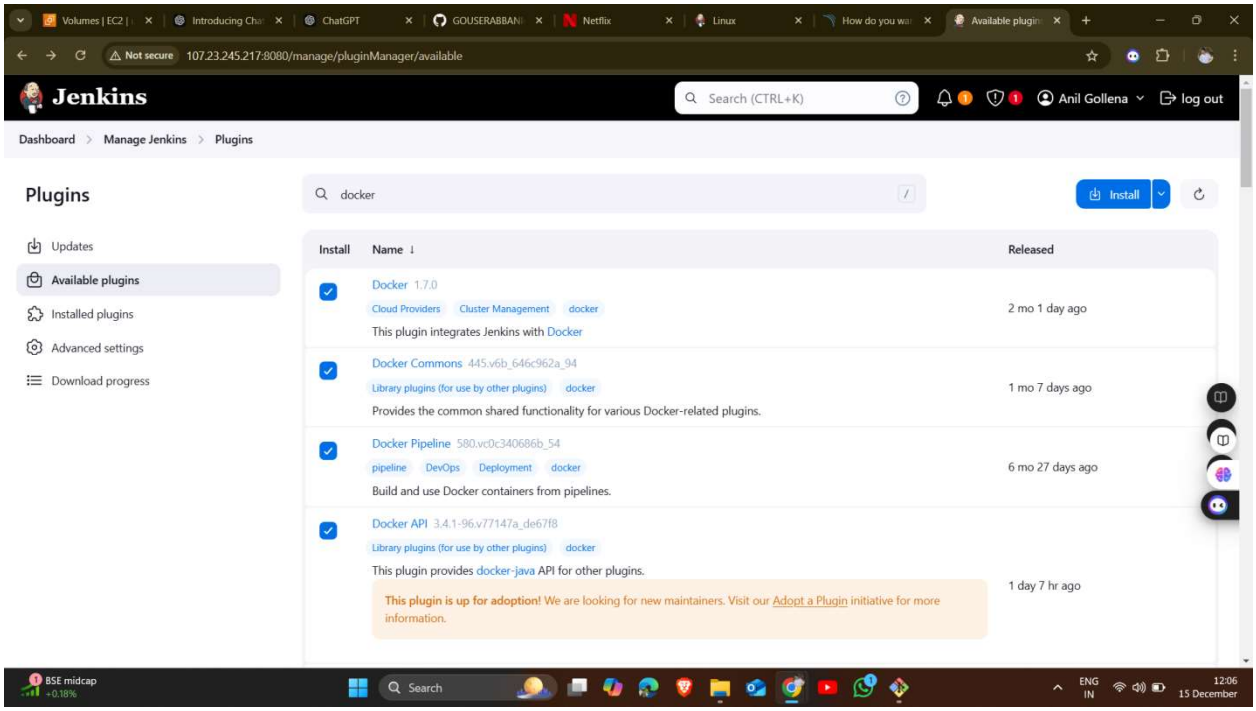


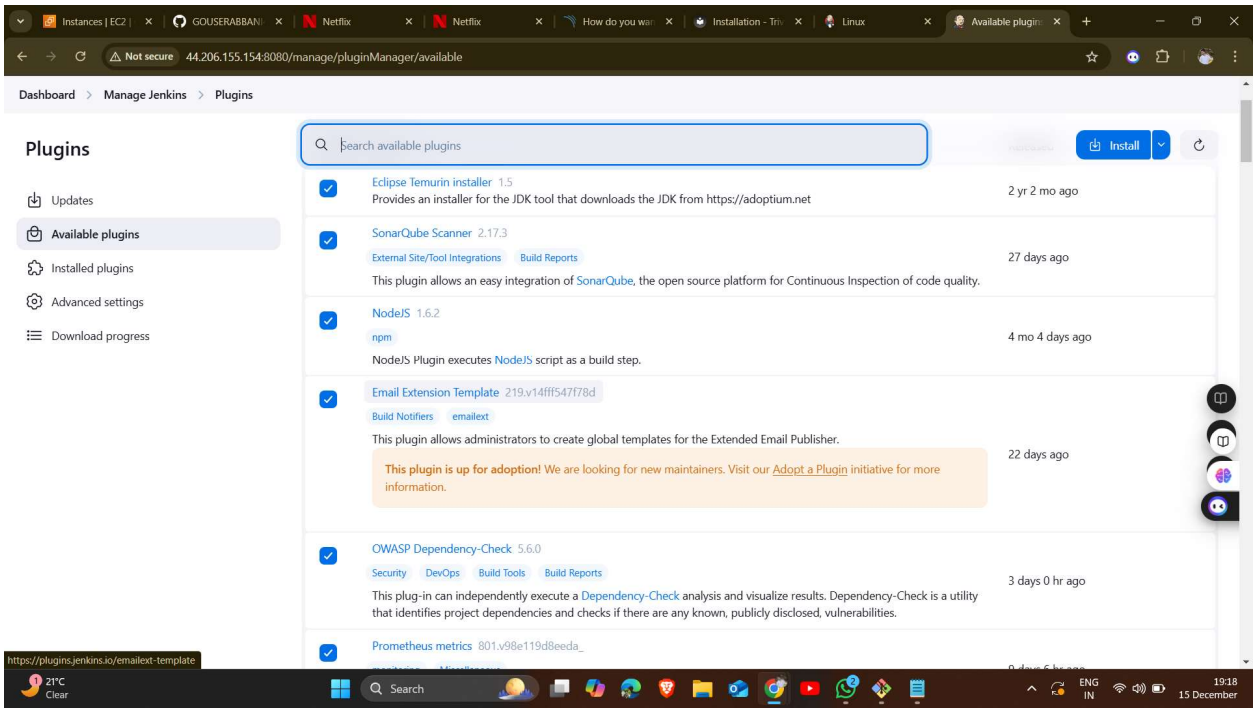


Initially, go to Dashboard→manage Jenkins→pluggins→Available pluggins

Here install few pluggins here are:

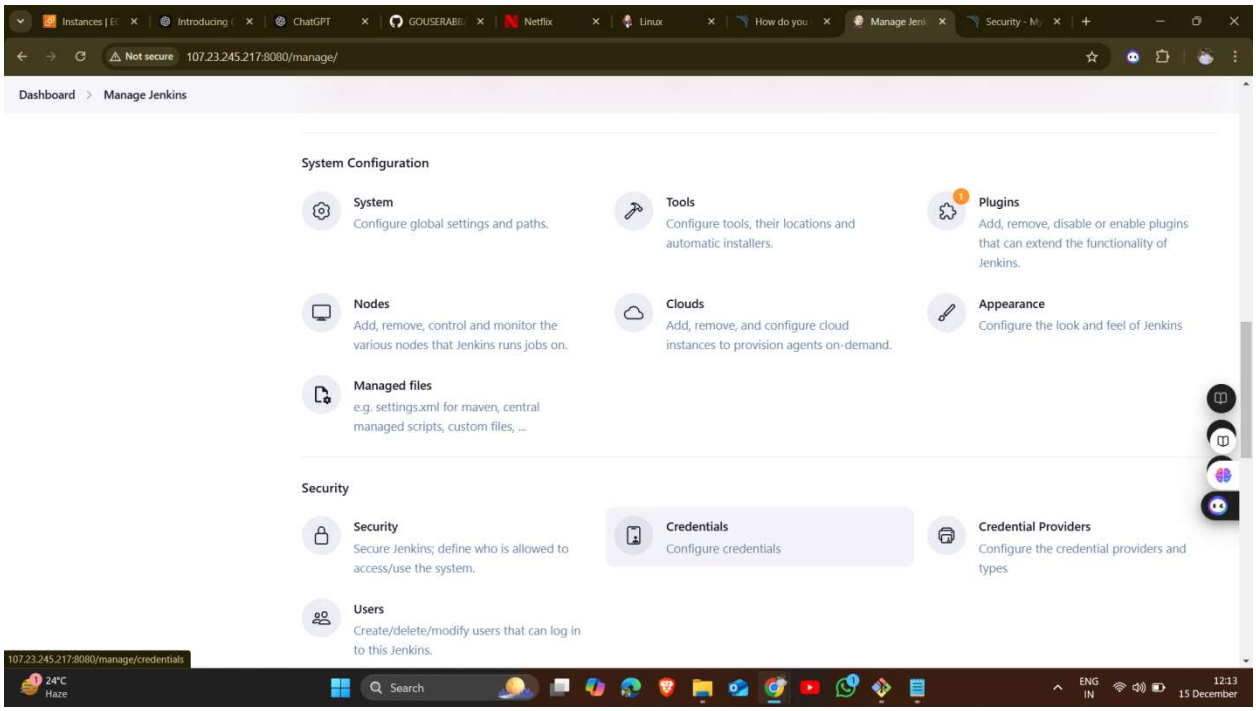
i. Eclipse temurin installer
ii. Sonarqube scanner
iii. NodeJS
iv. Email extension template
v. OWSP dependency check
vi. Prometheus metricks
vii. And all docker related pluggins



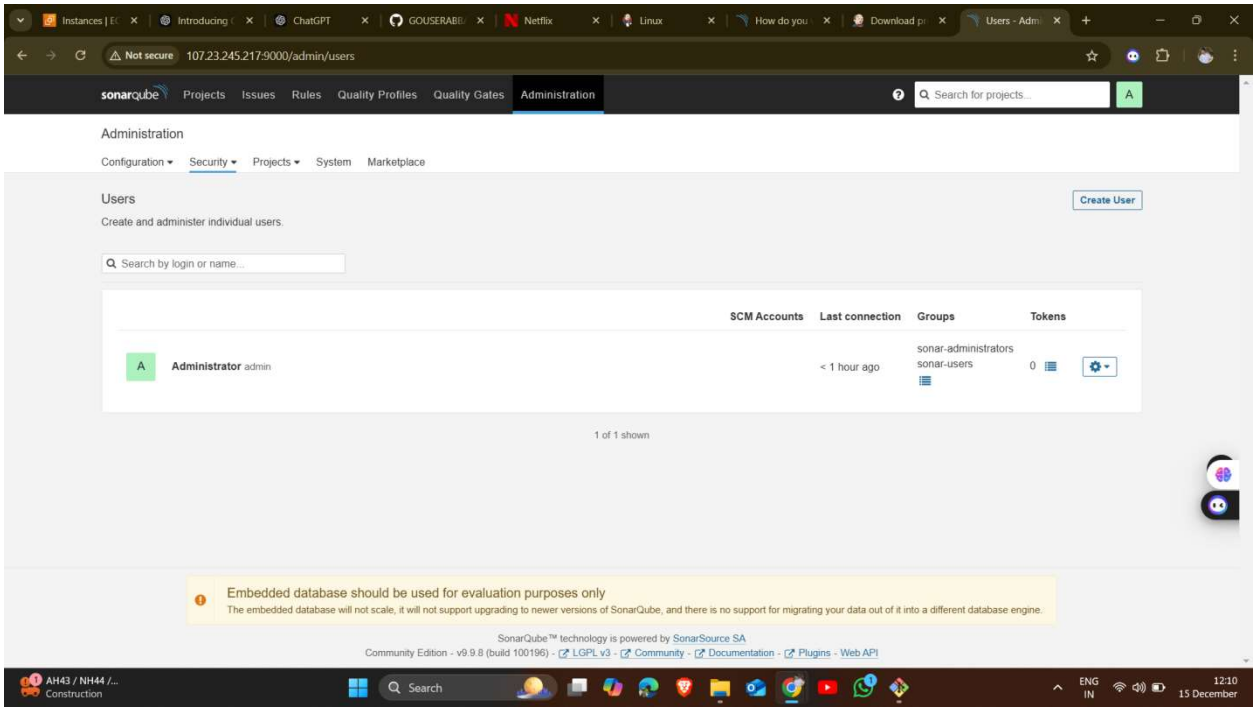


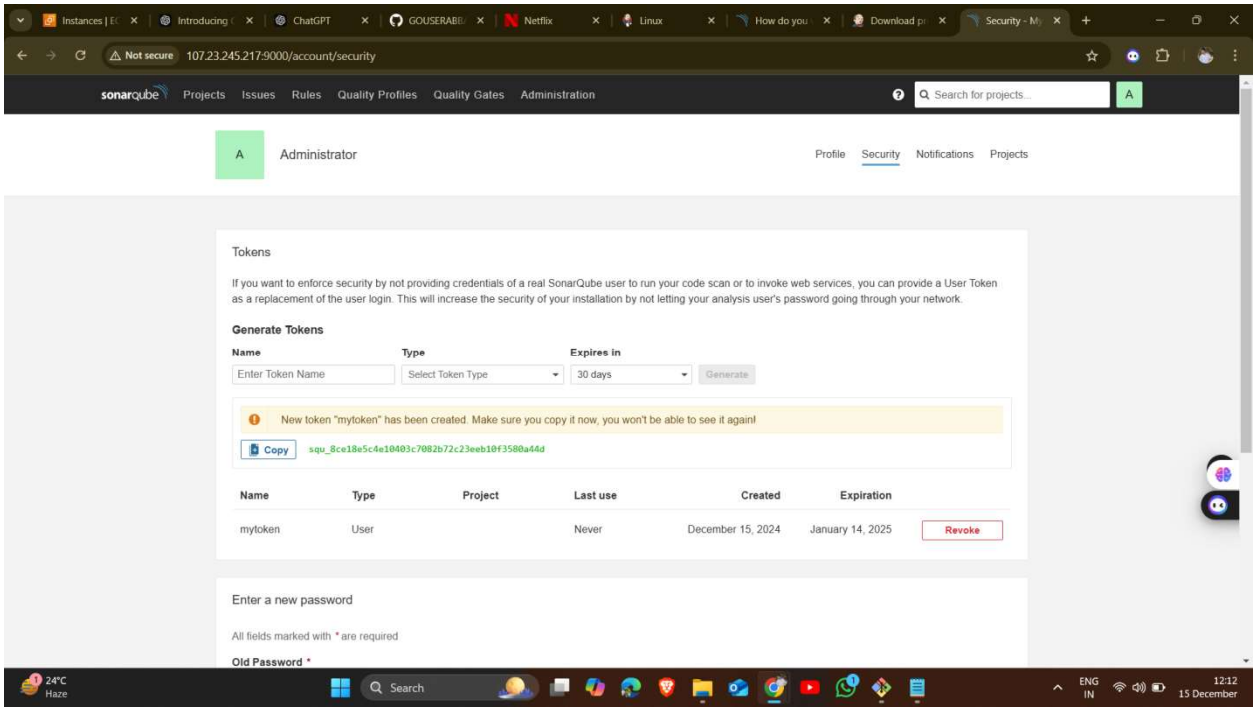
After installation of related plugins,

Now go to dashboard add sonarqube and docker credentials.

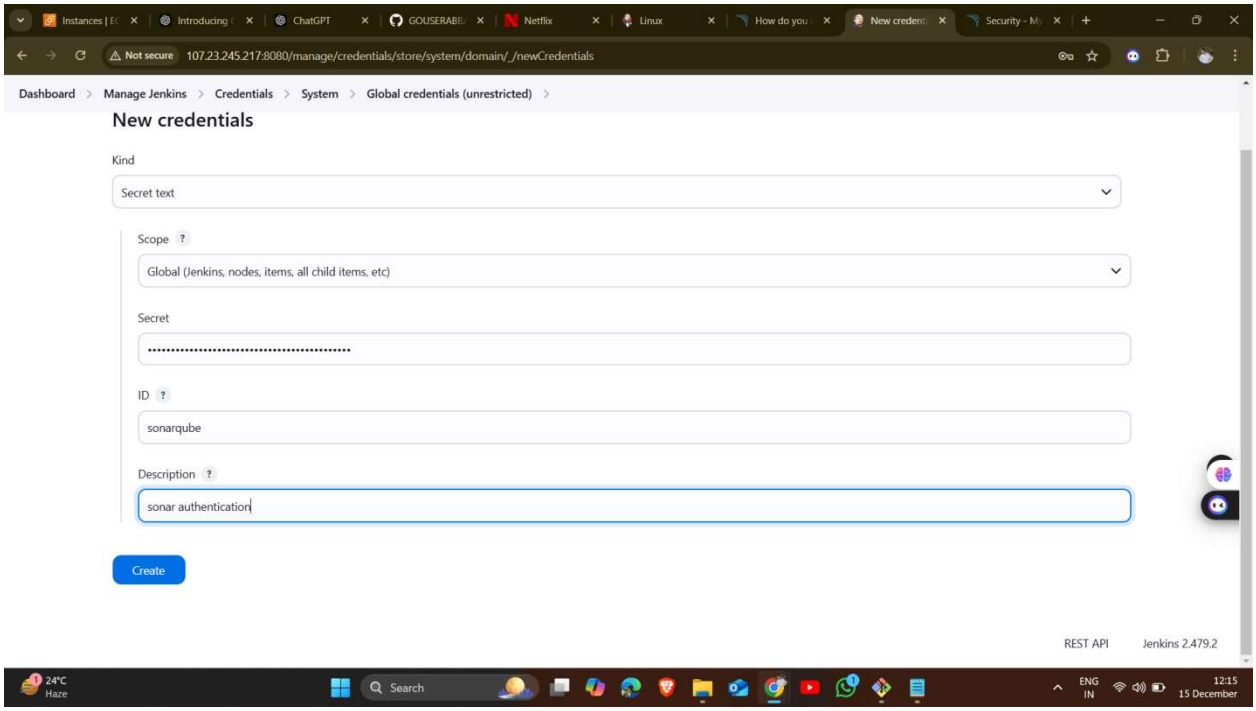


In the security field add credentials for that create a sonarqube token here is

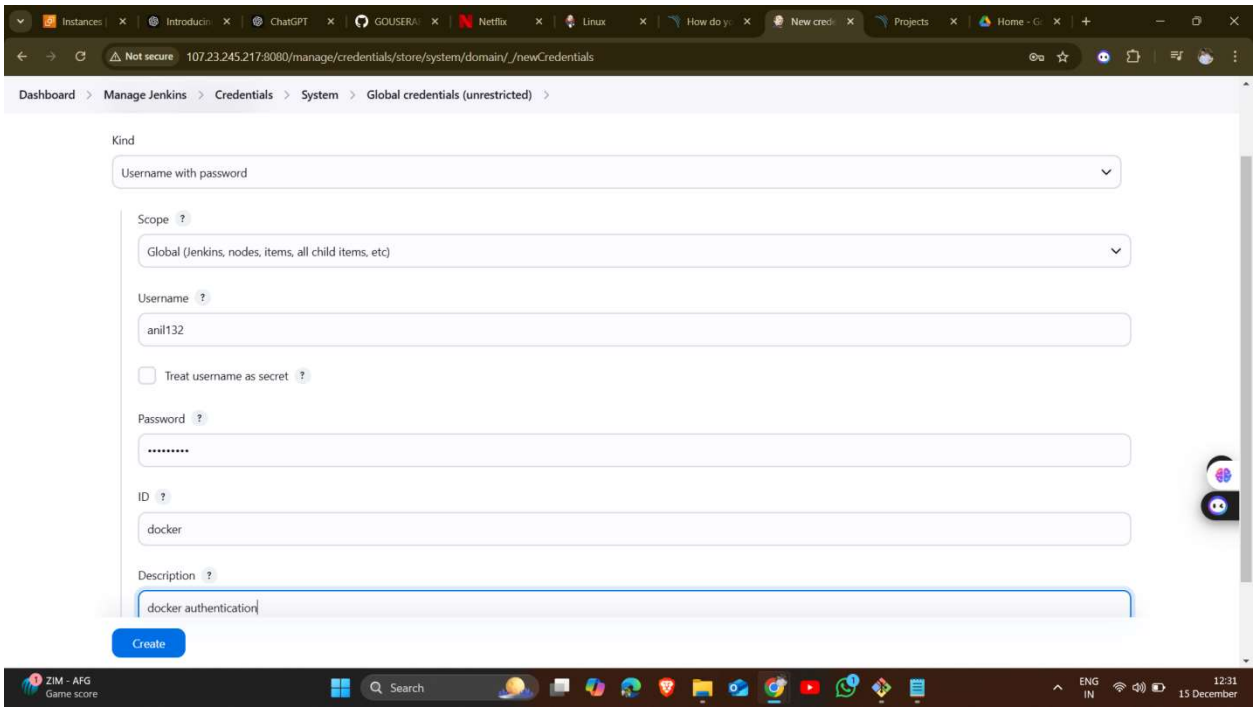




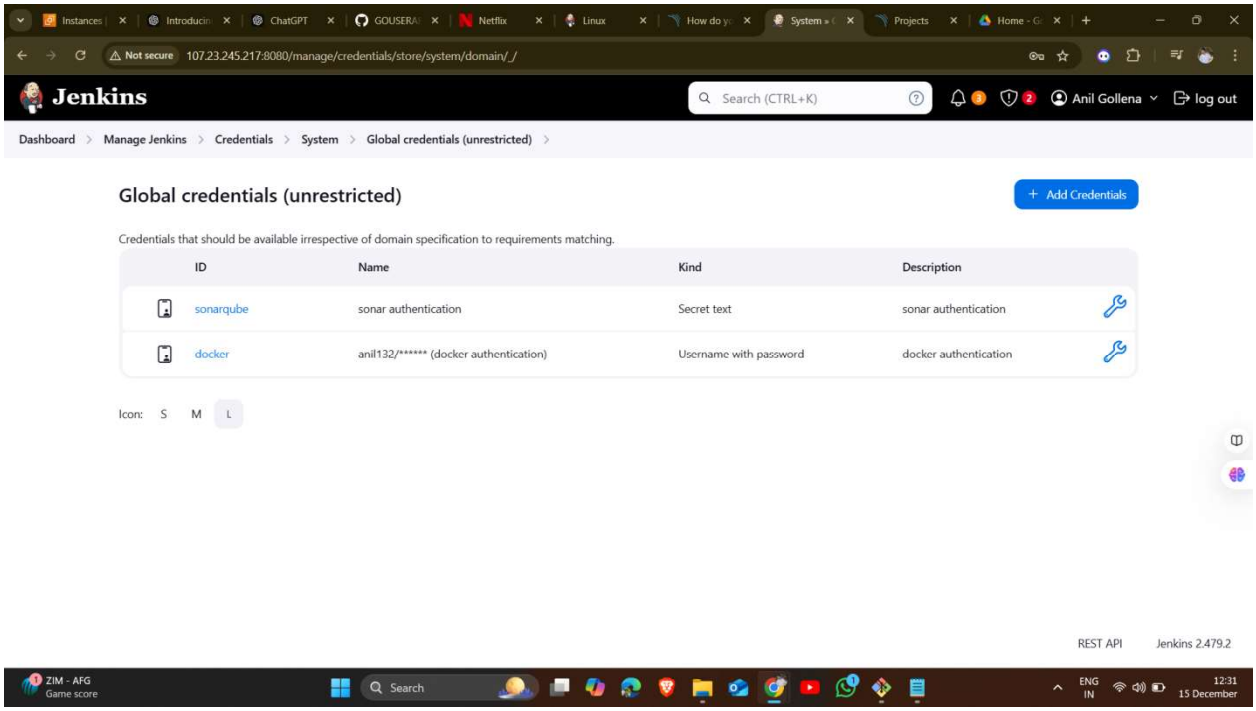
Copy that token and paste it in Jenkins



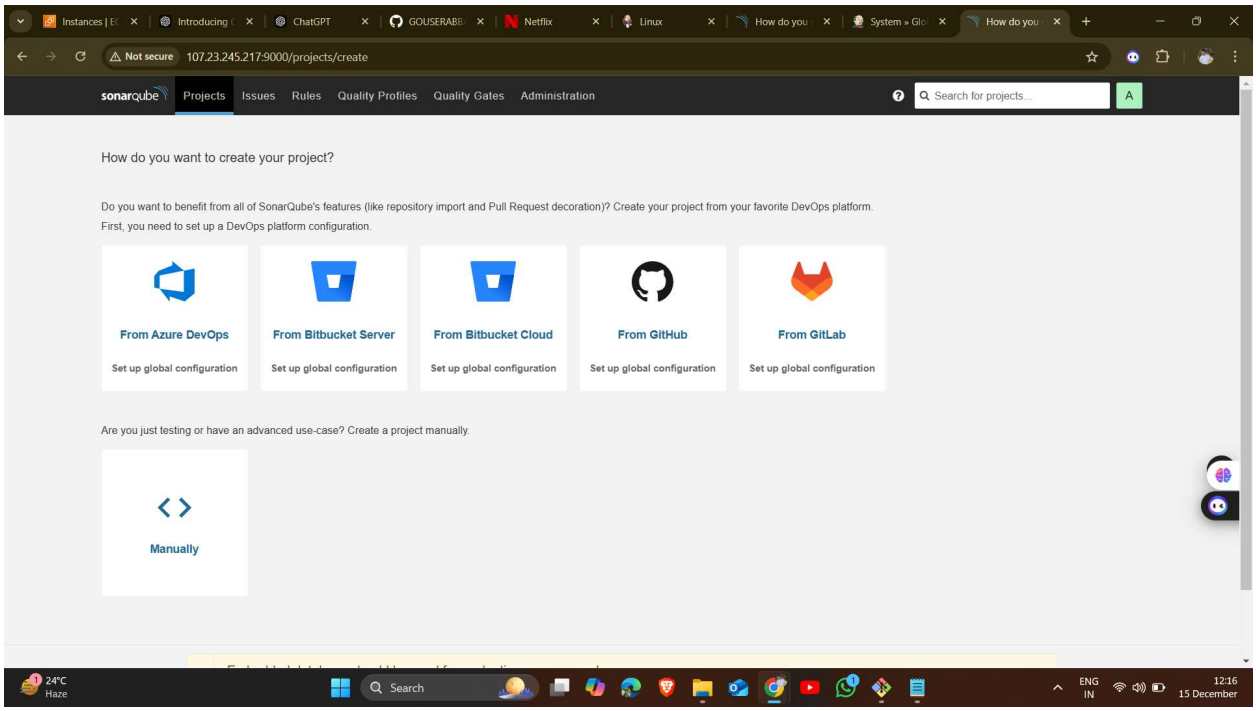
Now add docker credentials using username and password



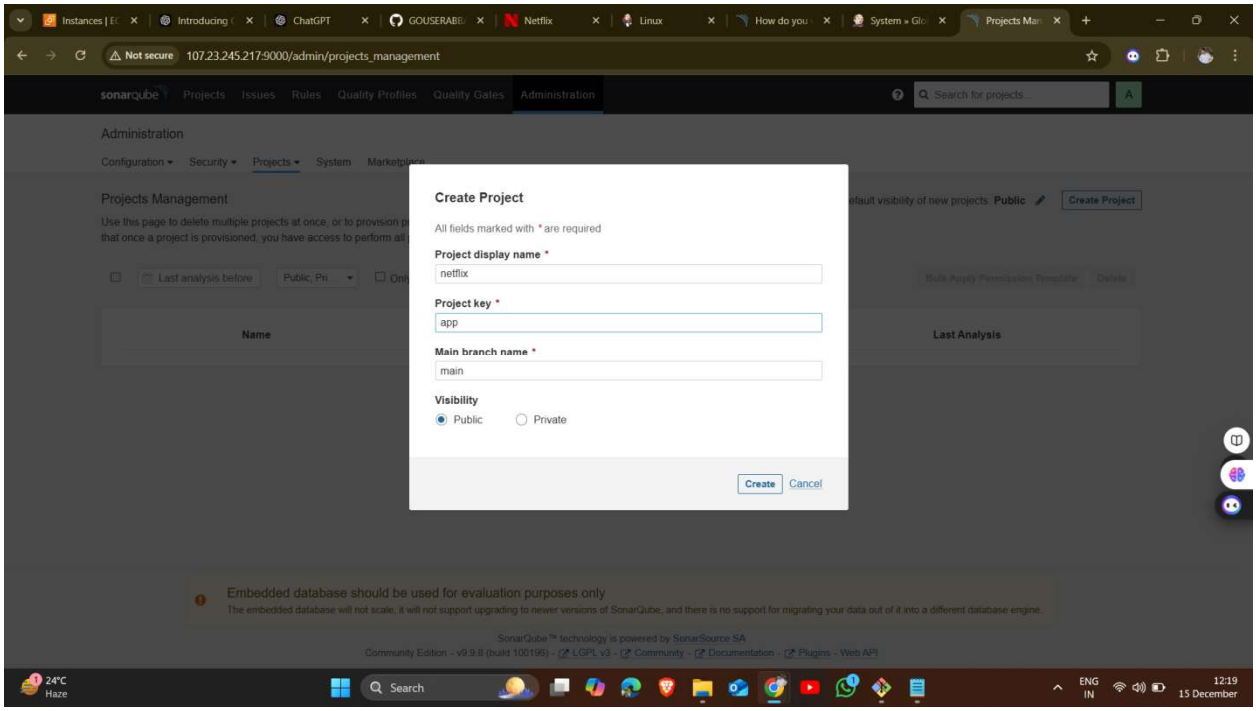
Here the credentials are added.



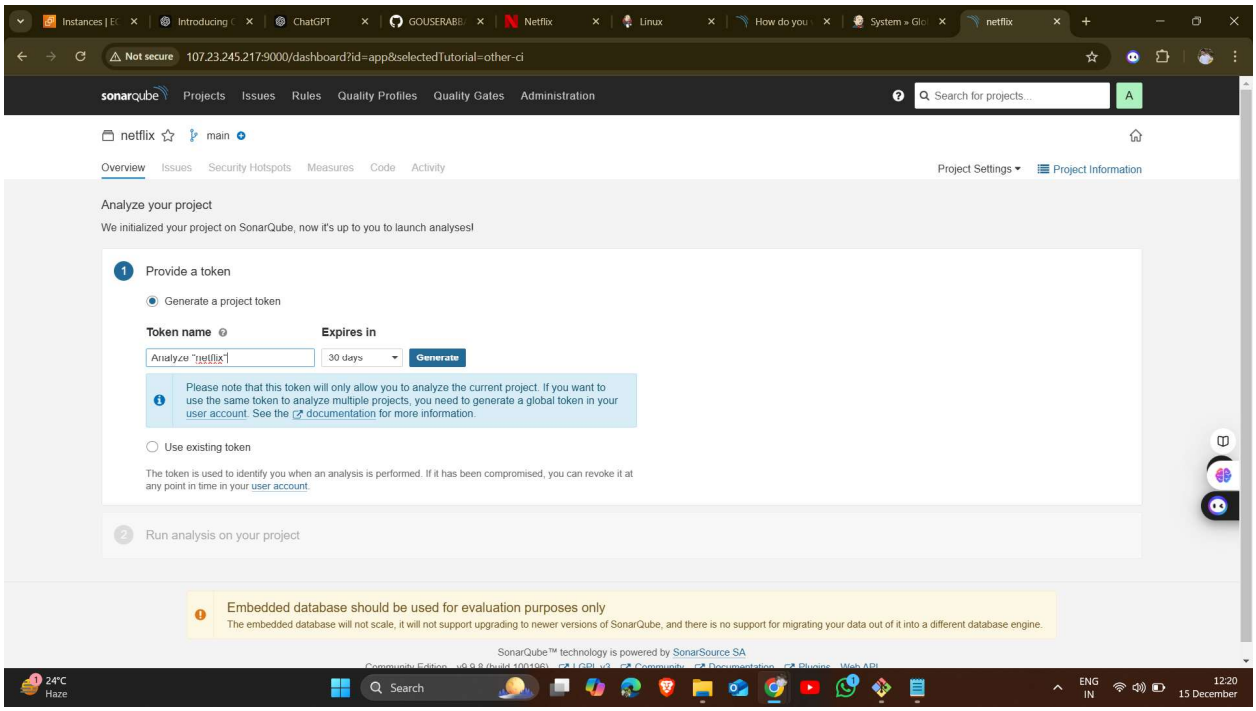
Now add the Netflix project into the sonarqube here is the steps to add.



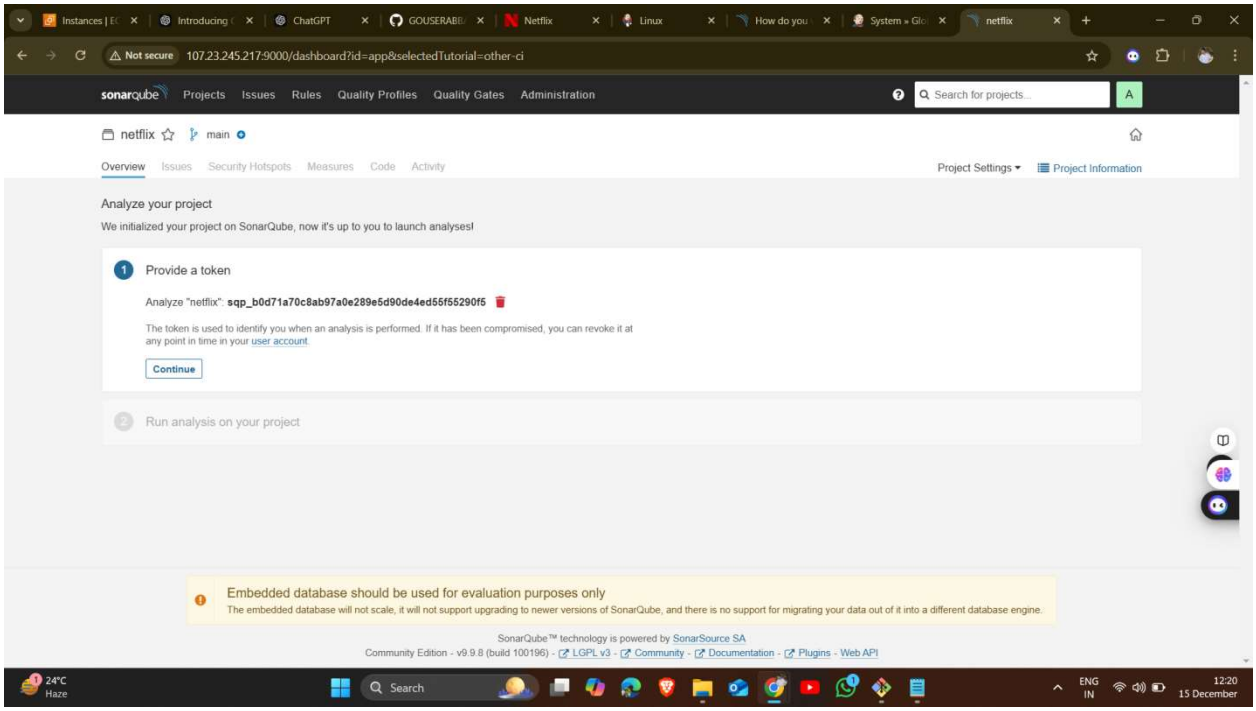
Now under projects→create a new project.



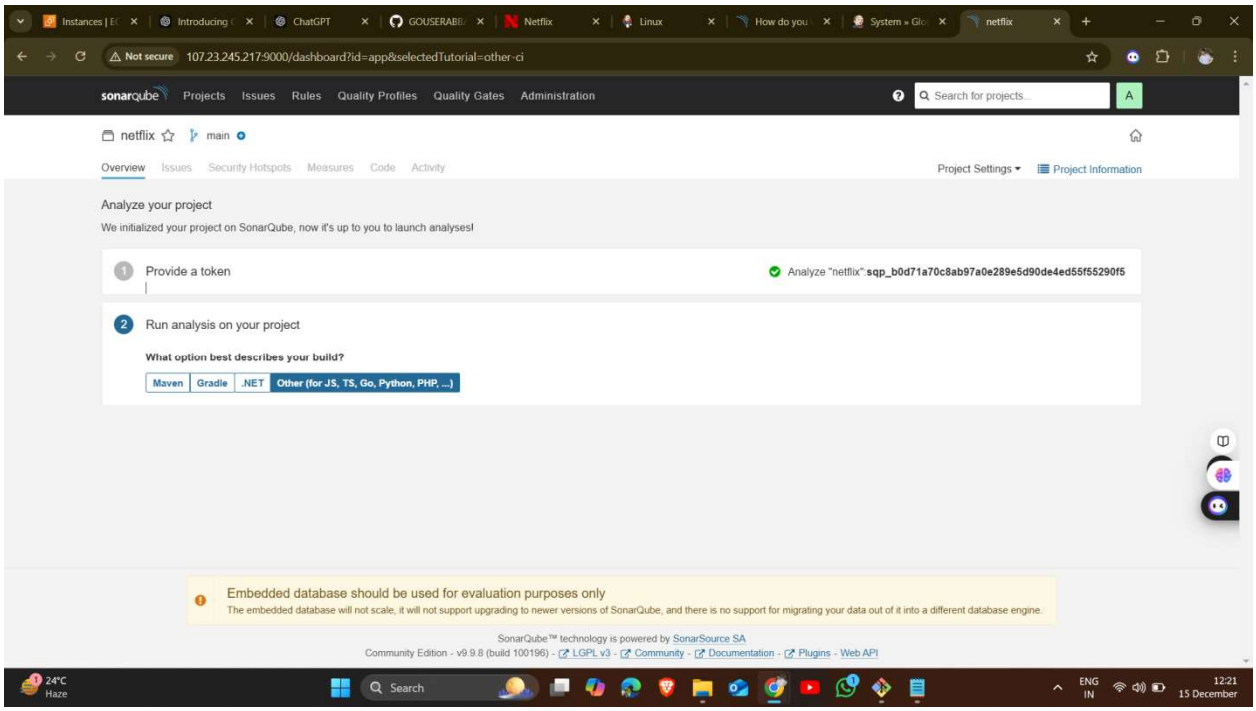
Now generate token



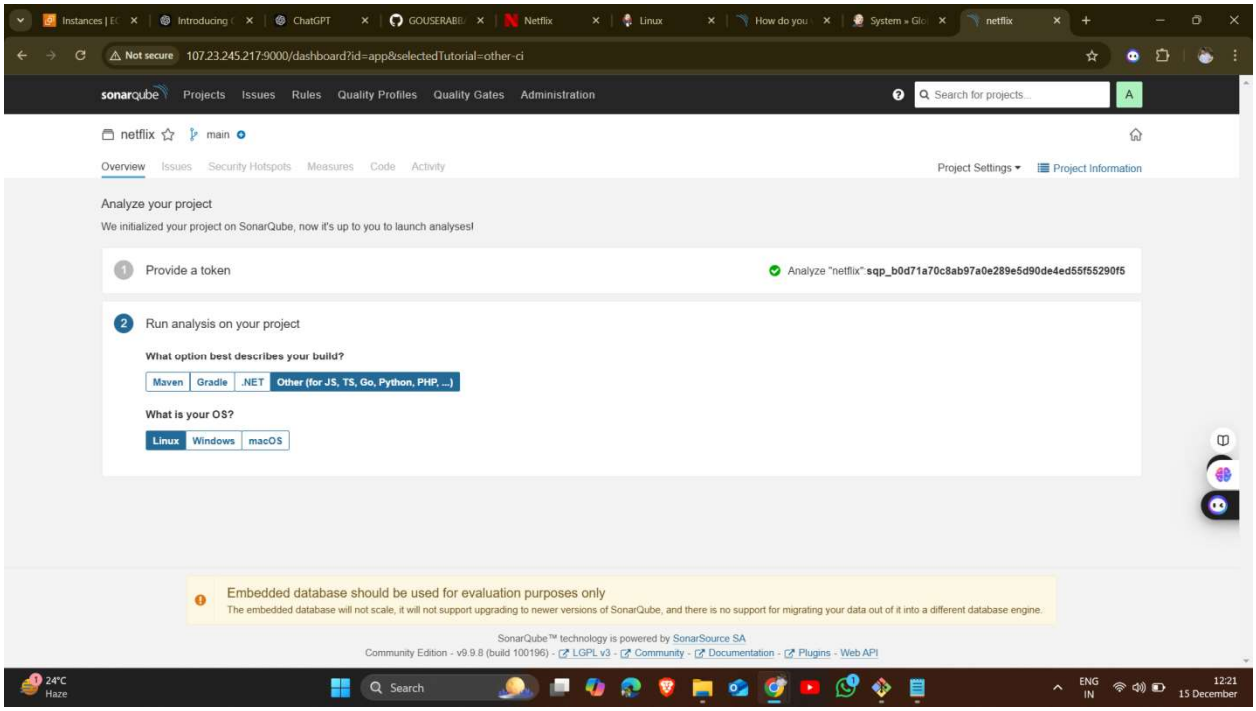
Now generate.



