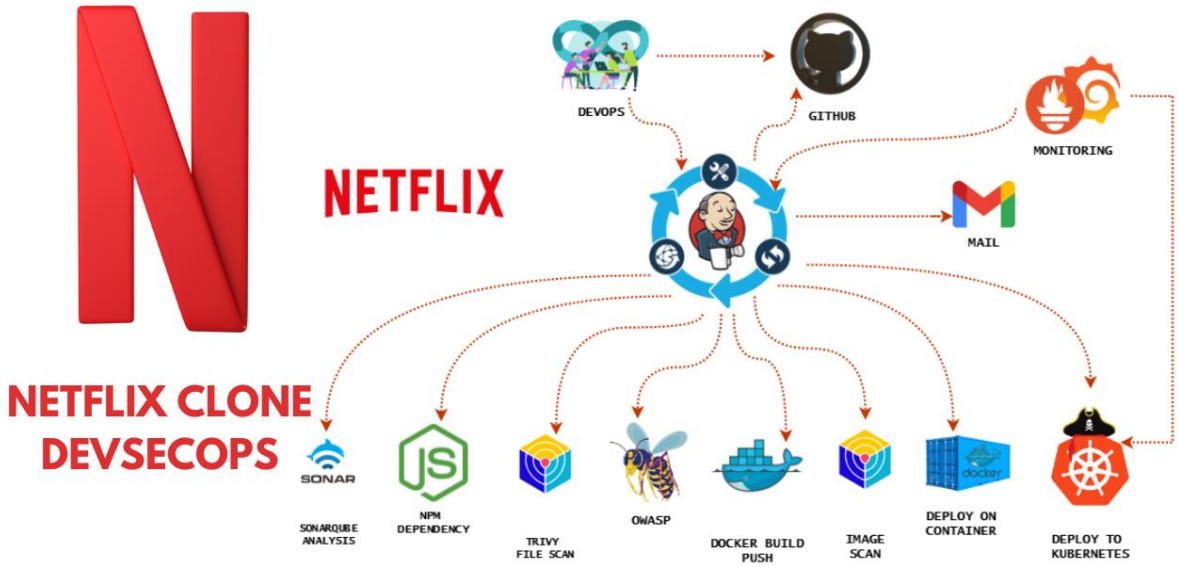
Netflix Clone CI/CD with Monitoring | Jenkins | Docker| Kubernetes| Monitoring | DevSecOps



Hello friends, we will be deploying a Netflix clone. We will be using Jenkins as a CICD tool and deploying our application on a Docker container and Kubernetes Cluster and we will monitor the Jenkins and Kubernetes metrics using Grafana, Prometheus and Node exporter. I Hope this detailed blog is useful.

CLICK HERE FOR GITHUB REPOSITORY

Steps:-

Step 1 — Launch an Ubuntu(22.04) T2 Large Instance

Step 2 — Install Jenkins, Docker and Trivy. Create a Sonarqube Container using Docker.

Step 3 — Create a TMDB API Key.

Step 4 — Install Prometheus and Grafana On the new Server.

Step 5 — Install the Prometheus Plugin and Integrate it with the Prometheus server.

Step 6 — Email Integration With Jenkins and Plugin setup.

Step 7 — Install Plugins like JDK, Sonarqube Scanner, Nodejs, and OWASP Dependency Check.

Step 8 — Create a Pipeline Project in Jenkins using a Declarative Pipeline

Step 9 — Install OWASP Dependency Check Plugins

Step 10 — Docker Image Build and Push

Step 11 — Deploy the image using Docker

Step 12 — Kubernetes master and slave setup on Ubuntu (20.04)

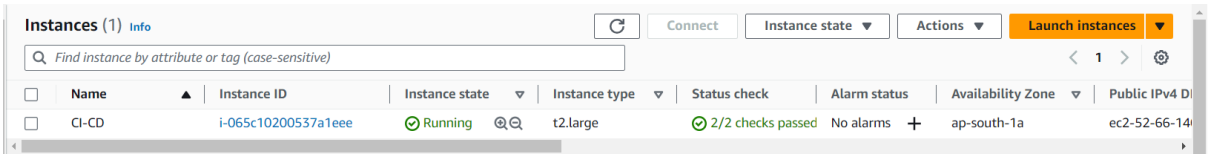
Step 13 — Access the Netflix app on the Browser.

Step 14 — Terminate the AWS EC2 Instances.

Now, let's get started and dig deeper into each of these steps:-

STEP1:Launch an Ubuntu(22.04) T2 Large Instance

Launch an AWS T2 Large Instance. Use the image as Ubuntu. You can create a new key pair or use an existing one. Enable HTTP and HTTPS settings in the Security Group and open all ports (not best case to open all ports but just for learning purposes it's okay).



