

Deploy Netflix Clone on

Kubernetes

Project link:

<https://medium.com/@aakibkhan1/project-4-deploy-netflix-clone-on-kubernetes-9ae6091b88bd>

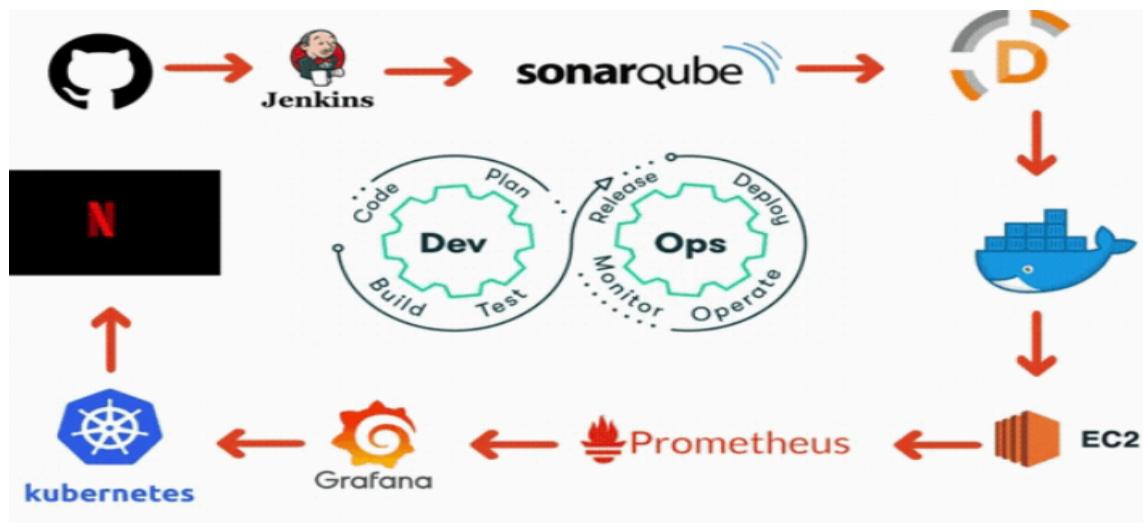
Phase 1 → deploy netflix locally on ec2 (t2.large)

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

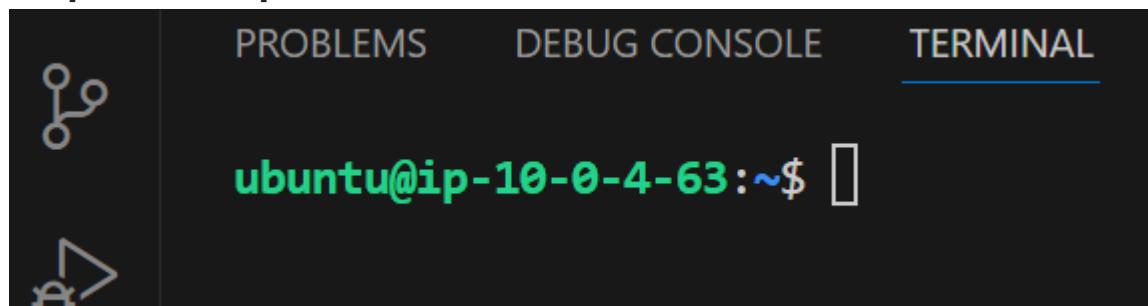
Phase 5 → Kubernetes →



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

1. go to aws console and launch ubuntu 22.04 with t2.large and 25 gb of storage allocated with it don't forget to enable public ip in vpc settings

Step 1 → setup ec2



3. run the following commands

a. sudo su

b. apt update

clone the github repo by

c. git clone

<https://github.com/Aakibgithuber/Deploy-Network-Clone-on-Kubernetes>

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

	<input type="checkbox"/> eksctl-EKS-1-cluster/NATIP	3.213.83.216	Public IP	eipalloc-03d5c01c1ce2bcd20
	<input checked="" type="checkbox"/> eksctl-netflix-cluster-original-cluster/N...	52.70.13.52	Public IP	eipalloc-0b30e2559729f968c

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 .**

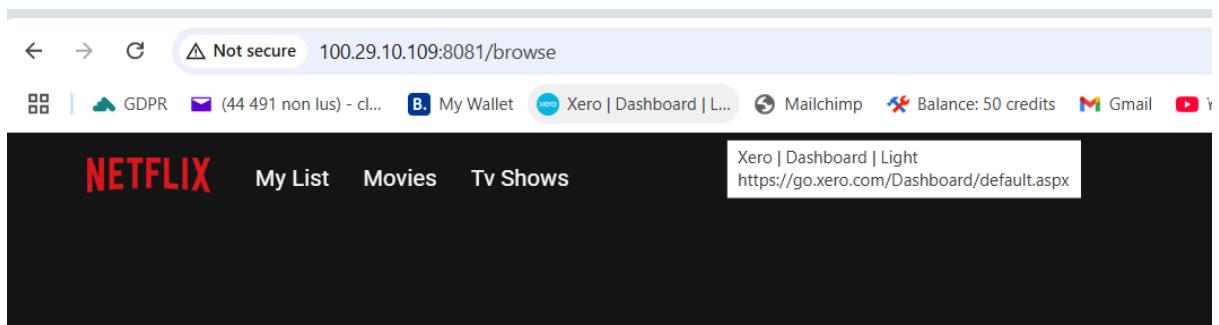
b. **docker run -d -- name netflix -p 8081:80 netflix:latest** → this maps the container port to your ec2 port

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

Edit inbound rules						
Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Info
sgr-05d564bbd7ce3c1a4	HTTP	TCP	80	Custom		<input type="text" value="0.0.0.0"/> <input type="button" value="X"/>
sgr-0ff22b11876bd6a05	Custom TCP	TCP	8081	Custom		<input type="text" value="0.0.0.0"/> <input type="button" value="X"/>
sgr-0c5156a5dfabc1691	Custom TCP	TCP	9000	Custom		<input type="text" value="0.0.0.0"/> <input type="button" value="X"/>
sgr-0b6f96b7ac2c2fe19	Custom TCP	TCP	9100	Custom		<input type="text" value="0.0.0.0"/> <input type="button" value="X"/>
sgr-07c8939cf6962a670	Custom TCP	TCP	8080	Custom		<input type="text" value="0.0.0.0"/> <input type="button" value="X"/>

ubuntu@ip-10-0-4-63:~\$ docker images					
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE	
guillou73/netflix	latest	aa8923ae55c6	5 days ago	52.5MB	
netflix	latest	aa8923ae55c6	5 days ago	52.5MB	

your application is running go to your ec2 copy public ip and browse http://your_public_ip:8081 and it's open like this



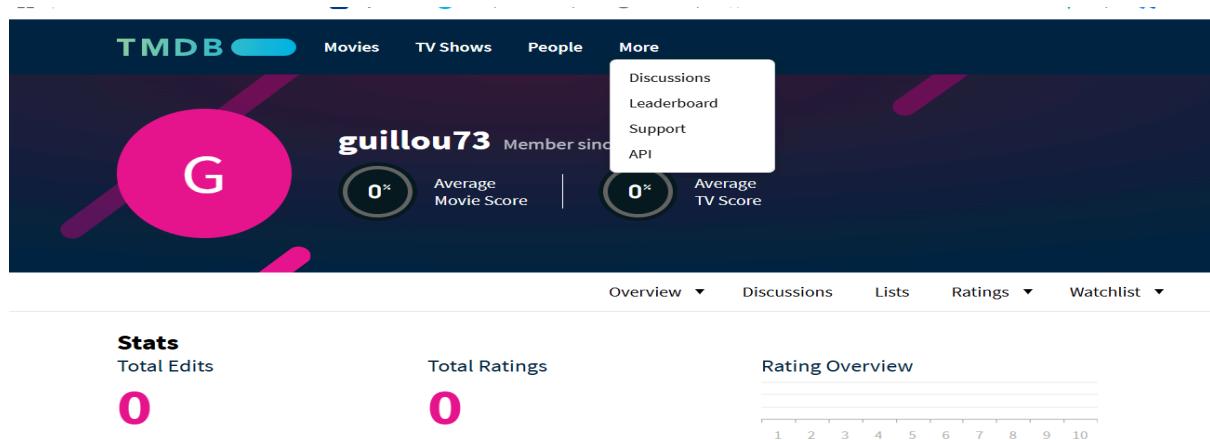
Its shows a blank page of netflix because you don't have a api that's communicate with netflix database let's solve this

what is an API→

An API (Application Programming Interface) is like a menu for a restaurant that lets you order food (data or functions) from a software system or application, allowing different programs to talk to each other and share information in a structured way.