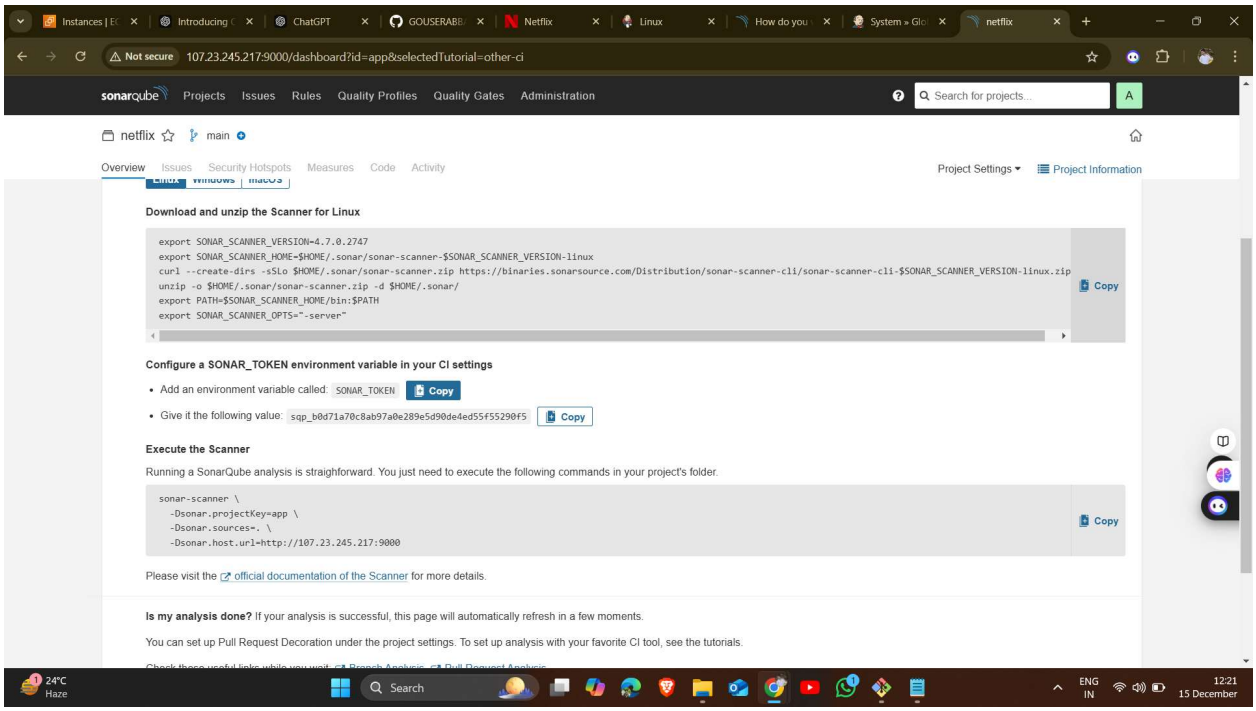
Now press continue



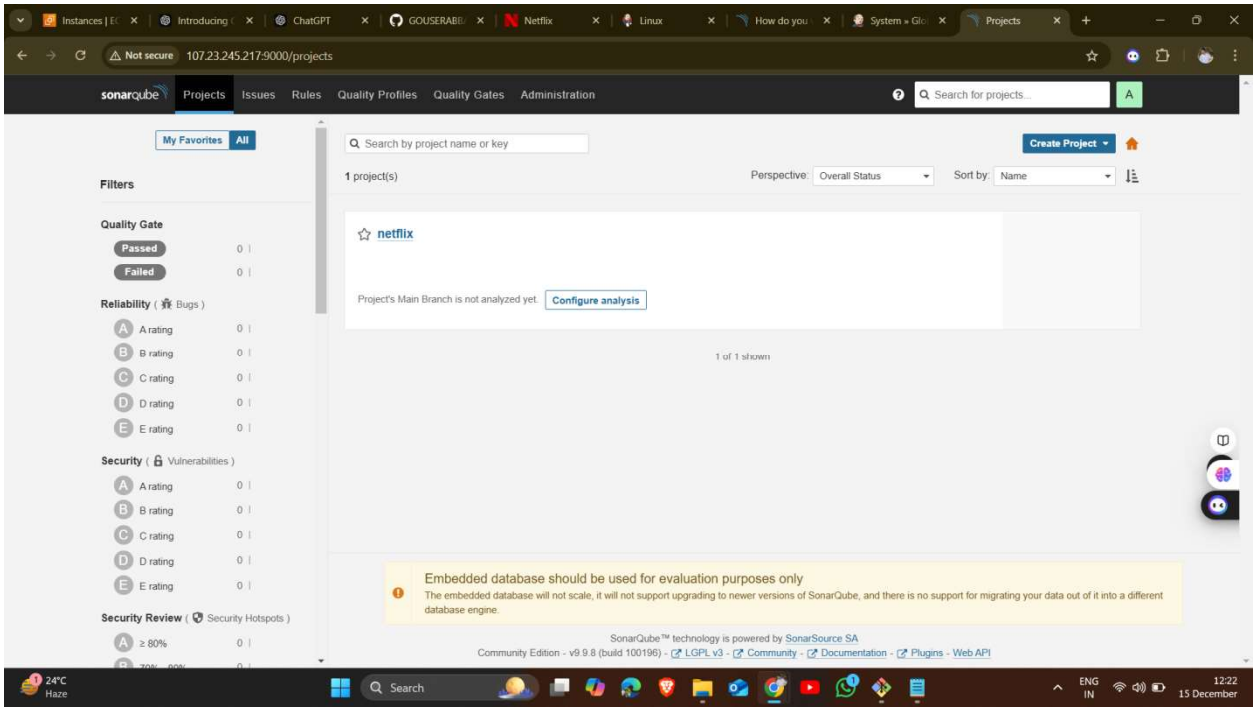
Now click other



After clicking the OS as linux it displays the commands and paste it on the local

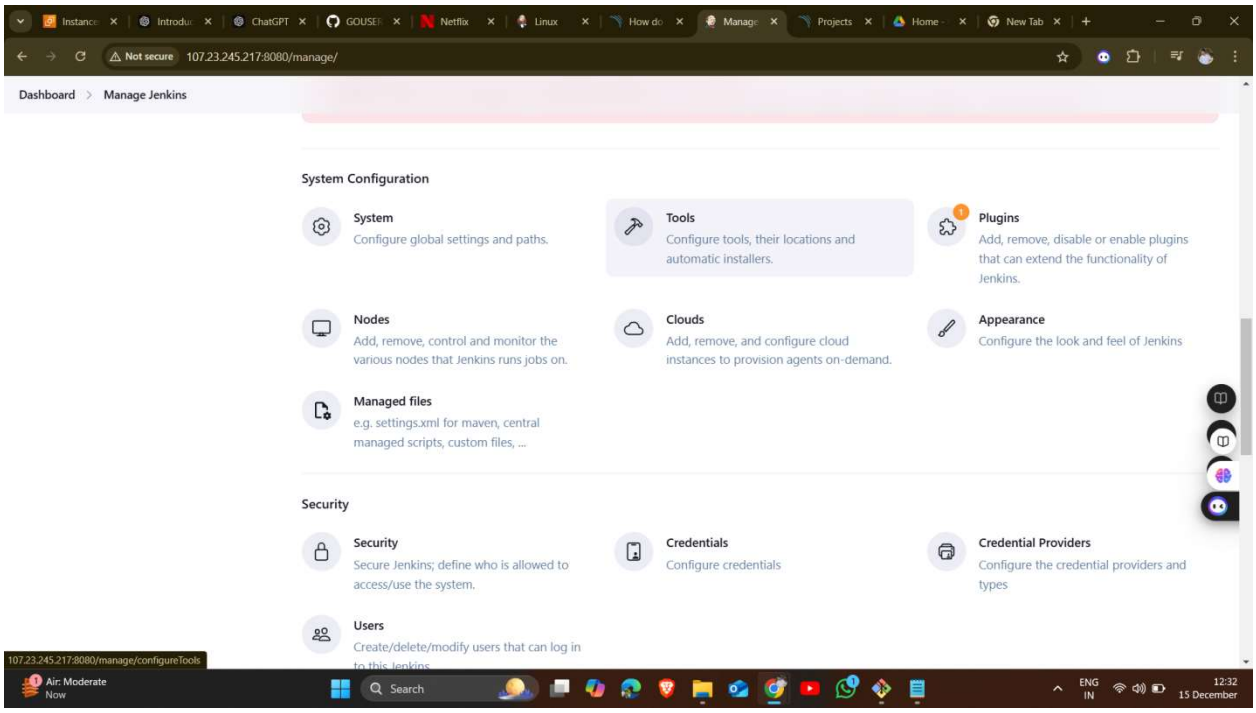


Here the project is added into the sonarqube

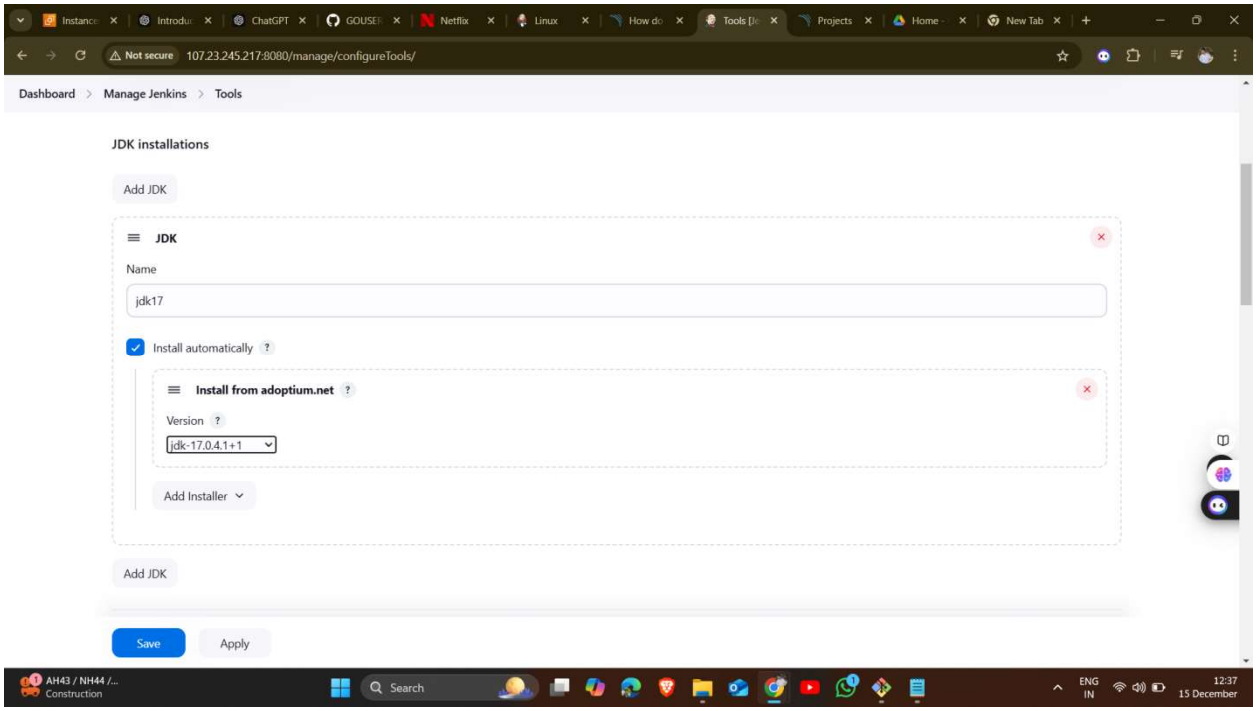


Now go to Jenkins dashboard→manage Jenkins→under tools

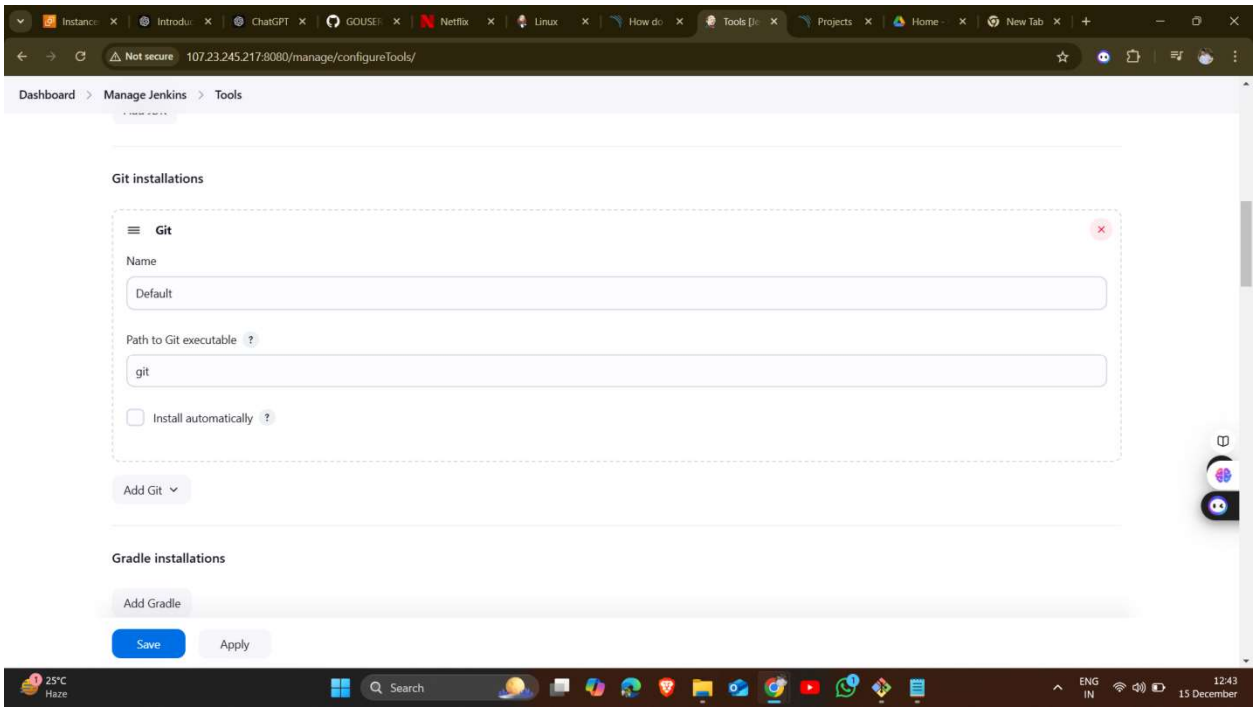
Add configuration



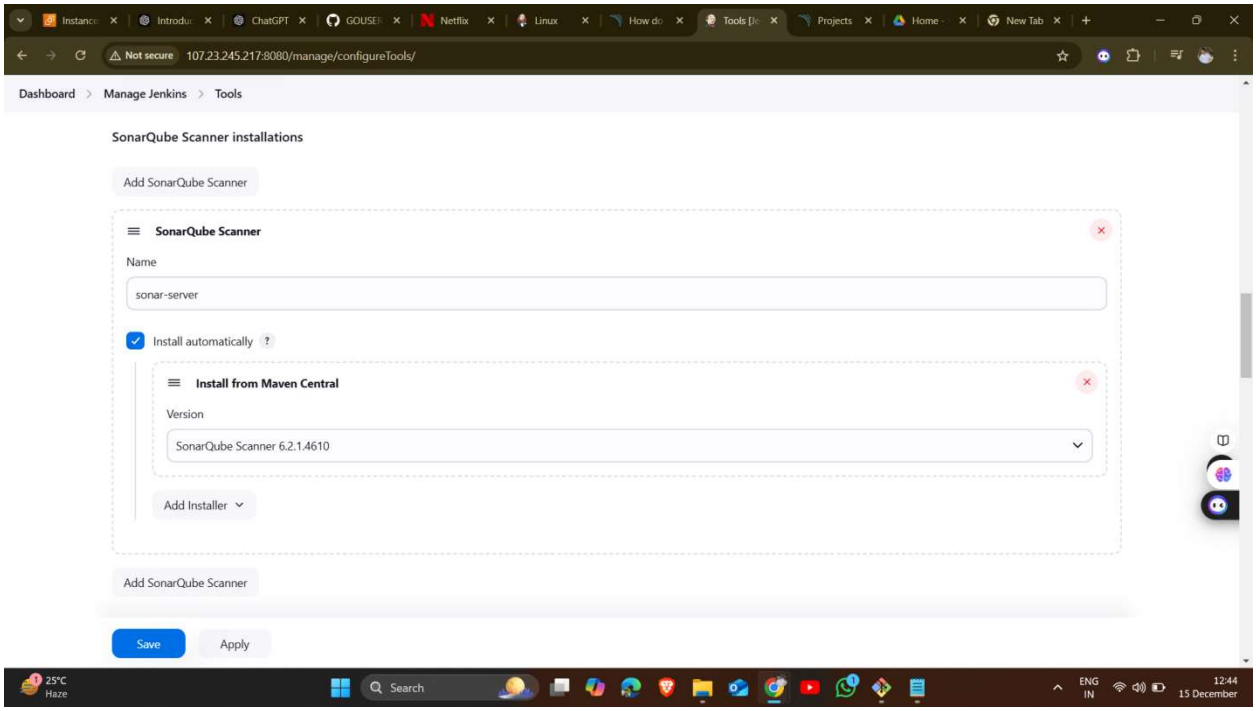
Add JDK



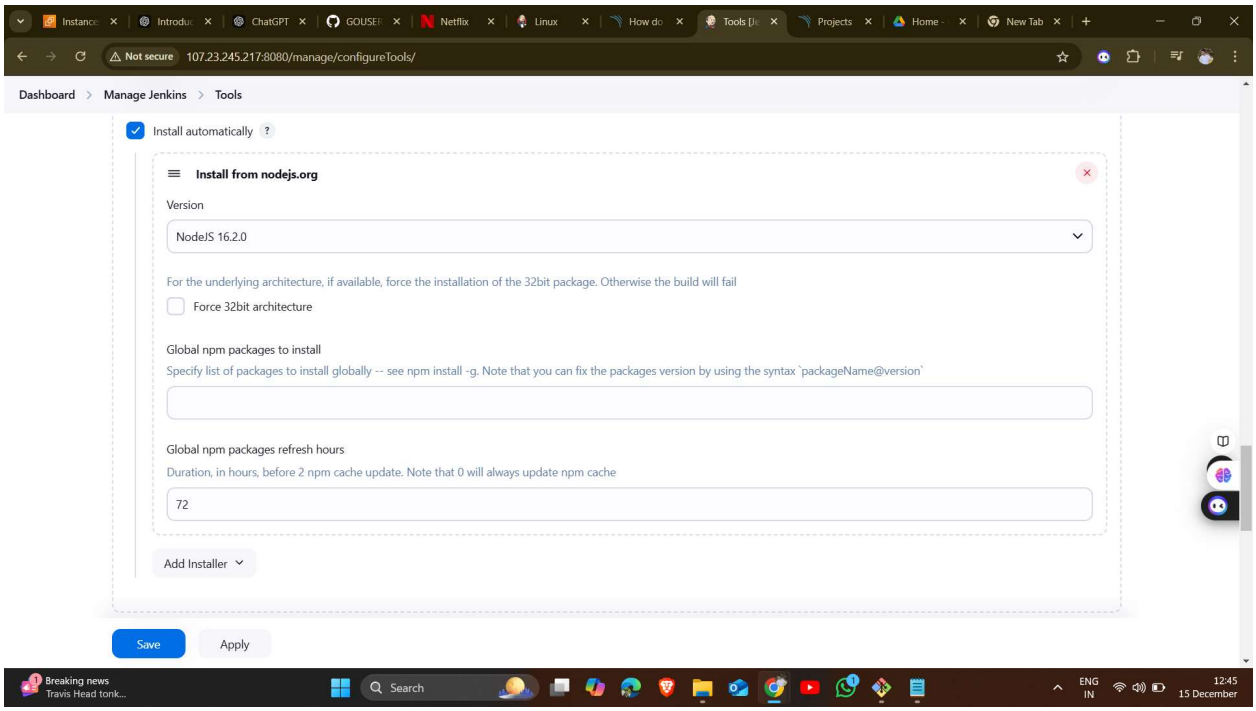
No need to change git installations



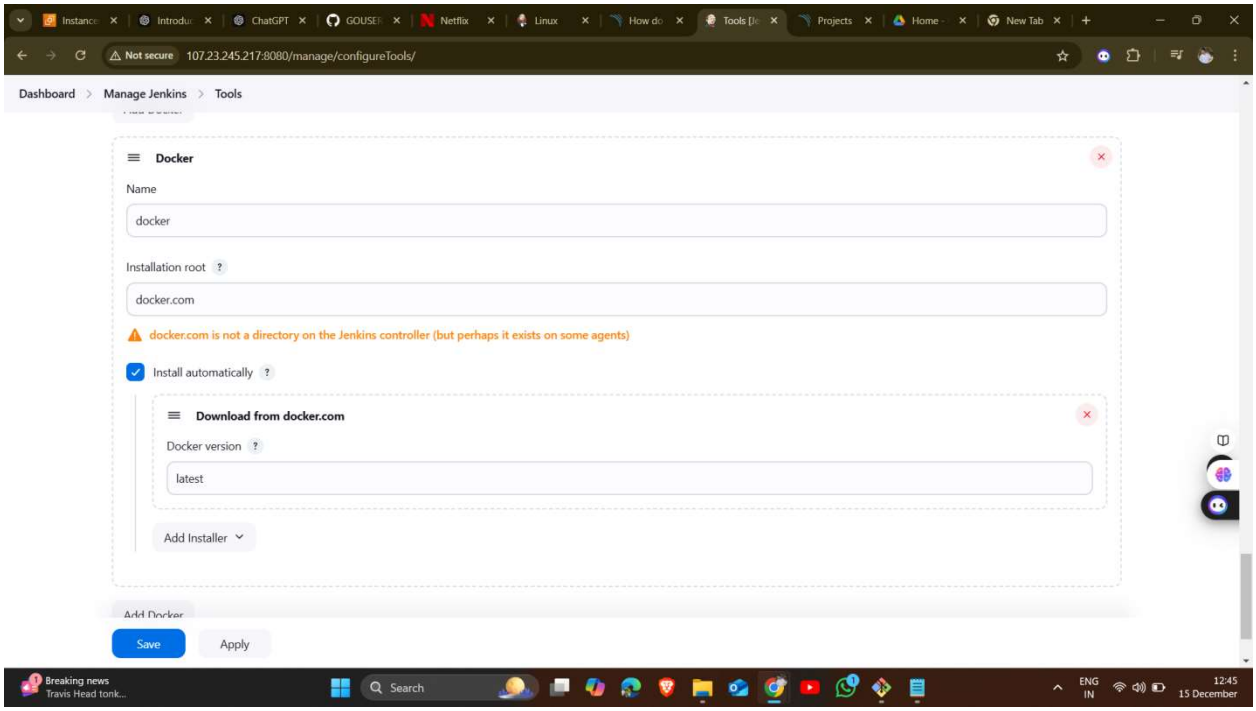
Add sonarqube scanner



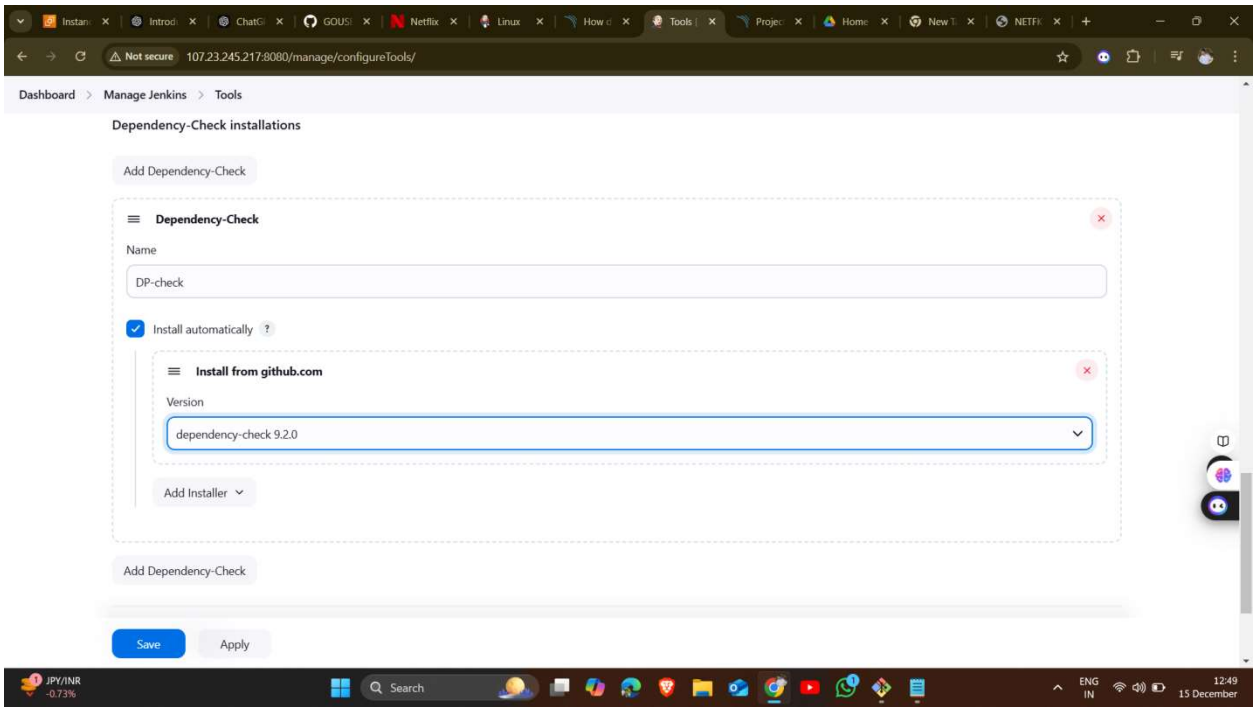
Add NodeJS



Add docker



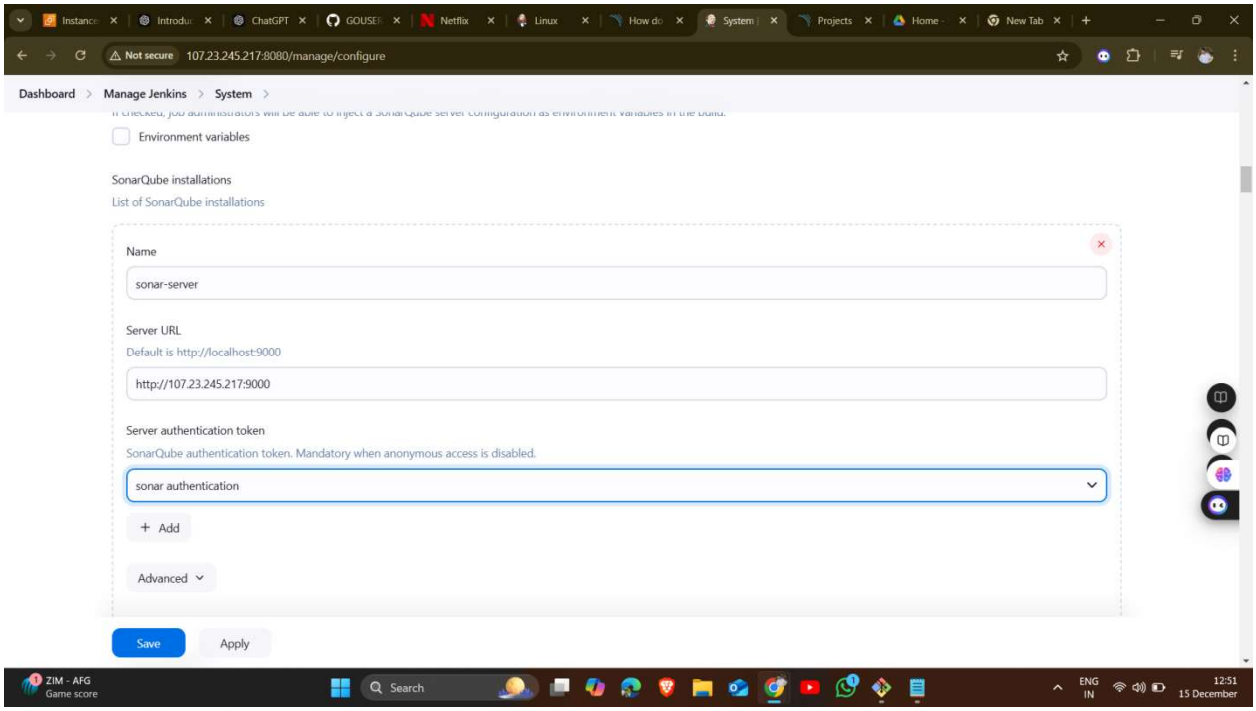
Add OWSP dependency check



Now save and apply

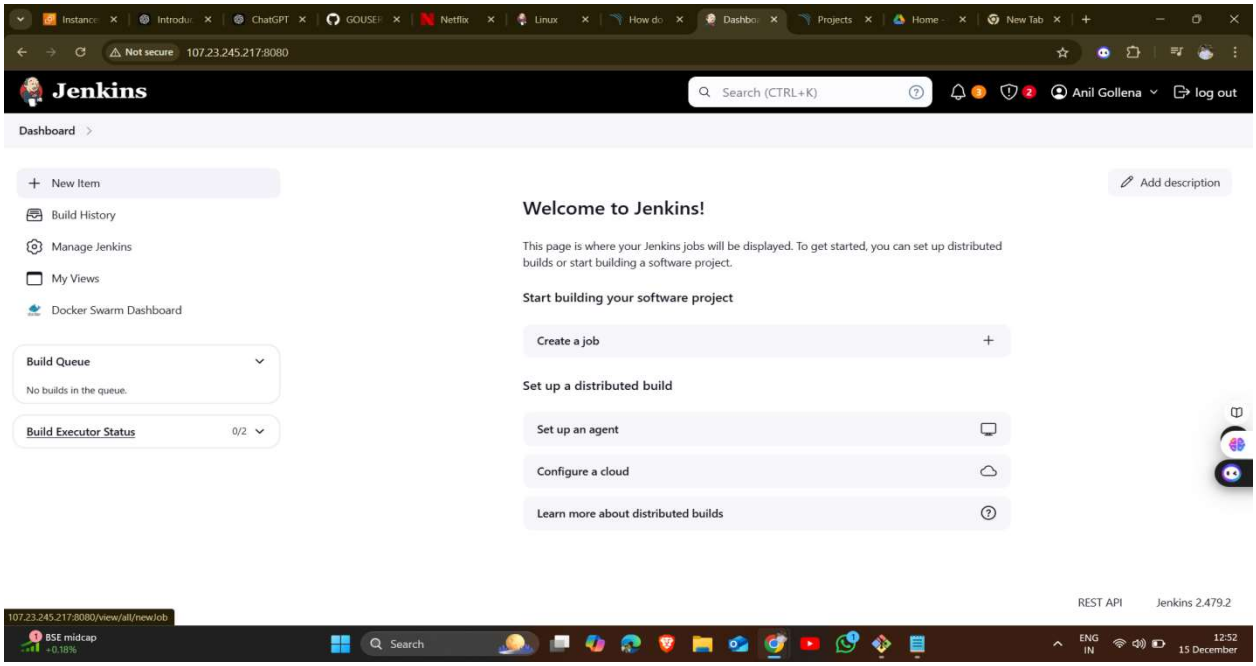
Now go to Jenkins dashboard→manage Jenkins→system

Add sonarqube installations

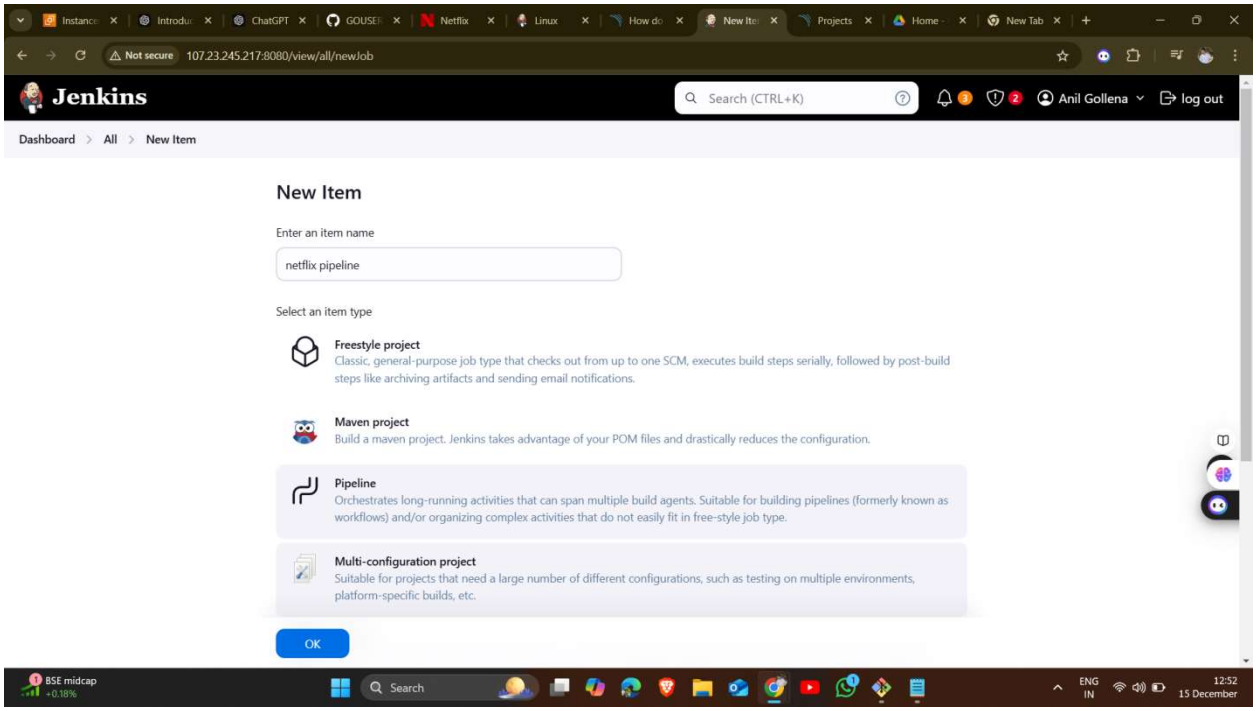


Now save and apply

Now back to dashboard create a new job for pipeline



New item→give pipeline name→select pipeline→OK.



Under pipeline script give script below:

#Groovy script:

```
pipeline {
    agent any
    tools {
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME = tool 'sonar-server'
    }
```

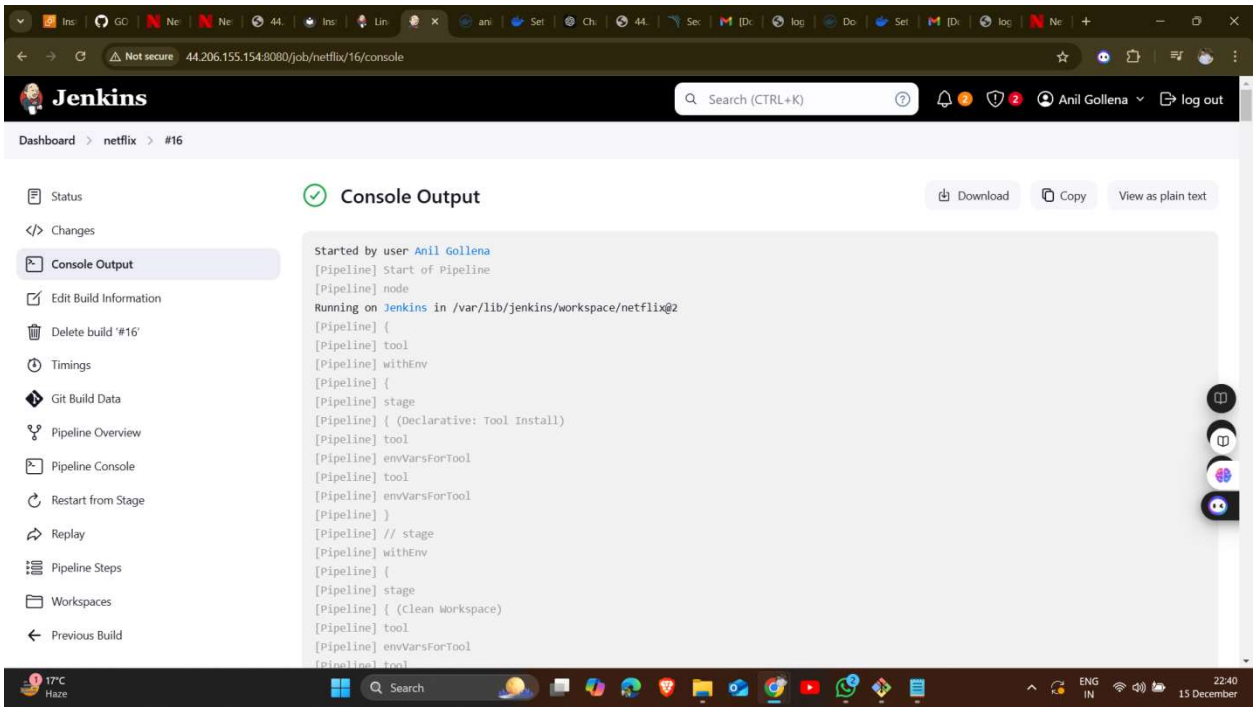
```
    stages {
        stage('Clean Workspace') {
            steps {
                cleanWs()
            }
        }
        stage('Checkout from Git') {
            steps {
                git branch: 'main', url:
'https://github.com/GOUSERABBANI44/Netflix.git'
            }
        }
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('sonar-server') {
                    sh '''
                        $SCANNER_HOME/bin/sonar-scanner \
                        -Dsonar.projectName=Netflix \
                        -Dsonar.projectKey=app
                    '''
                }
            }
```

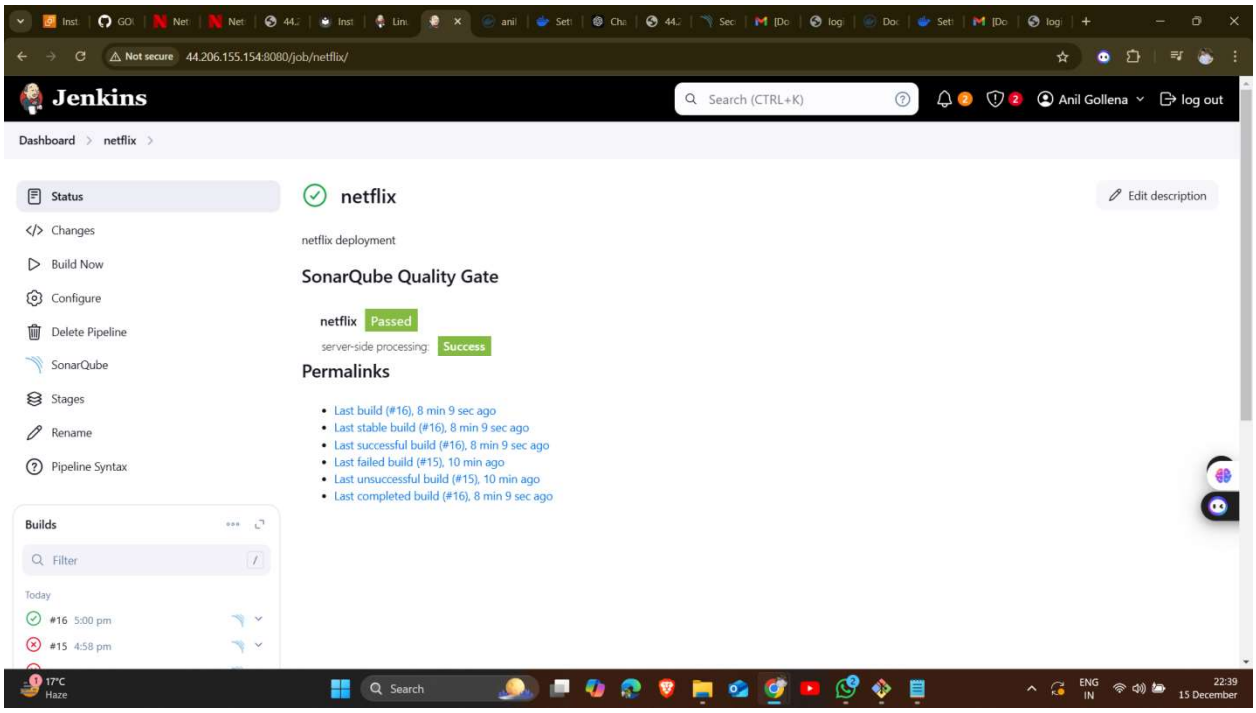
```
        }
        stage('Quality Gate') {
            steps {
                script {
                    // Wait for the quality gate to pass before proceeding
                    waitForQualityGate abortPipeline: false, credentialsId: 'sonarkey'
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh 'npm install'
            }
        }
        stage('OWASP Dependency Check') {
            steps {
                // Make sure that the Dependency-Check tool is configured correctly in Jenkins
                dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'DP-check'
            }
        }
```

```
        stage('TRIVY File System Scan') {
            steps {
                // Ensure Trivy is installed and available on the agent
                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=72cd1d5c019e34e3fd990b09f6788b25 -t netflix .
                            docker tag netflix anil132/netflix:latest
                            docker push anil132/netflix:latest
                        '''
                    }
                }
            }
        }
        stage('TRIVY Image Scan') {
            steps {
```

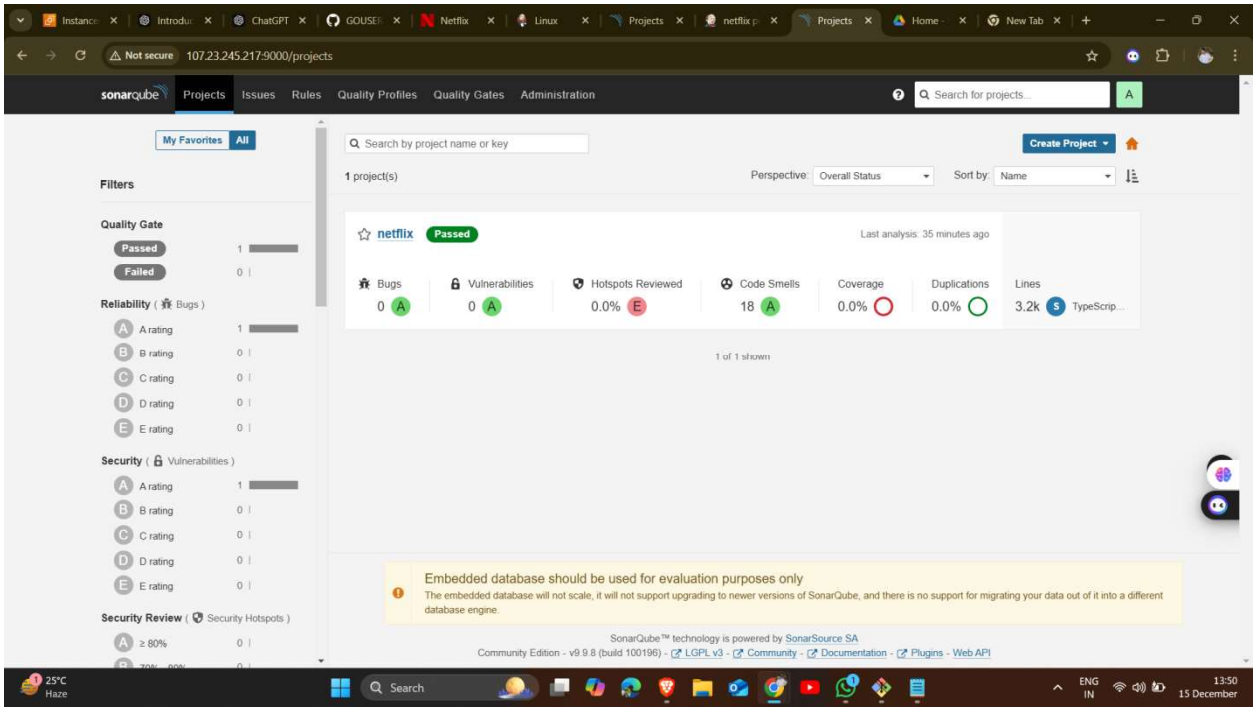
```
            sh 'trivy image anil132/netflix:latest > trivyimage.txt'
        }
    }
    stage('Deploy to Container') {
        steps {
            sh 'docker run -d --name netflixcont1 -p 8082:80 anil132/netflix:latest'
        }
    }
  }
}
```

Now save and apply the job.