Step 2 — Install Jenkins, Docker and Trivy

2A — To Install Jenkins

Connect to your console, and enter these commands to Install Jenkins

vi jenkins.sh #make sure run in Root (or) add at userdata while ec2 launch

#!/bin/bash

sudo apt update -y

#sudo apt upgrade -y

wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keyrings/adoptium.asc

echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list

sudo apt update -y

sudo apt install temurin-17-jdk -y

/usr/bin/java --version

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee \

/usr/share/keyrings/jenkins-keyring.asc > /dev/null

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

sudo apt-get update -y

sudo apt-get install jenkins -y

sudo systemctl start jenkins

sudo systemctl status jenkins

sudo chmod 777 jenkins.sh

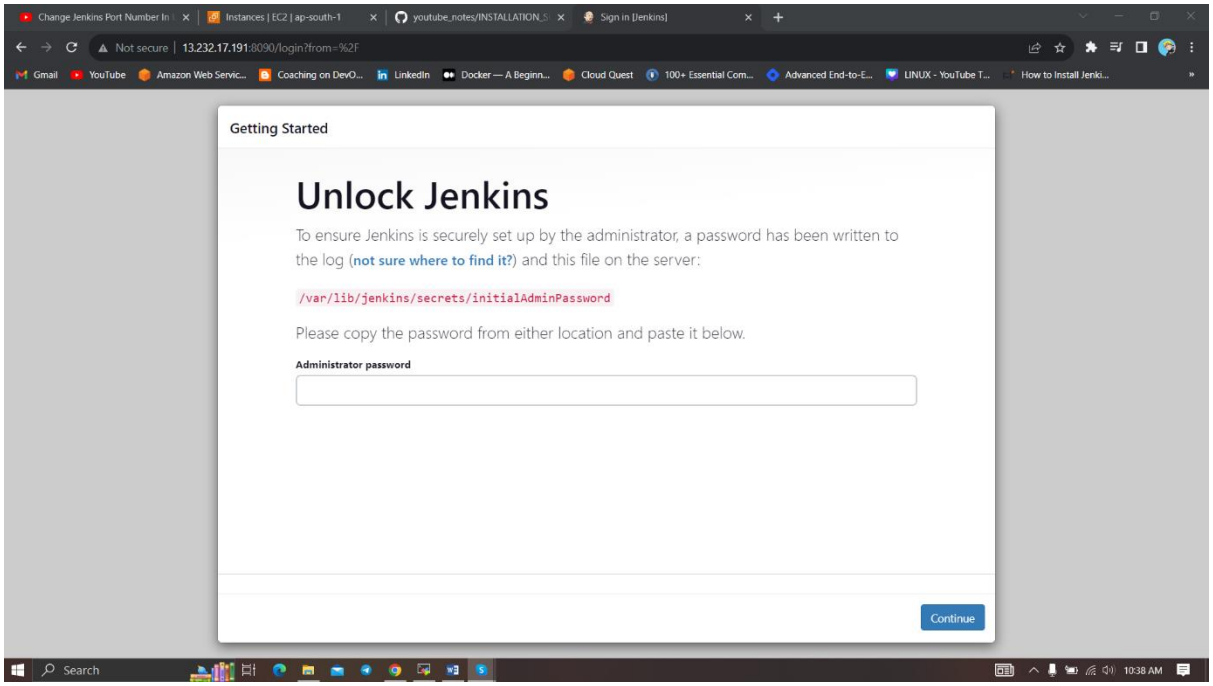
./jenkins.sh # this will installl jenkins

Once Jenkins is installed, you will need to go to your AWS EC2 Security Group and open Inbound Port 8080, since Jenkins works on Port 8080.