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



The screenshot shows the TMDB API Settings page. On the left, there's a sidebar with options like Edit Profile, Account Settings, Streaming Services, Notification Settings, Blocked Users, Import List, Sharing Settings, Sessions, API (which is selected), and Delete Account. The main content area has tabs for API, Documentation, Support, API Details, API Read Access Token, and API Key. The API Key tab is active, displaying a long API key: eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI2NzUwZDhiOWE1YzBiNzFmMjBmNzkzM2QlI1MTJjZjliMjkxNTRlNzMzCIsInNjb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uIjcsIjIwMjAxMSIsImtpZCI6ImRvbmV0aGUtZW1haWwuc2VydmljZS5jb20ifQ==. Below the key, there's a 'copy the api key' button.

Now delete the existing image by

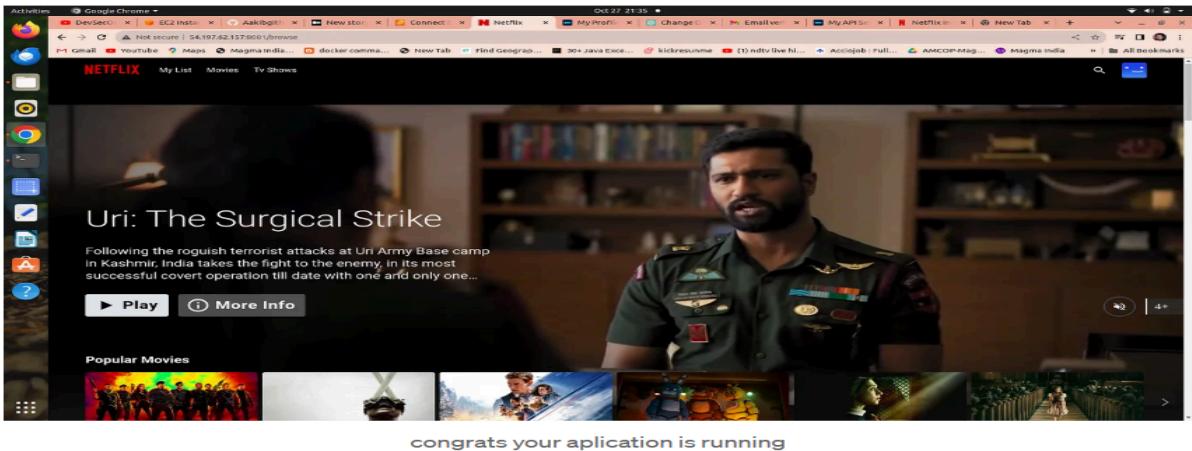
- a. **docker stop <containerid>**
- b. **docker rmi -f netflix**

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

what is sonarqube →

SonarQube is like a “code quality detective” tool for software developers that scans your code to find and report issues, bugs, security vulnerabilities, and code smells to help you write better and more reliable software. It provides insights and recommendations to improve the overall quality and maintainability of your codebase

what is trivy →

Trivy is like a “security guard” for your software that checks for vulnerabilities in the components (like libraries) your application uses, helping you find and fix potential security problems to keep your software safe from attacks and threats. It’s a tool that scans your software for known security issues and provides information on how to address them.

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**

The screenshot shows the SonarQube interface at the URL 100.29.10.109:9000/projects. The main view is for the 'Netflix' project, which has passed its quality gate. The dashboard provides a summary of code quality metrics:

- Bugs: 0
- Vulnerabilities: 0
- Hotspots Reviewed: 0.0%
- Code Smells: 18 (A)
- Coverage: 0.0%
- Duplications: 0.0%
- Lines: 3.2k (TypeScript)

On the left, there are filters for Quality Gate (Passed: 1, Failed: 0) and Reliability (A rating: 1, B rating: 0).