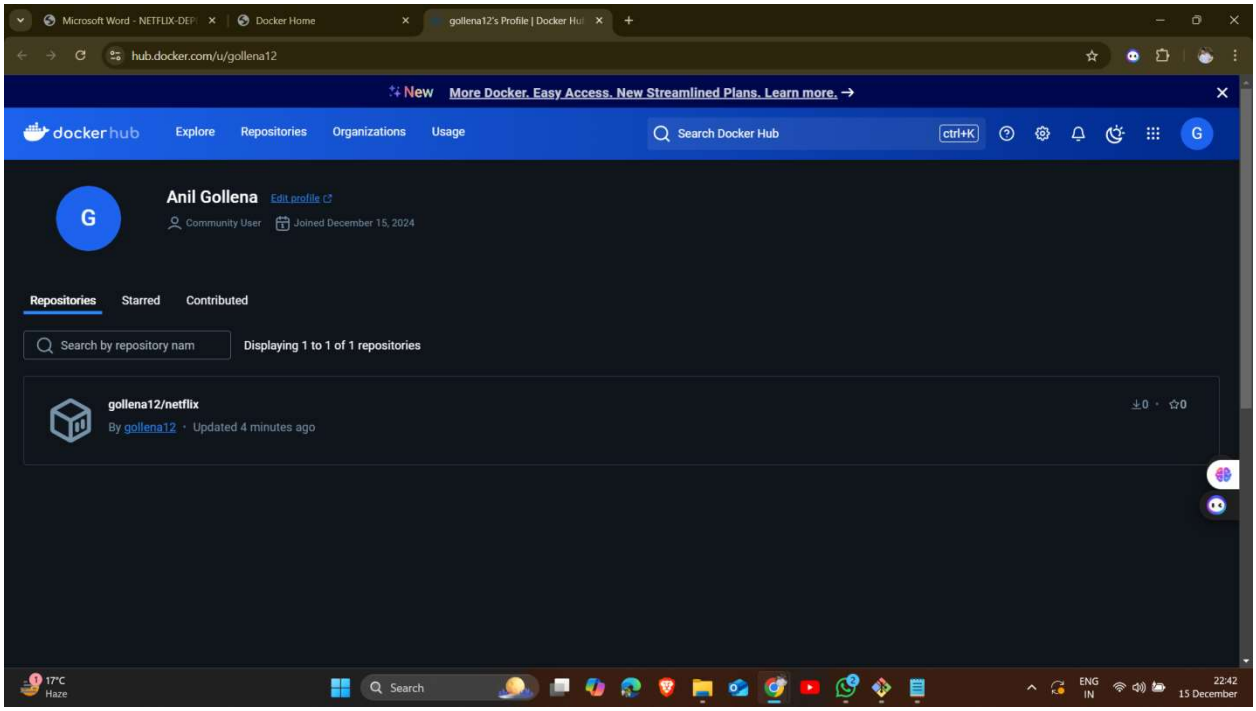


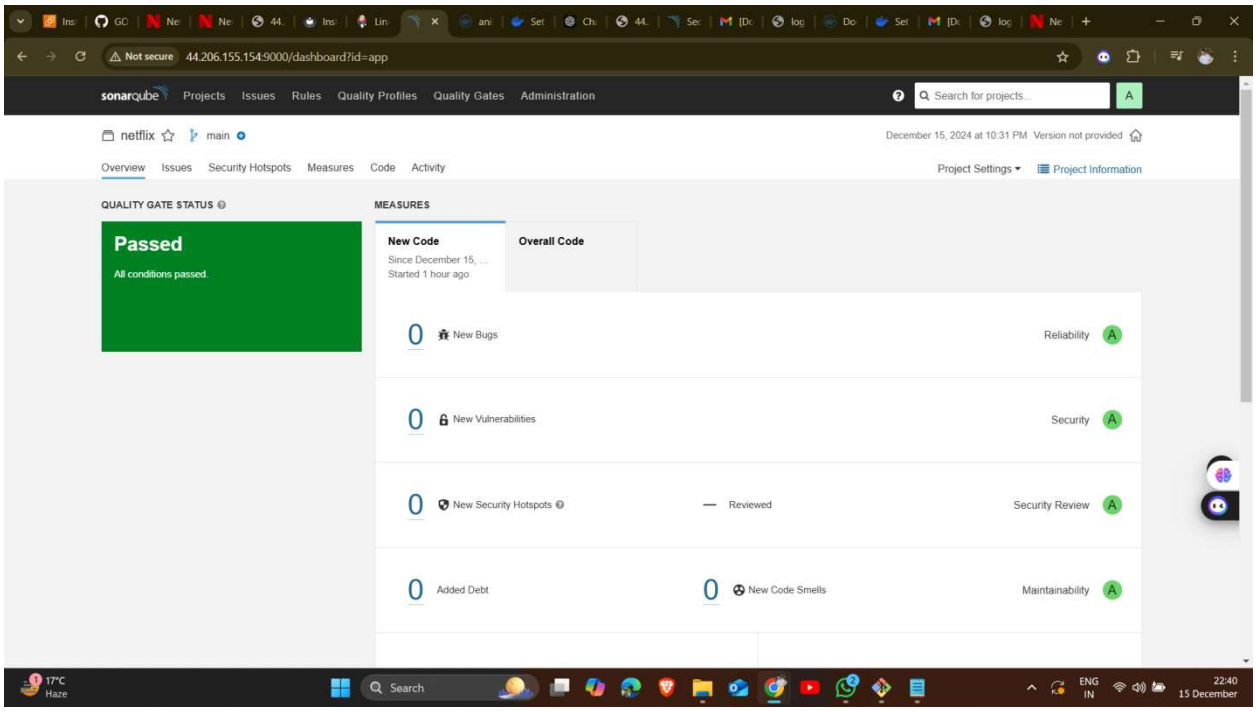


Now check the sonarqube, there you can see the output that all test case is passed.

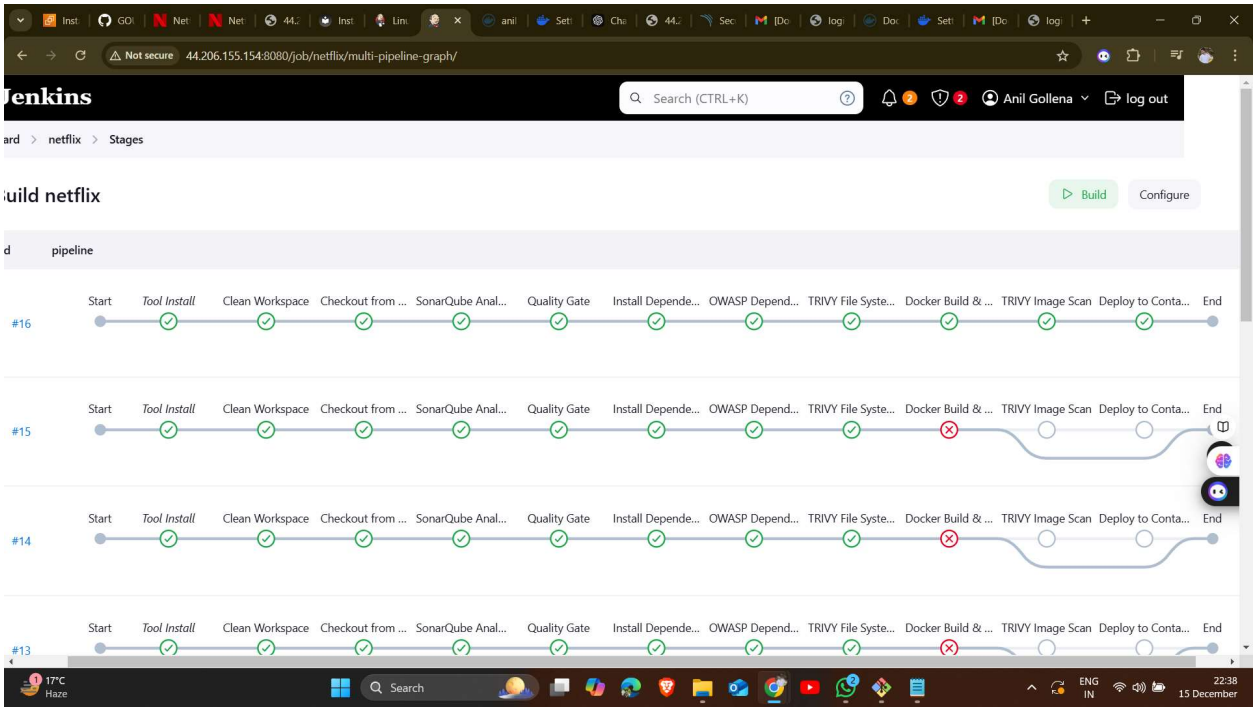


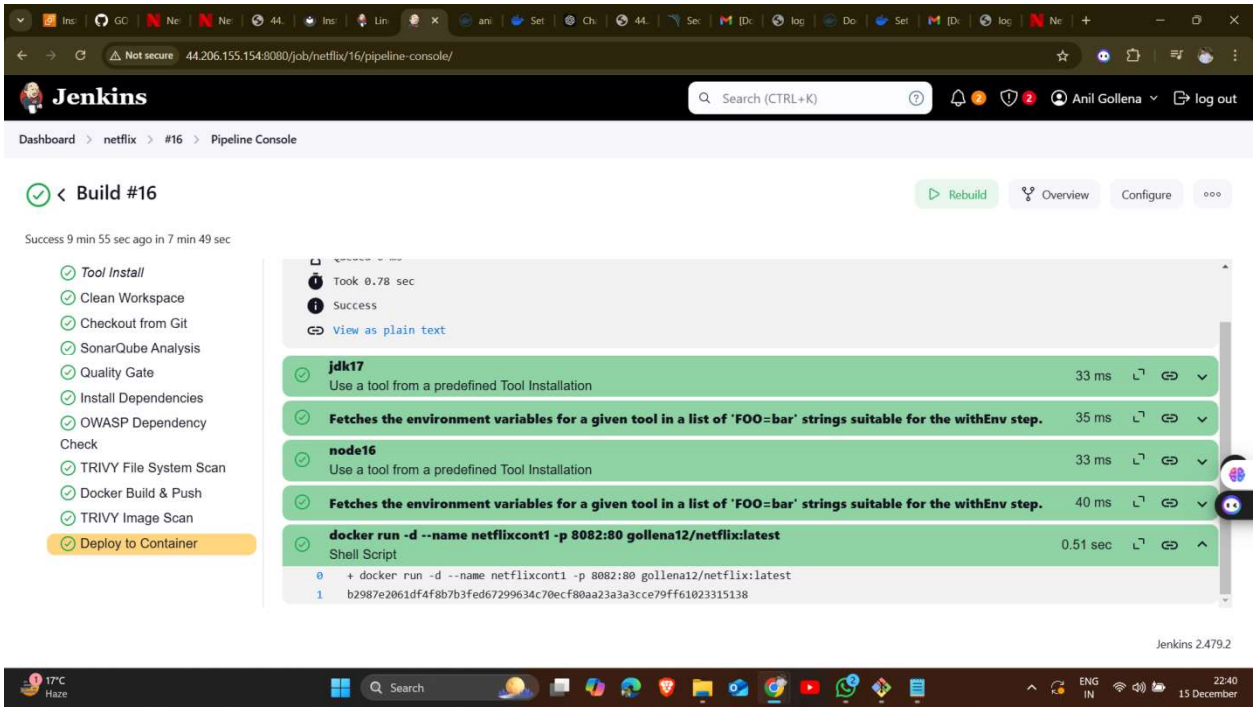
As well as, the image which has created in uploaded in docker hub.



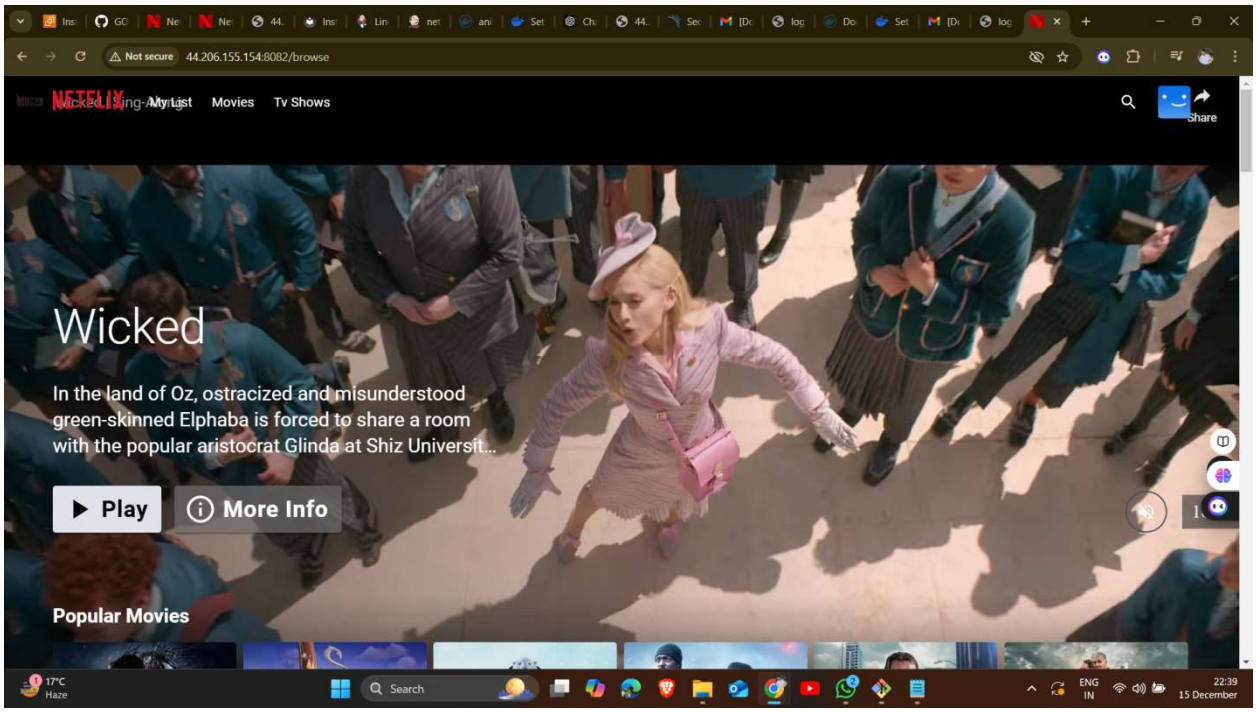


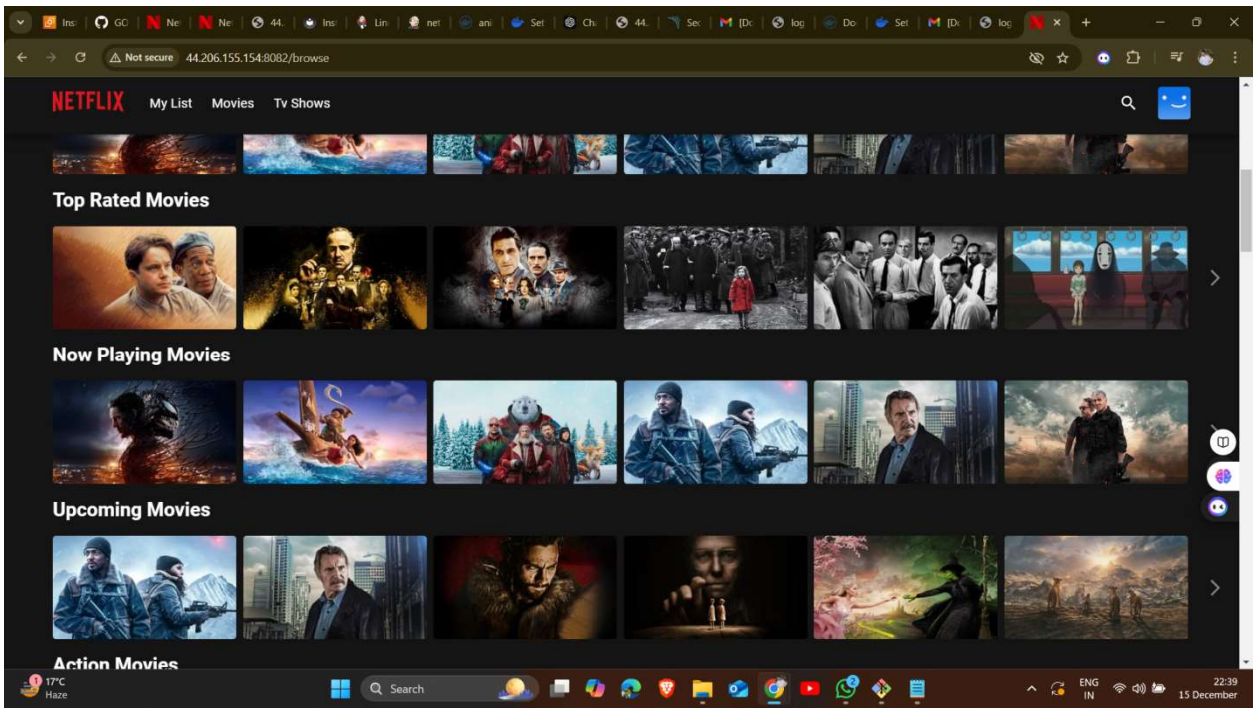
Here the stages that passed





Now copy the public of instance, now you can see that Netflix application is deployed.





PHASE 4: MONITERING THROUGH PROMETHEUS AND GRAFHANA

About Prometheus:

- **Prometheus** is an open-source monitoring and alerting toolkit designed for **metrics-based monitoring**.
- It collects, stores, and queries time-series data using a powerful **query language (PromQL)**.
- Key features include **multi-dimensional data collection**, **alerting**, and integrations with various systems.
- Commonly used for monitoring application performance, infrastructure, and services in cloud-native environments.

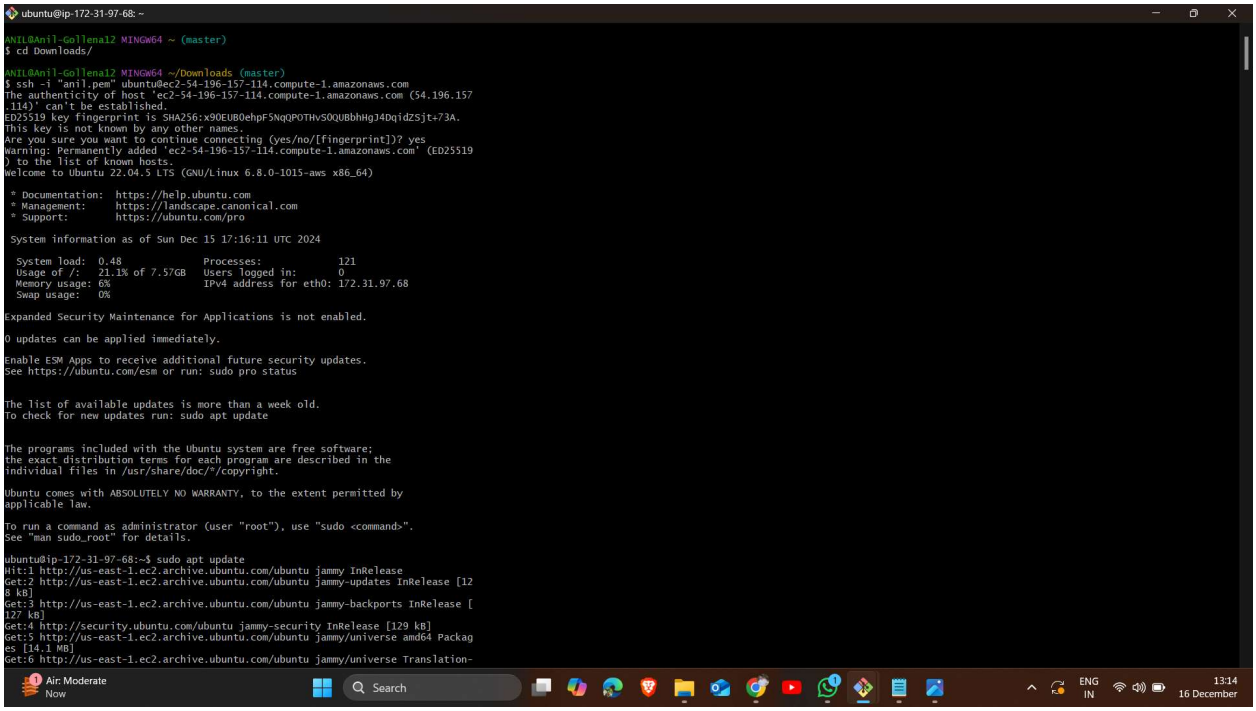
About Grafana:

- **Grafana** is an open-source visualization and analytics platform that allows users to create interactive **dashboards** from time-series data.
- It supports multiple data sources, including Prometheus, InfluxDB, and Elasticsearch.
- Key features include customizable visualizations like graphs, heatmaps, and tables, with real-time monitoring capabilities.
- Grafana makes complex metrics easier to interpret through intuitive dashboards.

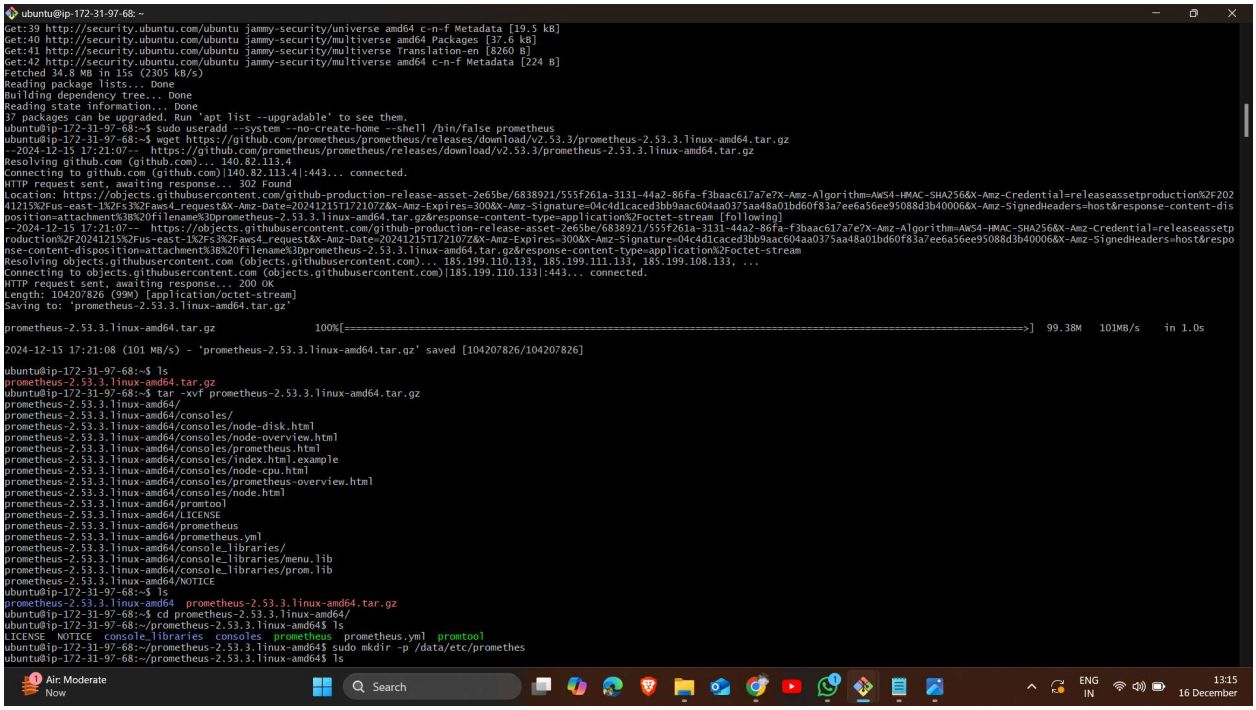
Prometheus + Grafana Integration:

1. **Data Collection**: Prometheus scrapes metrics from monitored targets and stores them in its time-series database.
2. **Visualization**: Grafana uses Prometheus as a data source to build interactive dashboards and visualize metrics.
3. **Alerting**: Prometheus sends alerts based on defined conditions, and Grafana can display alert notifications on dashboards.

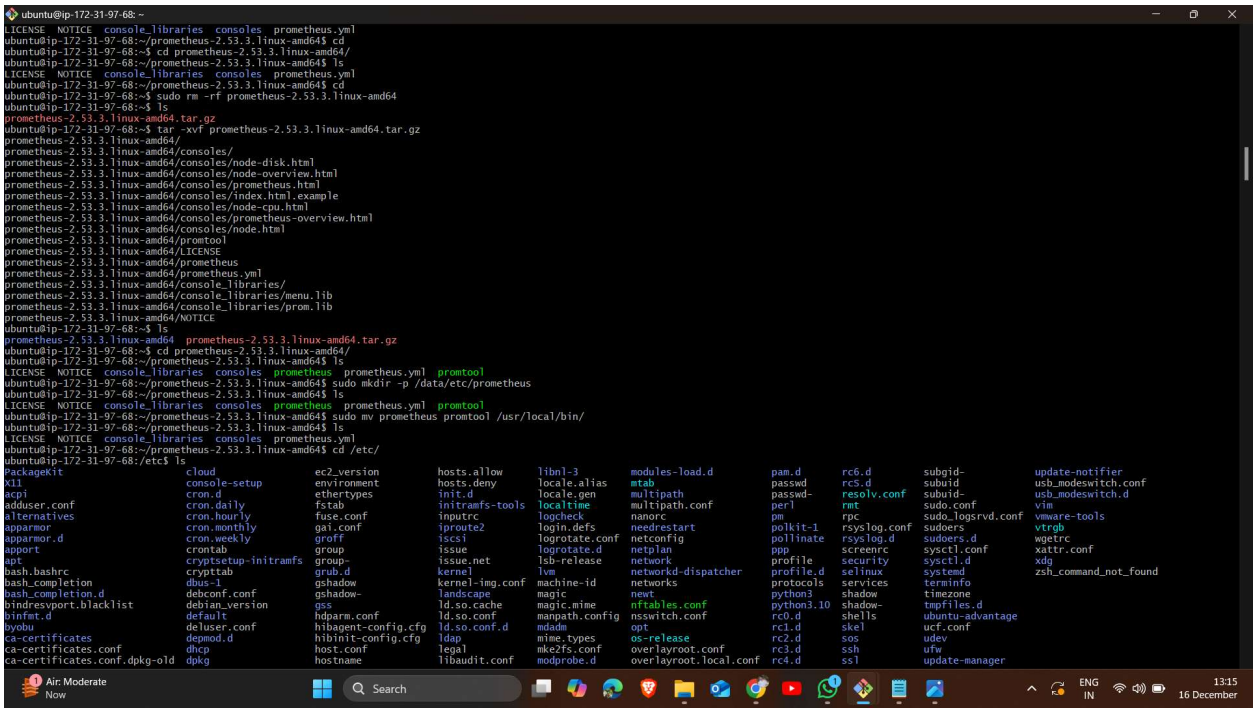
Now create a new instance using aws console and connect it with the server



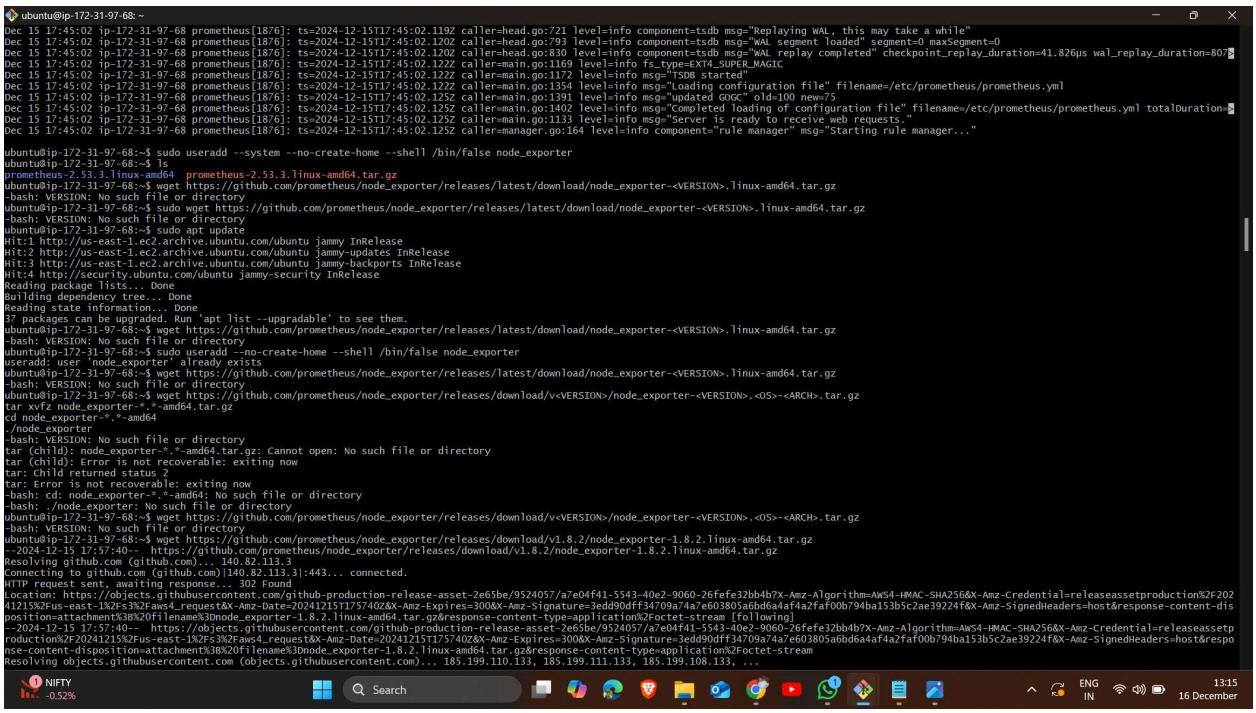
Now install Prometheus by using Prometheus official page copy the displayed commands



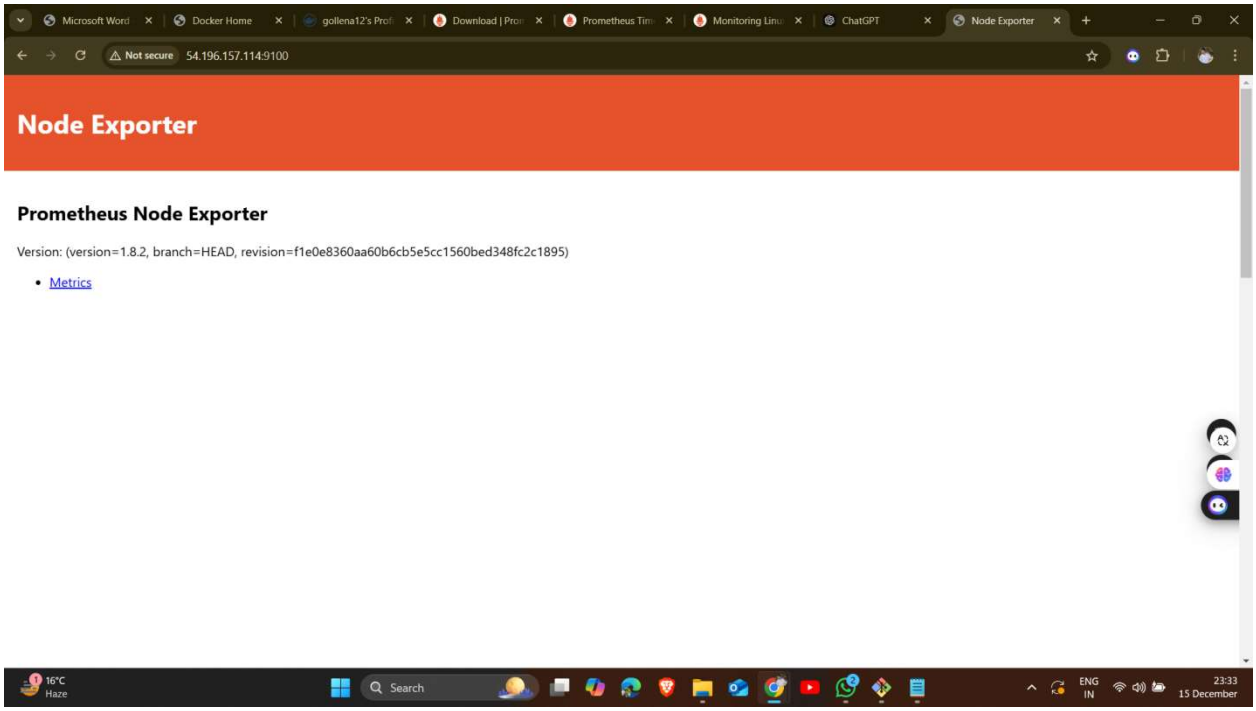
Now unzip the file



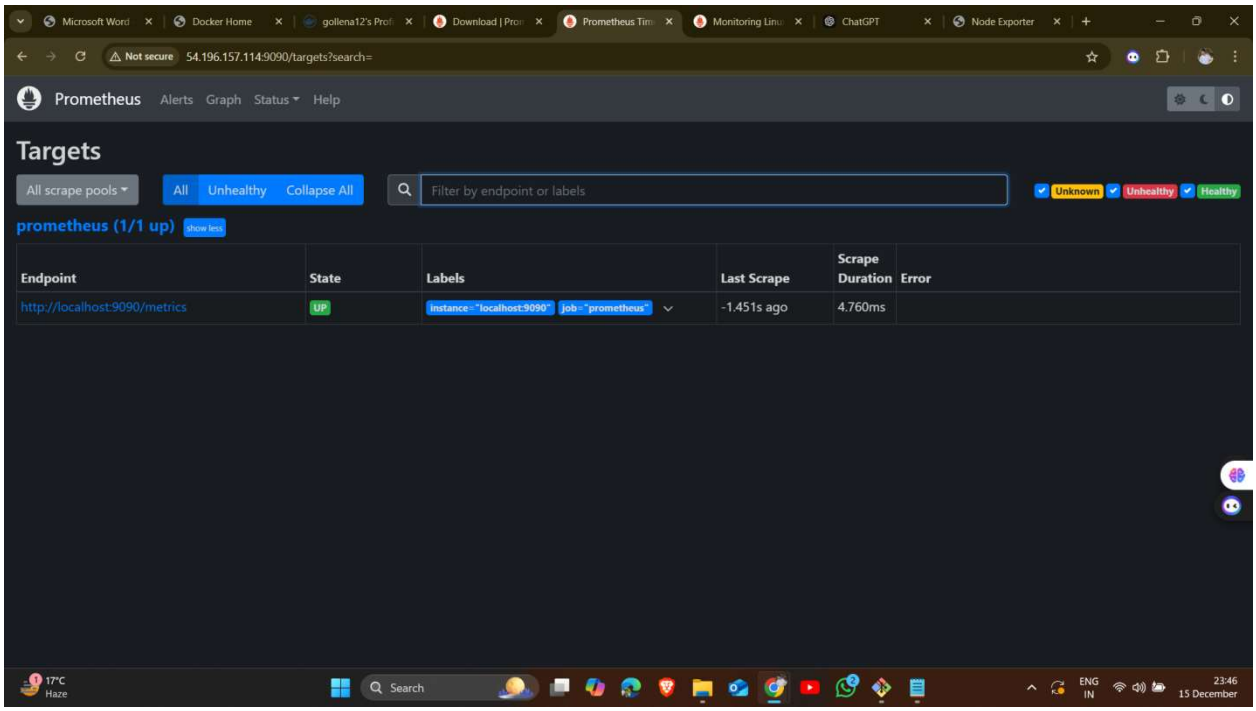
Now install node exporter from google copy the web link address