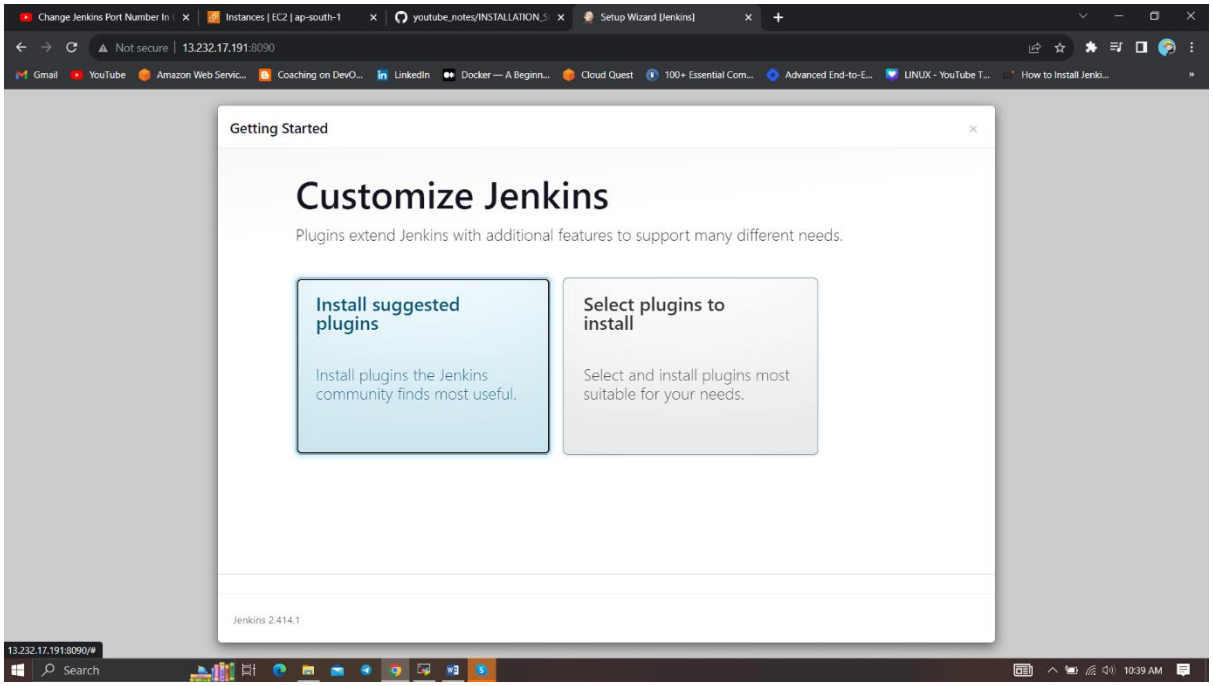
Now, grab your Public IP Address

<EC2 Public IP Address:8080>

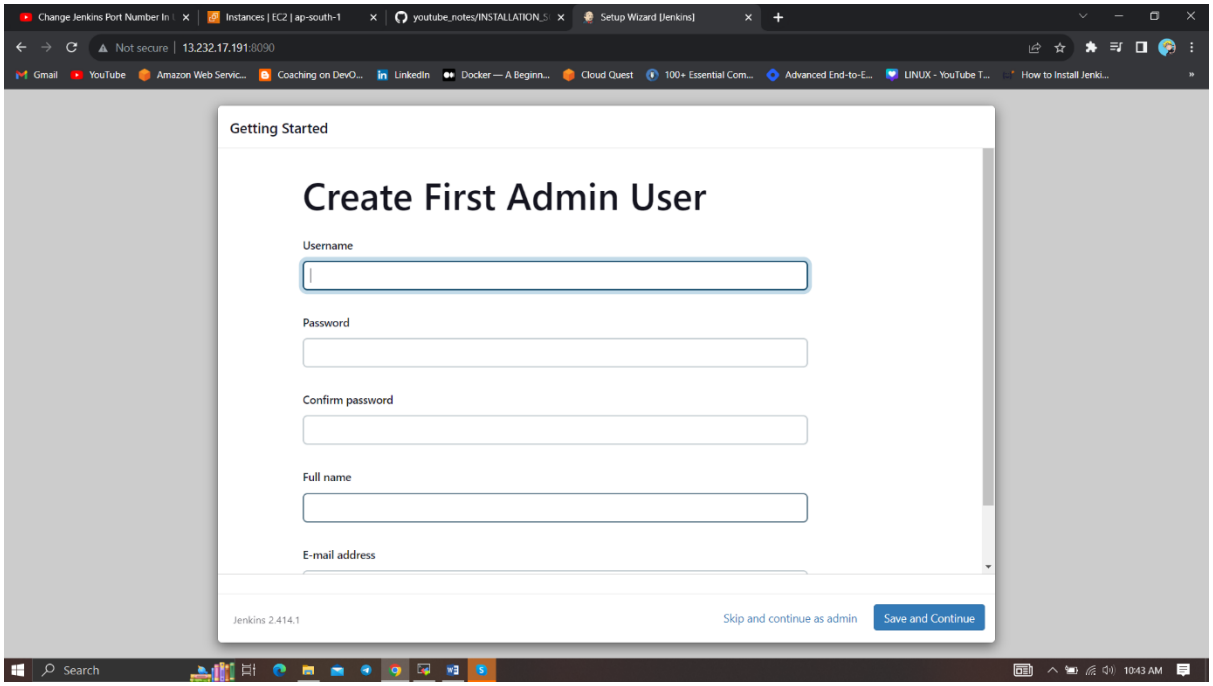
sudo cat /var/lib/jenkins/secrets/initialAdminPassword



Unlock Jenkins using an administrative password and install the suggested plugins.

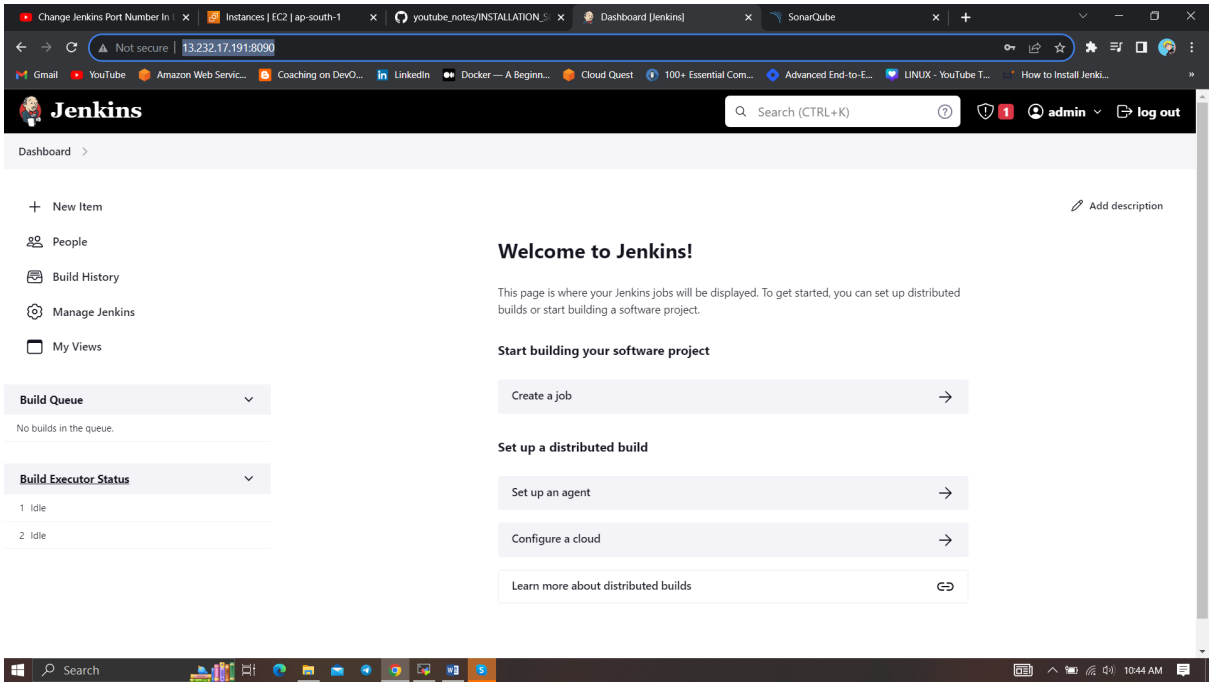


Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

Jenkins Getting Started Screen.