Step 2 → setup Trivy

run the following commands

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

b. wget -qO – <https://aquasecurity.github.io/trivy-repo/deb/public.key>

| sudo apt-key add -

c. echo deb <https://aquasecurity.github.io/trivy-repo/deb> \$(lsb_release

-sc) main | sudo tee -a /etc/apt/sources.list.d/trivy.list

d. sudo apt-get update

e. sudo apt-get install trivy

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>

```
ubuntu@ip-10-0-4-63:~$ trivy image aa8923ae55c6
2025-01-18T10:15:08Z  INFO  [vulndb] Need to update DB
2025-01-18T10:15:08Z  INFO  [vulndb] Downloading vulnerability DB...
2025-01-18T10:15:08Z  INFO  [vulndb] Downloading artifact...          repo="mirror.gcr.io/aquasec/trivy-db:2"
58.63 MiB / 58.63 MiB [-----] 100.00% 26.04 MiB p/s 2.5s
2025-01-18T10:15:12Z  INFO  [vulndb] Artifact successfully downloaded      repo="mirror.gcr.io/aquasec/trivy-db:2"
2025-01-18T10:15:12Z  INFO  [vuln] Vulnerability scanning is enabled
2025-01-18T10:15:12Z  INFO  [secret] Secret scanning is enabled
2025-01-18T10:15:12Z  INFO  [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-01-18T10:15:12Z  INFO  [secret] Please see also https://aquasecurity.github.io/trivy/v0.58/docs/scanner/secret#recommendation f
or faster secret detection
2025-01-18T10:15:13Z  INFO  Detected OS      family="alpine" version="3.20.5"
2025-01-18T10:15:13Z  INFO  [alpine] Detecting vulnerabilities...   os_version="3.20" repository="3.20" pkg_num=66
2025-01-18T10:15:13Z  INFO  Number of language-specific files      num=0

aa8923ae55c6 (alpine 3.20.5)

Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

here's the scan of your image of netflix

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

output →openjdk version “17.0.8” 2023-07-18

OpenJDK Runtime Environment (build 17.0.8+7-Debian-1deb12u1)

OpenJDK 64-Bit Server VM (build 17.0.8+7-Debian-1deb12u1, mixed mode, sharing)

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
5. sudo systemctl start jenkins
6. sudo systemctl enable jenkins
7. Access Jenkins in a web browser using the public IP of your EC2 instance
<http://publicIp:8080>

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)

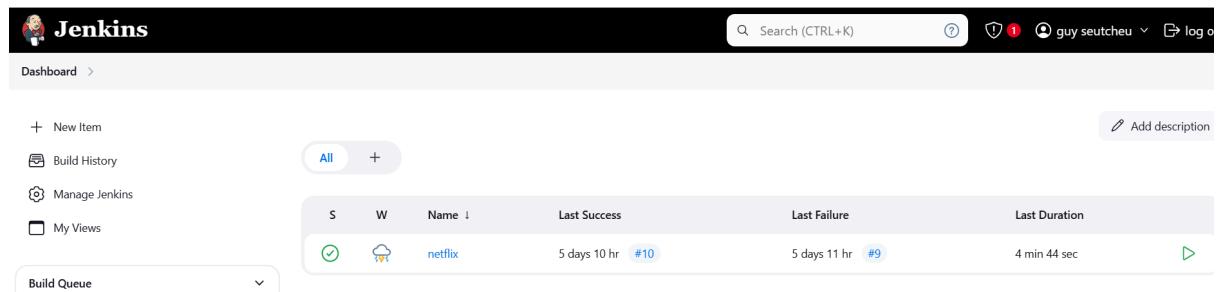
4 Email Extension Plugin

6. **owasp** →The OWASP Plugin in Jenkins is like a “security assistant” that helps you find and fix security issues in your software. It uses the knowledge and guidelines from the Open Web Application

Security Project (OWASP) to scan your web applications and provide suggestions on how to make them more secure. It's a tool to ensure that your web applications are protected against common security threats and vulnerabilities.

7. Prometheus metrics → to monitor jenkins on grafana dashboard

8. Download all the docker related plugins



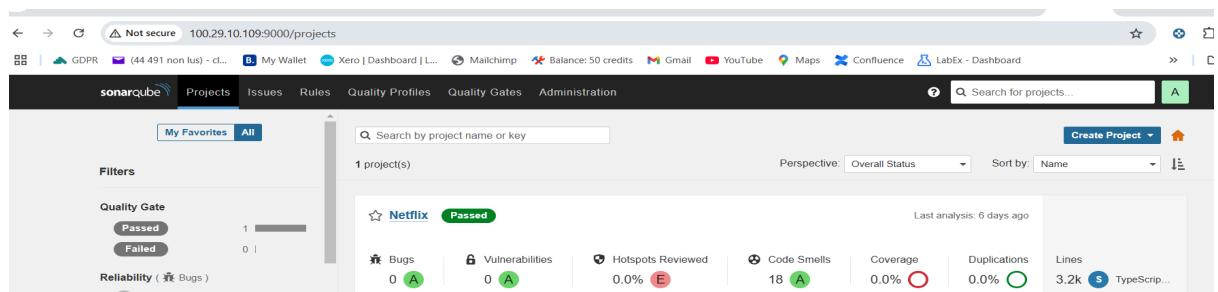
The screenshot shows the Jenkins dashboard with the 'Build Queue' section highlighted. A single build entry for the 'netflix' project is visible. The build status is 'S' (Success), with a green checkmark icon. The name is 'netflix'. The 'Last Success' timestamp is '5 days 10 hr #10'. The 'Last Failure' timestamp is '5 days 11 hr #9'. The 'Last Duration' is '4 min 44 sec'. Other options like 'New Item', 'Build History', and 'Manage Jenkins' are visible on the left.

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 security → users → token → generate token



The screenshot shows the Sonarqube 'Projects' dashboard for the 'Netflix' project. At the top, there are tabs for 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. The main area shows '1 project(s)' with the 'Netflix' project selected, which has a 'Passed' status. Below this, there are several metrics: Bugs (0), Vulnerabilities (0), Hotspots Reviewed (0.0%), Code Smells (18), Coverage (0.0%), Duplications (0.0%), and Lines (3.2k). On the left, there are filters for Quality Gate (Passed, Failed) and Reliability (Bugs). A search bar at the top right allows searching for projects.

The screenshot shows the SonarQube Administration interface at the URL 100.29.10.109:9000/admin/users. The top navigation bar includes links for GDPR, My Wallet, Xero, Mailchimp, Balance: 50 credits, Gmail, YouTube, Maps, Confluence, and LabEx - Dashboard. The main navigation menu has tabs for Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration, with Administration selected. A search bar at the top right says "Search for projects...". Below the navigation, there's a sub-navigation for Administration with tabs for Configuration, Security (which is selected), Projects, System, and Marketplace. The main content area is titled "Users" and contains the instruction "Create and administer individual users." There is a search bar labeled "Search by login or name...". A table lists a single user entry:

	SCM Accounts	Last connection	Groups	Tokens
A Administrator admin		< 1 hour ago	sonar-administrators sonar-users	2

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
3. in the name field type Netflix
4. click on set up

Create a project

All fields marked with * are required

Project display name *

Up to 255 characters. Some scanners might override the value you provide.

Project key *

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](#)

Are you just testing or have an advanced use-case? Analyze your project locally.



Locally

[click on this option](#)

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

Generate a project token

Token name [?](#)

 Analyze "Netflix"

Expires in

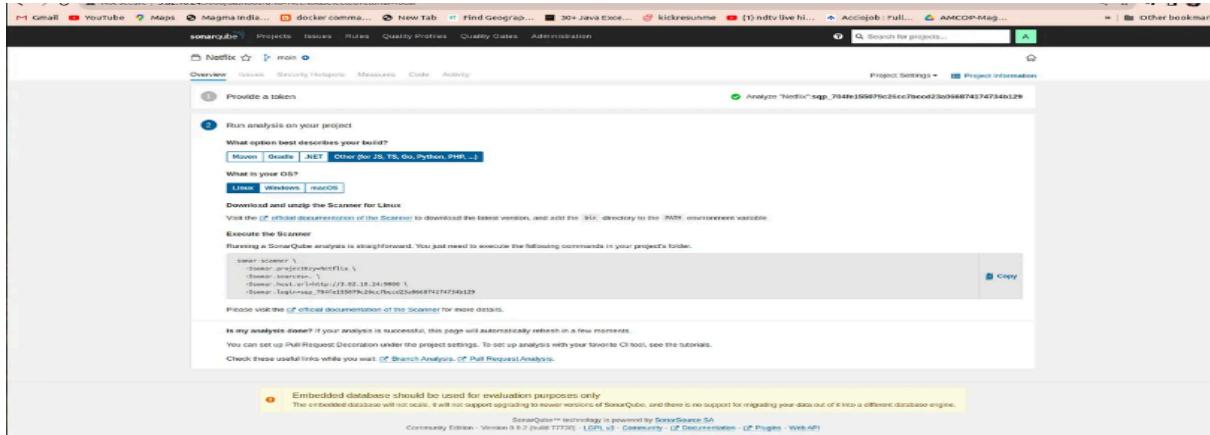
 30 days[Generate](#)

ⓘ Please note that this token will only allow you to analyze the current project. If you want to use the same token to analyze multiple projects, you need to generate a global token in your user account. See the [documentation](#) for more information.

Use existing token

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your [user account](#).

[click on generate](#)



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 dockerhub
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 jdk17

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

The screenshot shows the Jenkins global settings page. It includes three main sections: 'NodeJS installations' (with a dropdown menu and 'Edited' status), 'Dependency-Check installations' (with a dropdown menu and 'Edited' status), and 'Docker installations' (with a dropdown menu and 'Edited' status). Each section has a small icon and a link to 'Edit'.

Step 4 →Configure global setting for sonarube

1. go to manage jenkins →Configure global setting →add sonarqube servers
2. **name ==sonar-server**
3. **server_url==http://public_ip:9000**
4. **server authentication token == sonar-token**

The screenshot shows the 'SonarQube installations' configuration screen. It includes a checkbox for 'Environment variables' (unchecked) and a 'Name' field containing 'sonar-server'. Below it, there's a 'Server URL' field with a default value of 'http://localhost:9000' and a new value of 'http://52.91.65.44:9000'. A 'Server authentication token' field contains 'sonar-token'. There are 'Add' and 'Advanced' buttons at the bottom.

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 'jdk17'  
  
        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/Aakibgithuber/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
aakibkhan1212/netflix:latest "

                sh "docker push
aakibkhan1212/netflix:latest "

            }
        }
    }
}
```

```
        }

    }

}

stage("TRIVY") {

    steps{
        sh "trivy image nasi101/netflix:latest > trivyimage.txt"
    }

}

stage('Deploy to container') {

    steps{
        sh 'docker run -d --name netflix -p 8081:80 nasi101/netflix:latest'
    }

}
```

```
}
```

```
}
```

```
}
```

If you get docker login failed errorr

```
sudo su
```

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

```
sudo systemctl restart jenkins
```

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

The screenshot shows a Jenkins pipeline status page for the 'netflix' pipeline. On the left, there's a sidebar with various pipeline management options like 'Changes', 'Build Now', 'Configure', 'Delete Pipeline', 'Full Stage View', 'SonarQube', and 'Stages'. The main area displays the pipeline stages: 'Declarative: Tool Install', 'clean workspace', 'Checkout from Git', 'Sonarqube Analysis', 'quality gate', 'Install Dependencies', 'OWASP FS SCAN', 'TRIVY FS SCAN', 'Docker Build & Push', 'TRIVY', and 'Deploy to container'. The 'quality gate' stage is highlighted with a green checkmark icon and labeled 'Passed'. Below it, a SonarQube dependency check badge shows 'server-side processing: Success'. A 'Permalinks' section is also present.

This screenshot shows the 'Full Stage View' for the 'netflix' pipeline. It provides a detailed breakdown of the execution time for each stage. The stages and their average runtimes are: Declarative: Tool Install (2s), clean workspace (370ms), Checkout from Git (1s), Sonarqube Analysis (22s), quality gate (17h 51min), Install Dependencies (14s), OWASP FS SCAN (3min 37s), TRIVY FS SCAN (15s), Docker Build & Push (35s), TRIVY (489ms), and Deploy to container (454ms). A tooltip indicates an average full run time of approximately 4 minutes and 44 seconds. The 'Deploy to container' stage is currently running, as indicated by a red progress bar. A 'Logs' button is visible in the bottom right corner of the stage view.

Phase 4 →Monitoring via Prometheus and grafana

- **Prometheus** is like a detective that constantly watches your software and gathers data about how it's performing. It's good at collecting metrics, like how fast your software is running or how many users are visiting your website.
- **Grafana**, on the other hand, is like a dashboard designer. It takes all the data collected by Prometheus and turns it into easy-to-read charts and graphs. This helps you see how well your software is doing at a glance and spot any problems quickly.