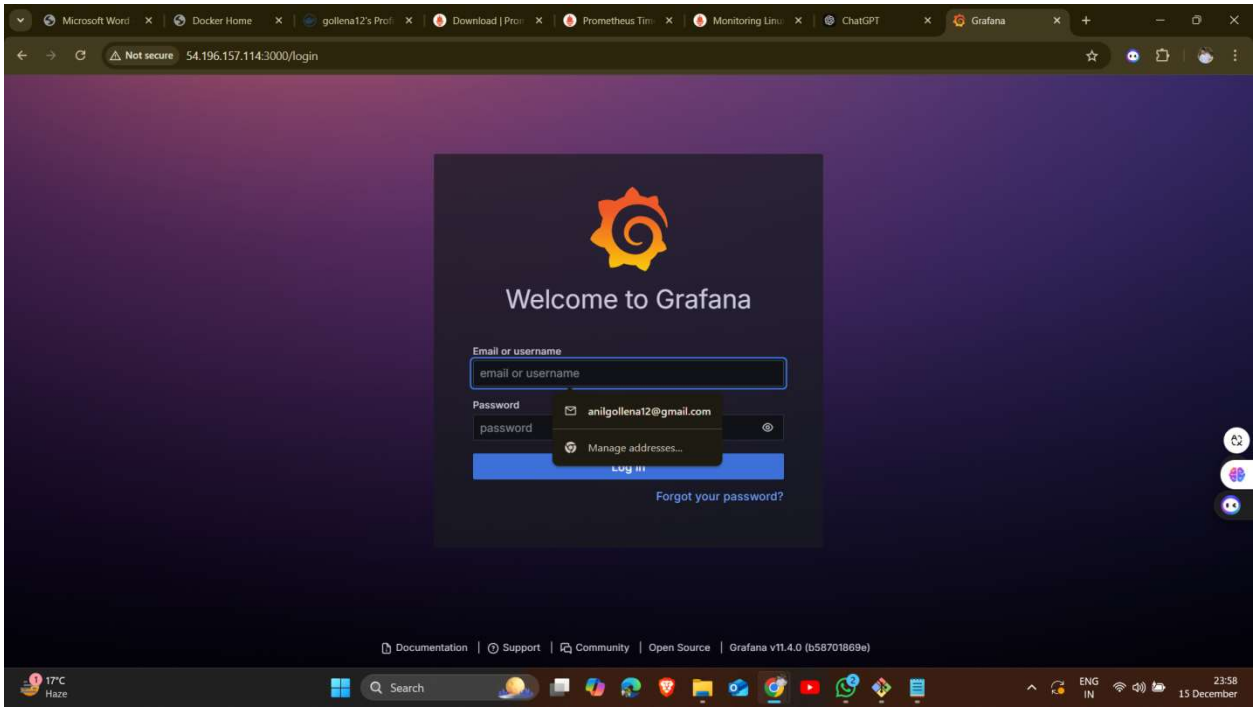
Now copy the public ip with port number 9100



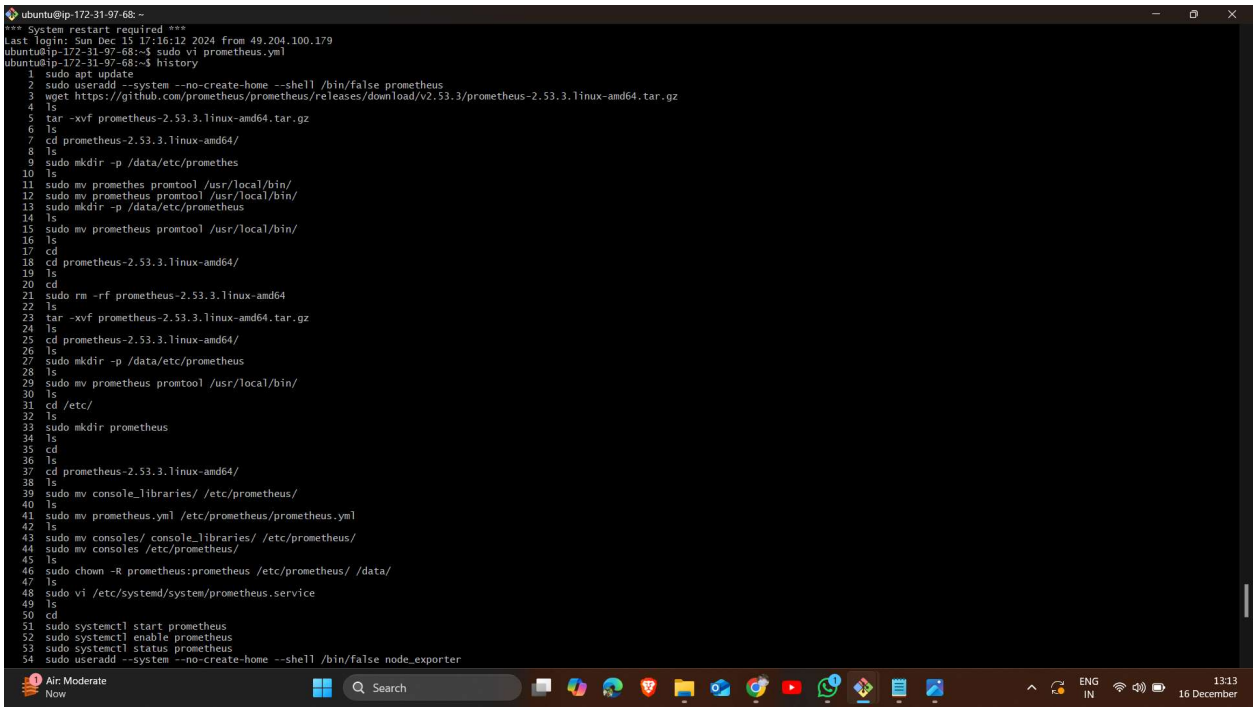
Now copy the public ip with port number 9090

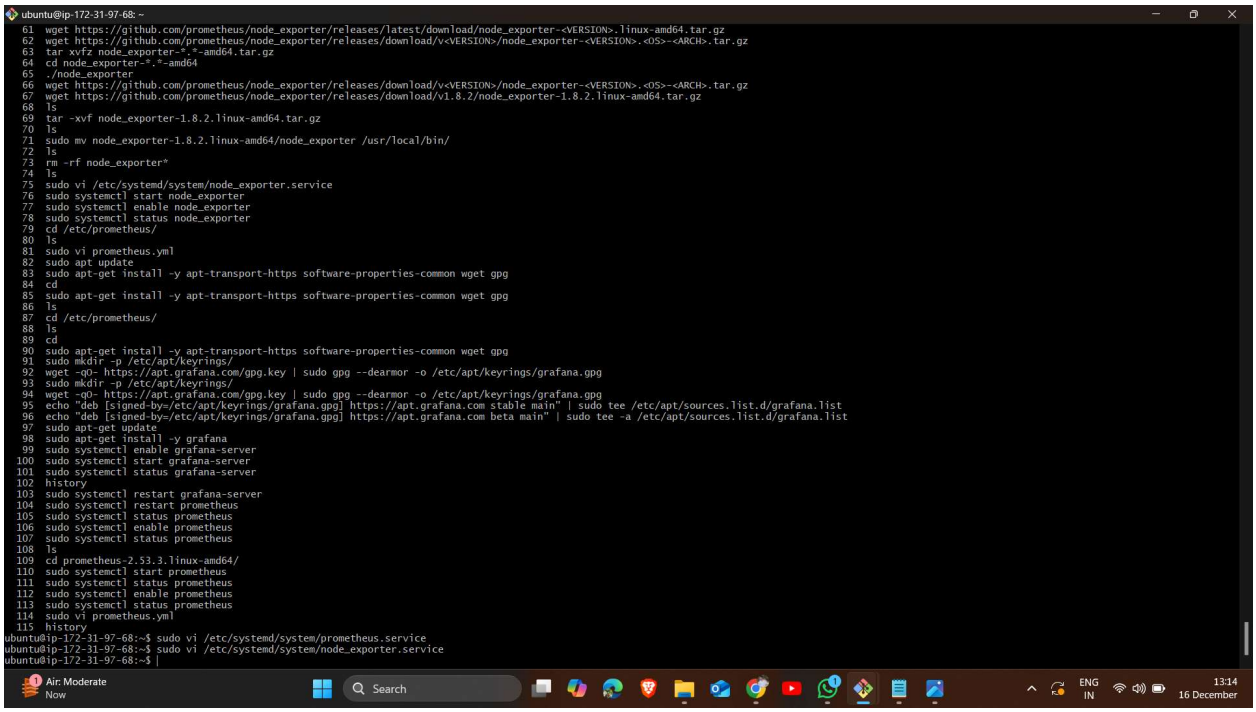


Copy the public ip with port number 3000

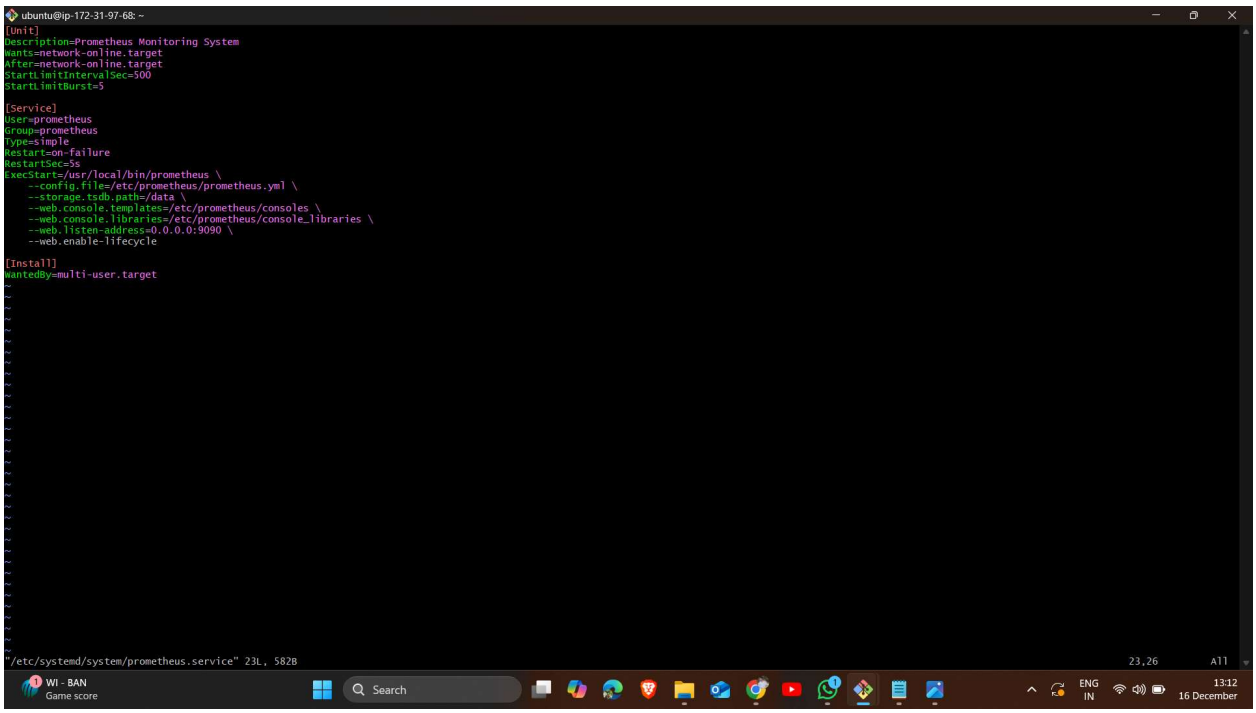


Here are the commands I used to setup these

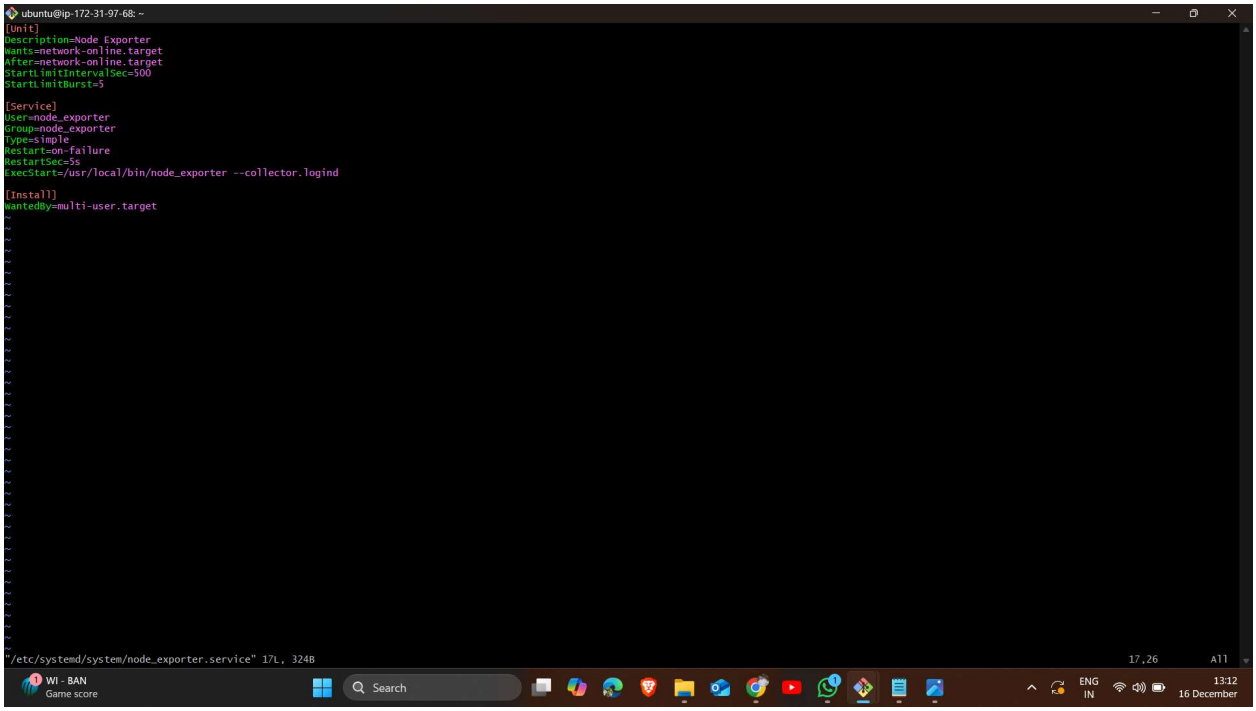


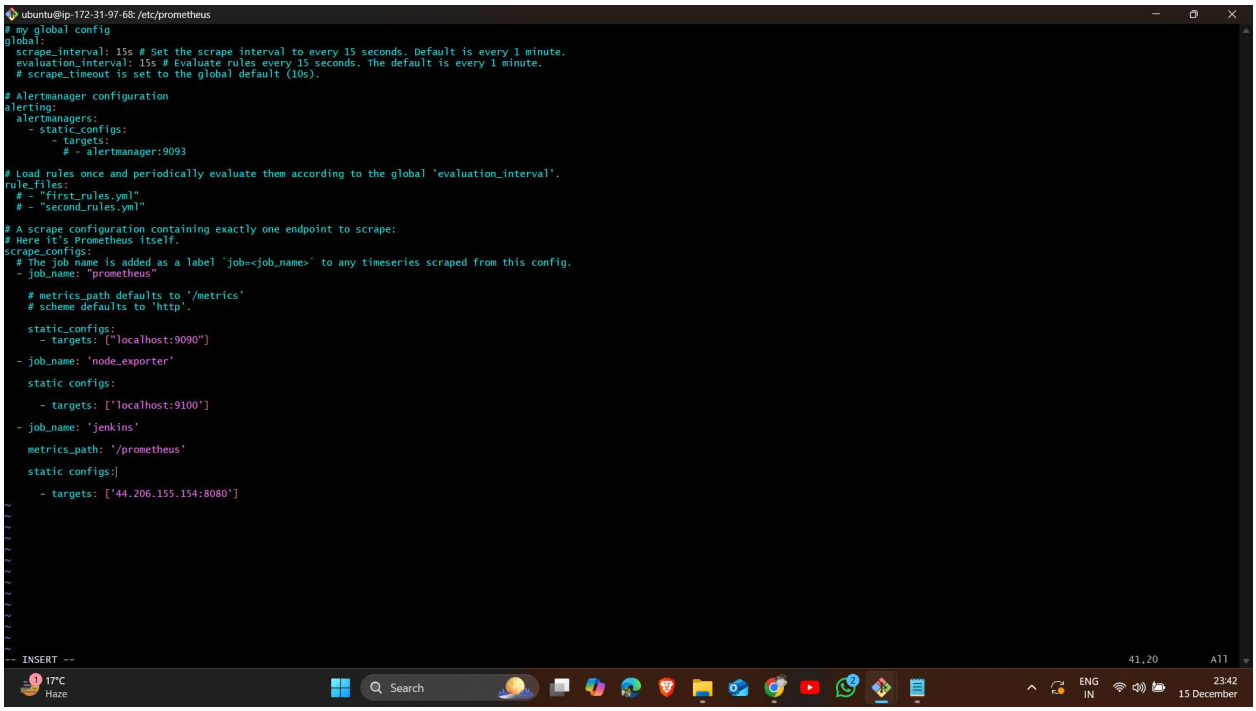


And configure Prometheus and Grafana files



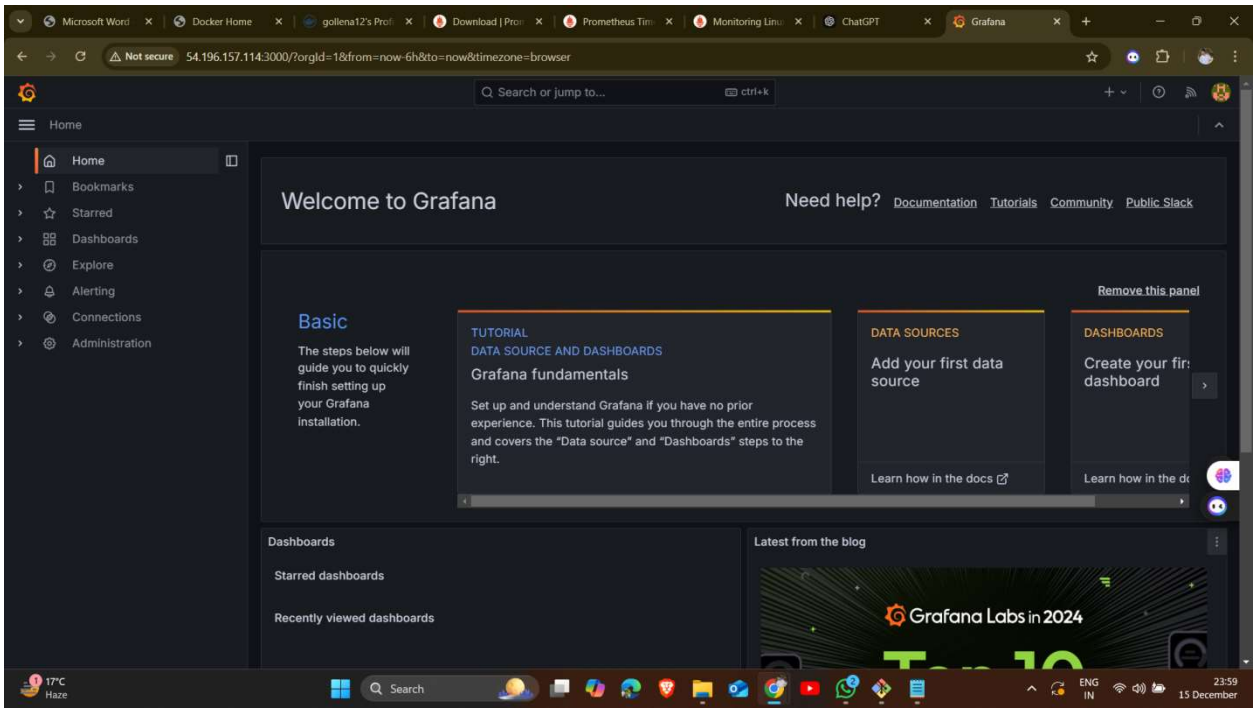
Node exporter configuration file.

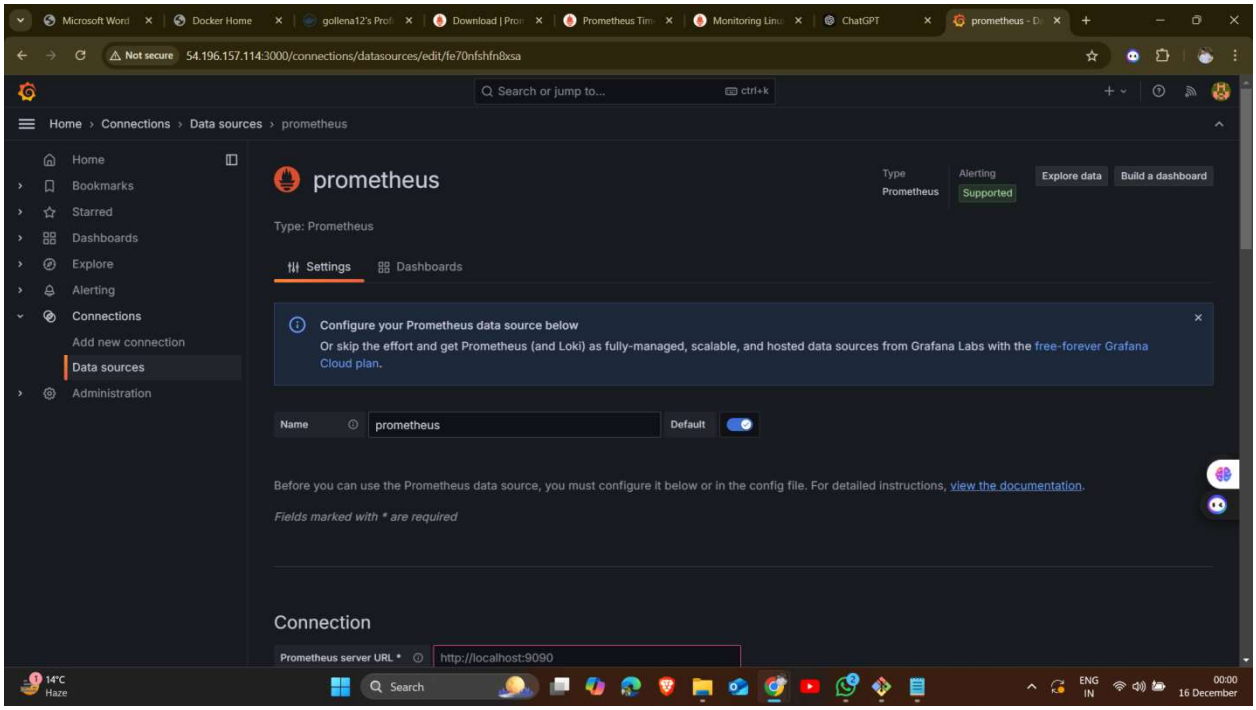




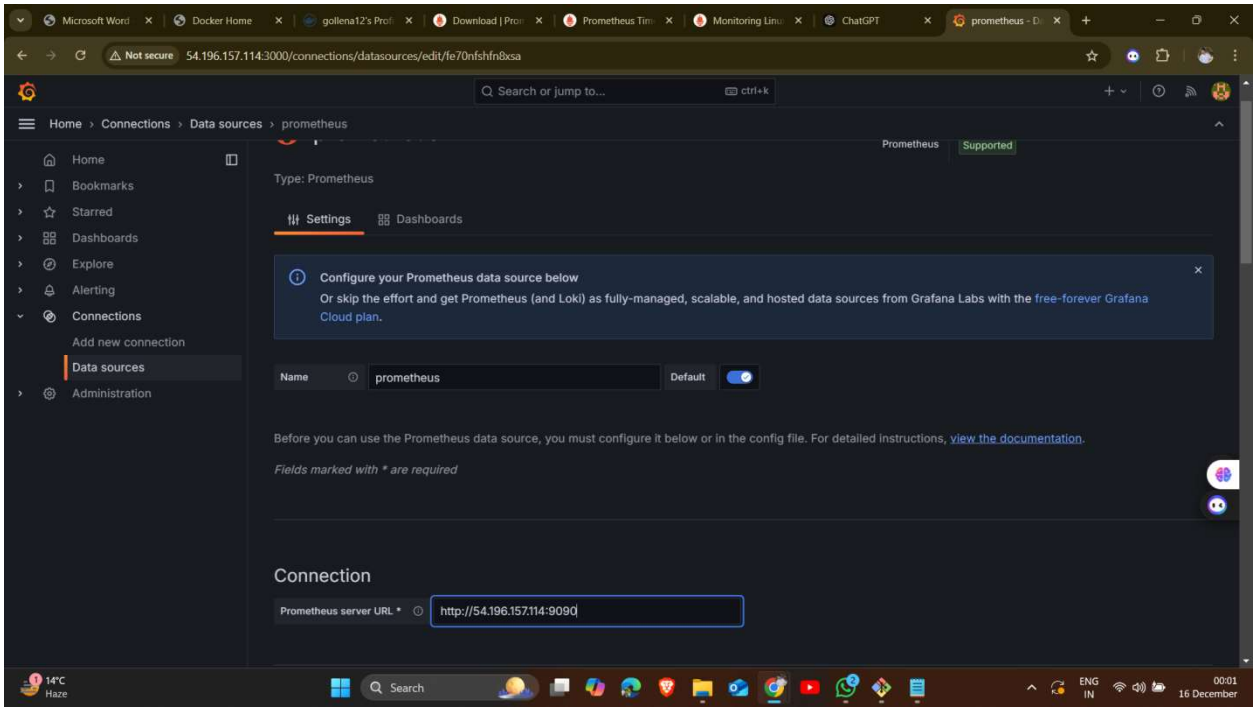
Now go to graphana page add some configuration

Now set up the Prometheus in Grafana for visualize and analyze data form Prometheus.



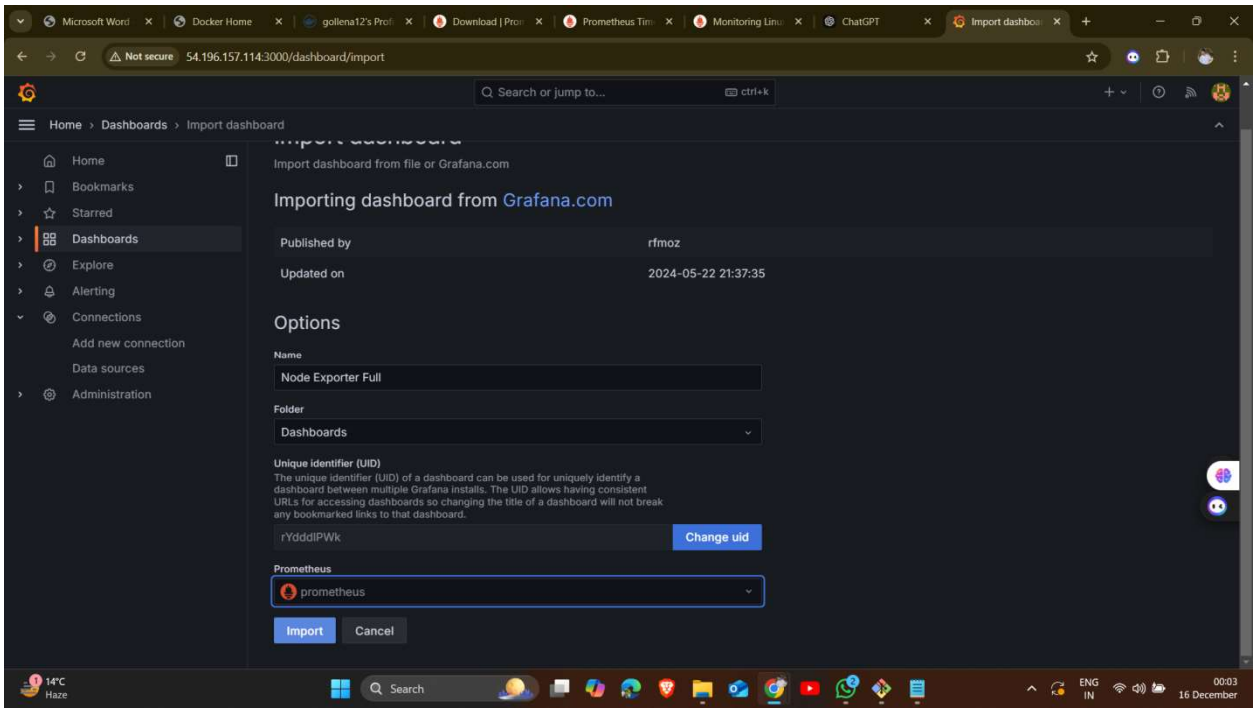


Add localhost

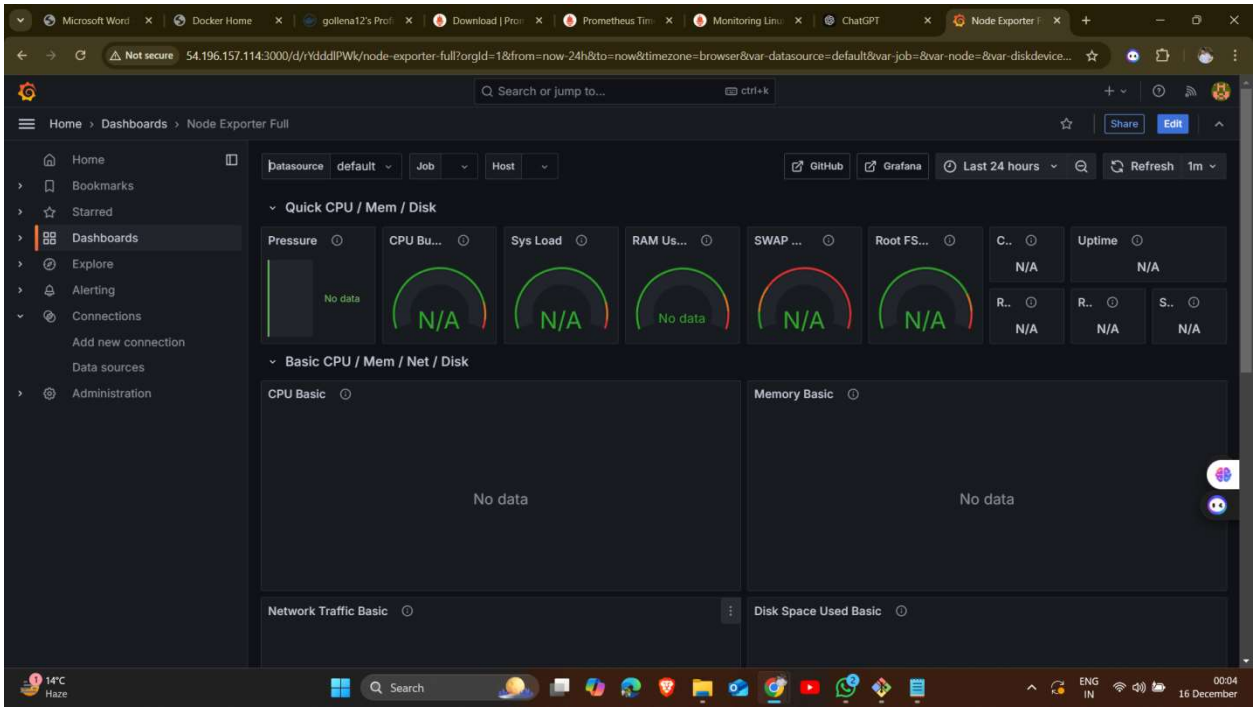


And save and apply the changes.

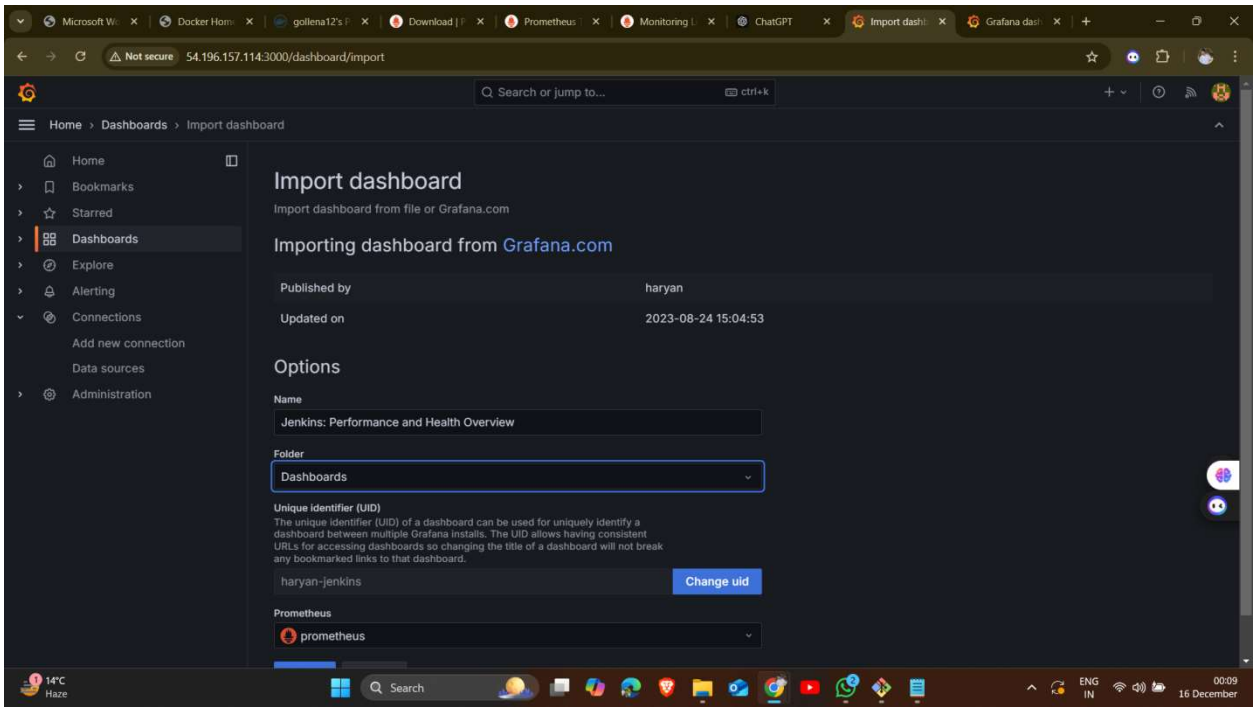
Now under dashboards→import dashboard



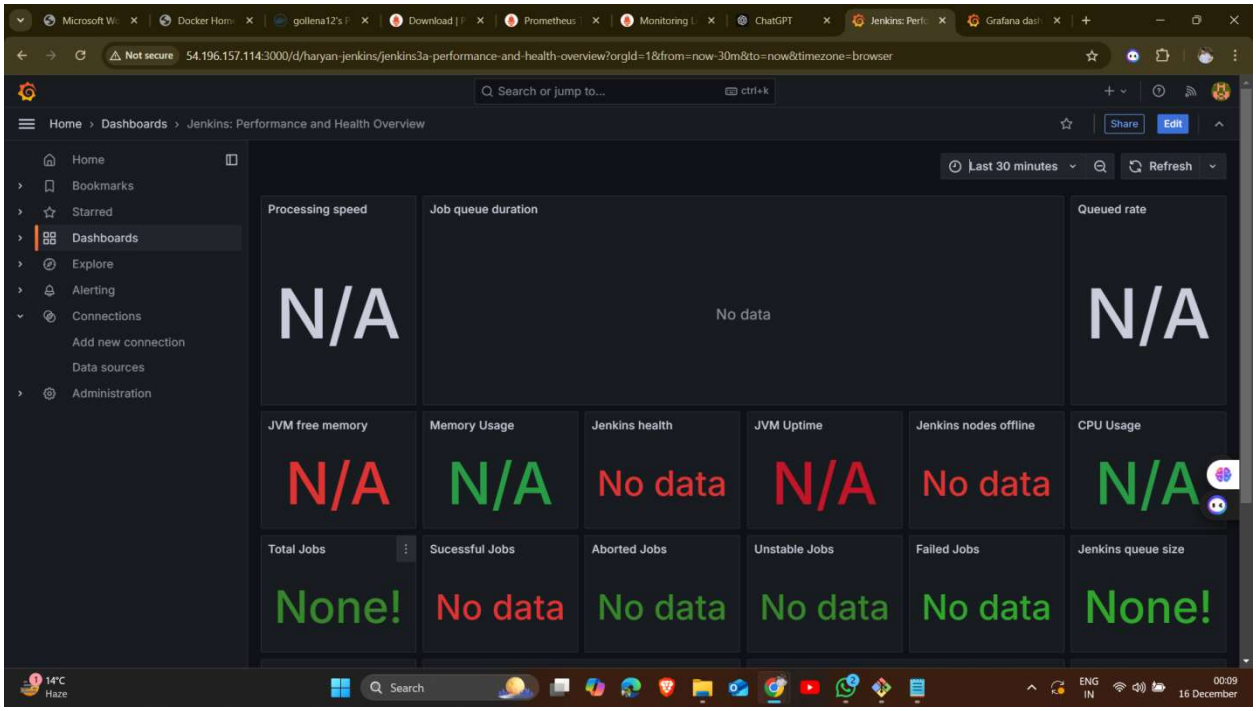
Now you can see the changes are made in node exporter



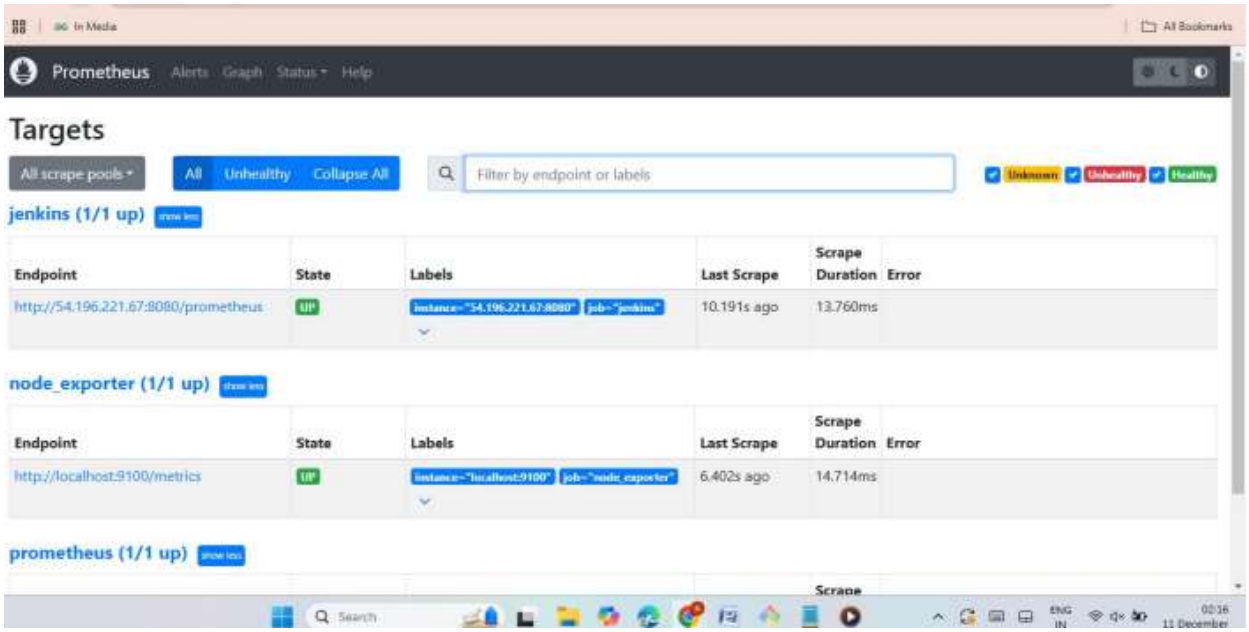
Now again import the Jenkins dashboard



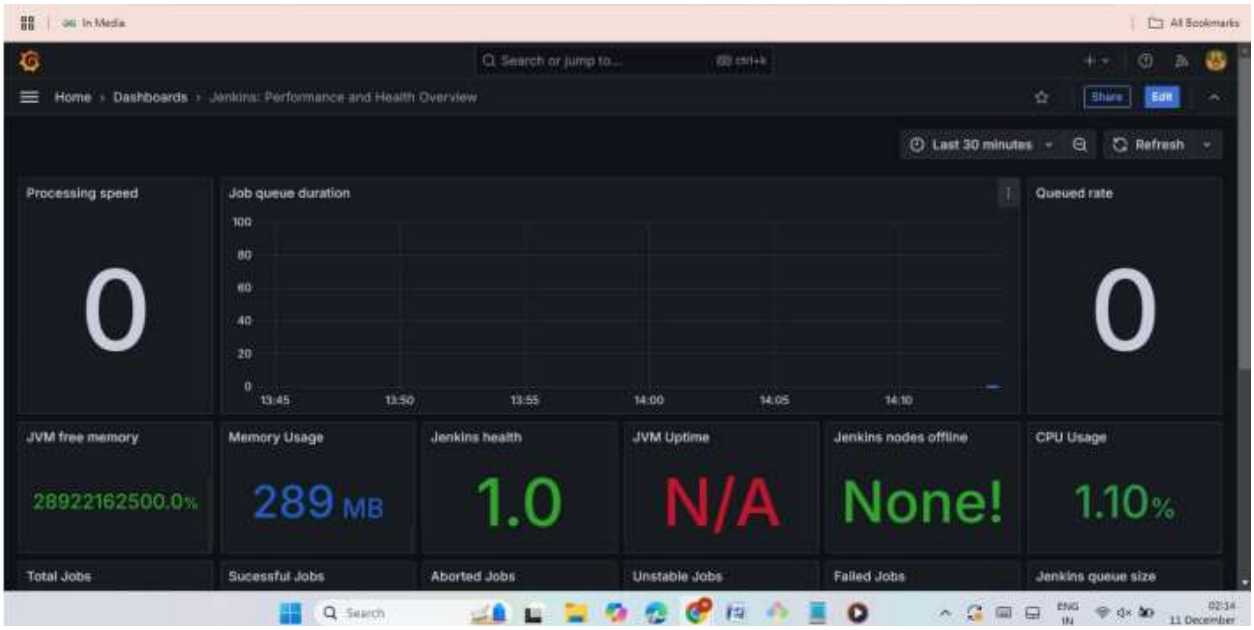
Now you can see the changes made by the Jenkins

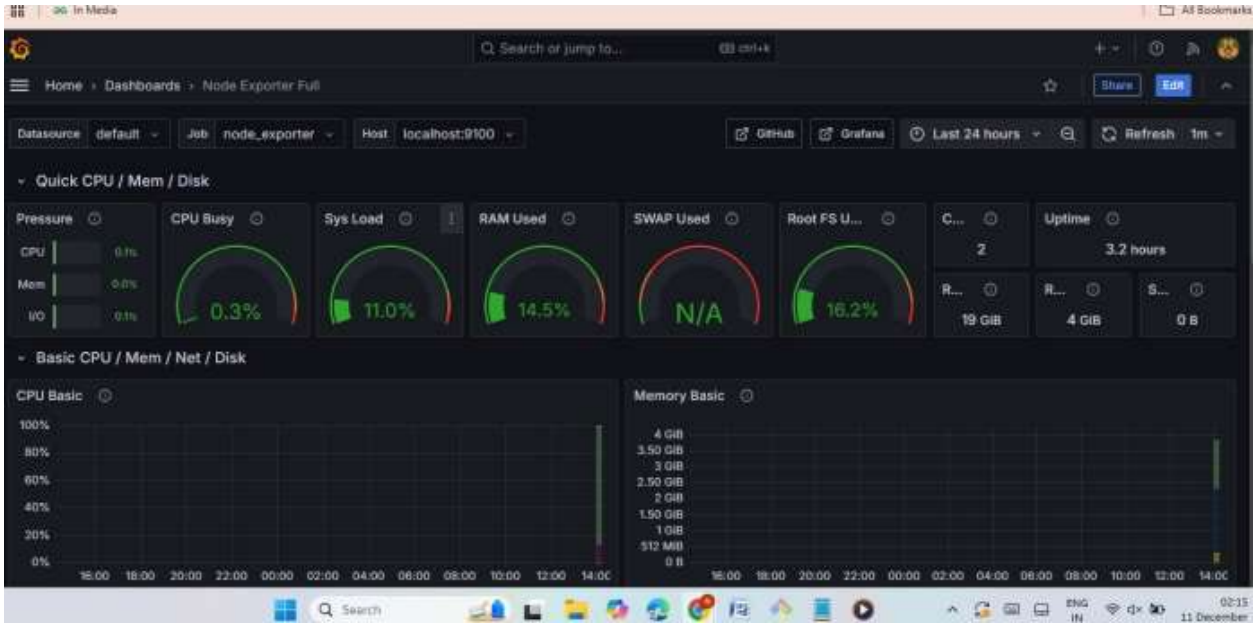


- Now check the plugins in Jenkins.
- Install Prometheus metrics then restart the Jenkins.
- Restart the Prometheus.yaml file.