2B — Install Docker

sudo apt-get update

sudo apt-get install docker.io -y

sudo usermod -aG docker $USER #my case is ubuntu

newgrp docker

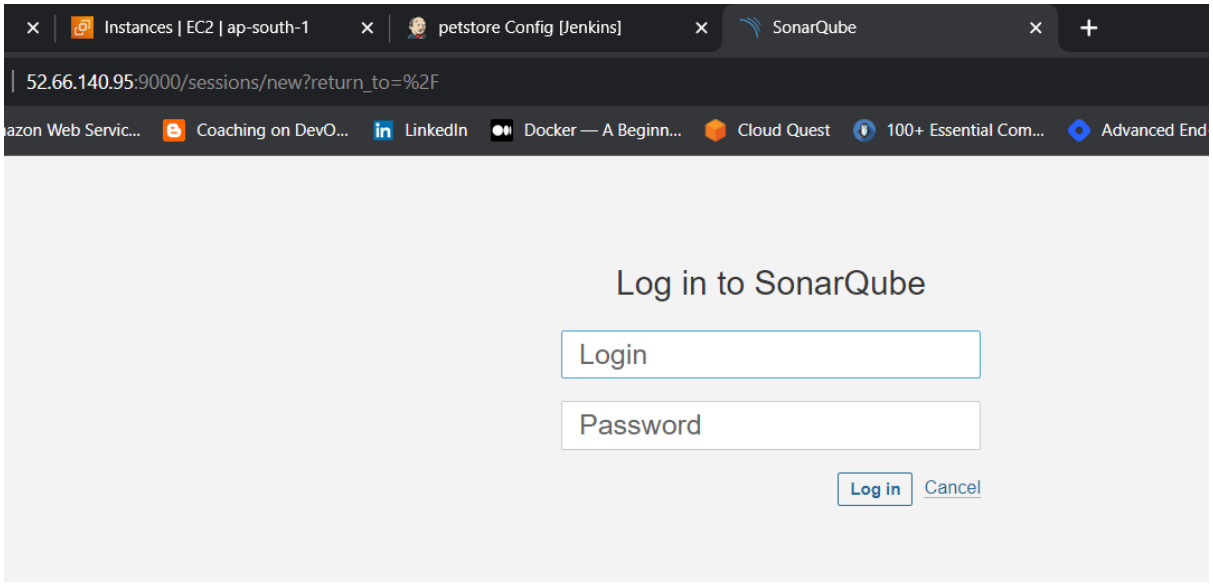
sudo chmod 777 /var/run/docker.sock

After the docker installation, we create a sonarqube container (Remember to add 9000 ports in the security group).

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

```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882f8eb: Pull complete
2cabec57fa36: Pull complete
c20481384b6a: Pull complete
bf7b17ee74f8: Pull complete
38617faac714: Pull complete
706f20f58f5e: Pull complete
65a29568c257: Pull complete
Digest: sha256:1a118f8ab960d6c3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436af7f752fdb04993092d0ca3065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID   IMAGE                       COMMAND                  CREATED         STATUS         PORTS                                                NAMES
4b60c96bf9ad   sonarqube:lts-community   "/opt/sonarqube/dock…"   9 seconds ago   Up 5 seconds   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   sonar
ubuntu@ip-172-31-42-253:~$
```