In other words, Prometheus collects the information, and Grafana makes it look pretty and understandable so you can make decisions about your software. They're often used together to monitor and manage applications and infrastructure.

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.
- 20 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
```

5. Enable and start Prometheus:

a. sudo systemctl enable prometheus

b. sudo systemctl start prometheus

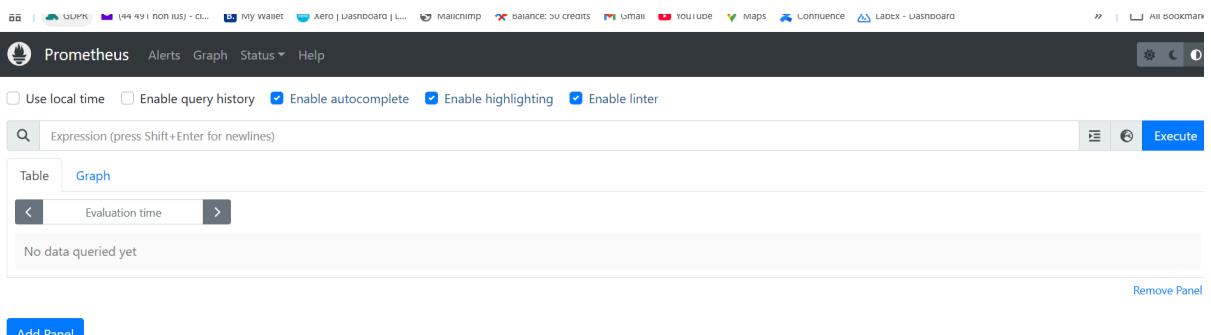
c. sudo systemctl status prometheus

```
ubuntu@ip-10-0-4-18:~$ systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2025-01-17 02:04:07 UTC; 2 days ago
     Main PID: 29094 (prometheus)
        Tasks: 8 (limit: 4676)
       Memory: 97.2M
          CPU: 2min 57.076s
        CGroup: /system.slice/prometheus.service
                  └─29094 /usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/data --

Jan 19 11:00:00 ip-10-0-4-18 prometheus[29094]: ts=2025-01-19T11:00:00.199Z caller=compact.go:523 level=info component=tsdb
Jan 19 11:00:00 ip-10-0-4-18 prometheus[29094]: ts=2025-01-19T11:00:00.202Z caller=head.go:1298 level=info component=tsdb
```

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



Step 3 → Installing Node Exporter:

Node exporter is like a “reporter” tool for Prometheus, which helps collect and provide information about a computer (node) so Prometheus can monitor it. It gathers data about things like CPU usage, memory, disk space, and network activity on that computer.

A Node Port Exporter is a specific kind of Node Exporter that is used to collect information about network ports on a computer. It tells Prometheus which network ports are open and what kind of data is going in and out of

those ports. This information is useful for monitoring network-related activities and can help you ensure that your applications and services are running smoothly and securely.

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
```

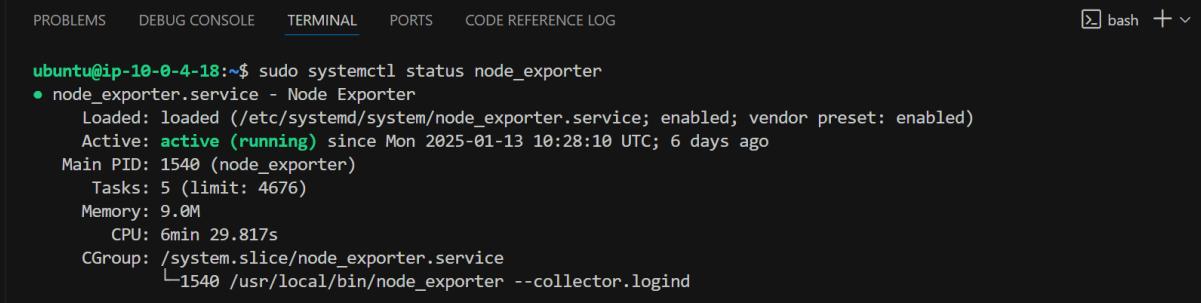
```
[Install]
```

WantedBy=multi-user.targetb. press → ctrl+o then ctrl+x

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



```
ubuntu@ip-10-0-4-18:~$ sudo systemctl status node_exporter
● node_exporter.service - Node Exporter
    Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)
      Active: active (running) since Mon 2025-01-13 10:28:10 UTC; 6 days ago
        Main PID: 1540 (node_exporter)
           Tasks: 5 (limit: 4676)
         Memory: 9.0M
            CPU: 6min 29.817s
          CGroup: /system.slice/node_exporter.service
                  └─1540 /usr/local/bin/node_exporter --collector.logind
```

You can access Node Exporter metrics in Prometheus.

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 monitor 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: ['<your-jenkins-ip>:<your-jenkins-port>']
```

add the above code with proper indentation like this →

```
>     # The job name is added as a label `job=<job_name>` to any timeseries so
>     - 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: ['100.29.10.109:8080']
>
>         - job_name: 'Netflix'
>             metrics_path: '/metrics'
>             static_configs:
>                 - targets: ['node1Ip:9001']
```

a. Check the validity of the configuration file →

```
promtool check config /etc/prometheus/prometheus.yml
```

o/p →success

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

```

ubuntu@ip-10-0-4-18:/etc/prometheus$ ls
a  console_libraries  consoles  prometheus.yml  -
ubuntu@ip-10-0-4-18:/etc/prometheus$ vi prometheus.yml
ubuntu@ip-10-0-4-18:/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

```

http://100.29.10.109:8080/prometheus	DOWN	instance="100.29.10.109:8080" job="jenkins"	9.821s ago	1.4
node_exporter (1/1 up) show less				
Endpoint	State	Labels	Last Scrape	Scr Du
http://localhost:9100/metrics	UP	instance="localhost:9100" job="node_exporter"	10.97s ago	12.1
prometheus (1/1 up) show less				
Endpoint	State	Labels	Last Scrape	Scr Du
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	7.218s ago	4.0

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 apt-key add -

Add the repository for Grafana stable releases:

a. echo "deb <https://packages.grafana.com/oss/deb> stable main" | sudo tee -a
/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

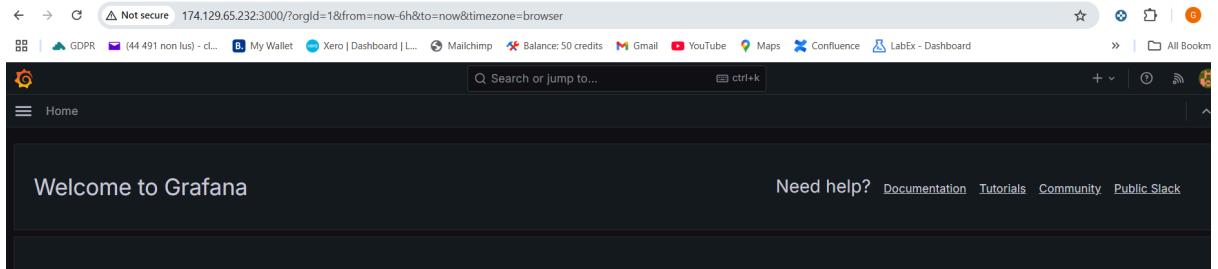
d. sudo systemctl start grafana-server

e. sudo systemctl status grafana-server