Here we can see the Jenkins performance and health overview in Grafana.





PHASE 5: DEPLOY NETFLIX APPLICATION WITH KUBERNETES EKS AND ARGOCD.

Kubernetes EKS:

- **Amazon Elastic Kubernetes Service (EKS)** is a **managed Kubernetes service** by AWS.
- Simplifies the deployment, scaling, and management of Kubernetes clusters in the cloud.
- Key features:
    - **Highly Available**: EKS manages the Kubernetes control plane across multiple availability zones.
    - **Secure**: Integrates with AWS Identity and Access Management (IAM) for access control.
    - **Scalable**: Easily scales worker nodes and applications as demand increases.
    - **Integrations**: Works seamlessly with AWS services like EC2, Fargate, and CloudWatch.

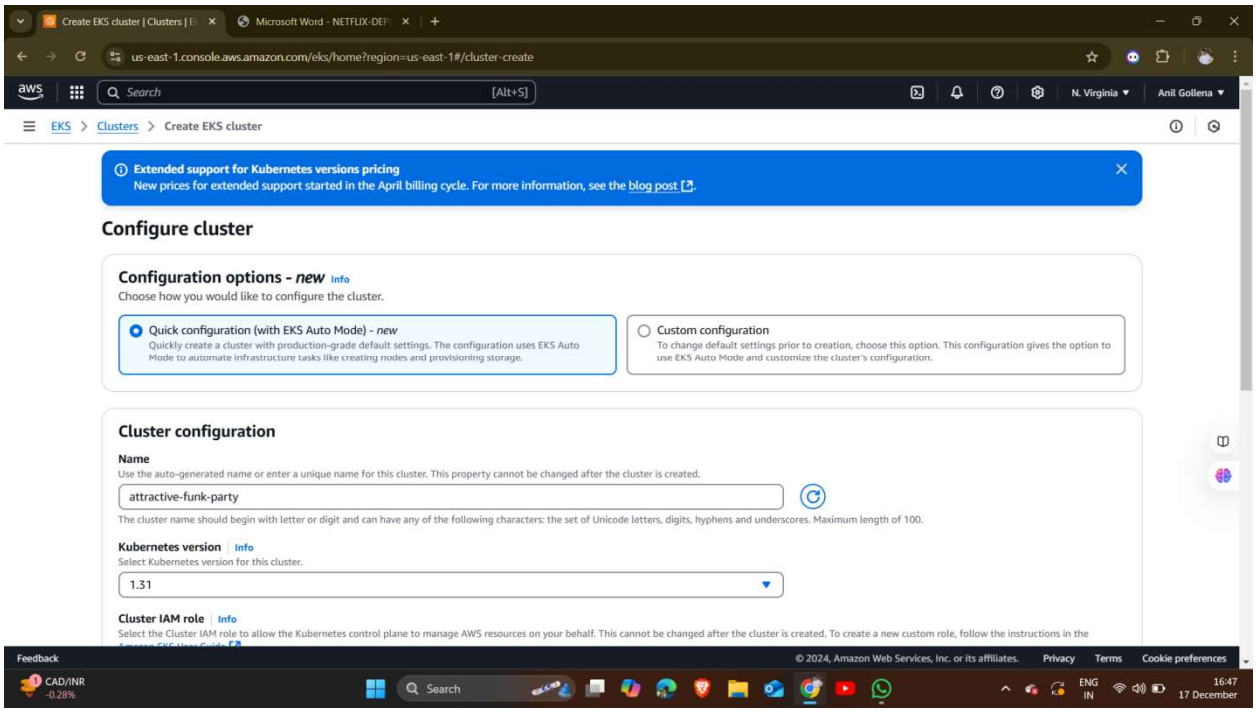
ArgoCD:

- **ArgoCD** is a declarative, GitOps-based **continuous delivery tool** for Kubernetes.
- Automates application deployment and lifecycle management in Kubernetes clusters.
- Key features:
    - **GitOps**: Uses Git repositories as the source of truth for application definitions and deployments.
    - **Sync and Drift Detection**: Continuously monitors the cluster state and identifies deviations from the desired state.
    - **Multi-Cluster Management**: Supports deploying applications to multiple Kubernetes clusters.
    - **UI and CLI**: Offers an intuitive web UI and CLI for managing deployments.

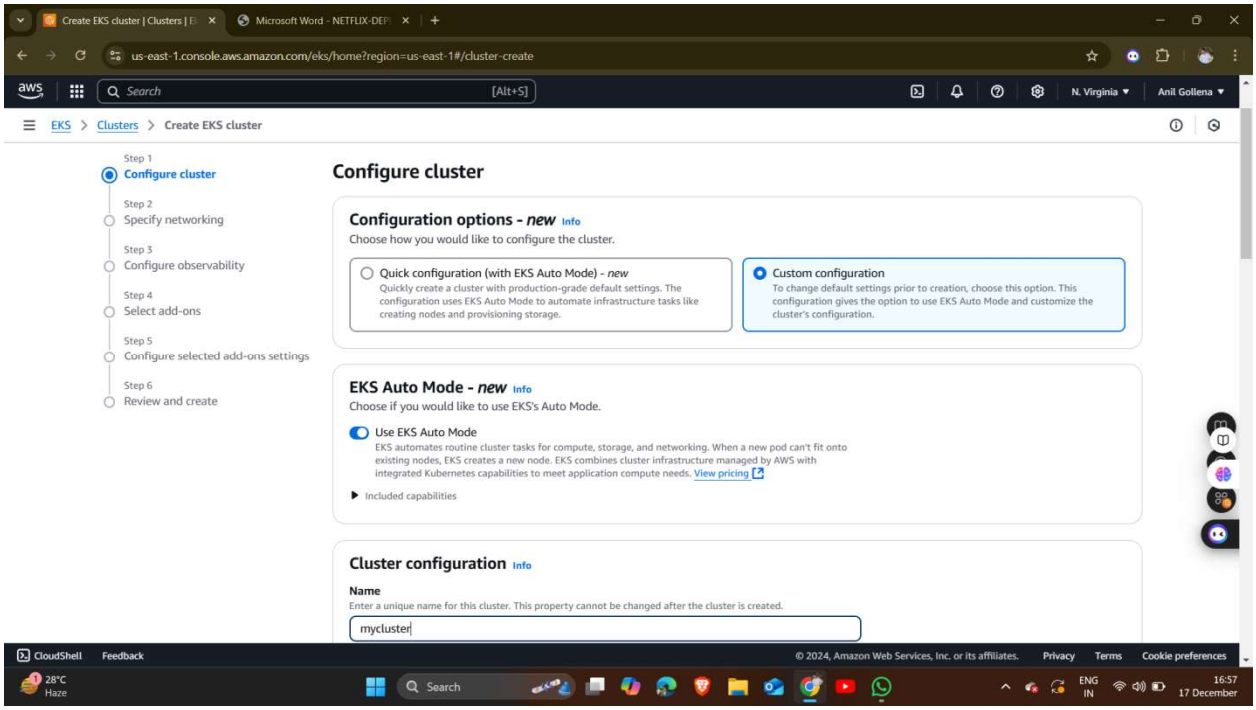
Kubernetes EKS + ArgoCD Integration:

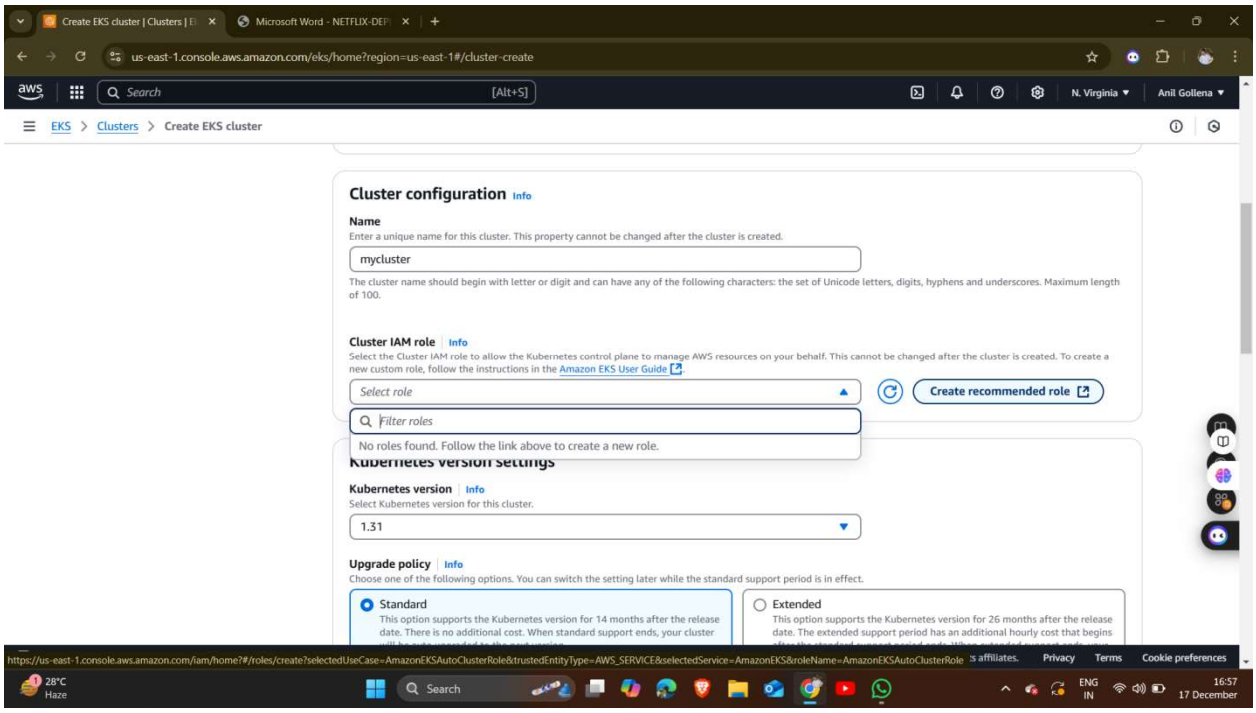
1. **Managed Clusters**: Use EKS for running Kubernetes clusters with minimal operational overhead.
2. **GitOps Workflow**: ArgoCD pulls deployment manifests from Git repositories and applies them to EKS clusters.
3. **Continuous Deployment**: Automate the deployment of new versions or configurations directly from Git.
4. **Real-Time Monitoring**: ArgoCD ensures EKS clusters match the desired state defined in Git.

Login to aws management console create a EKS cluster and node group for that here is the steps to create cluster.

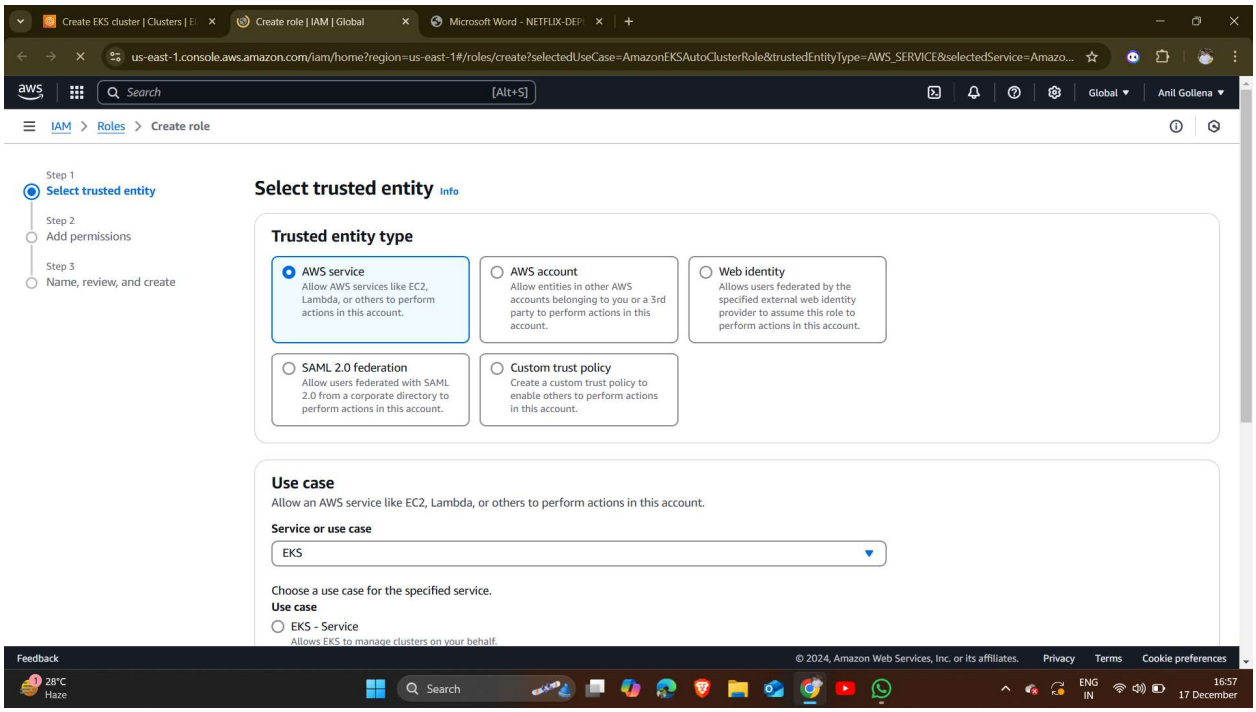


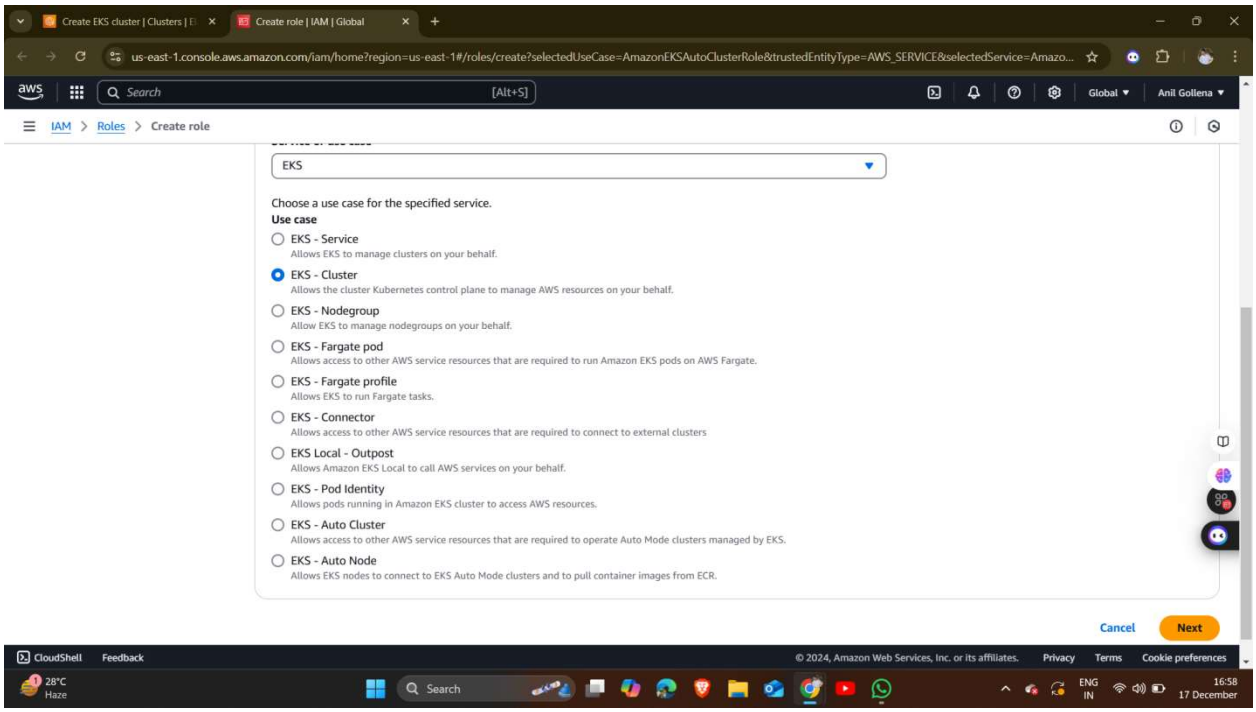
Select custom configuration.



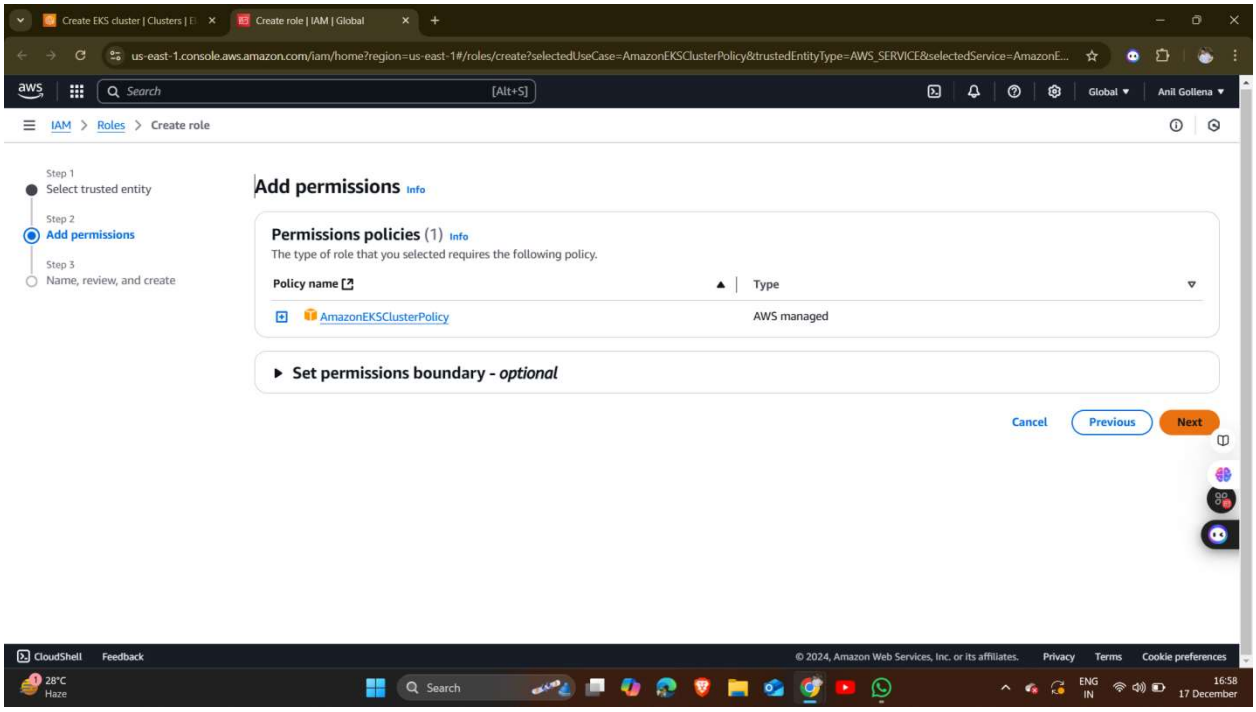


Now give cluster name and create IAM role for that click create recommended role.

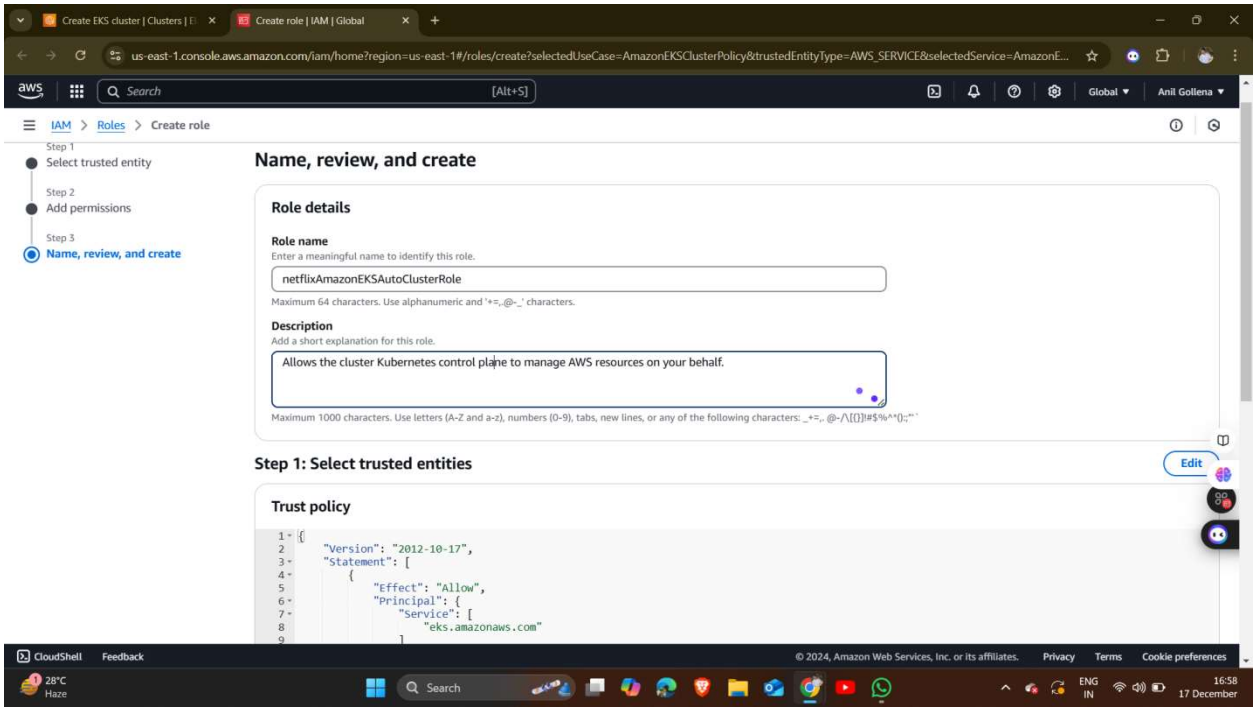




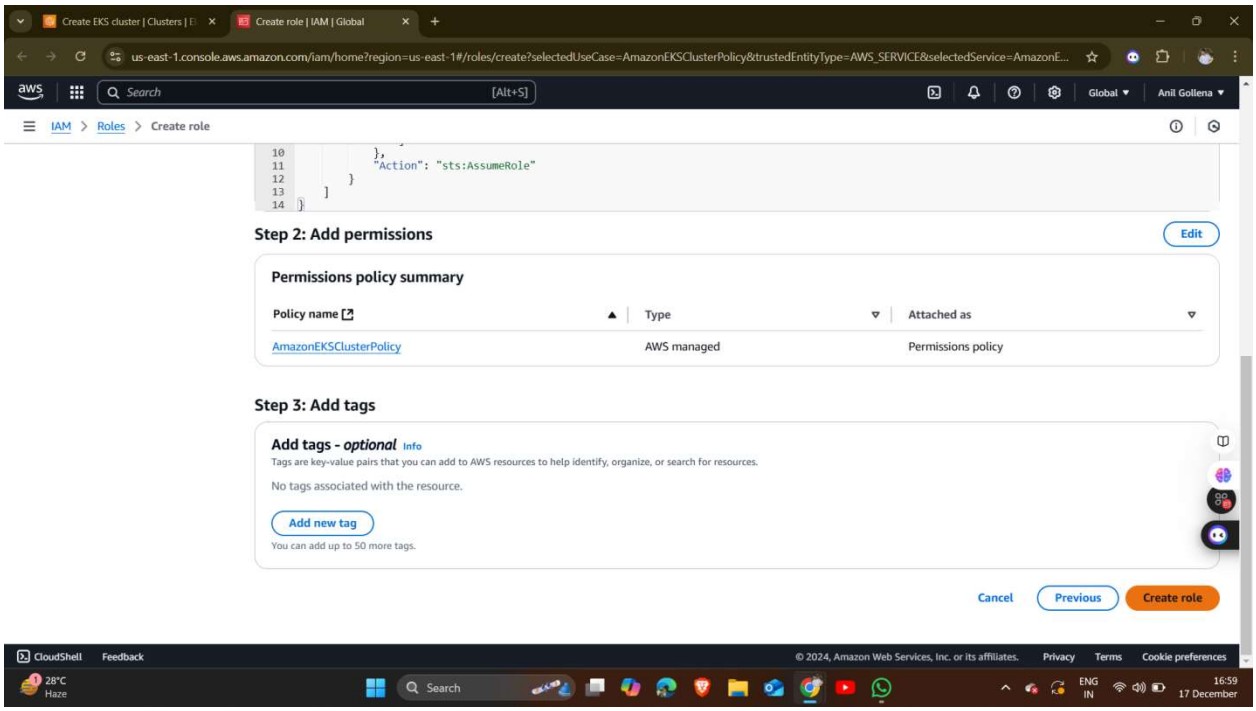
Now click on next.



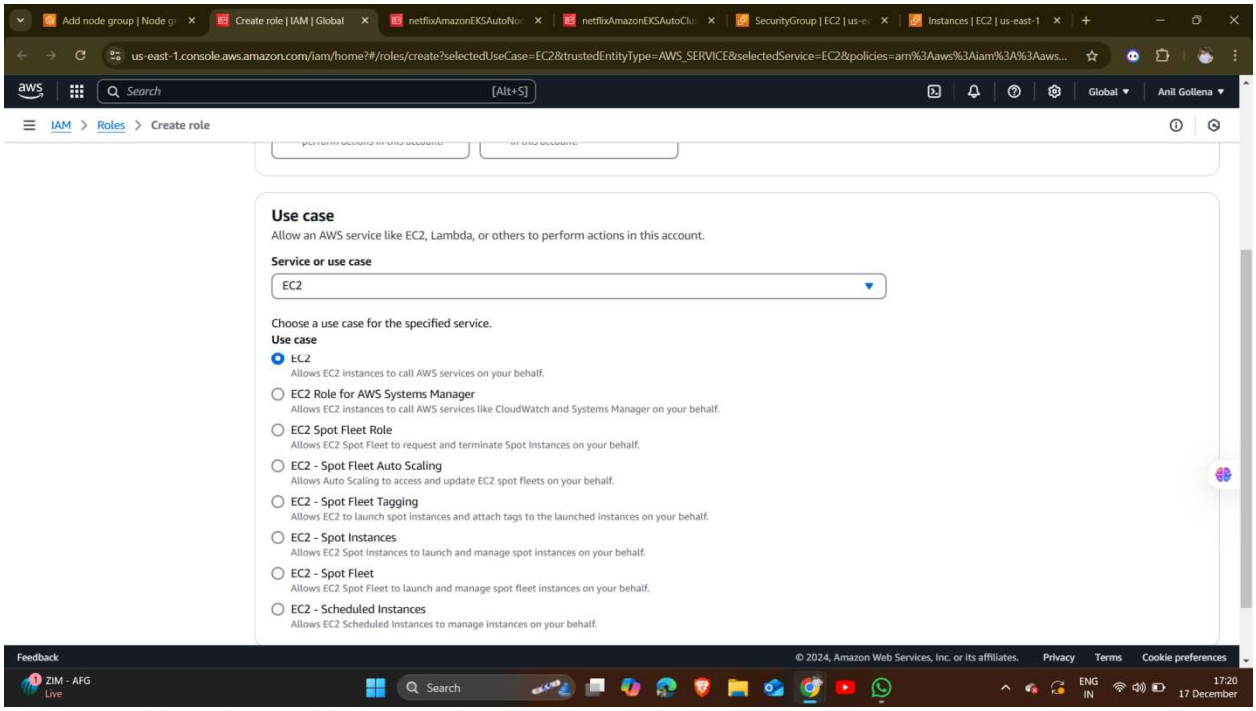
Now give role name.