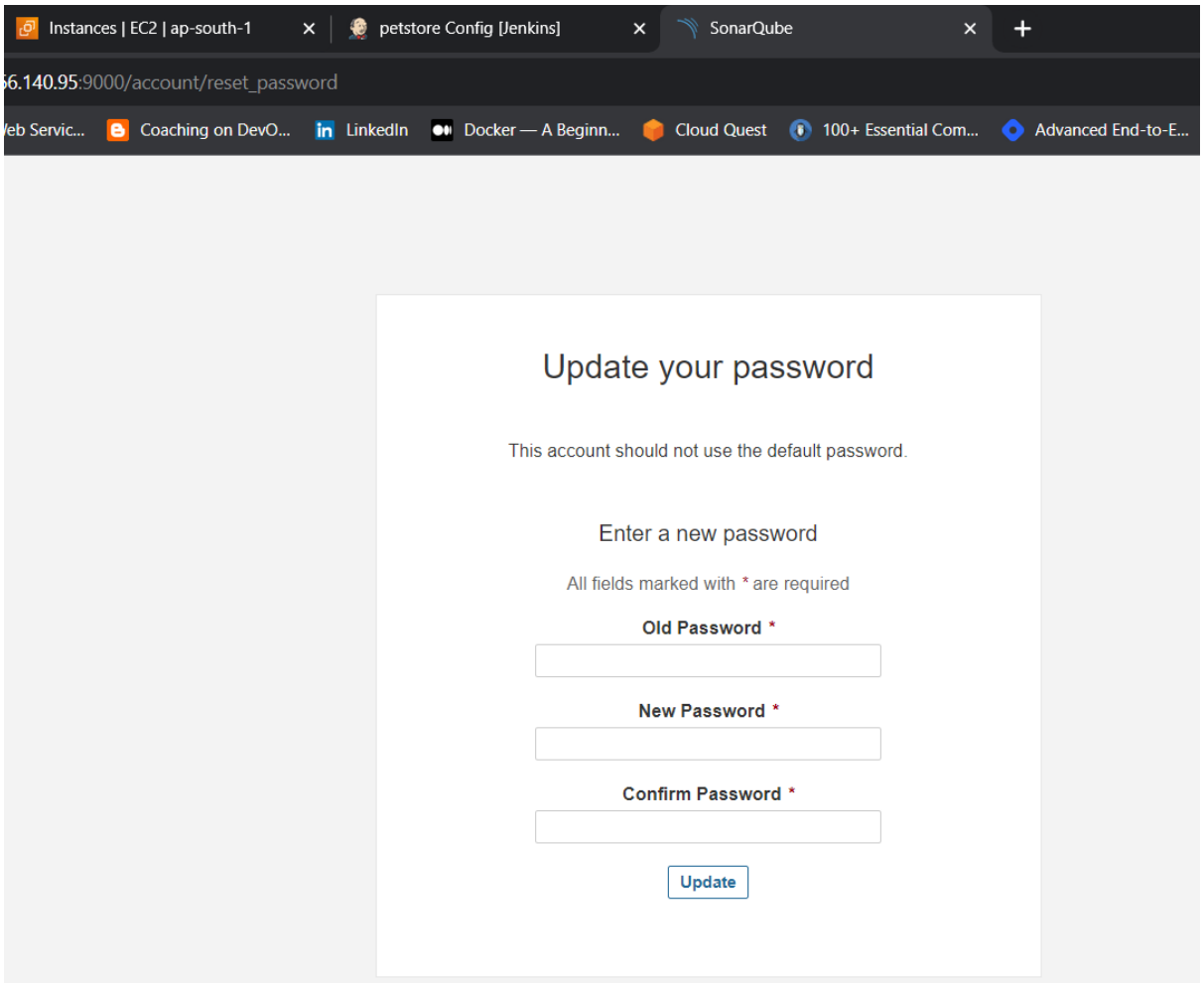
Now our sonarqube is up and running



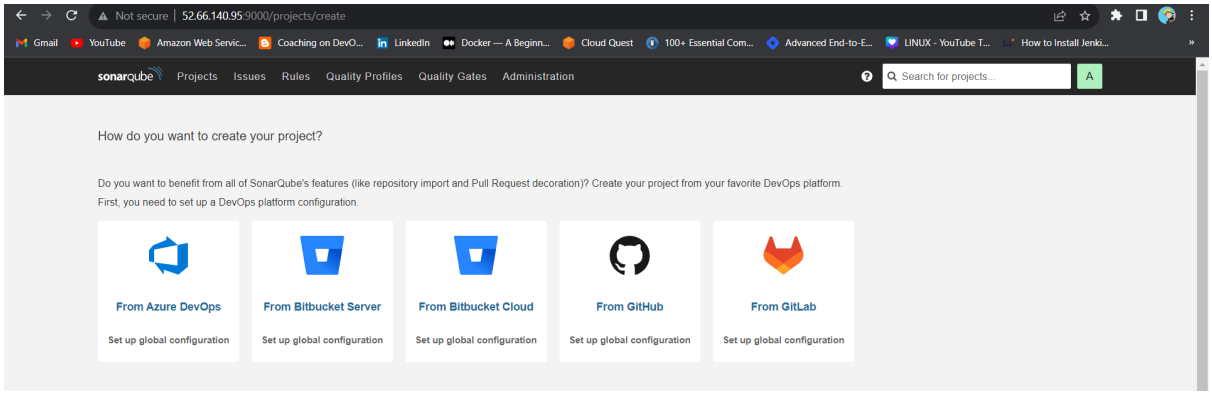
Enter username and password, click on login and change password

username admin

password admin



Update New password, This is Sonar Dashboard.