```
● grafana-server.service - Grafana instance
   Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2025-01-13 11:20:33 UTC; 6 days ago
     Docs: http://docs.grafana.org
 Main PID: 3710 (grafana)
    Tasks: 15 (limit: 4676)
   Memory: 67.3M
      CPU: 11min 55.206s
     CGroup: /system.slice/grafana-server.service
             └─3710 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/graf
Jan 19 14:51:25 ip-10-0-4-18 grafana[3710]: logger=infra.usagestats t=2025-01-19T14:51:25.25875801Z level=info
```

**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



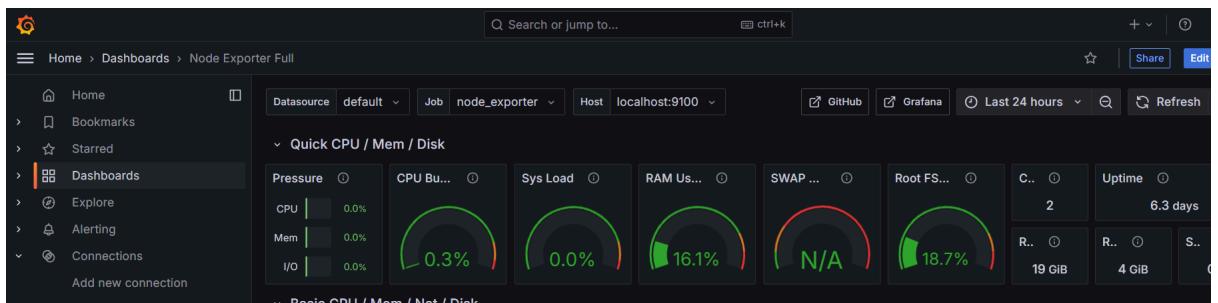
username = admin, password =admin

Step 6 → Add Prometheus Data Source:

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

Follow these steps:

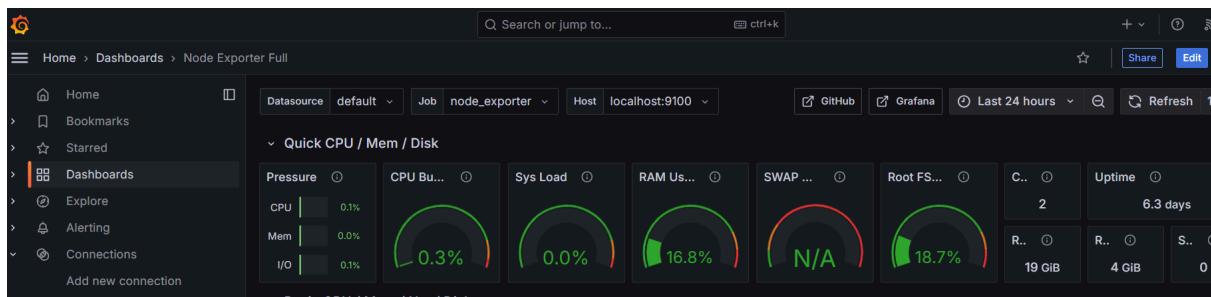
- Click on the gear icon (⚙️) in the left sidebar to open the “Configuration” menu.
- Select “Data Sources.”
- Click on the “Add data source” button.



Step 7 → Import a Dashboard

Importing a dashboard in Grafana is like using a ready-made template to quickly create a set of charts and graphs for monitoring your data, without having to build them from scratch.

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

Jenkins: Performance and Health Overview

- Home
- Bookmarks
- Starred
- Dashboards
- Explore
- Alerting
- Connections
- Add new connection
- Data sources
- Administration

Processing speed: N/A

Job queue duration: No data

JVM free memory: N/A

Memory Usage: N/A

Jenkins health: No data

JVM Uptime: N/A

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

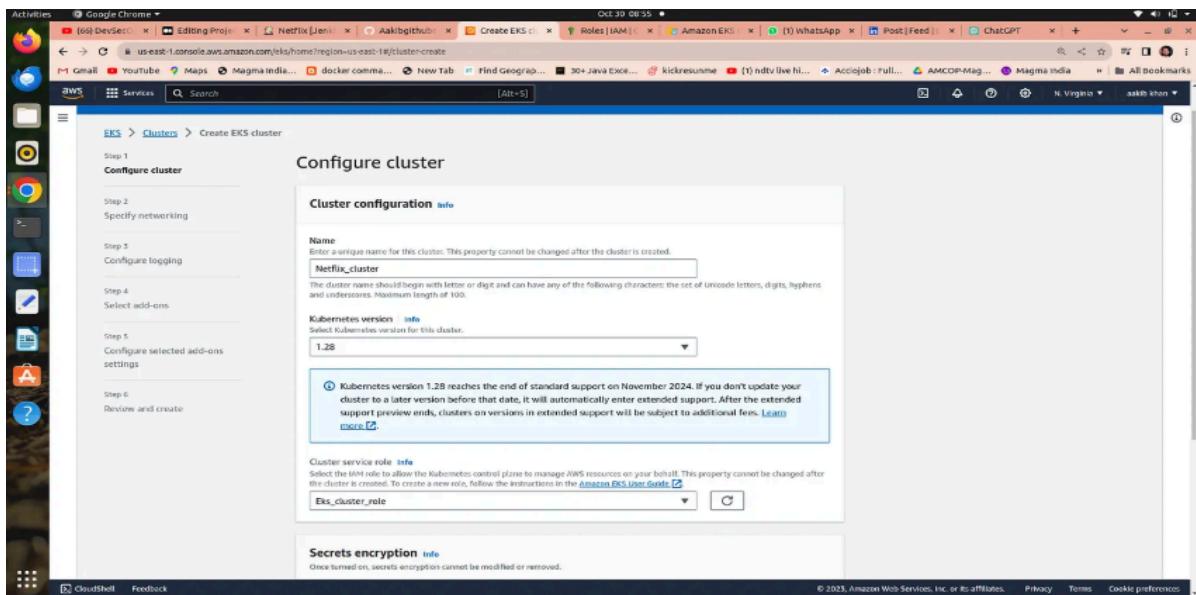


click on create

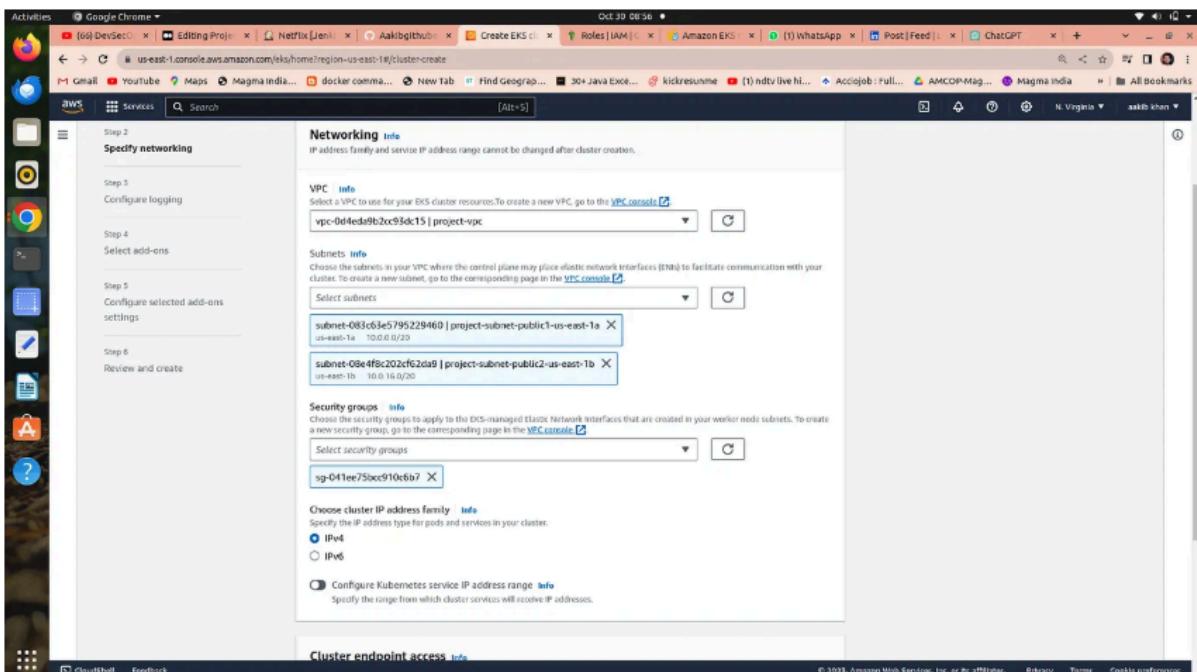
3. add a name to your cluster and choose a service role if you don't have then follow below documentation to create it

Amazon EKS cluster IAM role

The Amazon EKS cluster IAM role is required for each cluster. Kubernetes clusters managed by Amazon EKS use this role...

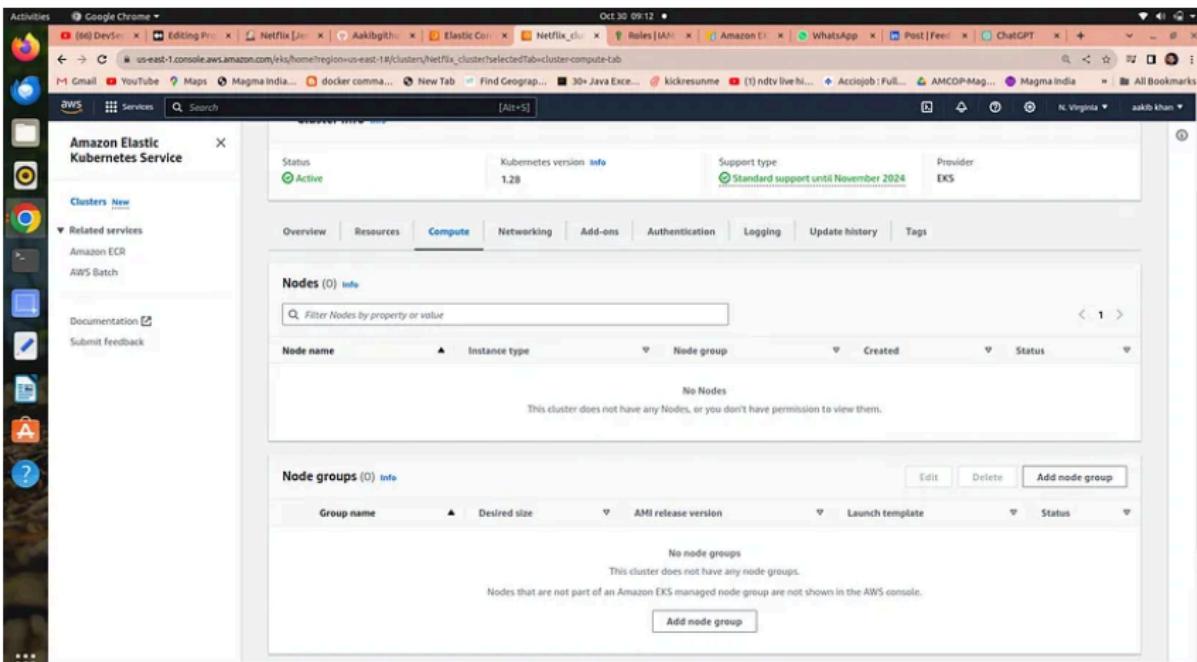


4. click on next and choose the default vpc and in subnet option make sure to remove all the private subnet and remain everything as it is



5. click on next →next →create

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



7. choose the below options

A node group is a group of EC2 instances that supply compute capacity to your Amazon EKS cluster. You can add multiple node groups to your cluster.

Node group configuration
These properties cannot be changed after the node group is created.

Name
Assign a unique name for this node group.
The node group name should begin with letter or digit and can have any of the following characters: the set of Unicode letters, digits, hyphens and underscores. Maximum length of 63.

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](#).
 [C](#)

ⓘ The selected role must not be used by a self-managed node group as this could lead to a service interruption upon managed node group deletion.

[Learn more](#) [?](#)

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.

Capacity type
Select the capacity purchase option for this node group.

Instance types [Info](#)
Select instance types you prefer for this node group.
 [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.

Remain everything as it is →click on create

The screenshot shows the AWS CloudWatch Metrics interface. A single metric named 'Lambda Function Invocations' is displayed with a value of 1000. The time range is set to '1 hour'. The chart has a light blue background with a white grid. The Y-axis is labeled 'Value' with a scale from 0 to 1000. The X-axis shows time intervals. The metric name is at the top left, and the unit is 'Count'.

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

2. scroll down to access keys and click on create

The screenshot shows the AWS Identity and Access Management (IAM) console. On the left, there's a sidebar with navigation links like Dashboard, Access management, Access reports, and Related consoles. The main area displays account information: Account name (aakib khan), Email address (dethiluniversityyy@gmail.com), AWS account ID (731234214840), and Canonical user ID (cbe1e015e1838237df49fac2ace12abeb4fb36d0352b67df586f253be508d978). Below this, the Multi-factor authentication (MFA) section shows one virtual device assigned. The Access keys section indicates 'No access keys' available. At the bottom, there are buttons for 'Create access key' and 'Create access key'.

copy and paste it to terminal

The screenshot shows the AWS Elastic Kubernetes Service (EKS) console. On the left, there's a sidebar with links for Amazon Elastic Kubernetes Service, Clusters, Amazon EKS Anywhere, Related services (Amazon ECR, AWS Batch), and Console settings. The main area shows the 'netflix' cluster details. It includes sections for Cluster info (Status: Active, Kubernetes version: 1.31, Support period: Standard support until November 26, 2025, Provider: EKS), Cluster health issues (0), Upgrade insights (5), and Node health issues (1).

Run the following command

1. **aws eks update-kubeconfig — name YourClusterName — region us-east-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 "us-east-" region, allowing you to interact with the cluster using `kubectl`.

Step 3→Install helm →

- 1. curl <https://baltocdn.com/helm/signing.asc> | gpg — dearmor | sudo tee /usr/share/keyrings/helm.gpg > /dev/null**
- 2. sudo apt-get install apt-transport-https — yes**
- 3. 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**
- 4. sudo apt-get install helm**

Step →4 Install Node Exporter using Helm

To begin monitoring your Kubernetes cluster, you'll install the Prometheus Node Exporter by Using Helm to install Node Exporter in Kubernetes makes it easy to set up and manage the tool for monitoring your servers by providing a pre-packaged, customizable, and consistent way to do it.

1. Add the Prometheus Community Helm repository:

- helm repo add prometheus-community**

<https://prometheus-community.github.io/helm-charts>

1. Create a Kubernetes namespace for the Node Exporter:

- kubectl create namespace prometheus-node-exporter**

1. Install the Node Exporter using Helm:

- helm install prometheus-node-exporter**

**prometheus-community/prometheus-node-exporter — namespace
prometheus-node-exporter**