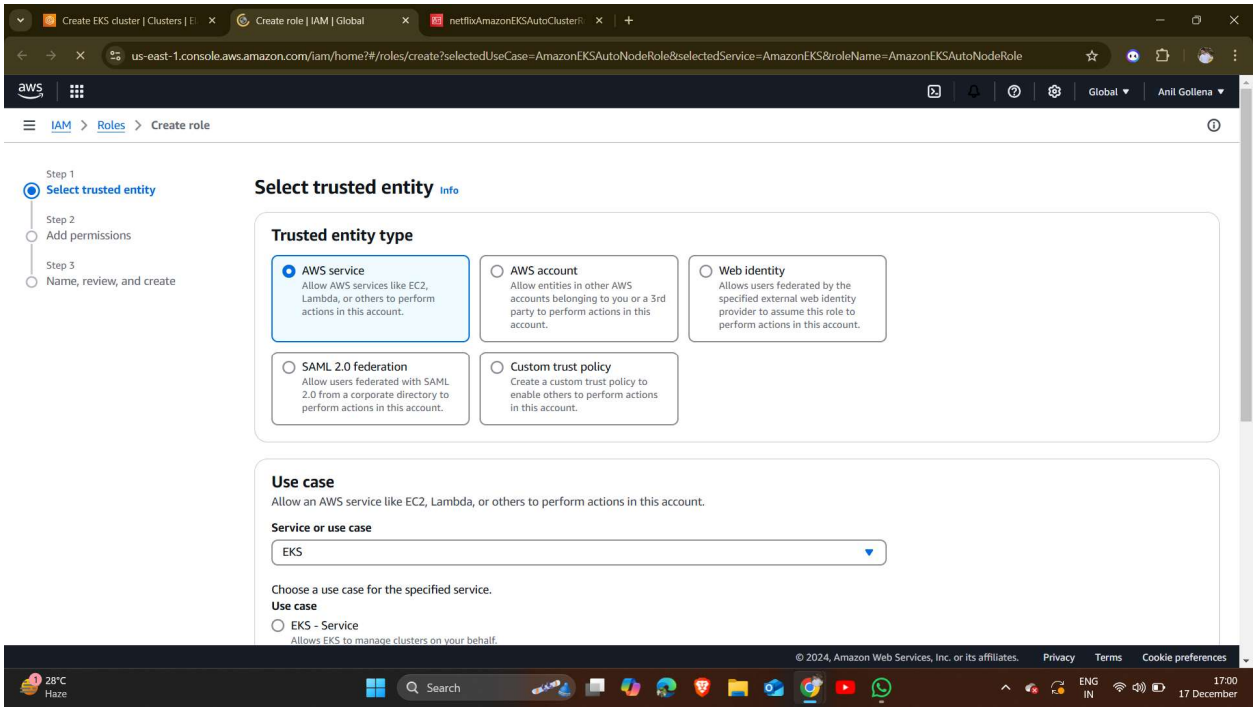


Create role

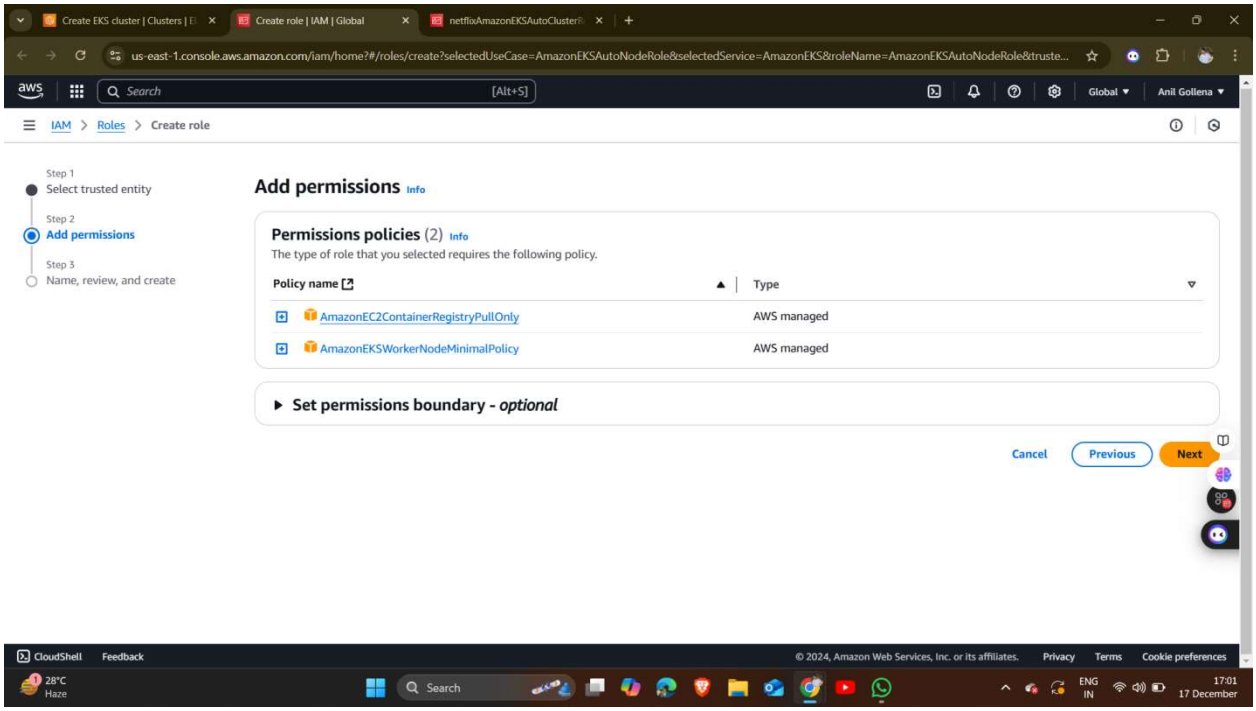


Now create another role for EC2

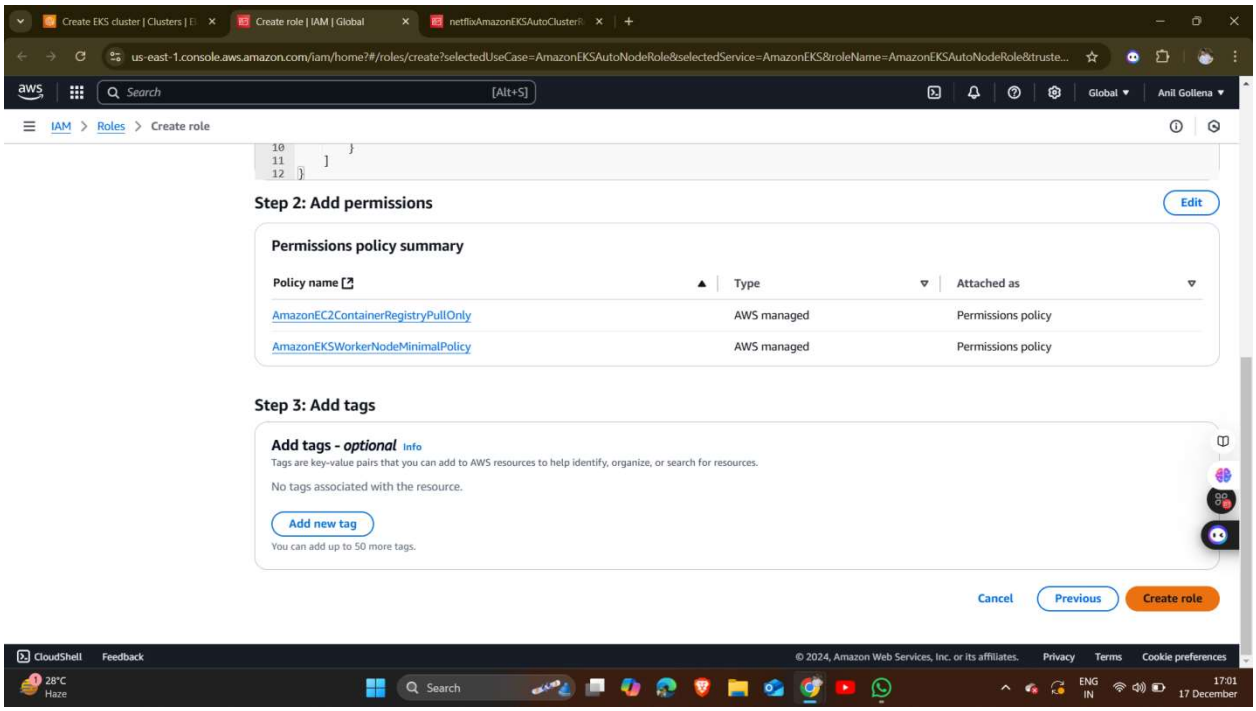




Now select ec2

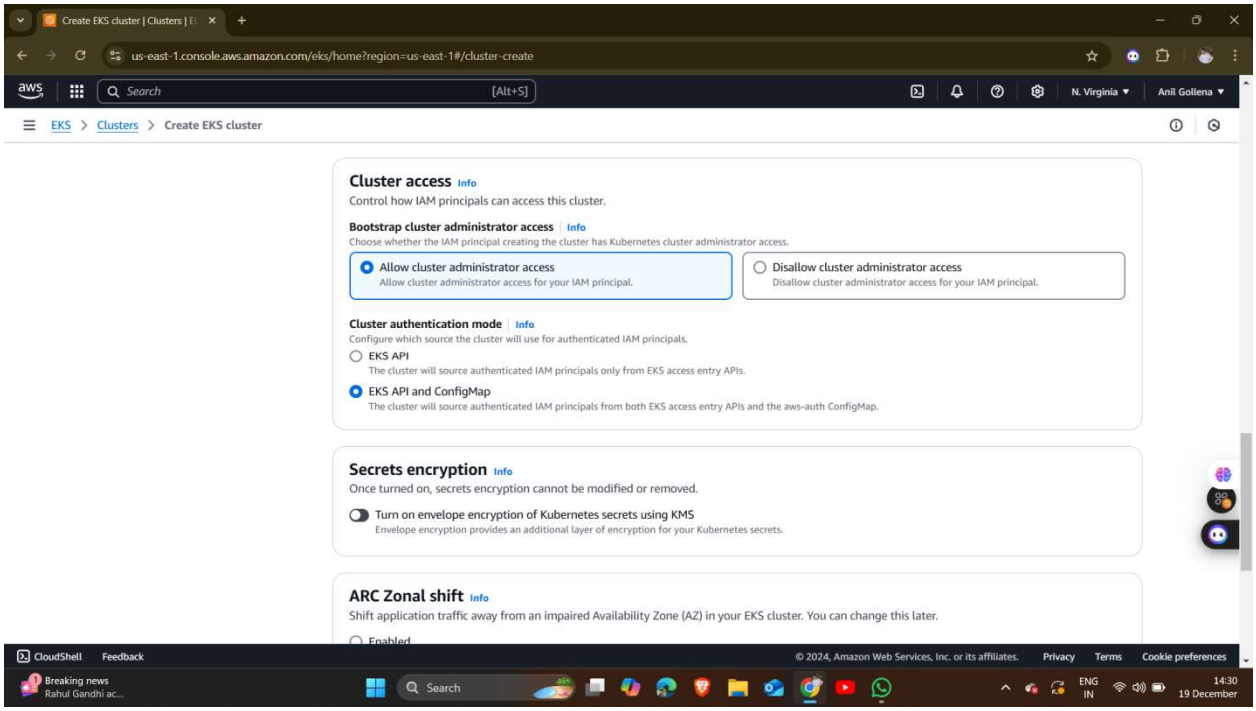


Create role

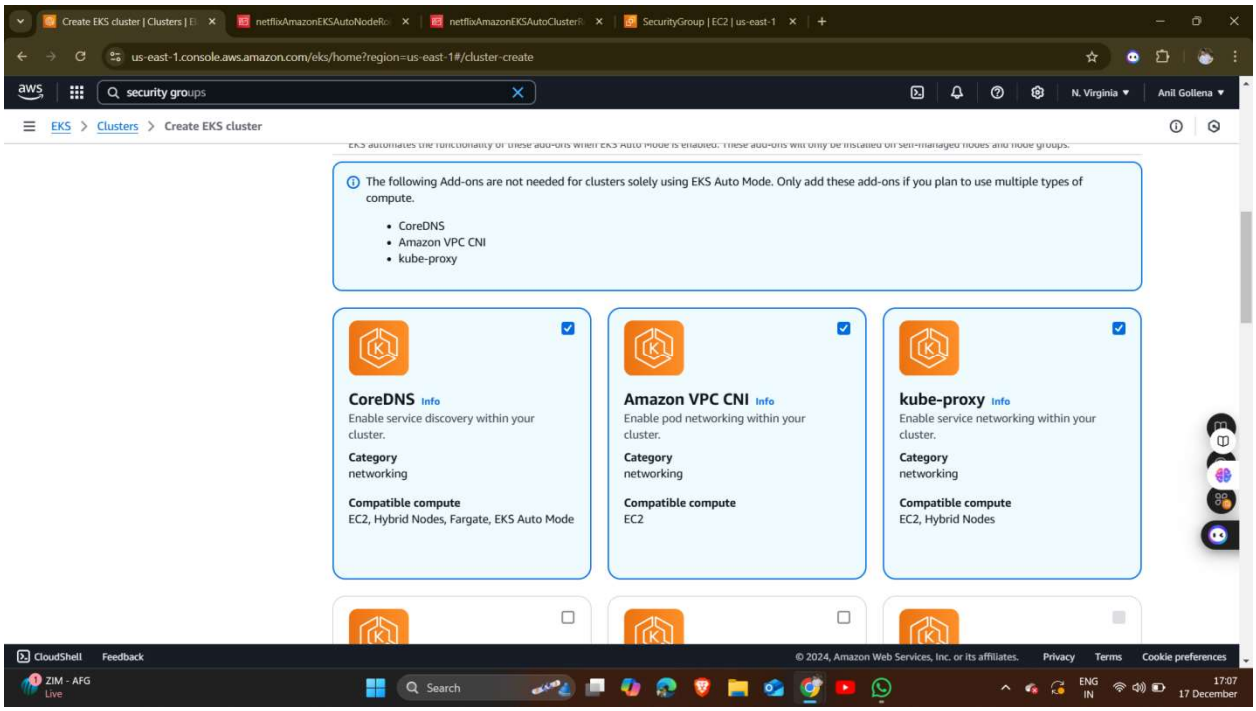


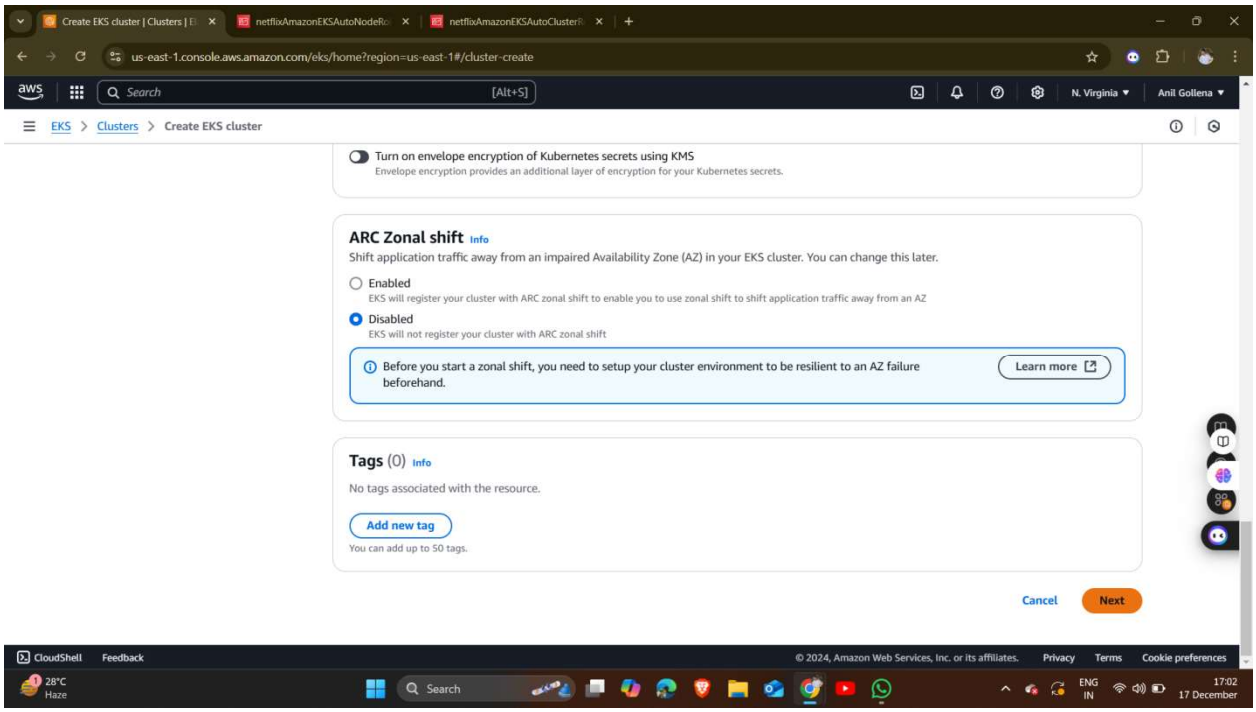
Now refresh the page so that, the roles are redirected to role place

Now give the cluster access

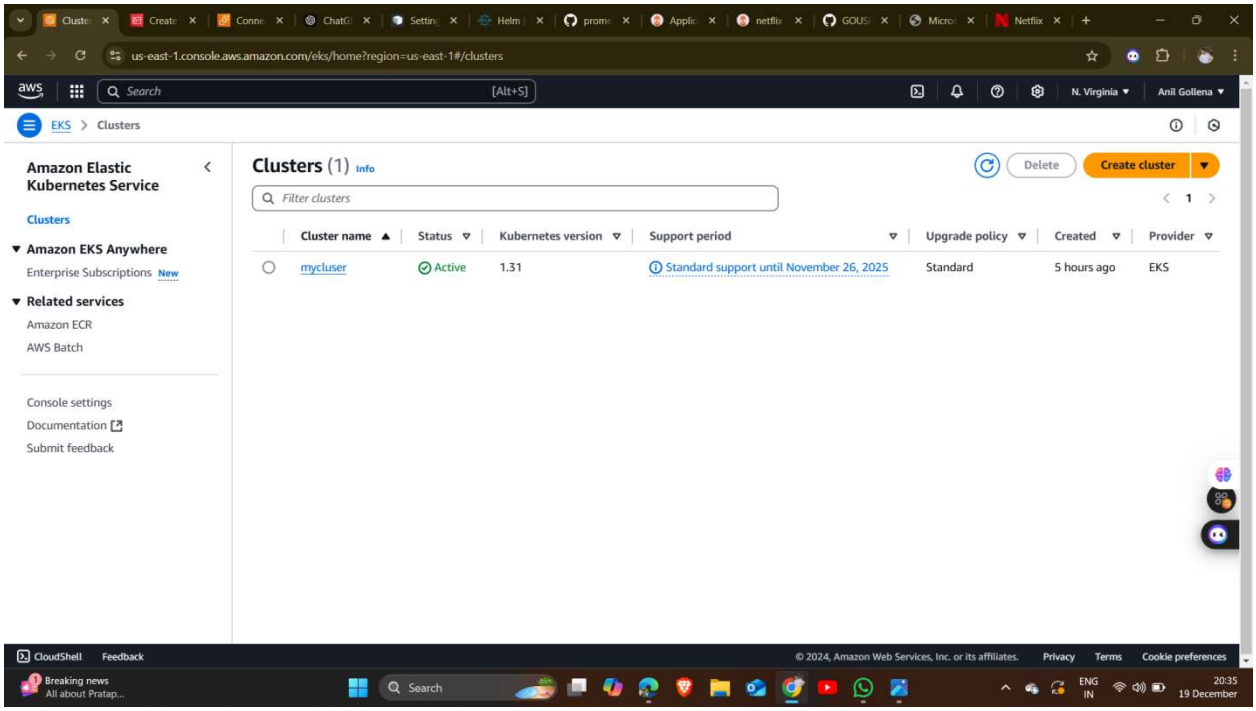


In the additional configuration select these

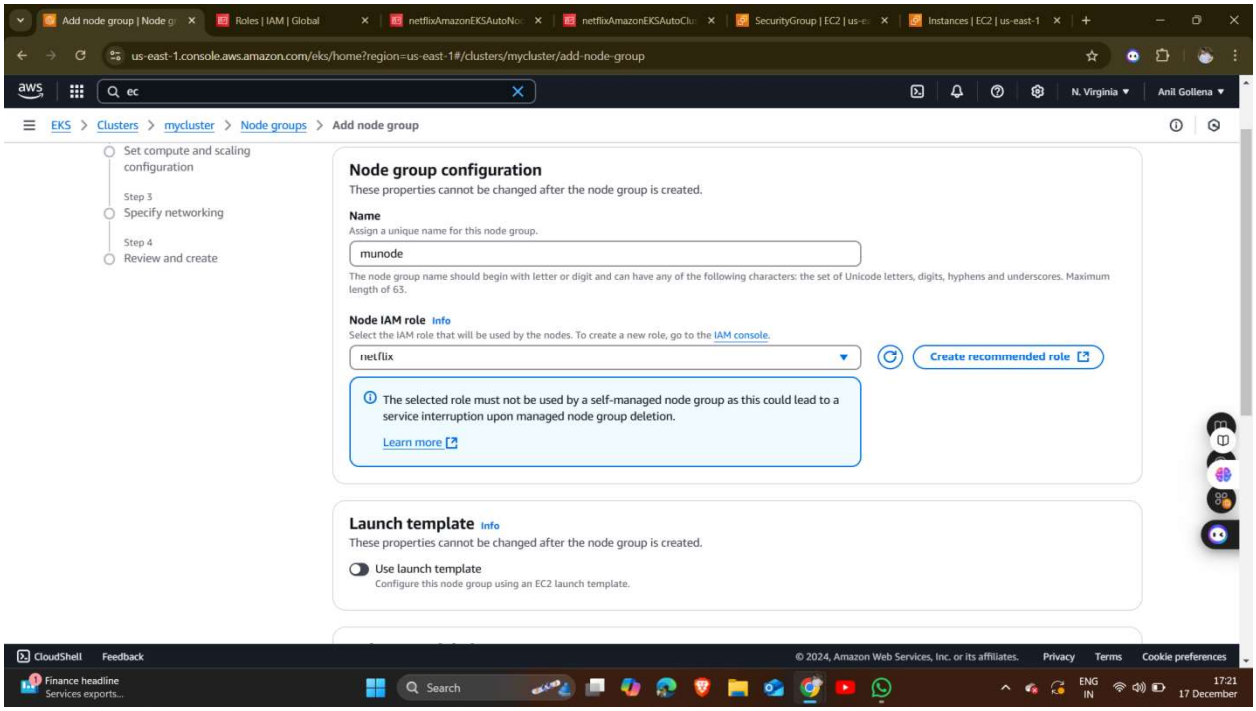




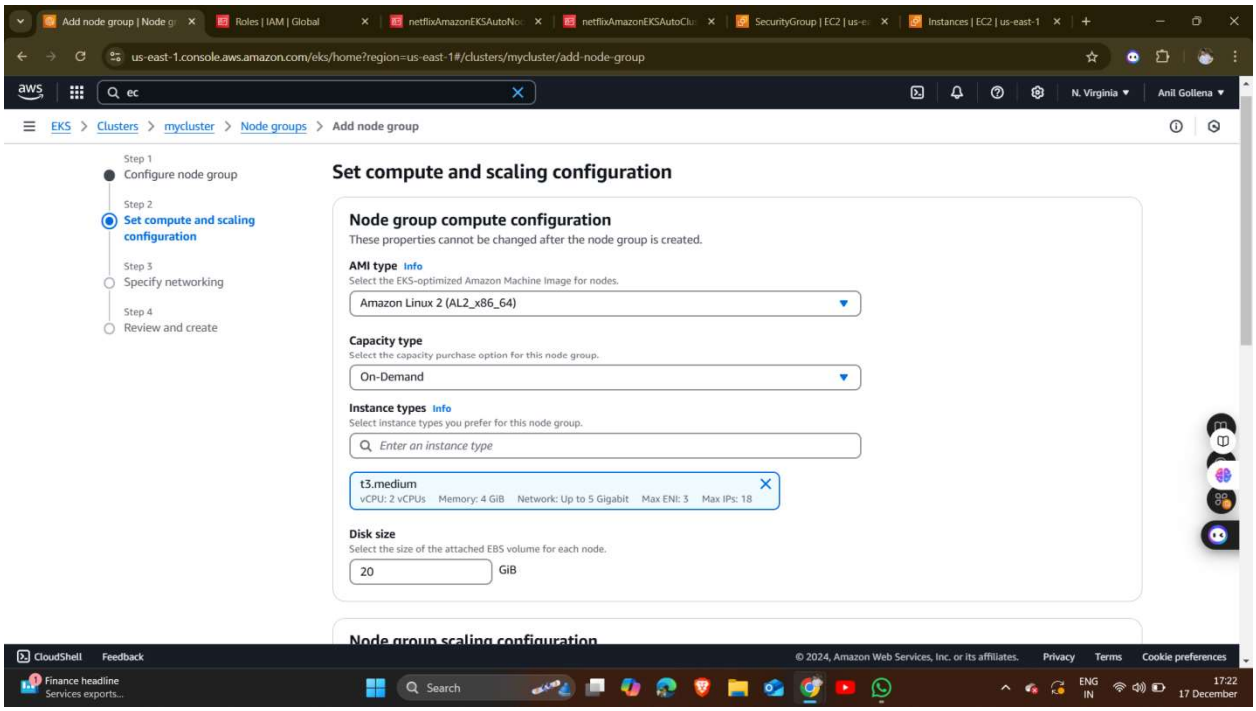
It may take few minutes to create cluster.



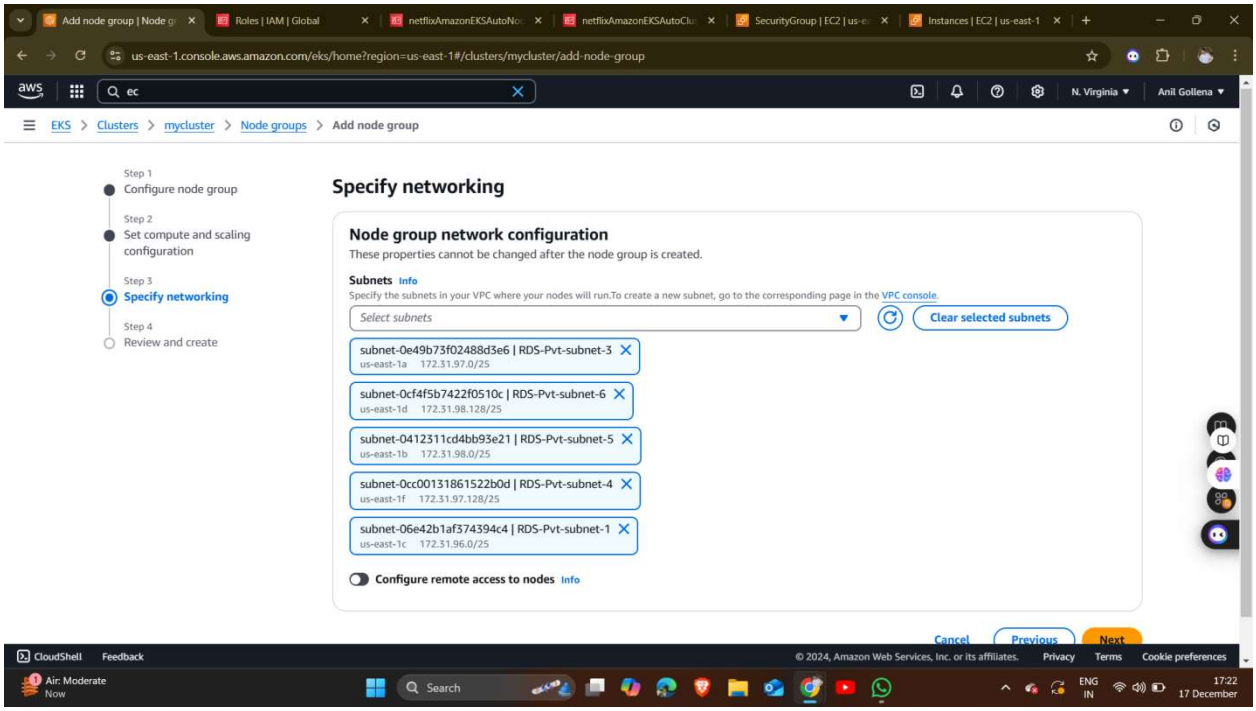
Now the cluster is in active state now create node group.

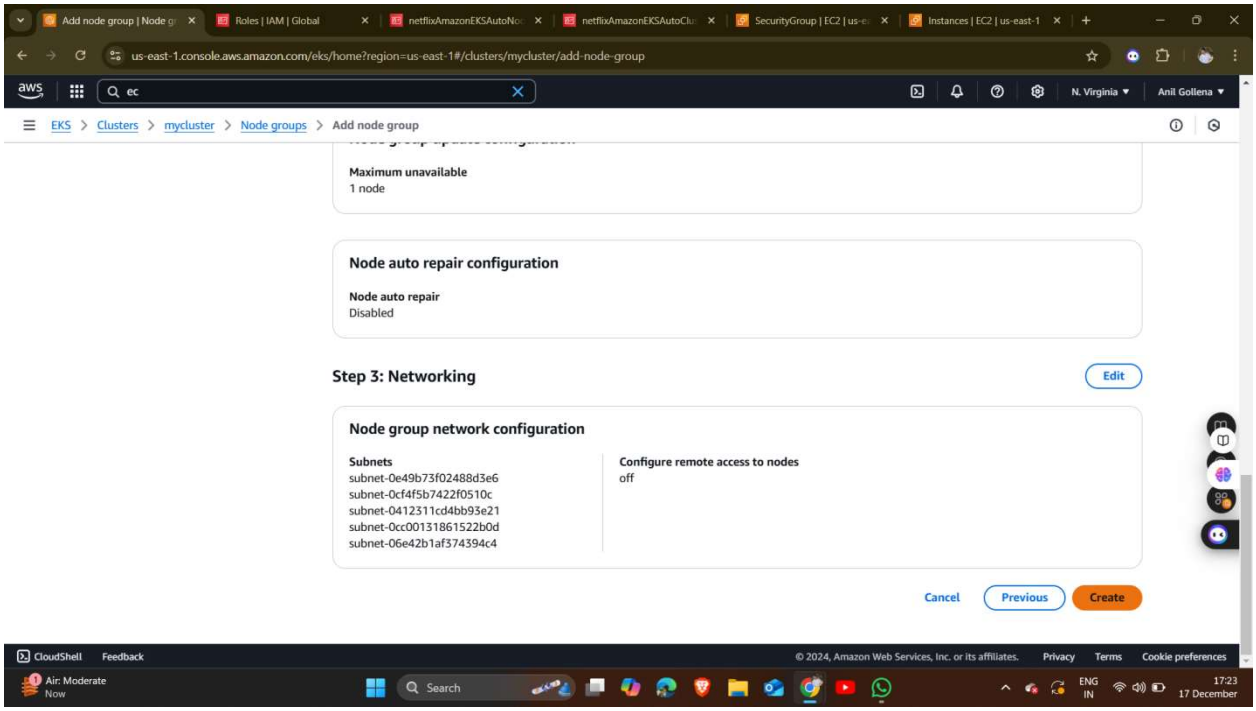


Now create role and attach it with the node.

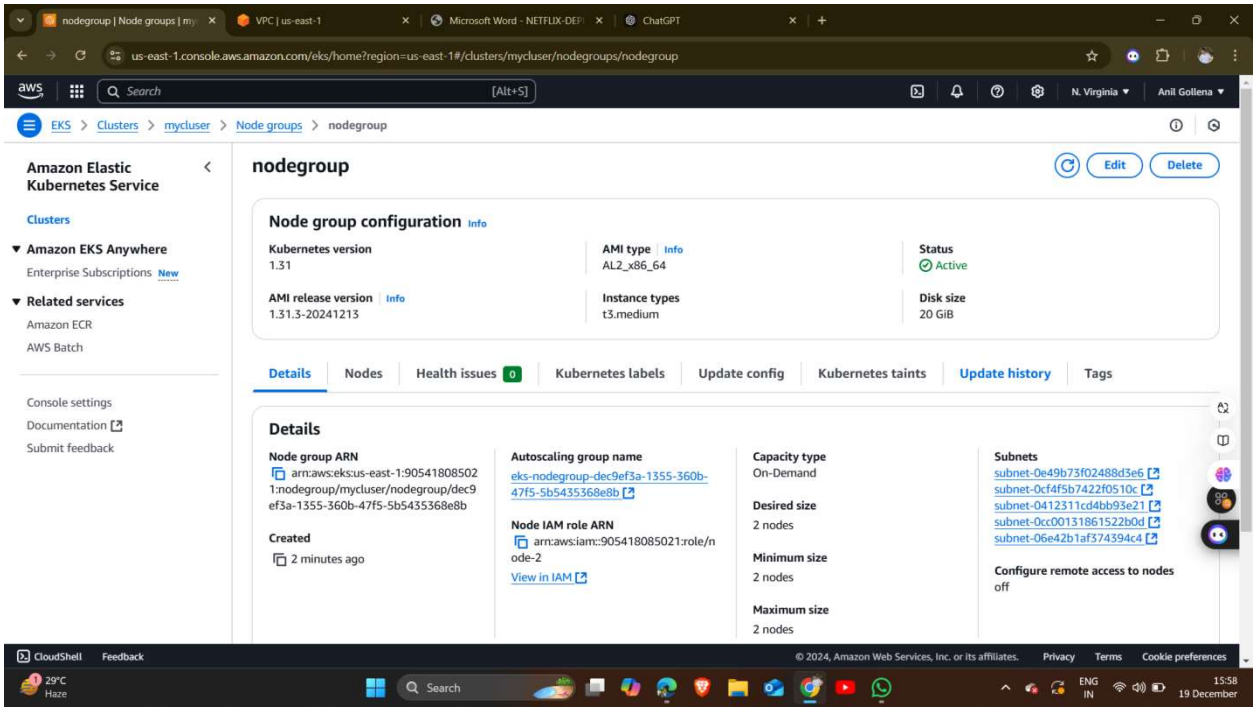


Now add subnets except us-east-1e

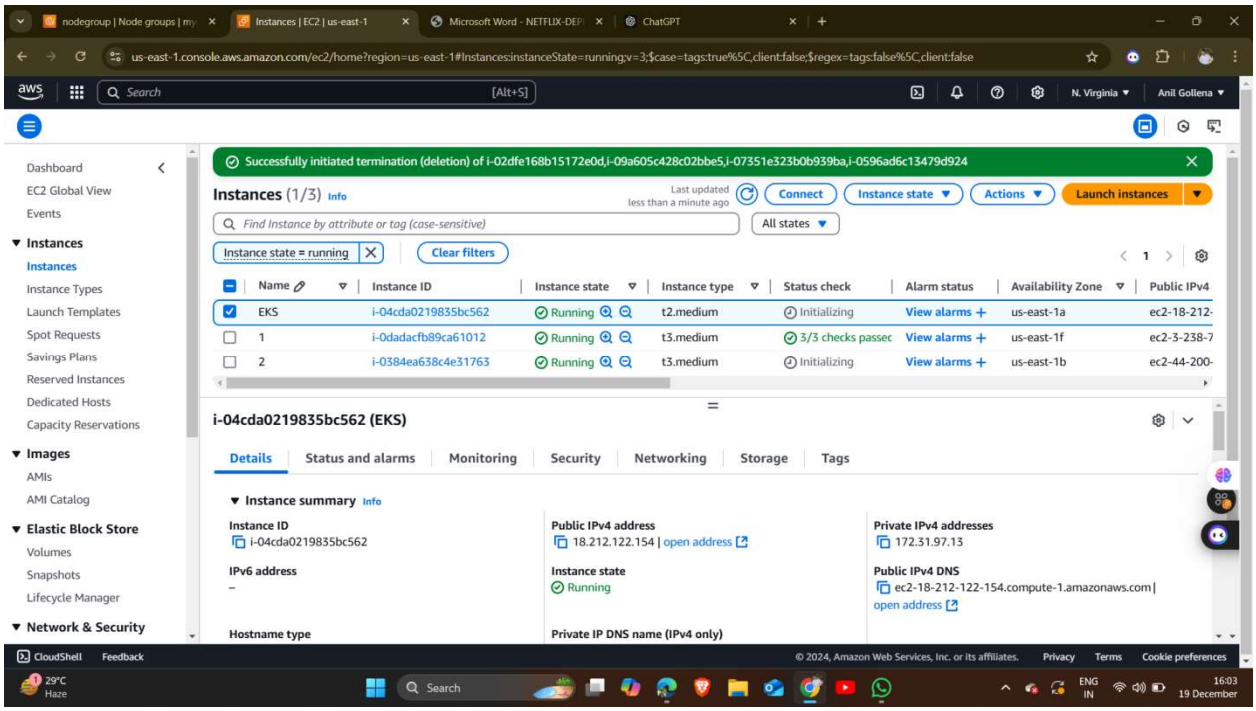




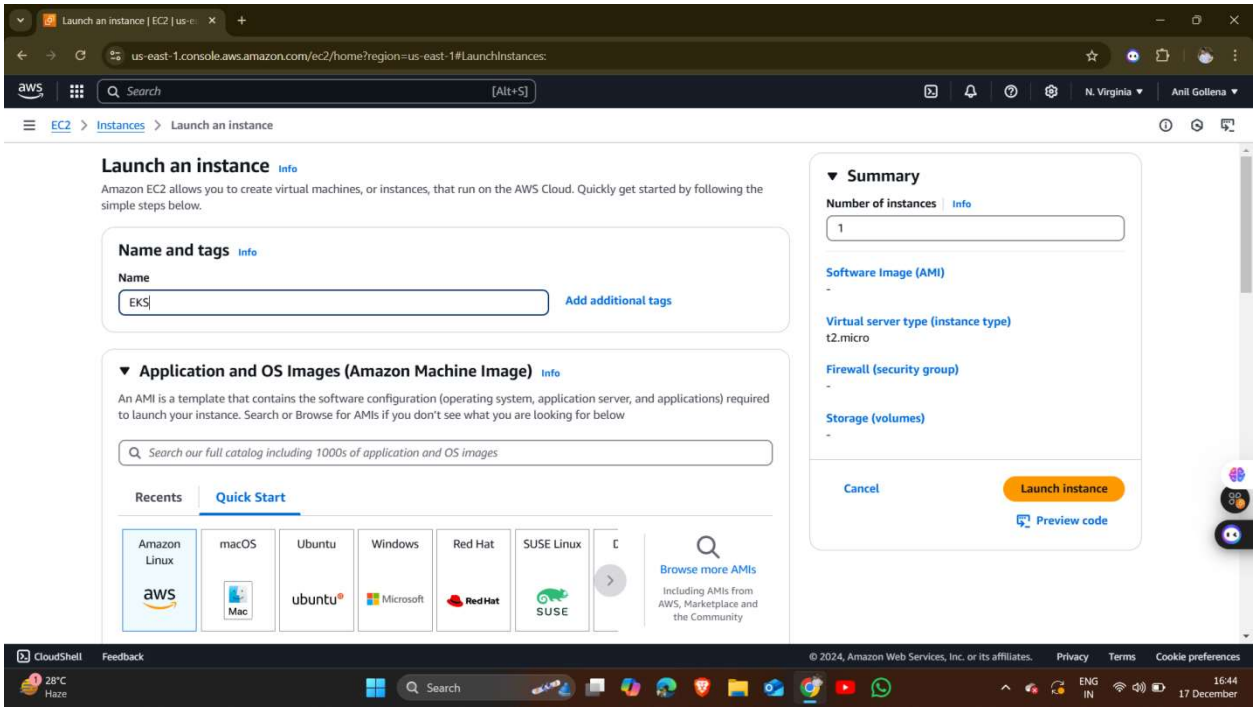
Now node group is in active state.



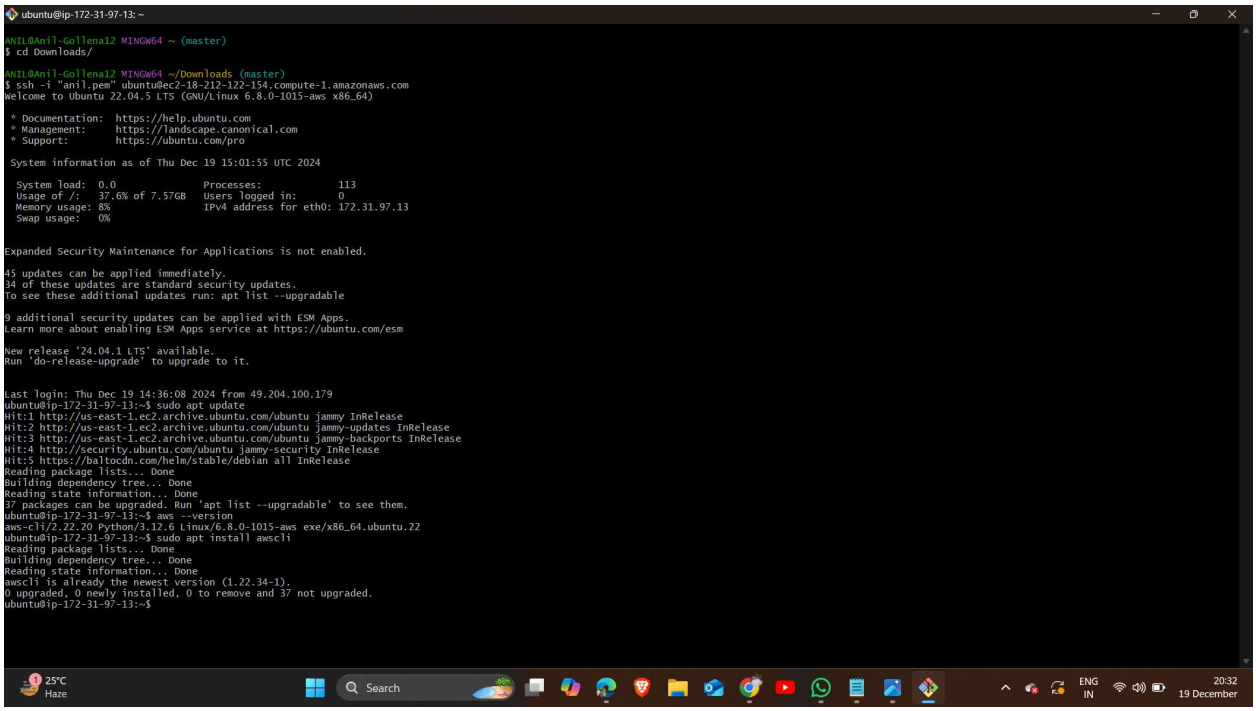
Here new two instance are created.