2C — Install Trivy

vi trivy.sh

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

wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null

echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list

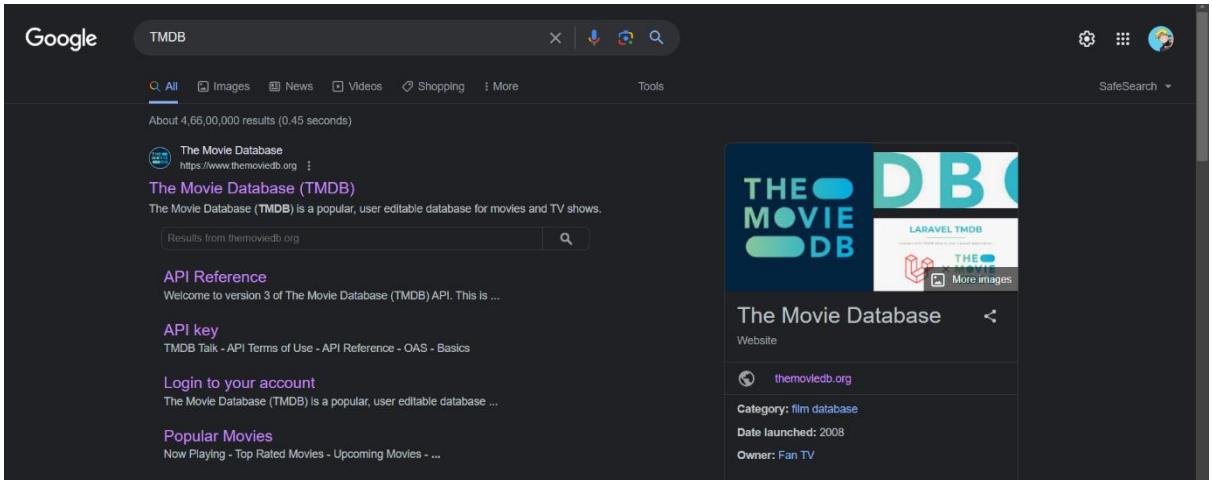
sudo apt-get update

sudo apt-get install trivy -y

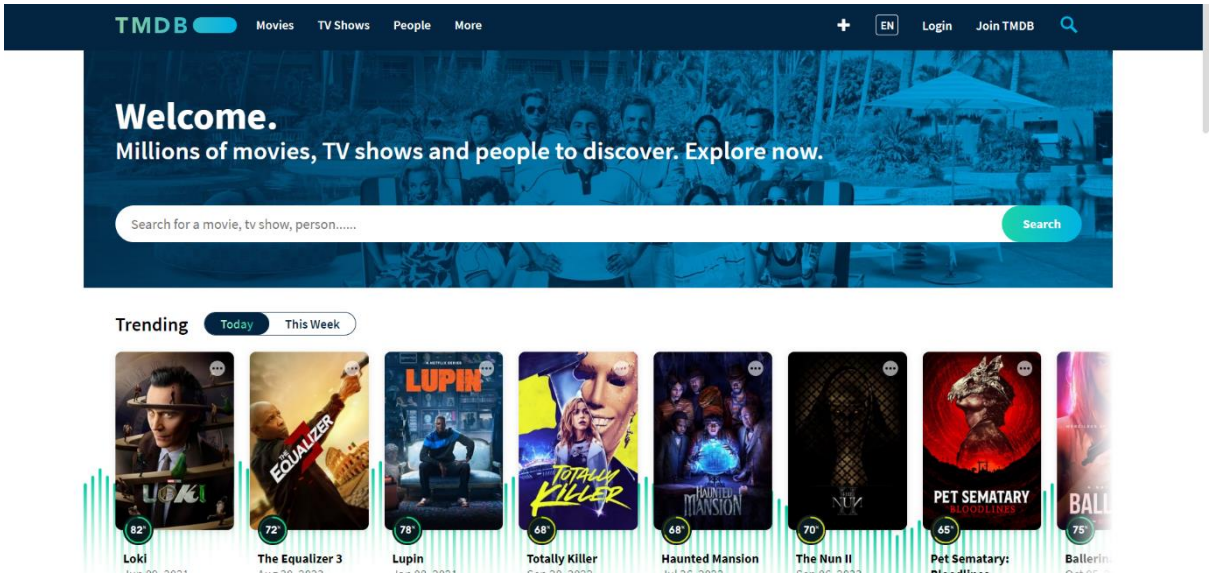
Step 3: Create a TMDB API Key

Next, we will create a TMDB API key

Open a new tab in the Browser and search for TMDB

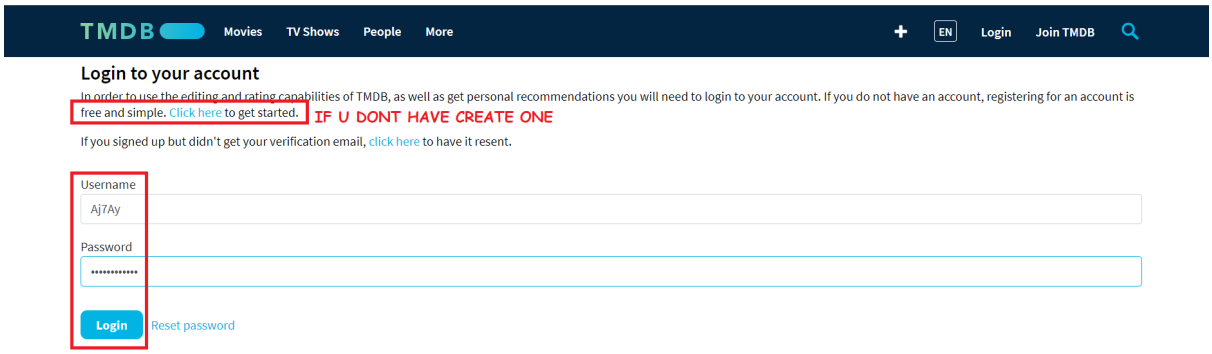


Click on the first result, you will see this page

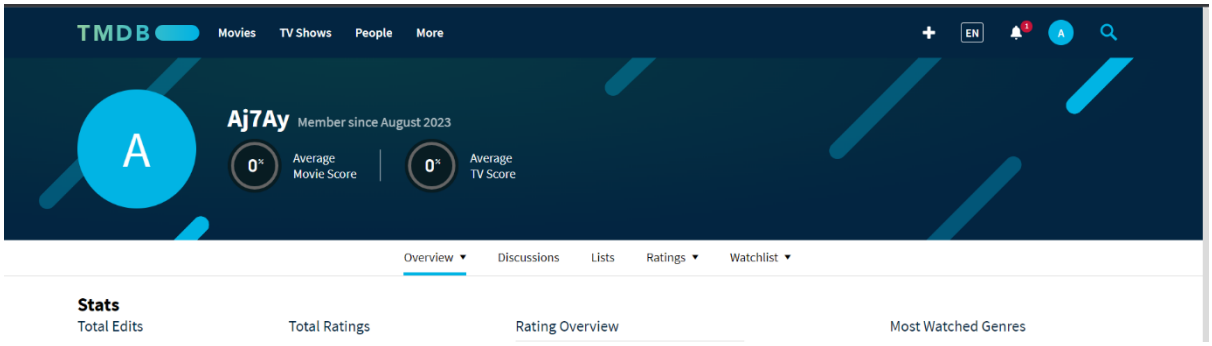


Click on the Login on the top right. You will get this page.