```
guyse@SEUTCHEU MINGW64 ~
$ kubectl get pods -n prometheus-node-exporter
NAME                               READY   STATUS    RESTARTS   AGE
prometheus-node-exporter-6w7fv   1/1     Running   0          45h
prometheus-node-exporter-9t2cw   1/1     Running   0          45h
prometheus-node-exporter-wr6mg   1/1     Running   0          2d14h

guyse@SEUTCHEU MINGW64 ~
$
```

Add a Job to Scrape Metrics on nodeip:9001/metrics in prometheus.yml:

Update your Prometheus configuration (prometheus.yml) to add a new job for scraping metrics from nodeip:9001/metrics. You can do this by adding the following configuration to your prometheus.yml file:

```
- job_name: 'Netflix'

metrics_path: '/metrics'

static_configs:

- targets: ['node1Ip:9100']
```

Replace 'your-job-name' with a descriptive name for your job. The static_configs section specifies the targets to scrape metrics from, and in this case, it's set to nodeip:9001.

Don't forget to reload or restart Prometheus to apply these changes to your configuration Deploy Application with ArgoCDon.

Step 5→Install Argo cd

Argo CD is a tool that helps software developers and teams manage and deploy their applications to Kubernetes clusters more easily. It simplifies the process of keeping your applications up to date and in sync with your desired configuration by automatically syncing your code with what's running in your Kubernetes environment. It's like a traffic cop for your chmod 700 get_helm.shapplications on Kubernetes, ensuring they are always in the right state without you having to manually make changes.

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

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/argocd-applicationset-controller	1/1	1	1	2d14h
deployment.apps/argocd-dex-server	1/1	1	1	2d14h
deployment.apps/argocd-notifications-controller	1/1	1	1	2d14h
deployment.apps/argocd-redis	1/1	1	1	2d14h
deployment.apps/argocd-repo-server	1/1	1	1	2d14h
deployment.apps/argocd-server	1/1	1	1	2d14h
NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/argocd-applicationset-controller-68dd598849	1	1	1	2d14h
replicaset.apps/argocd-dex-server-86877d5f88	1	1	1	2d14h
replicaset.apps/argocd-notifications-controller-74584f69d8	1	1	1	2d14h
replicaset.apps/argocd-redis-95497c66	1	1	1	2d14h
replicaset.apps/argocd-repo-server-689dd9d6d4	1	1	1	2d14h
replicaset.apps/argocd-server-6997845b6	1	1	1	2d14h
NAME	READY	AGE		
statefulset.apps/argocd-application-controller	1/1	2d14h		

Expose argocd-server

By default argocd-server is not publicaly 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"}}'`

```
export ARGOCD_SERVER=$(kubectl get svc argocd-server -n argocd -o json | jq
--raw-output '.status.loadBalancer.ingress[0].hostname')
```

```
echo $ARGOCD_SERVER
```

```
export ARGO_PWD=$(kubectl -n argocd get secret argocd-initial-admin-secret -o
jsonpath=".data.password" | base64 -d)
```