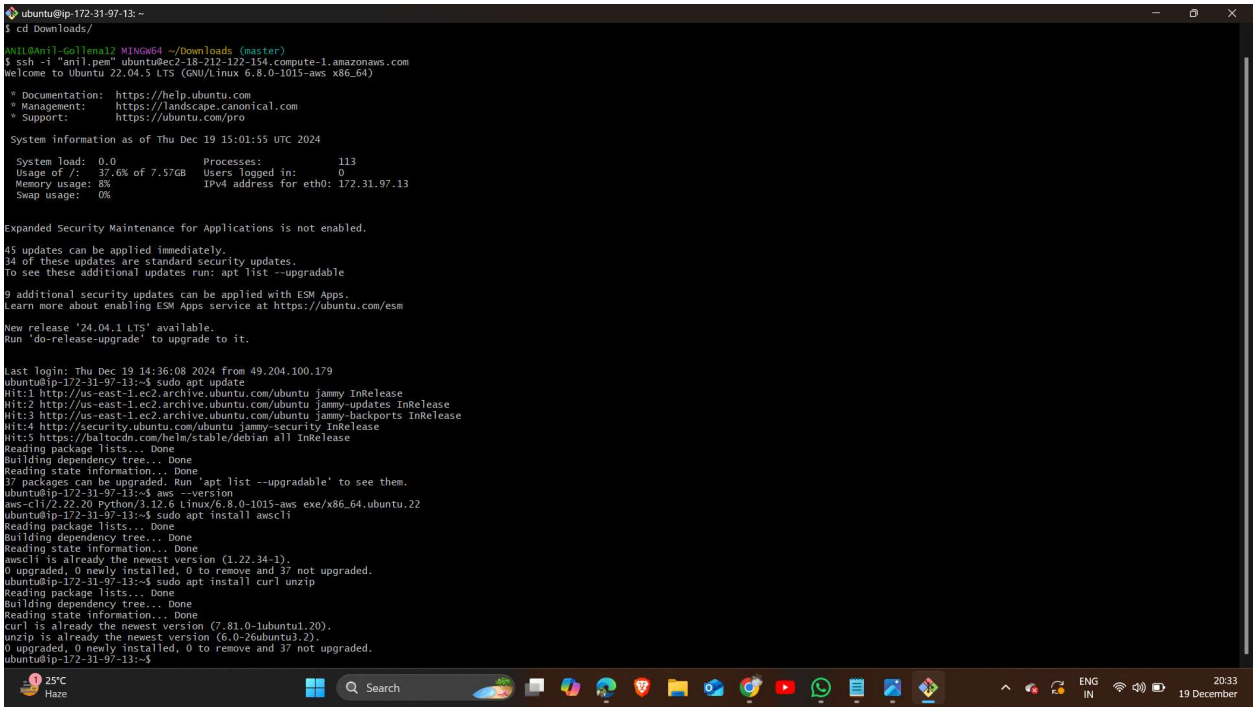
Now back to ec2 dashboard create a new instance



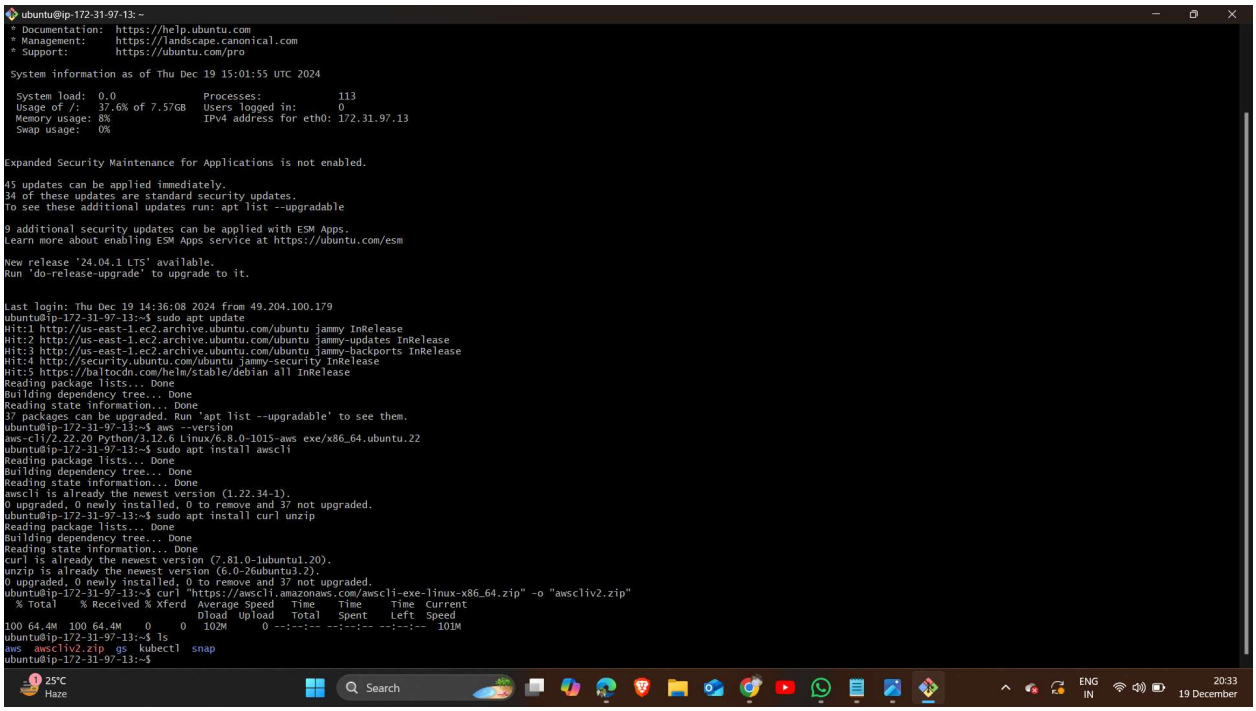
Now connect it with the server



And update the ubuntu install updated CLI

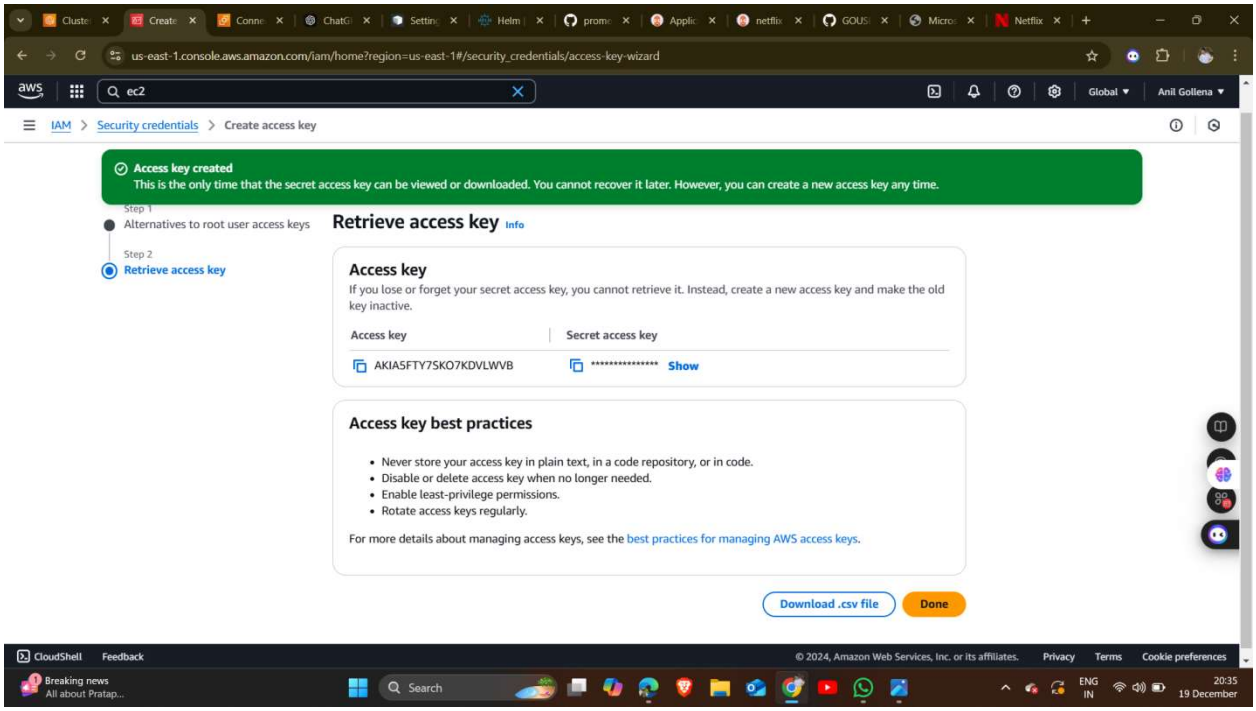


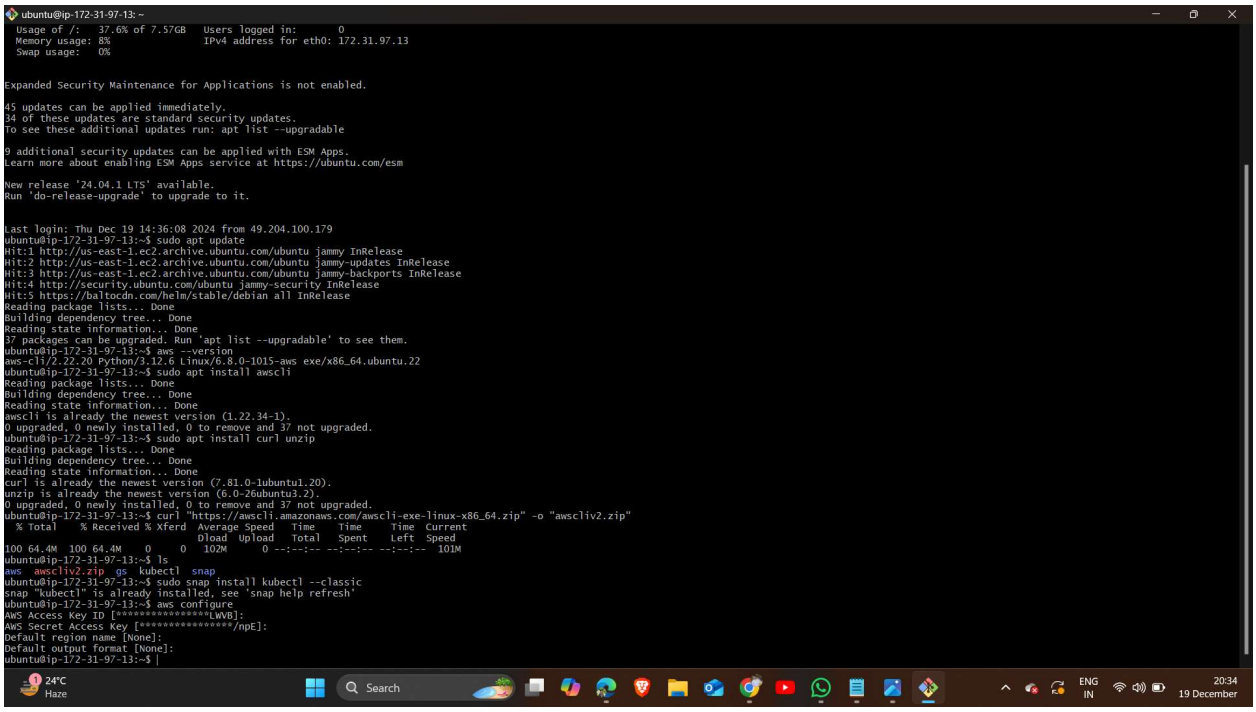
Now install kubectl unzip the files

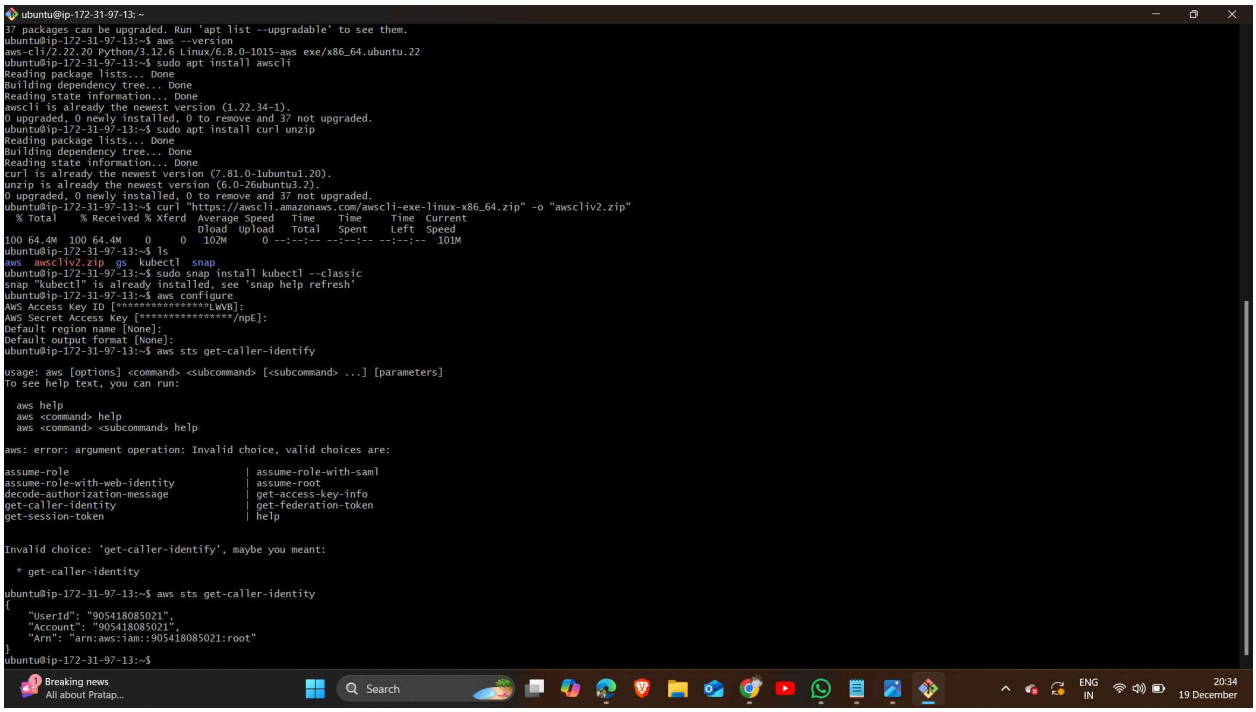


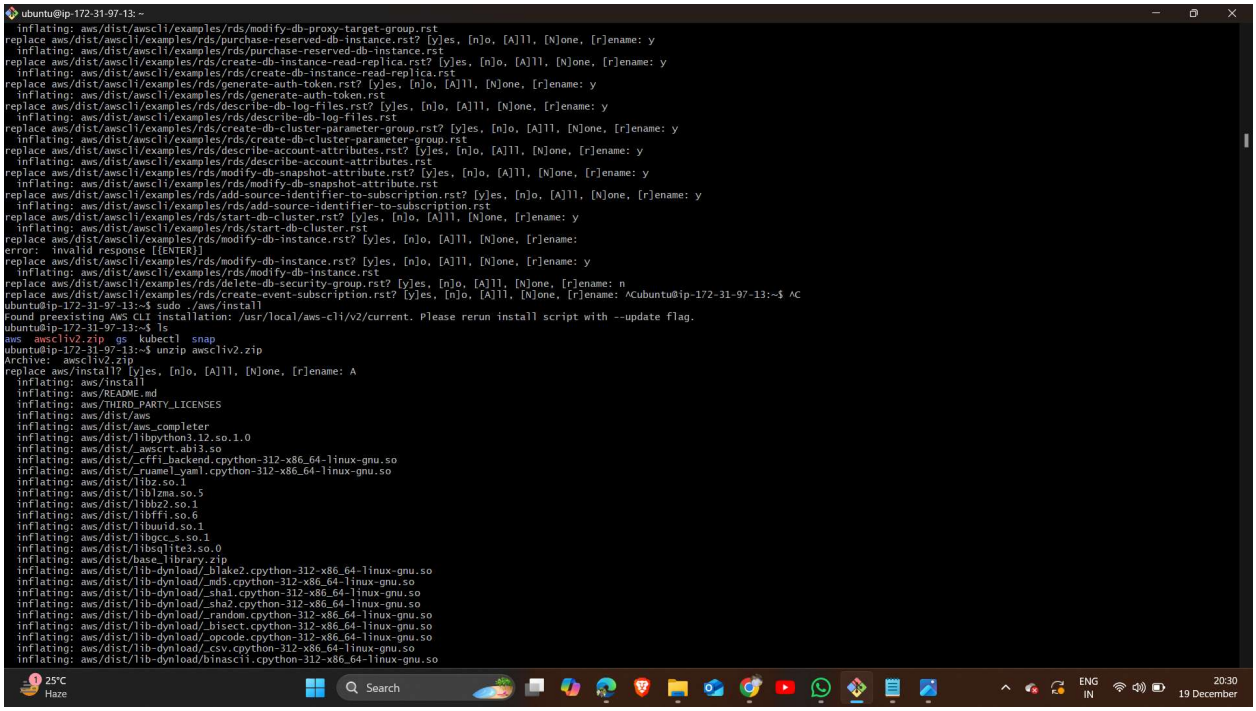
Now connect it with the aws management console by using aws configure

Add secret key and access key





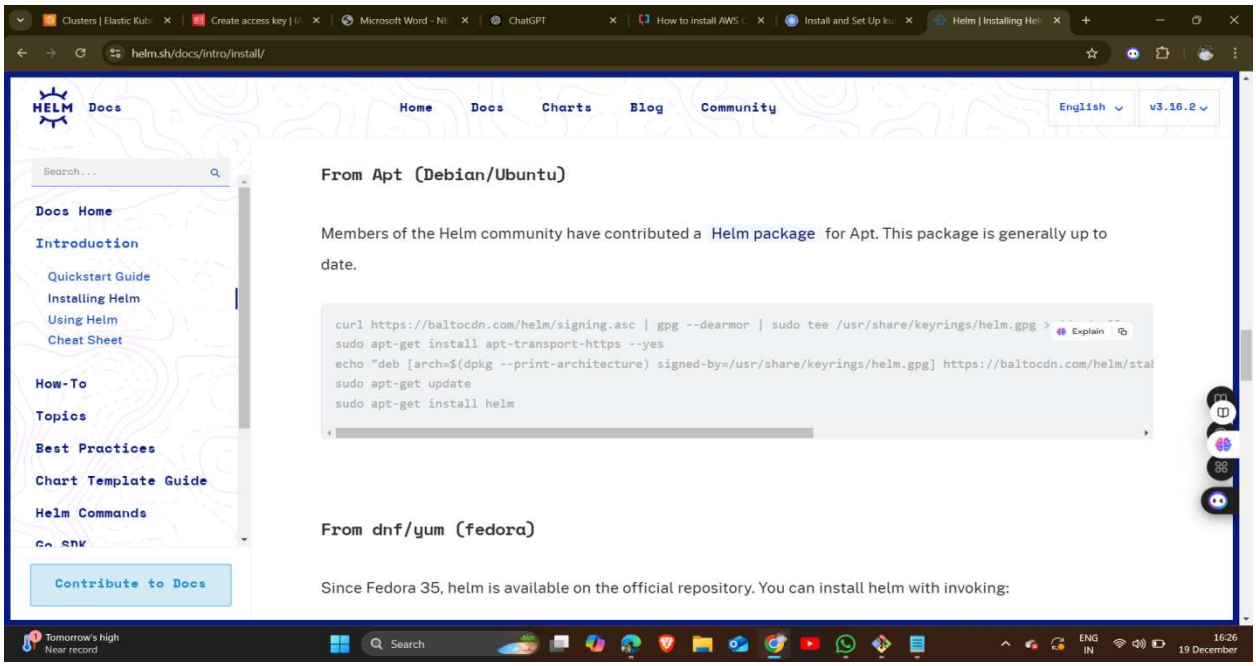

ubuntu@ip-172-31-97-13: ~
37 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-97-13:~$ aws --version
aws-cli/2.22.20 Python/3.12.6 Linux/6.8.0-1015-aws exe/x86_64.ubuntu.22
ubuntu@ip-172-31-97-13:~$ sudo apt install awscli
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
awscli is already the newest version (1.22.34-1).
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.
ubuntu@ip-172-31-97-13:~$ sudo apt install curl unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (7.81.0-1ubuntu1.20).
unzip is already the newest version (6.0-26ubuntu3.2).
0 upgraded, 0 newly installed, 0 to remove and 37 not upgraded.
ubuntu@ip-172-31-97-13:~$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
100 64.4M 100 64.4M 0 0 102M 0 --:--:-- --:--:-- --:--:-- 101M
ubuntu@ip-172-31-97-13:~$ ls
aws awscliv2.zip gs kubectl snap
ubuntu@ip-172-31-97-13:~$ sudo snap install kubectl --classic
snap "kubectl" is already installed, see 'snap help refresh'
ubuntu@ip-172-31-97-13:~$ aws configure
AWS Access Key ID [**********LWVB]:
AWS Secret Access Key [**********/npE]:
Default region name [None]:
Default output format [None]:
ubuntu@ip-172-31-97-13:~$ aws sts get-caller-identity
usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:
aws help
aws <command> help
aws <command> <subcommand> help
aws: error: argument operation: Invalid choice, valid choices are:
assume-role | assume-role-with-saml
assume-role-with-web-identity | assume-root
decode-authorization-message | get-access-key-info
get-caller-identity | get-federation-token
get-session-token | help
Invalid choice: 'get-caller-identify', maybe you meant:
* get-caller-identity
ubuntu@ip-172-31-97-13:~$ aws sts get-caller-identity
{
"UserId": "905418085021",
"Account": "905418085021",
"Arn": "arn:aws:iam::905418085021:root"
}
ubuntu@ip-172-31-97-13:~$
Breaking news
All about Pratap...
Search
ENG
IN
20:34
19 December


ubuntu@ip-172-31-97-13: ~
inflating: aws/dist/awscli/examples/rds/modify-db-proxy-target-group.rst
replace aws/dist/awscli/examples/rds/purchase-reserved-db-instance.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/purchase-reserved-db-instance.rst
replace aws/dist/awscli/examples/rds/create-db-instance-read-replica.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/create-db-instance-read-replica.rst
replace aws/dist/awscli/examples/rds/generate-auth-token.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/generate-auth-token.rst
replace aws/dist/awscli/examples/rds/describe-db-log-files.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/describe-db-log-files.rst
replace aws/dist/awscli/examples/rds/create-db-cluster-parameter-group.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/create-db-cluster-parameter-group.rst
replace aws/dist/awscli/examples/rds/describe-account-attributes.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/describe-account-attributes.rst
replace aws/dist/awscli/examples/rds/modify-db-snapshot-attribute.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/modify-db-snapshot-attribute.rst
replace aws/dist/awscli/examples/rds/add-source-identifier-to-subscription.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/add-source-identifier-to-subscription.rst
replace aws/dist/awscli/examples/rds/start-db-cluster.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/start-db-cluster.rst
replace aws/dist/awscli/examples/rds/modify-db-instance.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename:
error: invalid response [{ENTER}]
replace aws/dist/awscli/examples/rds/modify-db-instance.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
inflating: aws/dist/awscli/examples/rds/modify-db-instance.rst
replace aws/dist/awscli/examples/rds/delete-db-security-group.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace aws/dist/awscli/examples/rds/create-event-subscription.rst? [y]es, [n]o, [A]ll, [N]one, [r]ename: ^Cubuntu@ip-172-31-97-13:~$ ^C
ubuntu@ip-172-31-97-13:~$ sudo ./aws/install
Found preexisting AWS CLI installation: /usr/local/aws-cli/v2/current. Please rerun install script with --update flag.
ubuntu@ip-172-31-97-13:~$ ls
aws awscliv2.zip gs kubectl snap
ubuntu@ip-172-31-97-13:~$ unzip awscliv2.zip
Archive: awscliv2.zip
replace aws/install? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
inflating: aws/install
inflating: aws/README.md
inflating: aws/THIRD_PARTY_LICENSES
inflating: aws/dist/aws
inflating: aws/dist/aws_completer
inflating: aws/dist/libpython3.12.so.1.0
inflating: aws/dist/_awscrt.abi3.so
inflating: aws/dist/_cffi_backend.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/_ruamel_yaml.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/libz.so.1
inflating: aws/dist/liblzma.so.5
inflating: aws/dist/libbz2.so.1
inflating: aws/dist/libffi.so.6
inflating: aws/dist/libuuid.so.1
inflating: aws/dist/libgcc_s.so.1
inflating: aws/dist/libsqlite3.so.0
inflating: aws/dist/base_library.zip
inflating: aws/dist/lib-dynload/_blake2.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/lib-dynload/_md5.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/lib-dynload/_sha1.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/lib-dynload/_sha2.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/lib-dynload/_random.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/lib-dynload/_bisect.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/lib-dynload/_opcode.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/lib-dynload/_csv.cpython-312-x86_64-linux-gnu.so
inflating: aws/dist/lib-dynload/binascii.cpython-312-x86_64-linux-gnu.so
25°C
Haze
Search
ENG
IN
20:30
19 December

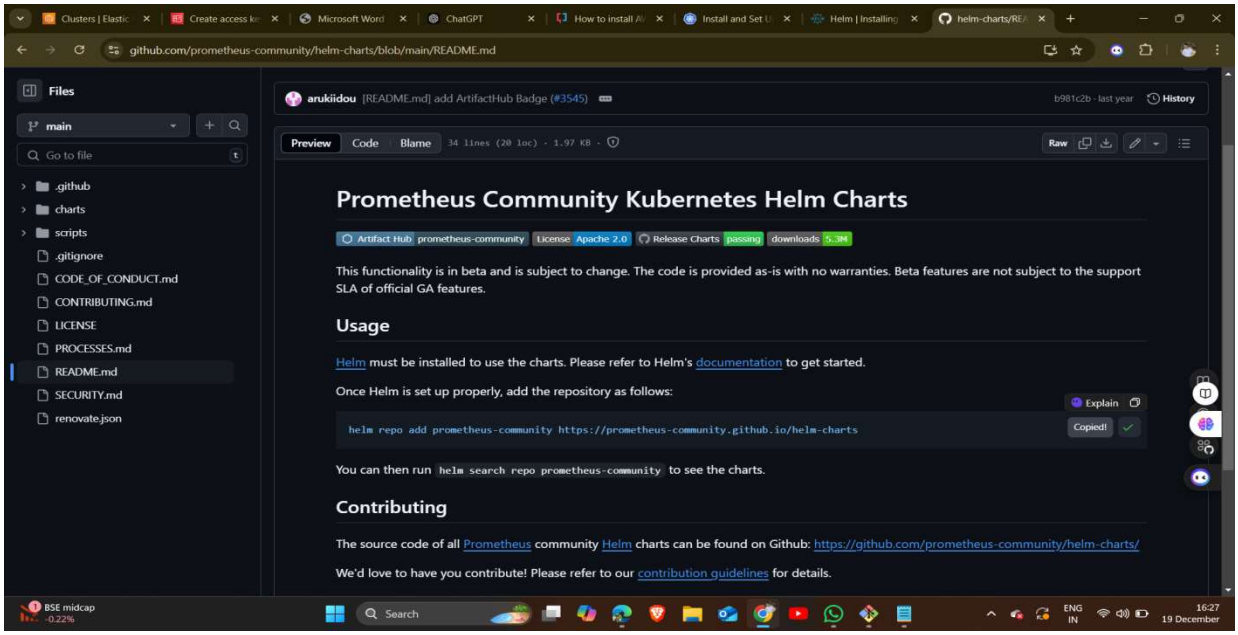
Now install HELM form Google.

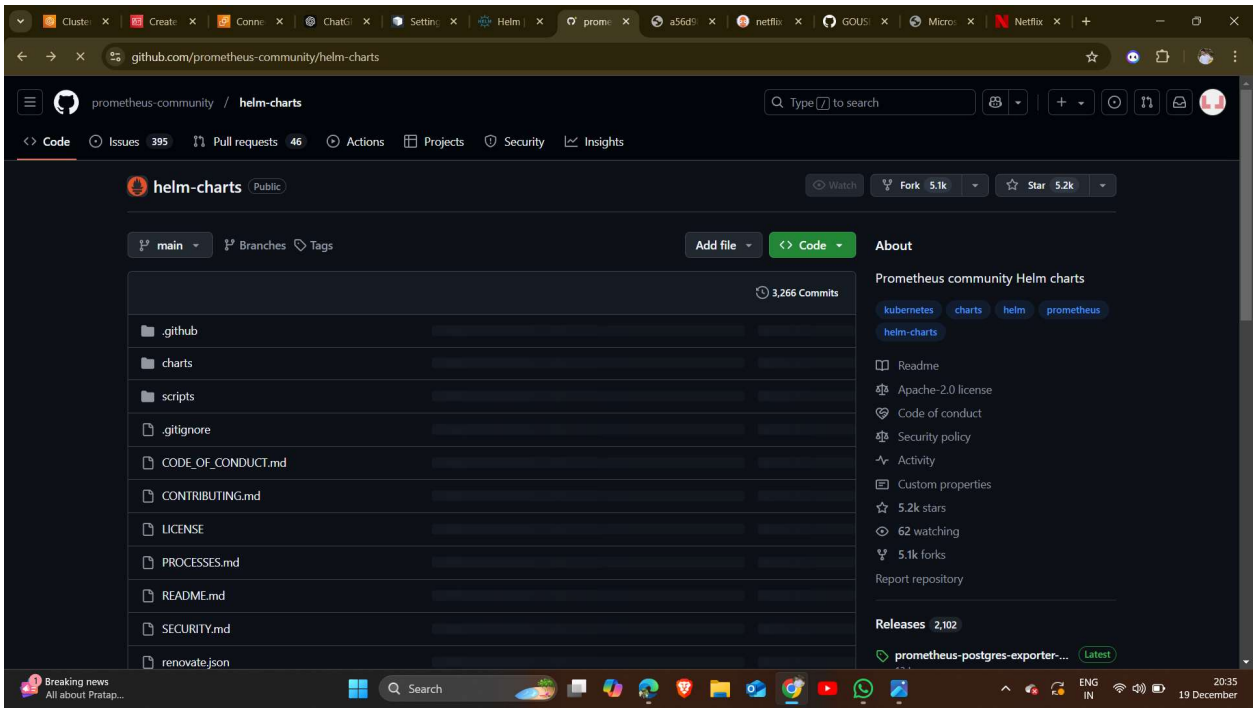
Helm is a package manager for Kubernetes that simplifies the process of deploying,managing, and configuring applications on Kubernetes clusters.

Think of it as a tool to automate Kubernetes deployments using pre-configured templates and reusable components called Helm Charts.

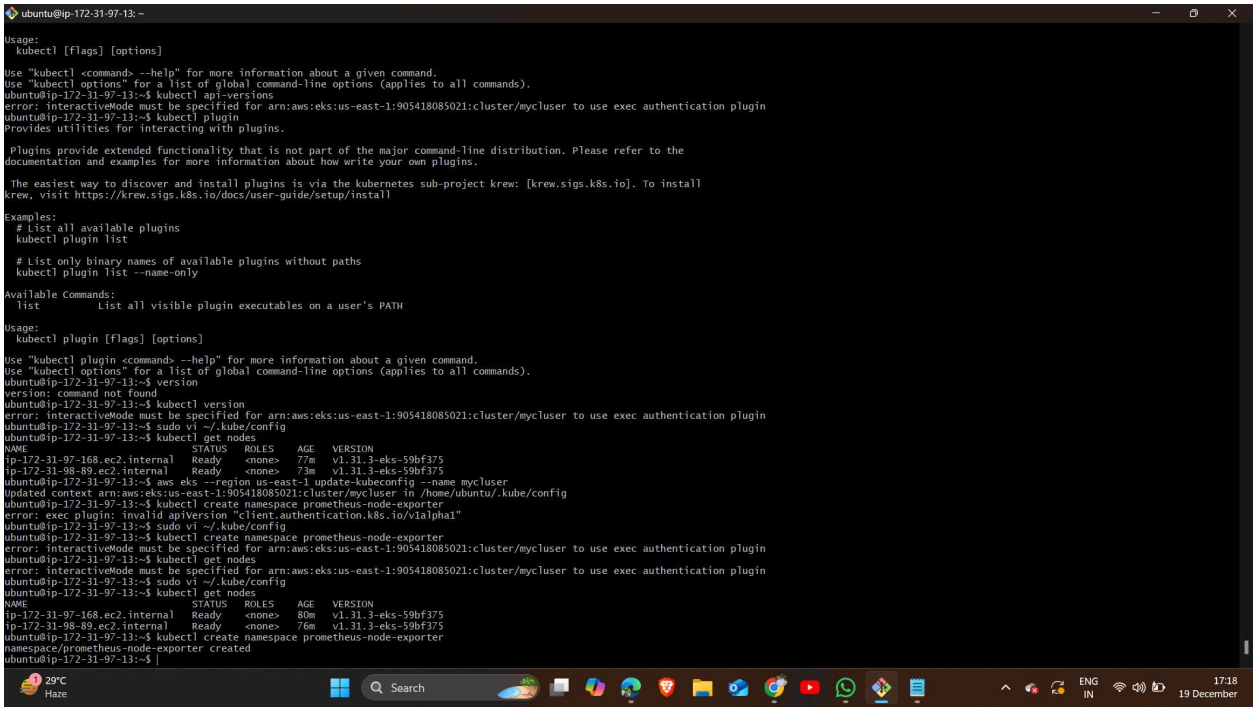


Now set up helm chart

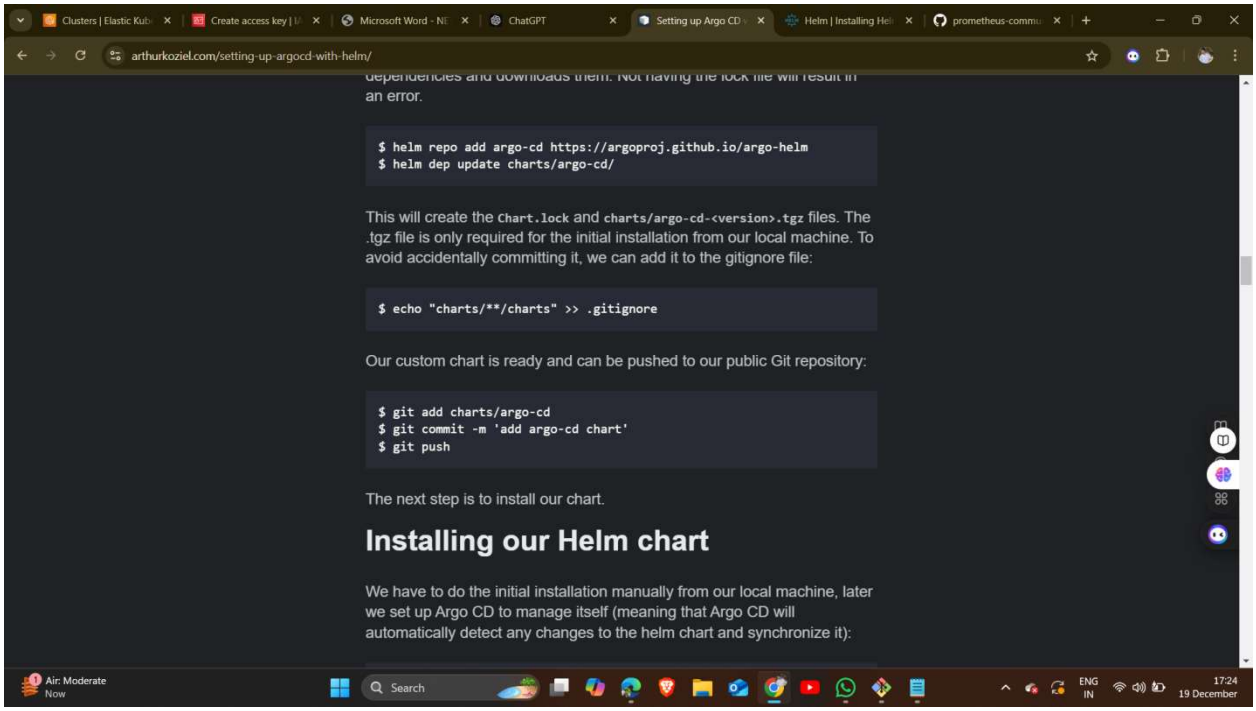


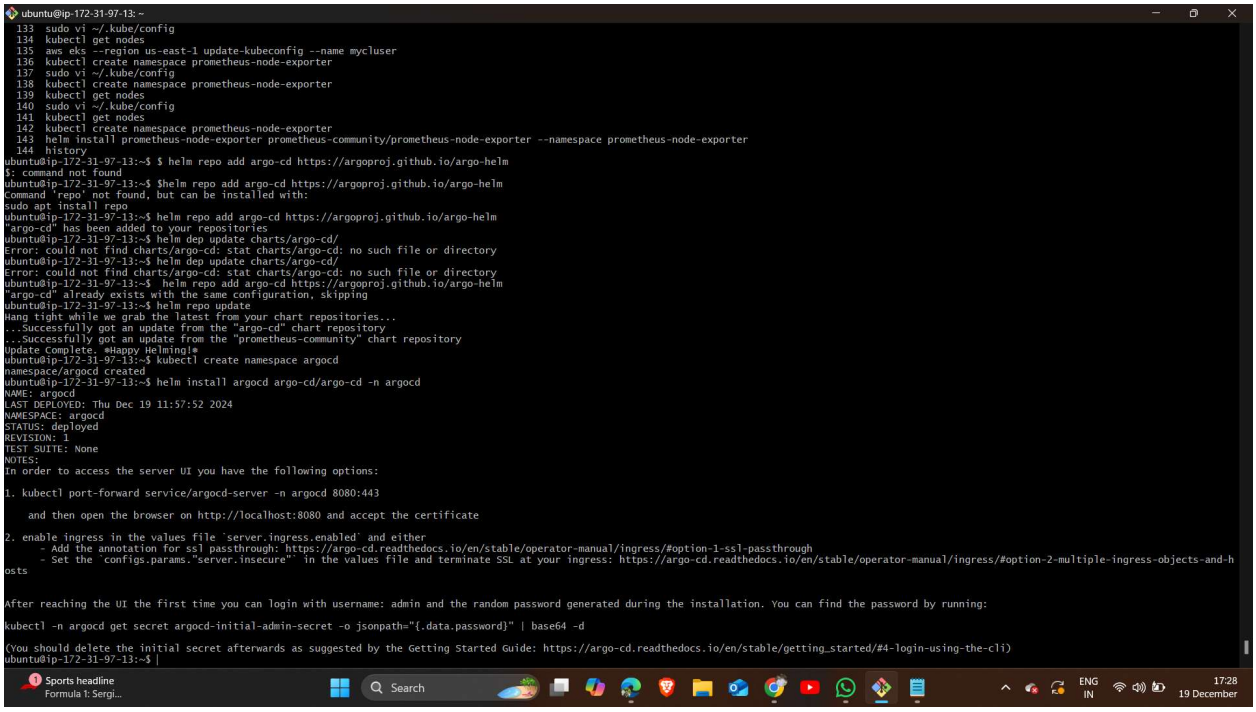


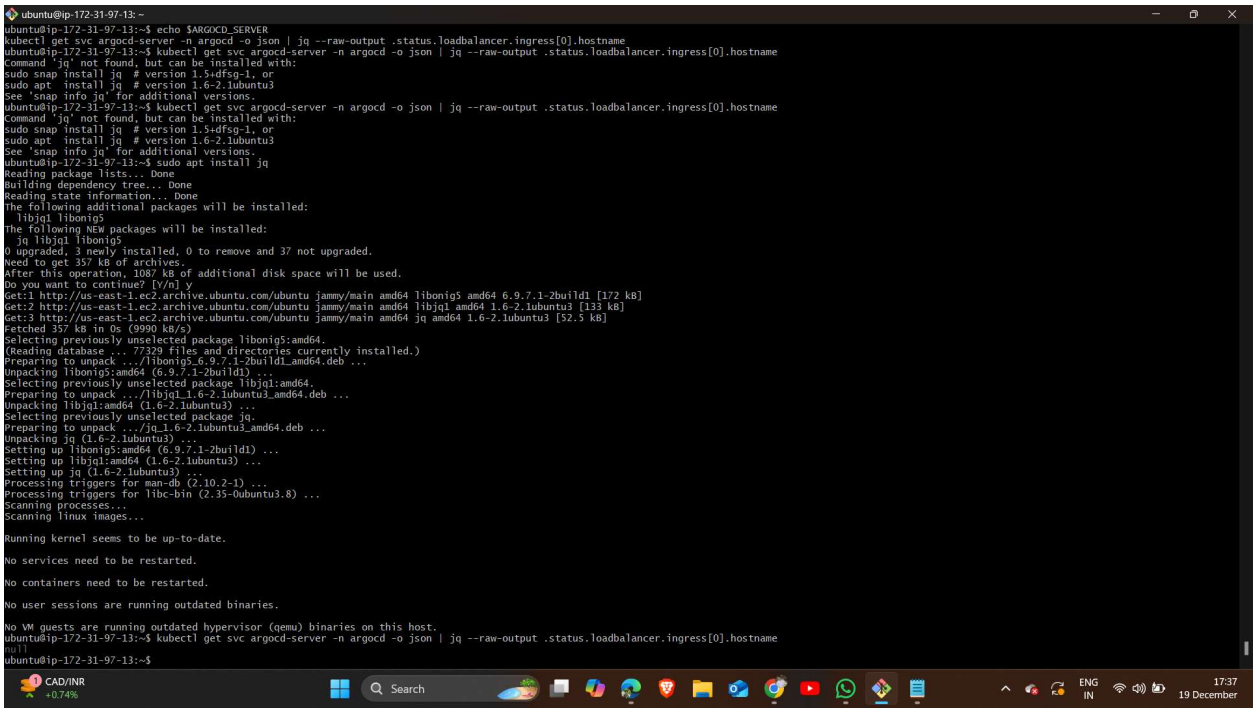
- Now Change API version to v1 then add interactiveMode: IfAvailable in config file.
- Then AWS configure with the access key and secret key.
- Now setup Prometheus node exporter namespace.