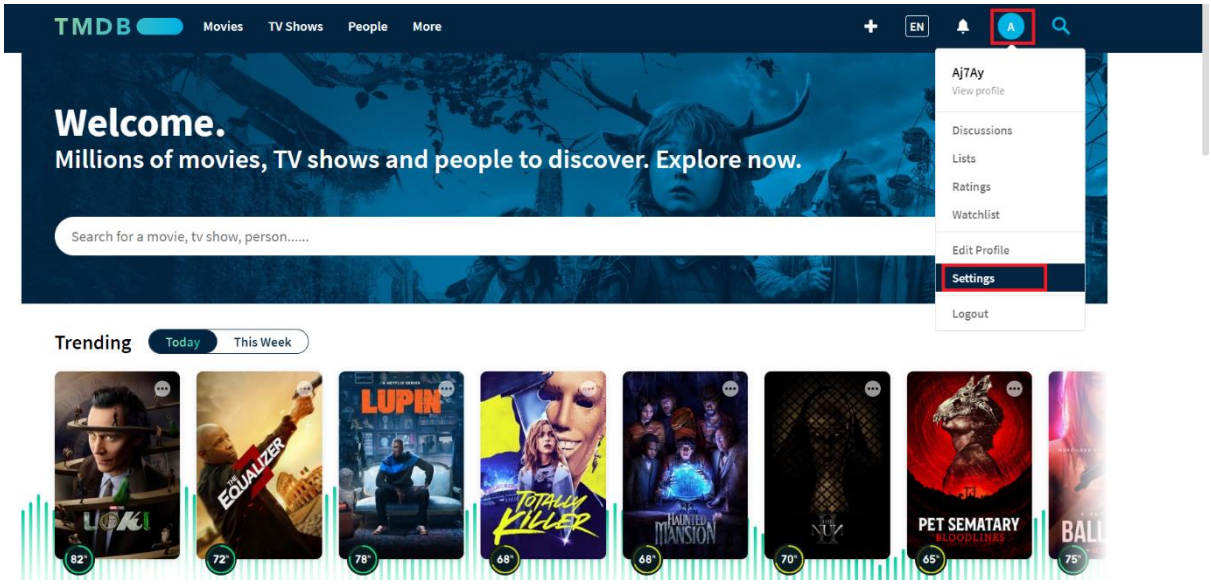
You need to create an account here. click on click here. I have account that's why i added my details there.



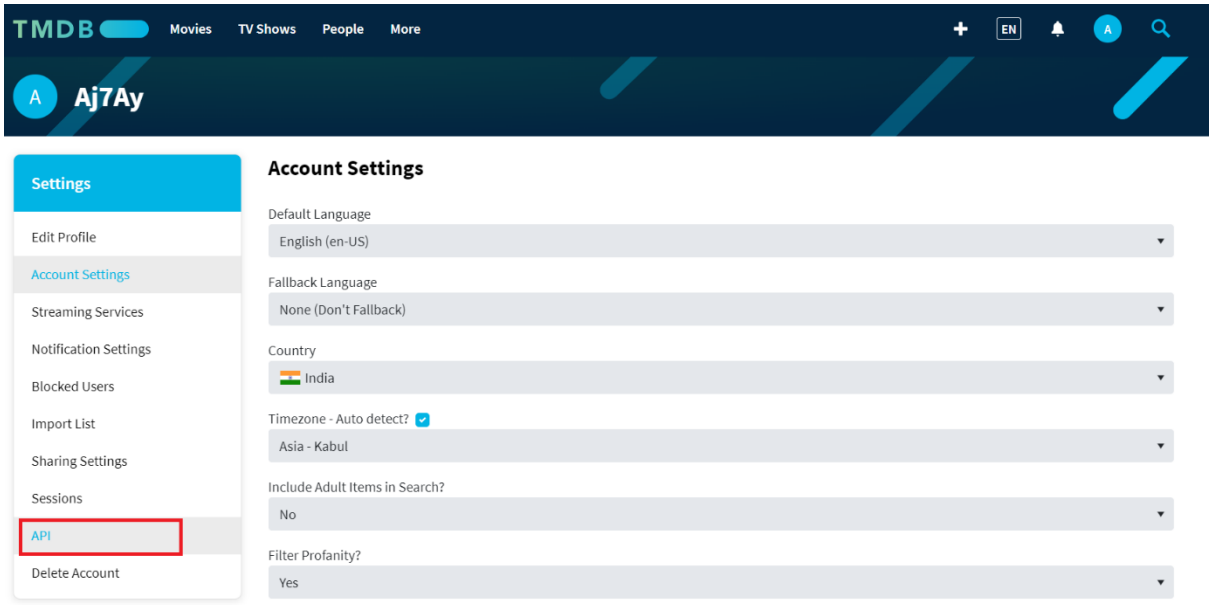
once you create an account you will see this page.