```
echo $ARGO_PWD
```

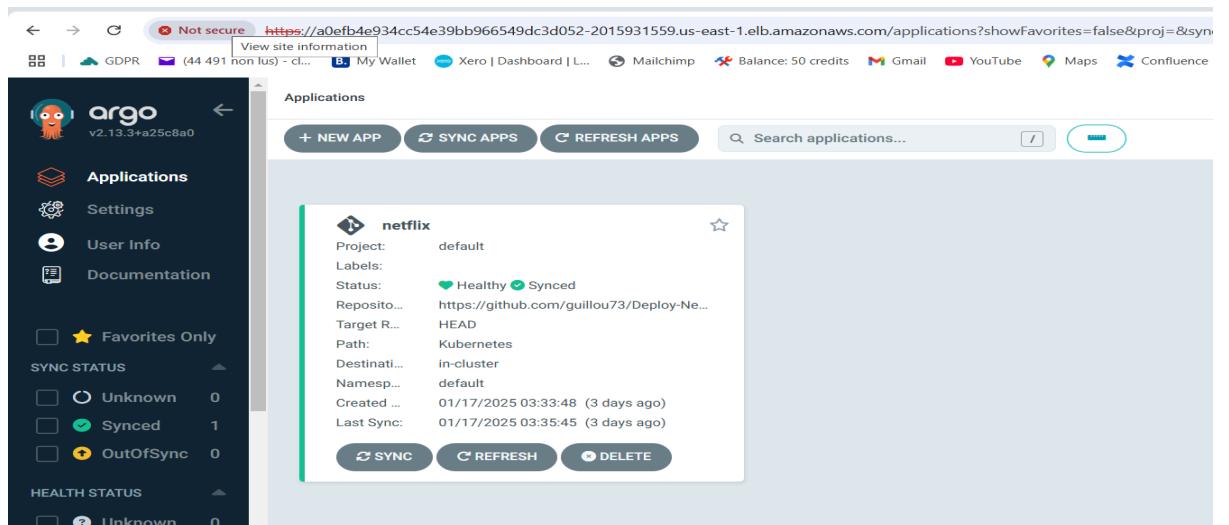
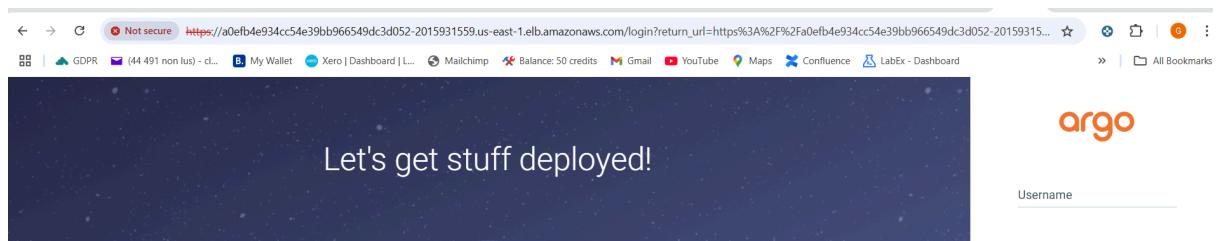
```

guyse@SEUTCHEU MINGW64 ~
$ export ARGOCD_SERVER=$(kubectl get svc argocd-server -n argocd -o json | jq --raw-output '.status.loadBalancer.ingress[0].hostName')
guyse@SEUTCHEU MINGW64 ~
$ echo $ARGOCD_SERVER
a0efb4e934cc54e39bb966549dc3d052-2015931559.us-east-1.elb.amazonaws.com

guyse@SEUTCHEU MINGW64 ~
$ export ARGO_PWD=$(kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath=".data.password" | base64 -d)
guyse@SEUTCHEU MINGW64 ~
$ echo $ARGO_PWD
kJ1k1USrJZU0cBKS