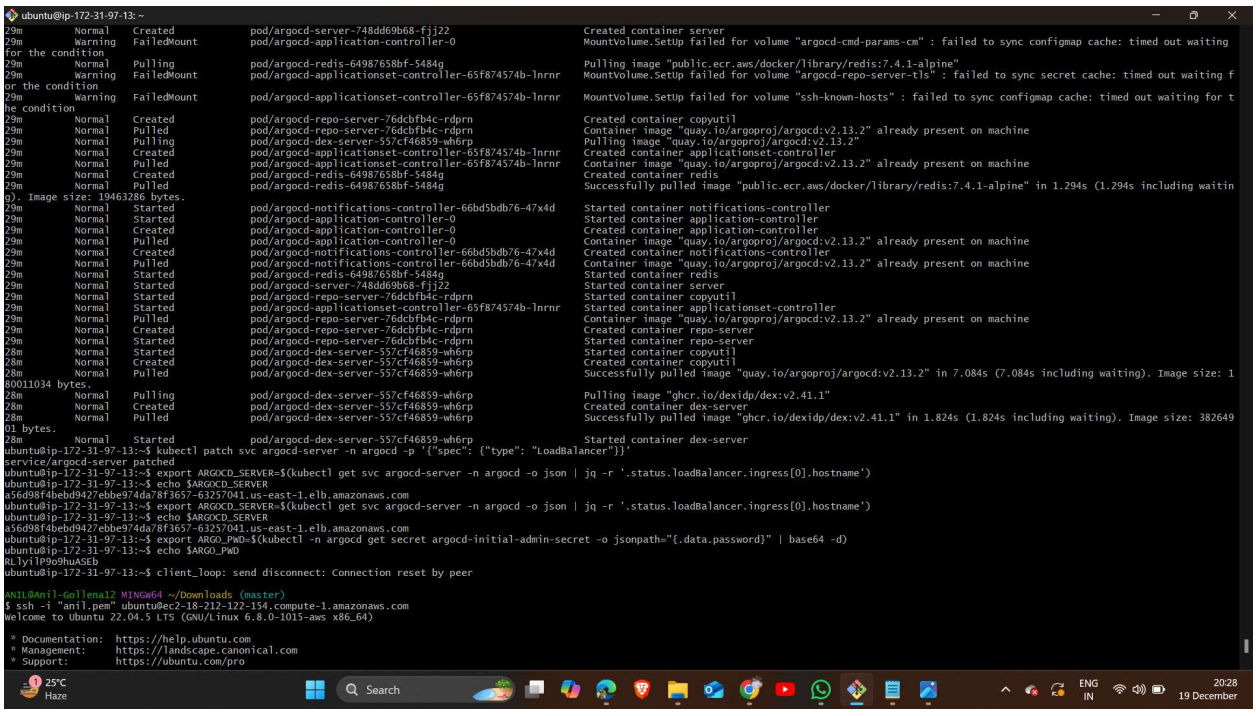
Now setup argoCD for that get commands form google



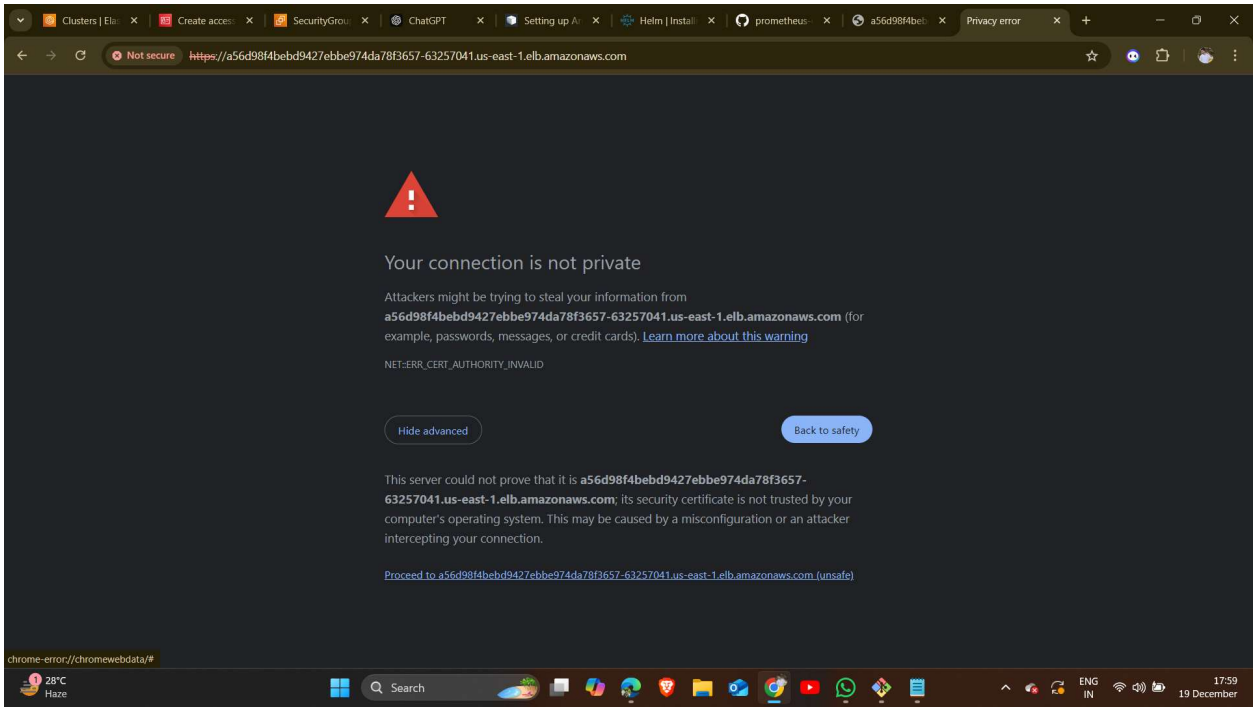


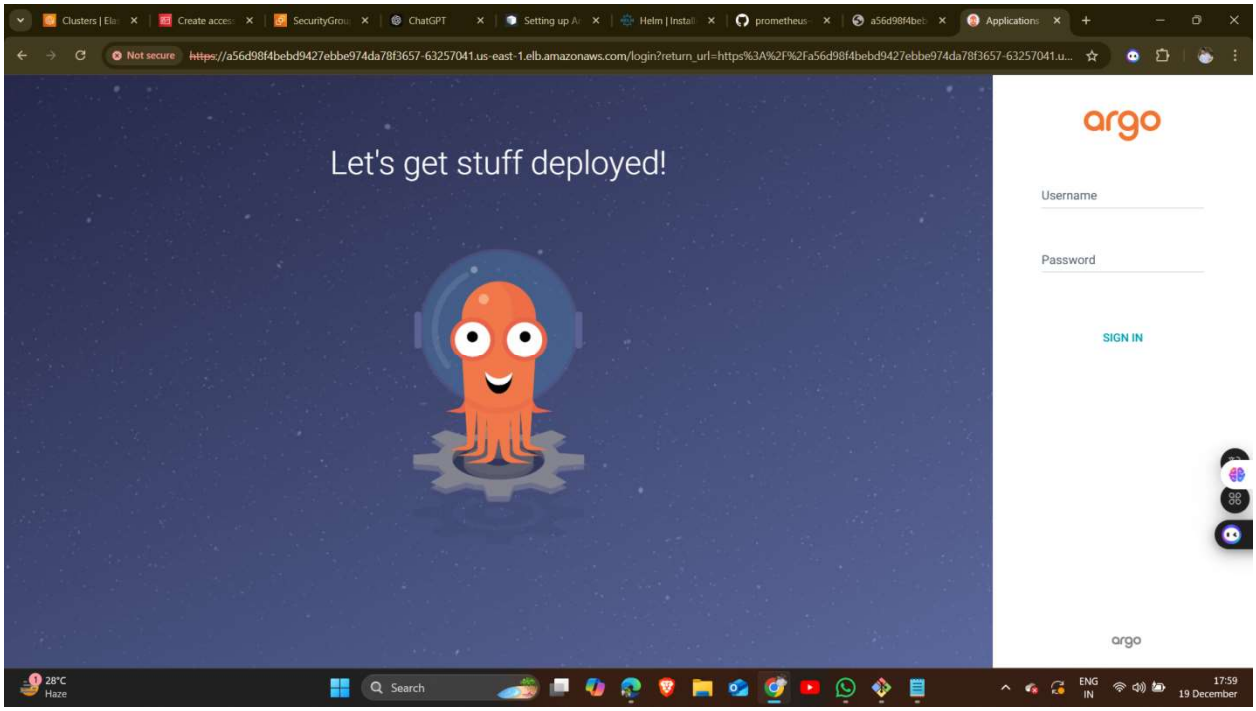


Here a new loadbalancer is created



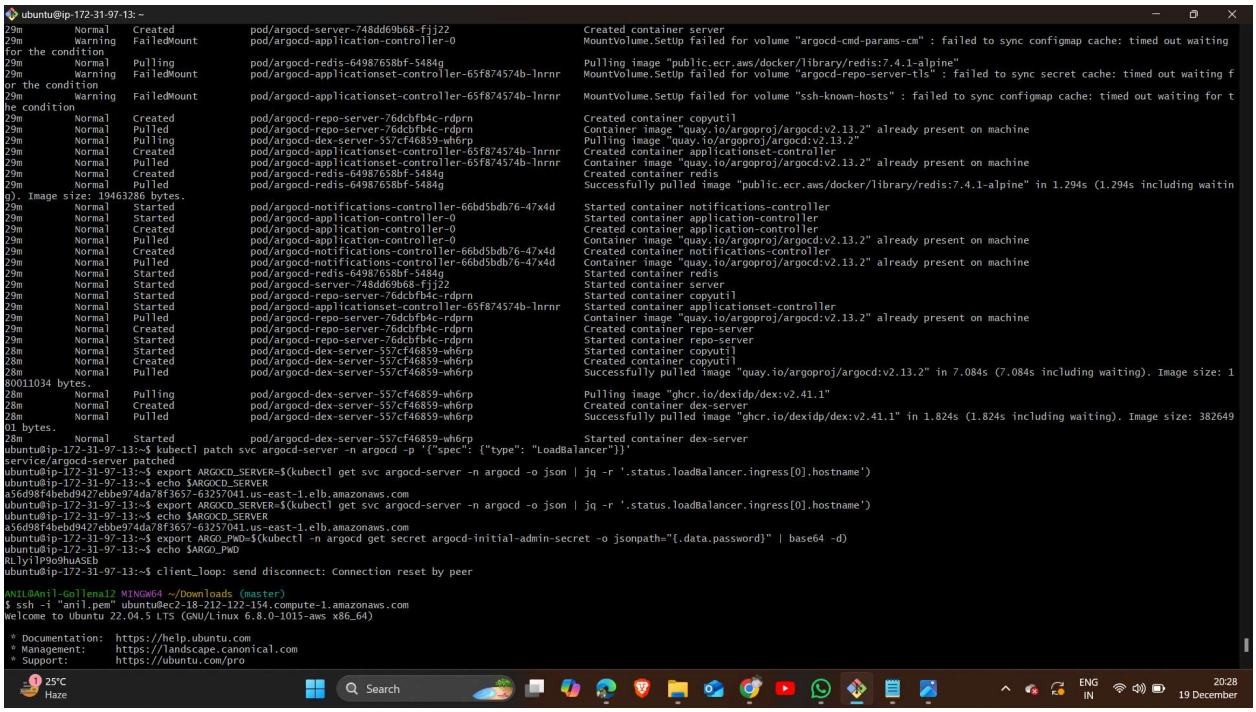
Copy the load balancer arn allow the port number



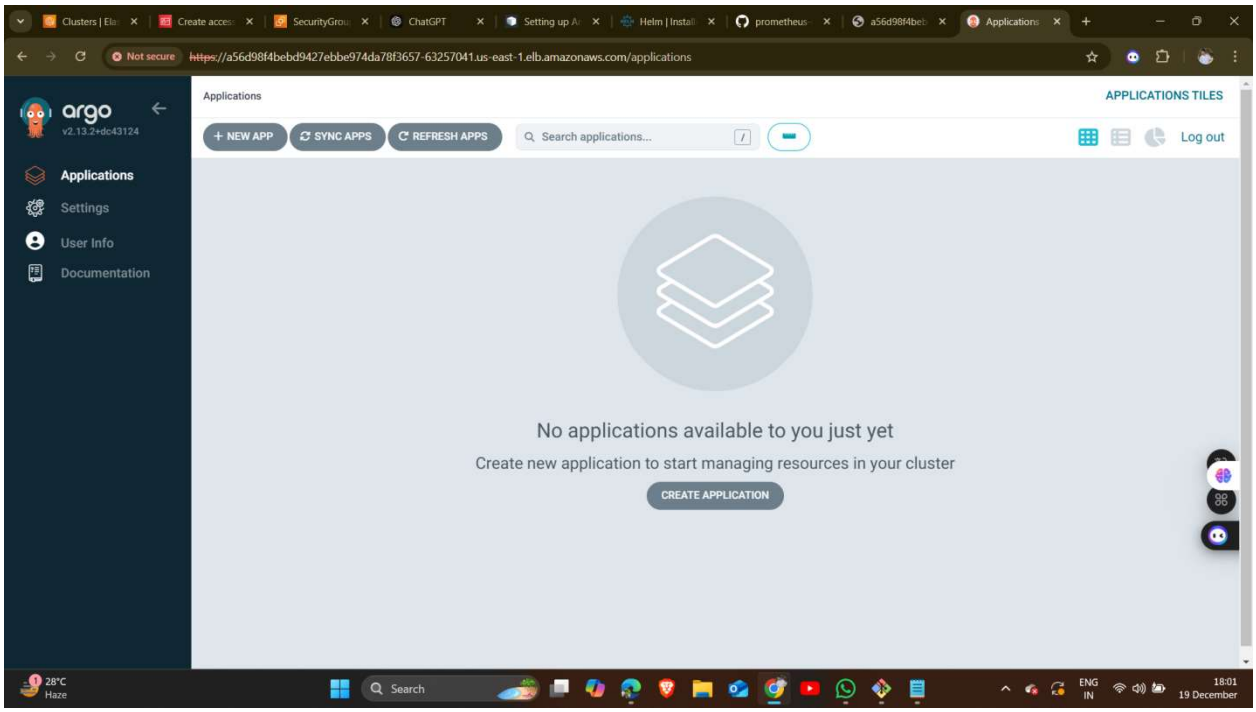


Here the user name is “admin” and for password run the command as

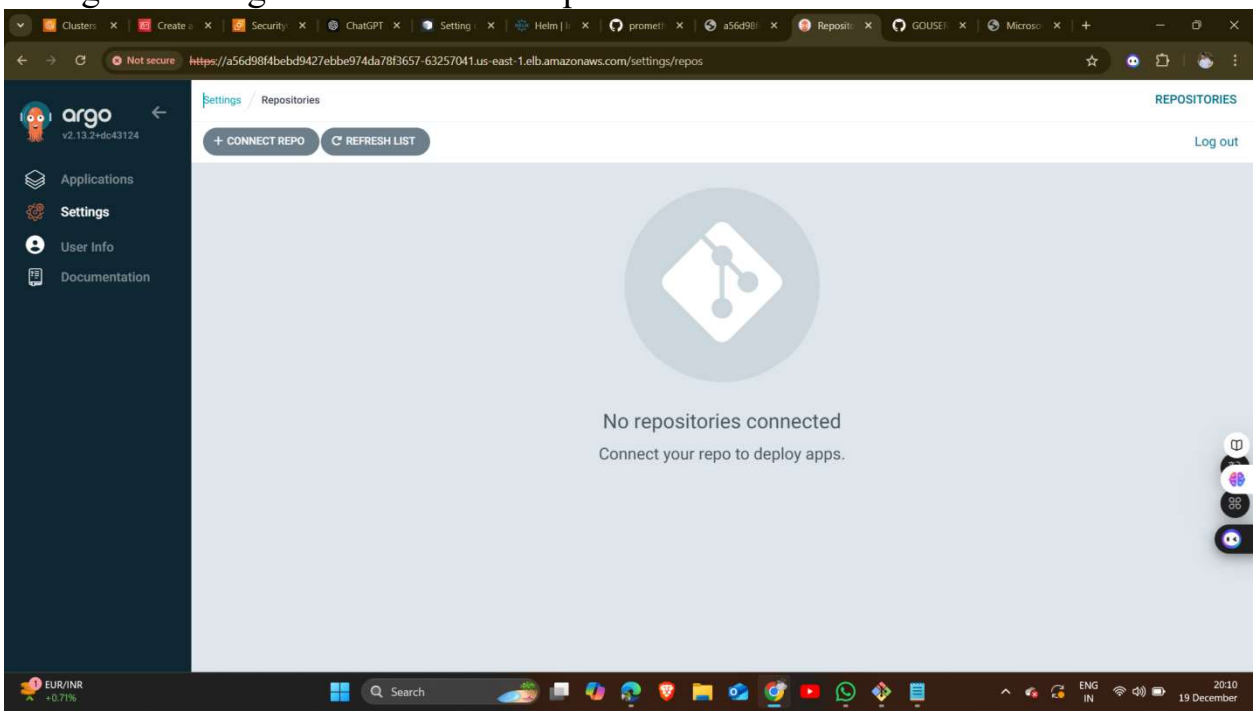
“echo $ARGO _PWD” there you get password copy that password login to the page.



Here are the commands I used to implement the whole process

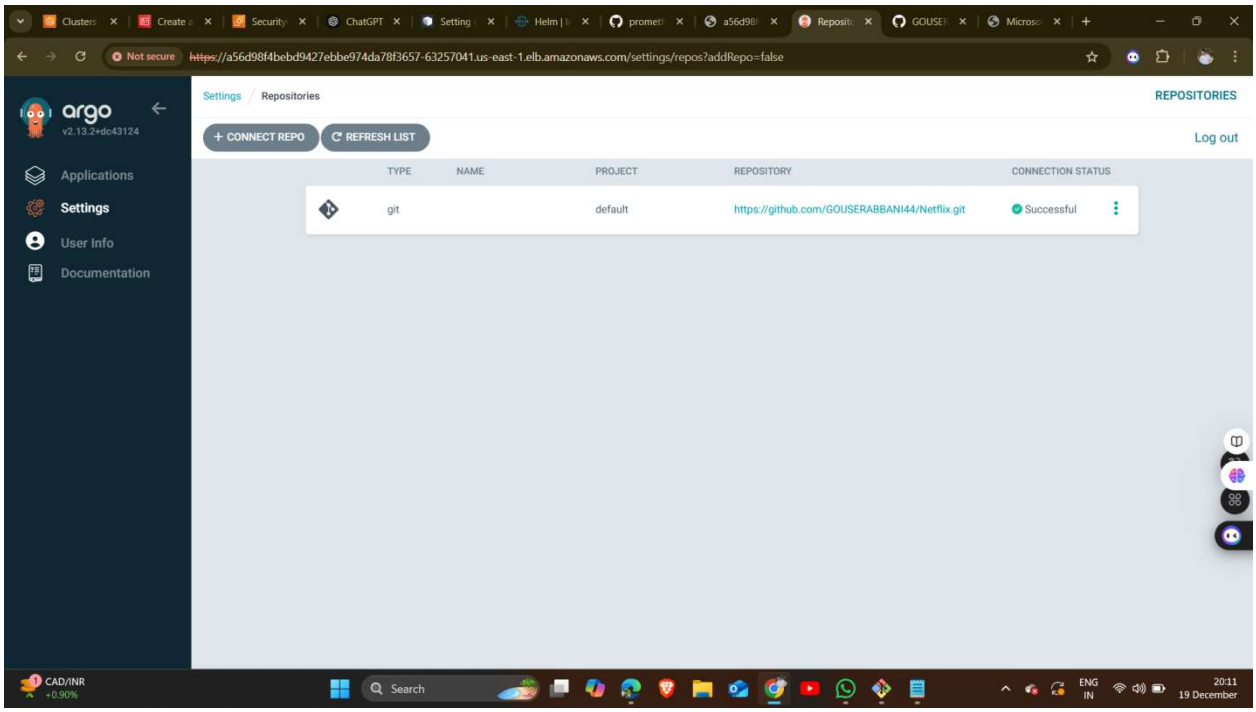


Now go to settings and connect the repo

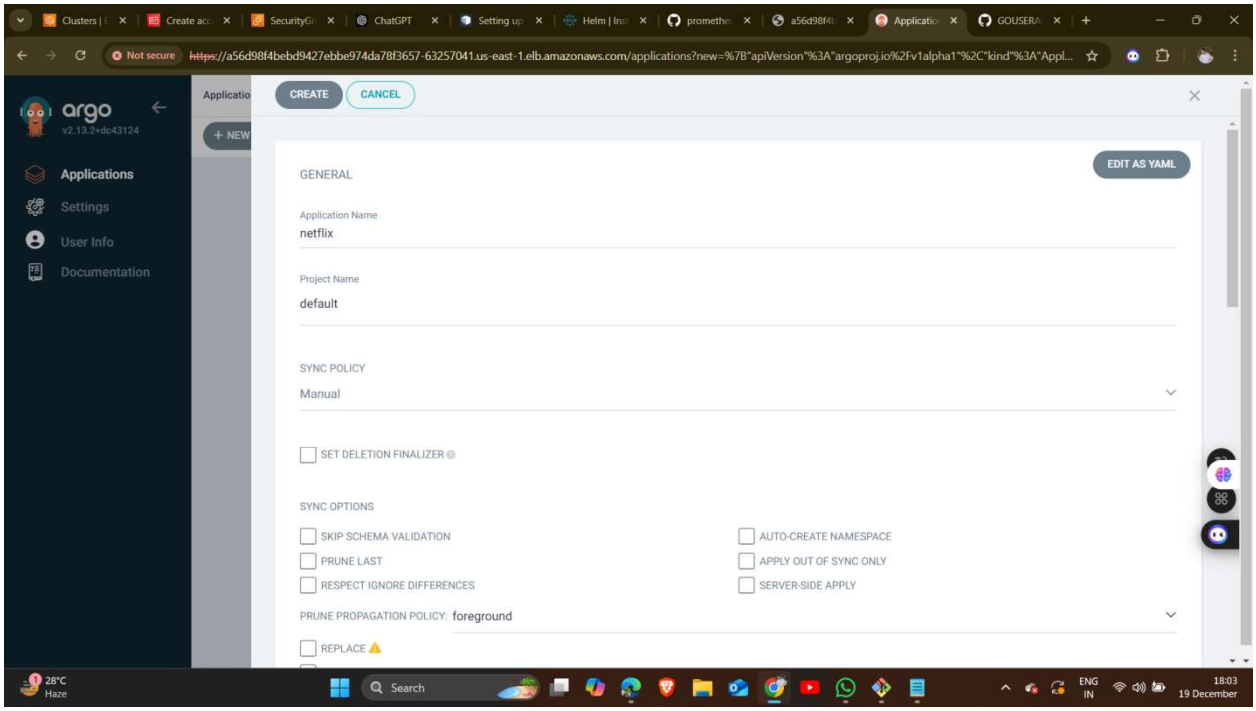


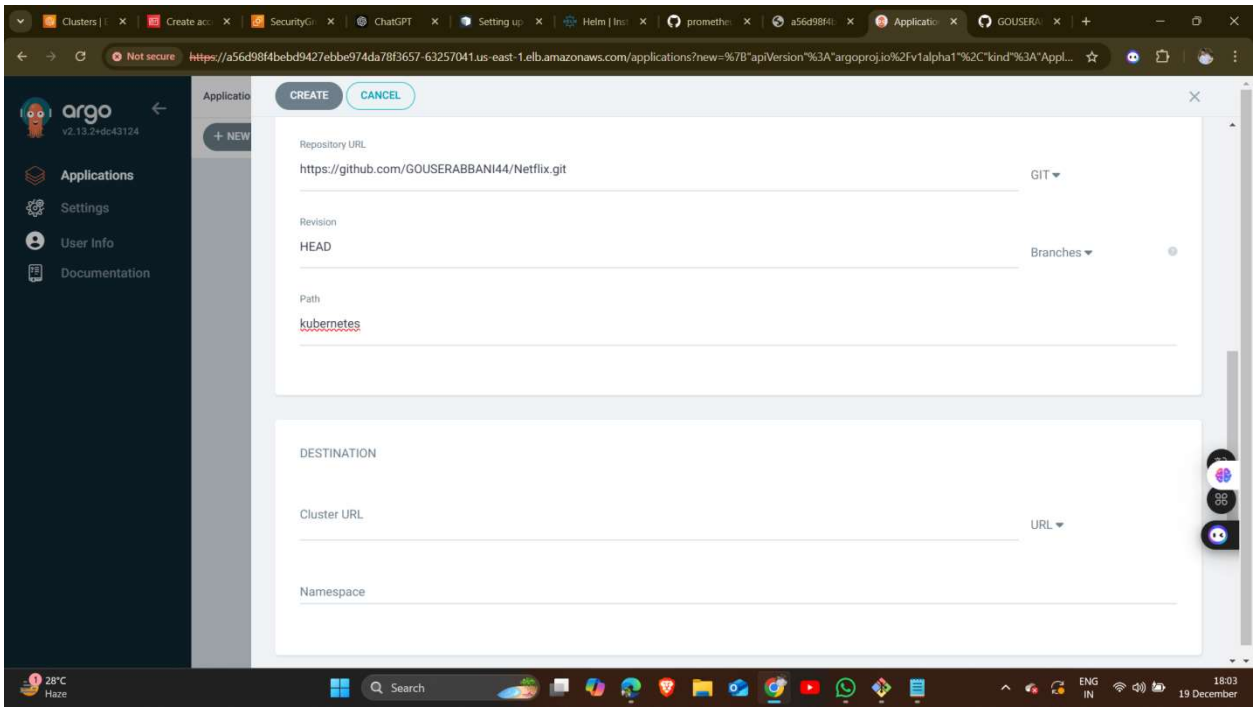
Click the connect repo

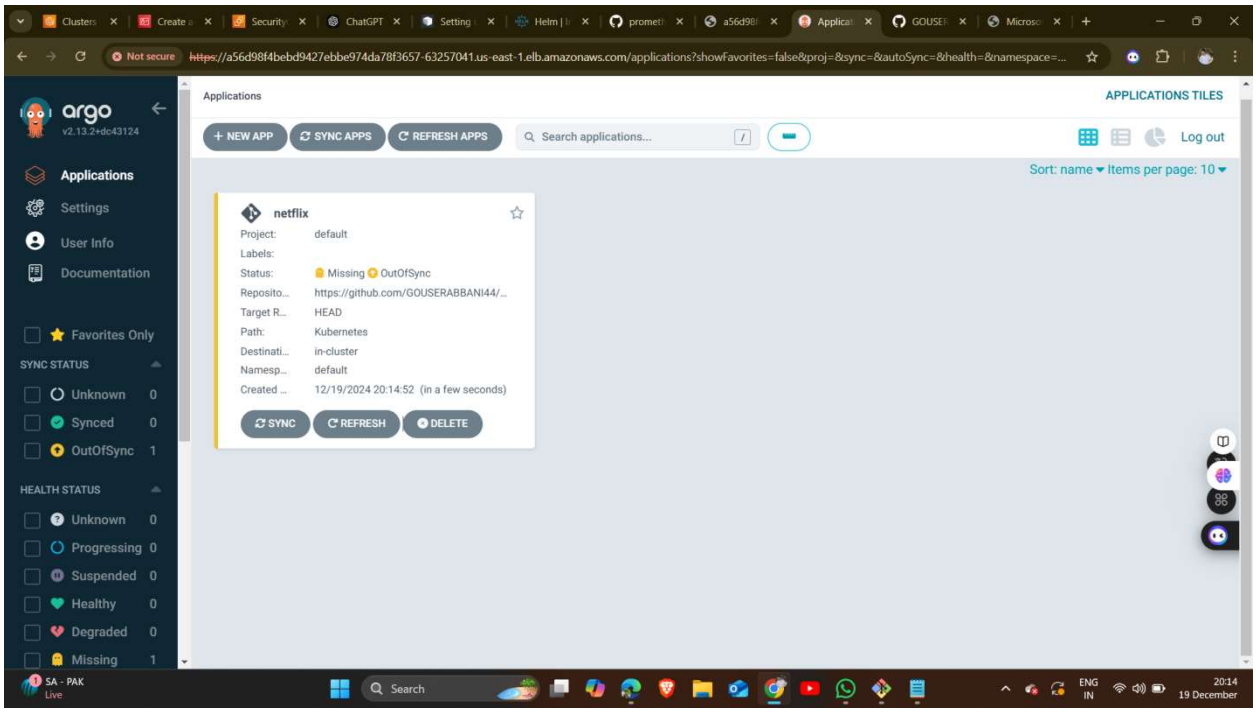
Now connect via HTTP, name as git and copy the Netflix application files under the git URL make project as default



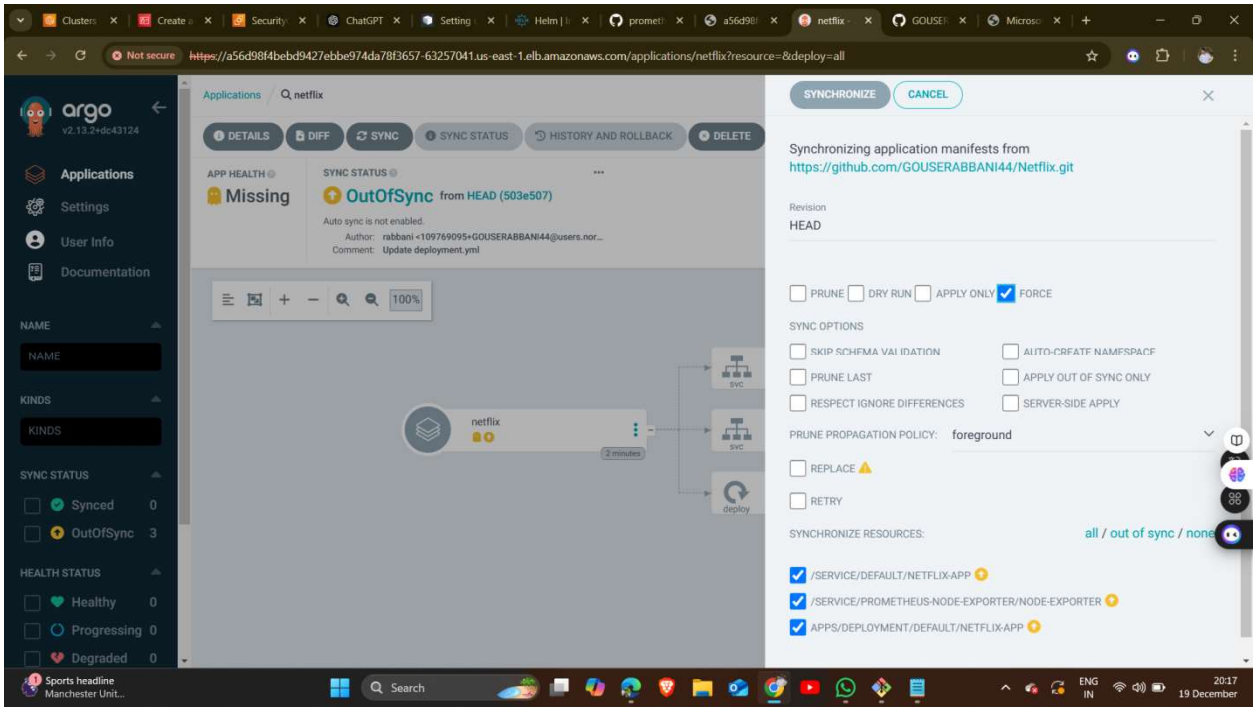
Now create application

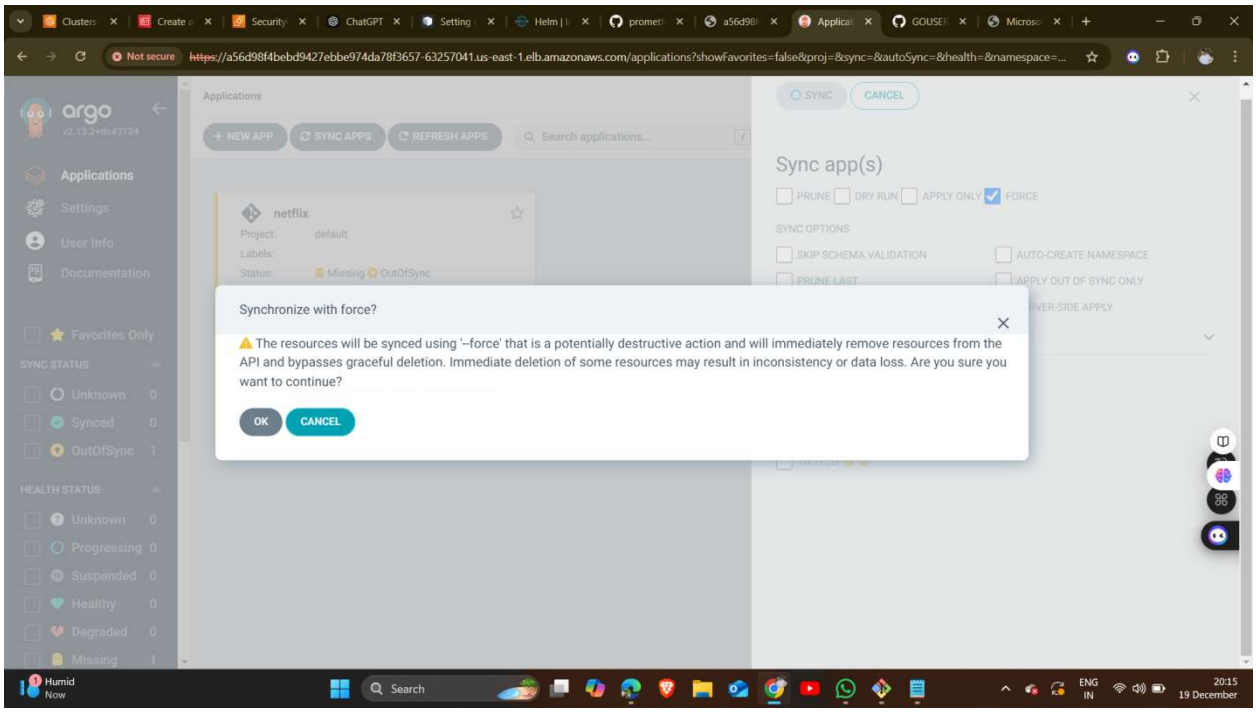




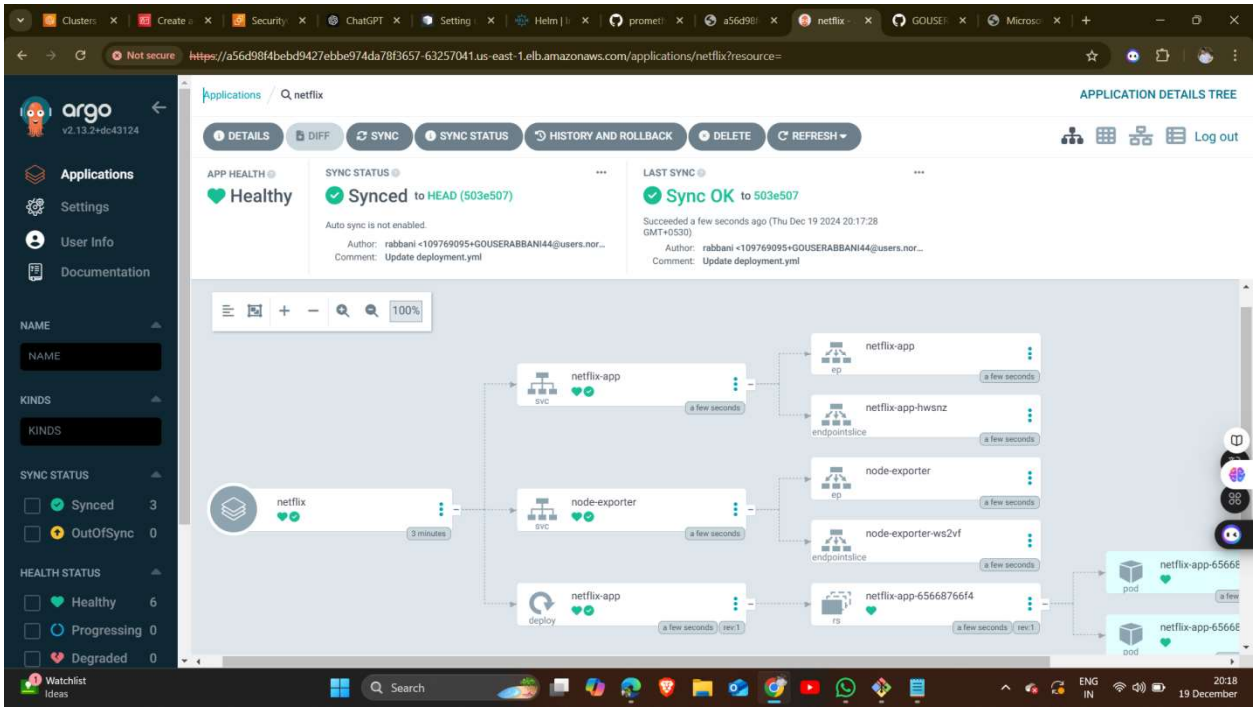


Now sync the application

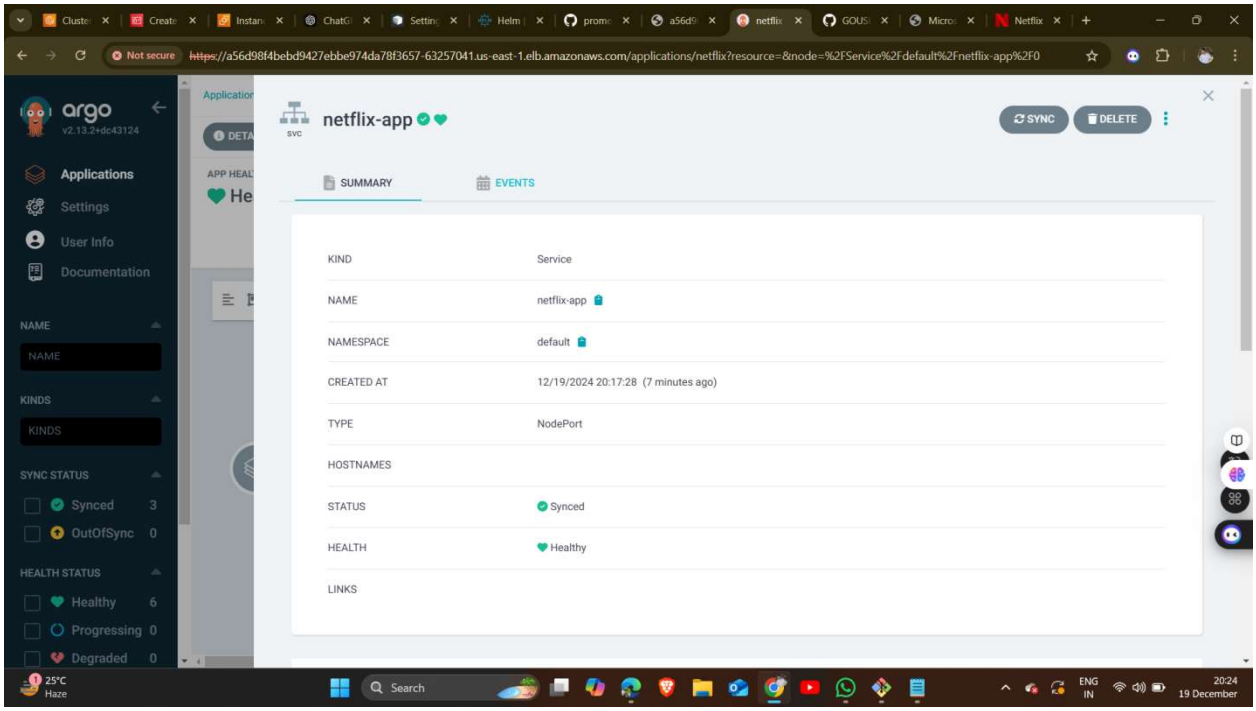



Synchronize with force?
The resources will be synced using '--force' that is a potentially destructive action and will immediately remove resources from the API and bypasses graceful deletion. Immediate deletion of some resources may result in inconsistency or data loss. Are you sure you want to continue?
OK
CANCEL
Sync app(s)
SYNC
CANCEL

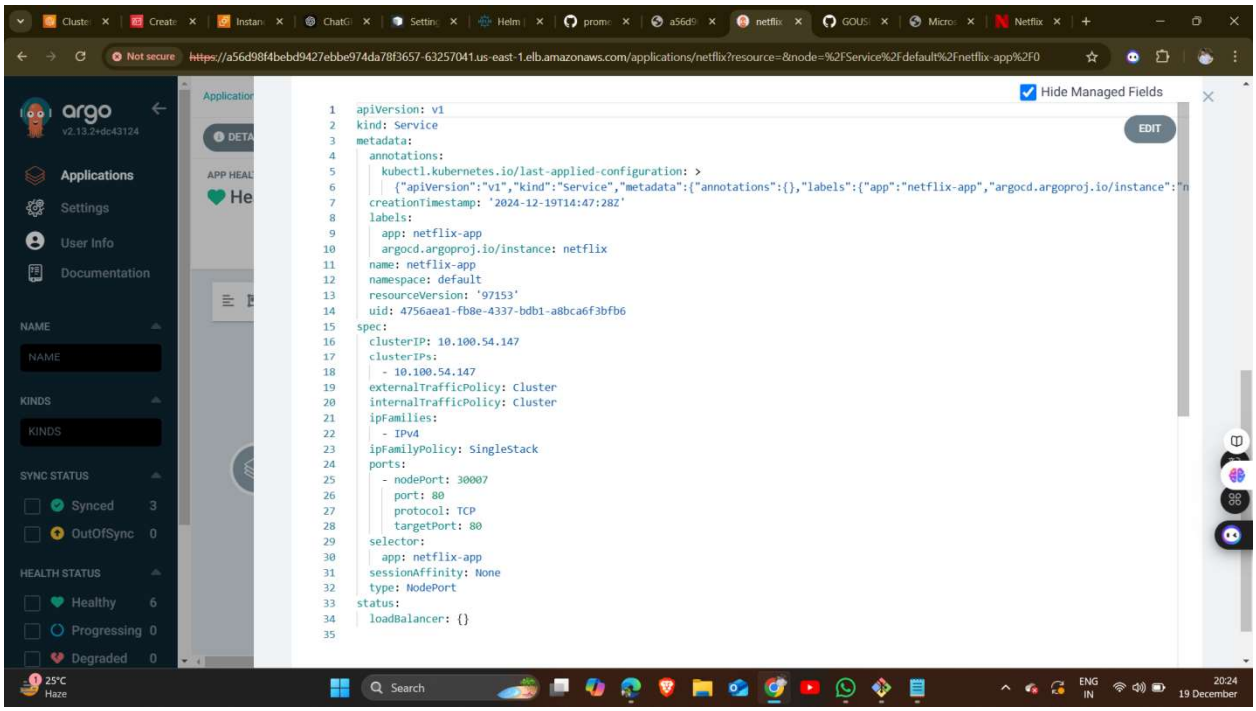
Click on ok


APPLICATION DETAILS TREE
DETAILS
DIFF
SYNC
SYNC STATUS
HISTORY AND ROLLBACK
DELETE
REFRESH
Log out
Healthy
Synced to HEAD (503e507)
Sync OK to 503e507
netflix
netflix-app
netflix-app-hwsnz
node-exporter
node-exporter-ws2vf
netflix-app-65668766f4

Here is the summary of the application



Here is the deployment file.



Now copy the public IP along with the port number as 30007