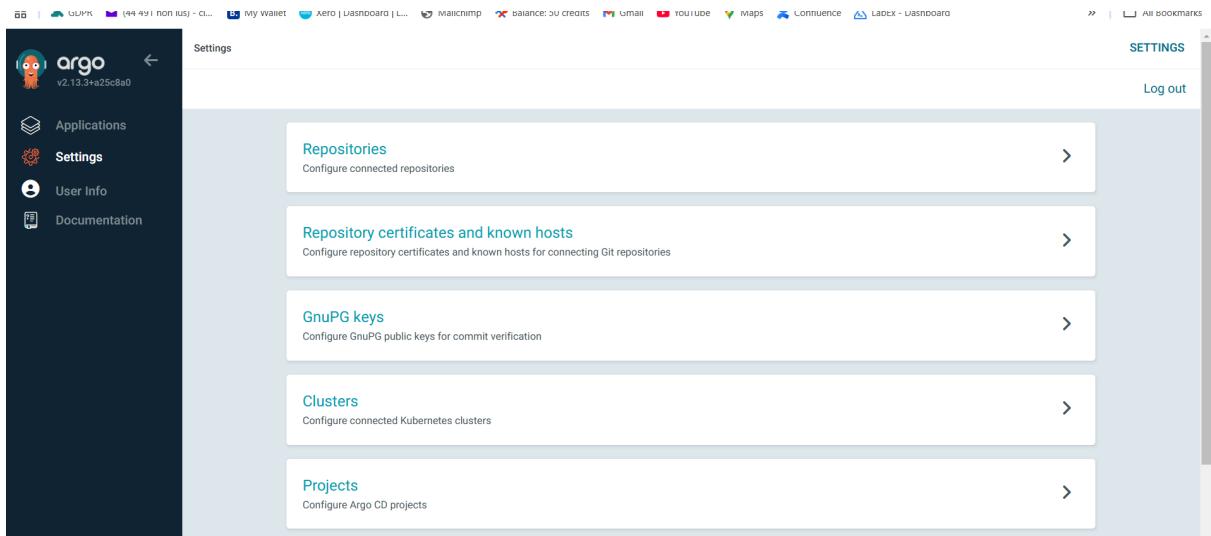
```

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



Step 5 → Deploy Application with ArgoCD

Sign into argo cd using above steps →

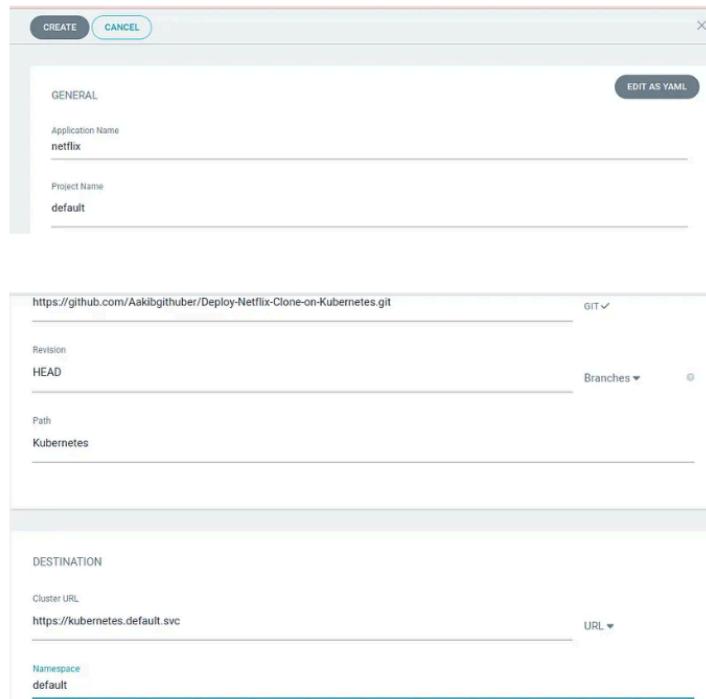


1. go to setting → repositories → connect repo

TYPE	NAME	PROJECT	REPOSITORY	CONNECTION STATUS	⋮
git OCI		default	github.com/Aakibgithuber/Deploy-Network-Clone-on...	✗ Failed	⋮
git		default	https://github.com/Aakibgithuber/Deploy-Network-Clone-on...	✗ Failed	⋮
git		default	https://github.com/guilou73/Deploy-Network-Clone-on...	✓ Successful	⋮

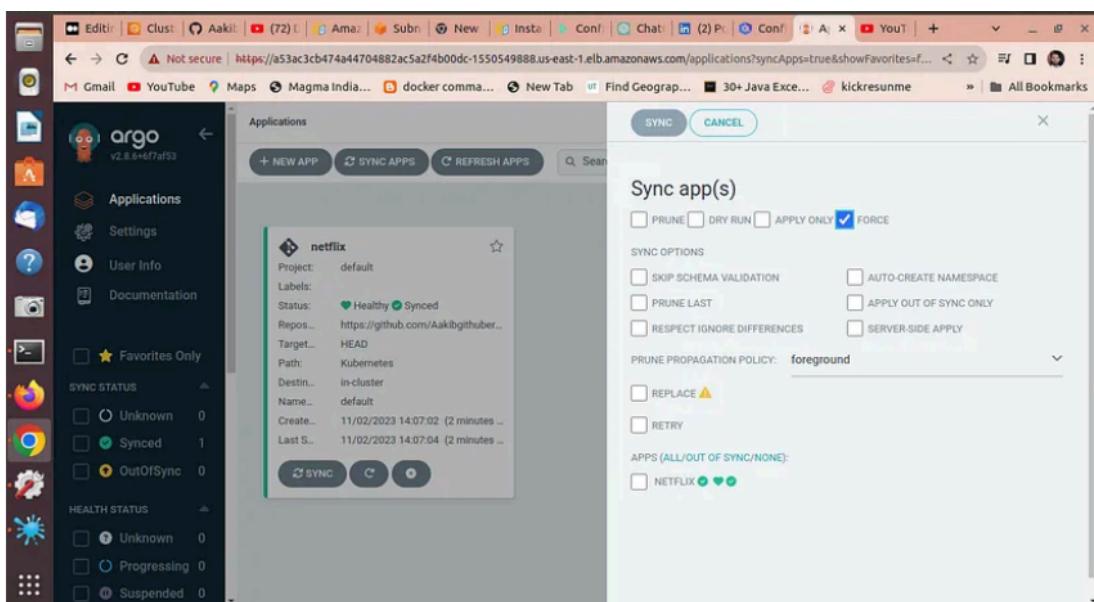
2. click on connect output → successful

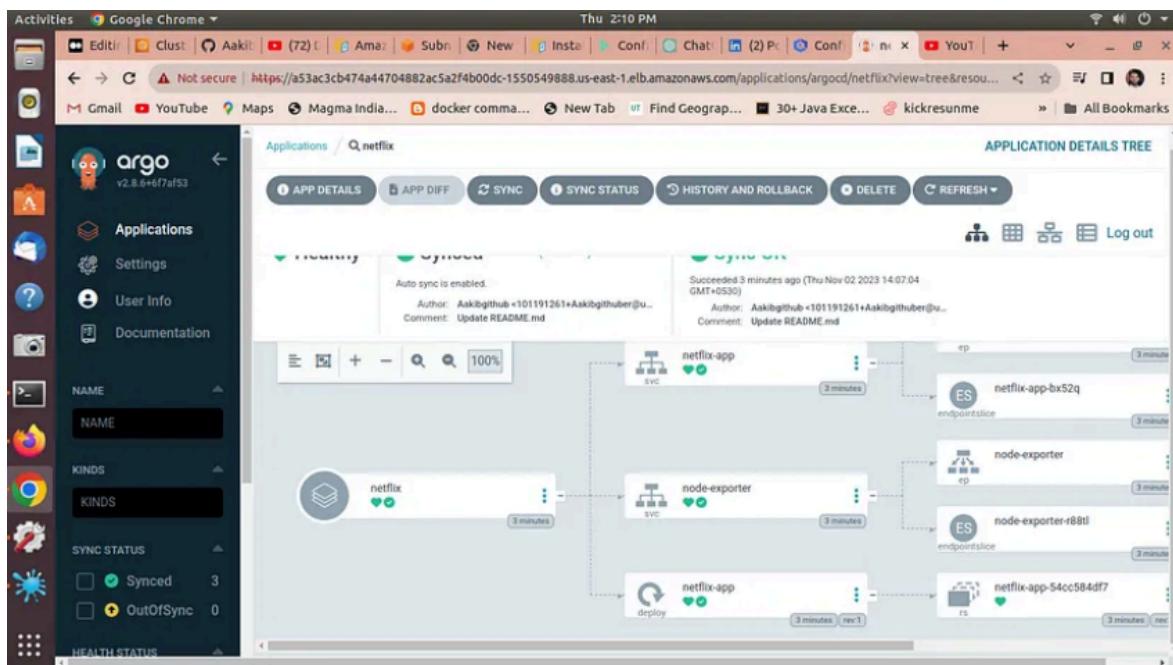
3. Now go to application → newapp



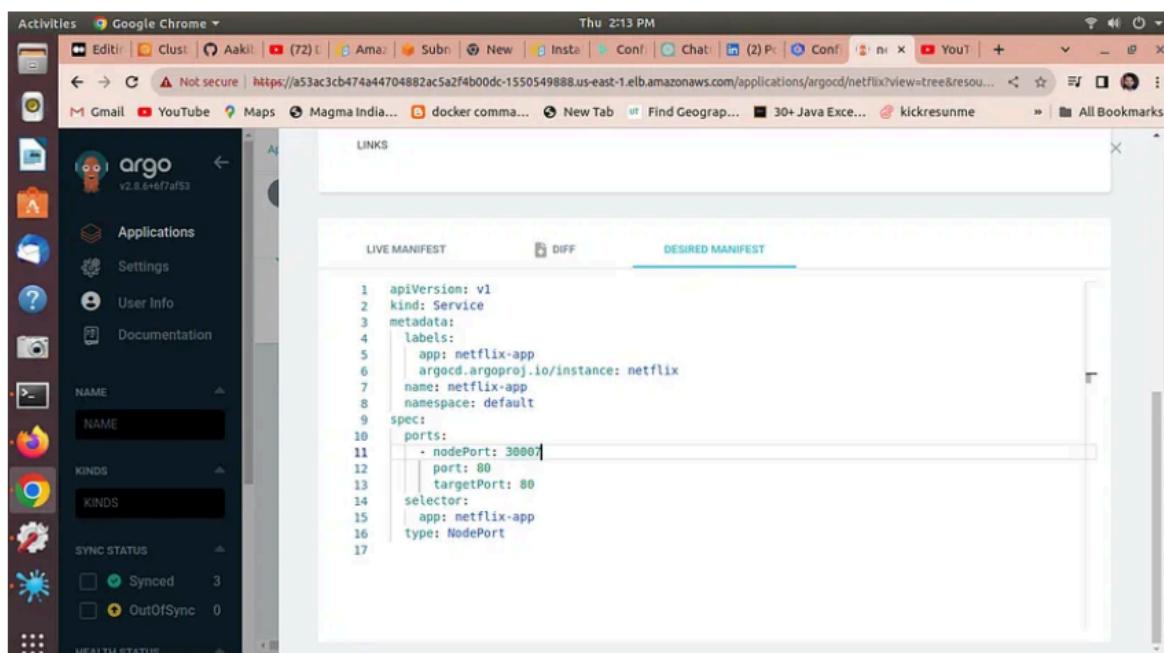
4. select the above options and click on create

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

The screenshot shows the AWS EKS Node details page for the node `ip-10-0-10-59.ec2.internal`. The node is in a **Ready** state. Key details include:

- Status:** Ready
- Kernel version:** 5.10.197-186.748.amzn2.x86_64
- Created:** An hour ago
- Last transition time:** An hour ago
- Node group:** Netnode
- Container runtime:** containerd://1.6.19
- OS (Architecture):** Linux (amd64)
- Instance:** i-038c4252e923ded8e
- Kubelet version:** v1.28.2-eks-a5df82a
- OS Image:** Amazon Linux 2
- Instance type:** t3.medium

now browse the application by Publicip:30007

The screenshot shows the AWS EKS Cluster details page for the cluster `netflix`. The cluster has the following metrics:

- Cluster health issues:** 0
- Upgrade insights:** 5
- Node health issues:** 1

The **Compute** tab is selected, showing the **Nodes** section with three nodes listed:

Node name	Instance type	Compute	Managed by	Created	Status
i-08ab0e8b55e52e7ae	c6a.large	Auto Mode	general-purpose	Created January 17, 2025, 19:01 (UTC+00:00)	Ready
i-0a997da398f740236	c6g.large	Auto Mode	system	Created January 17, 2025, 19:01 (UTC+00:00)	Ready

The screenshot shows the AWS EC2 Instances details page for the instance `i-08ab0e8b55e52e7ae`. The instance summary includes:

- Updated 4 minutes ago
- Instance ID:** i-08ab0e8b55e52e7ae
- IPv6 address:** -
- Public IPv4 address:** 14.223.106.128 (with a link to open address)
- Instance state:** Running

On the right, it lists network interfaces:

- Private IPv4 addresses:** 192.168.34.2
- Public IPv4 DNS:** ec2-44-223-106-128.compute-1.amazonaws.com