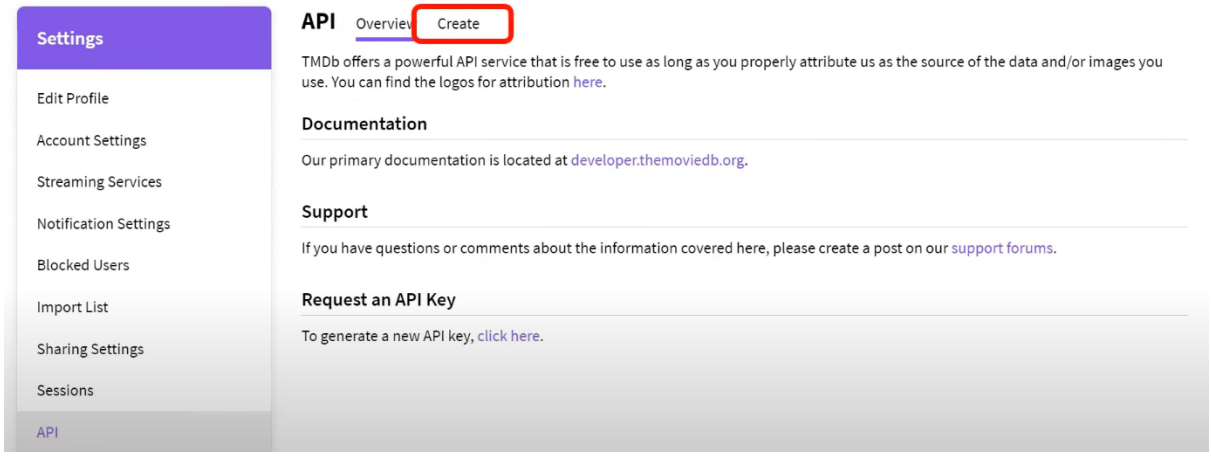
Let's create an API key, By clicking on your profile and clicking settings.



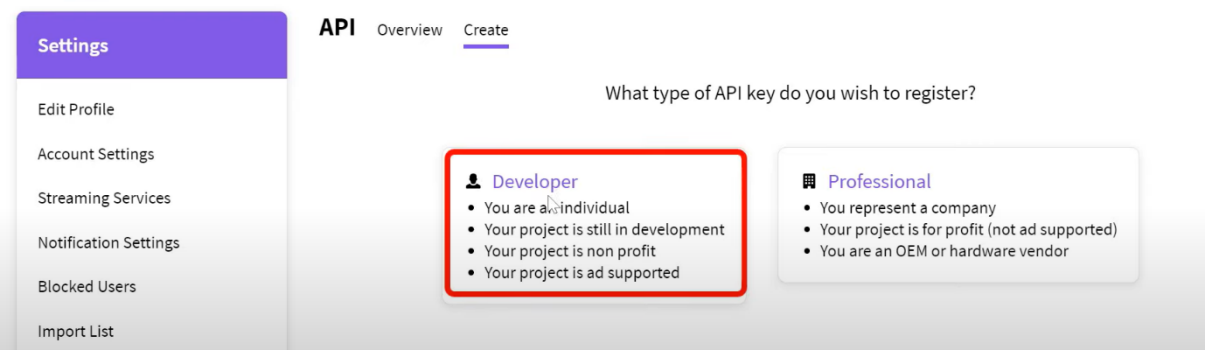
Now click on API from the left side panel.



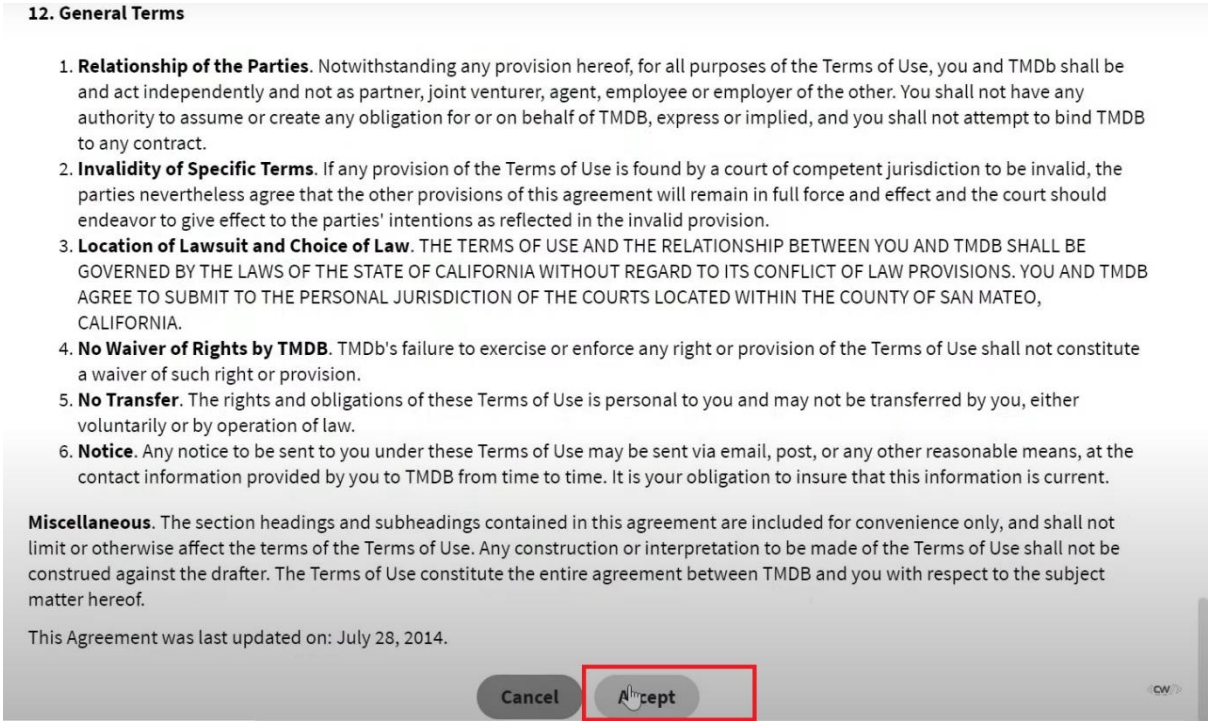
Now click on create



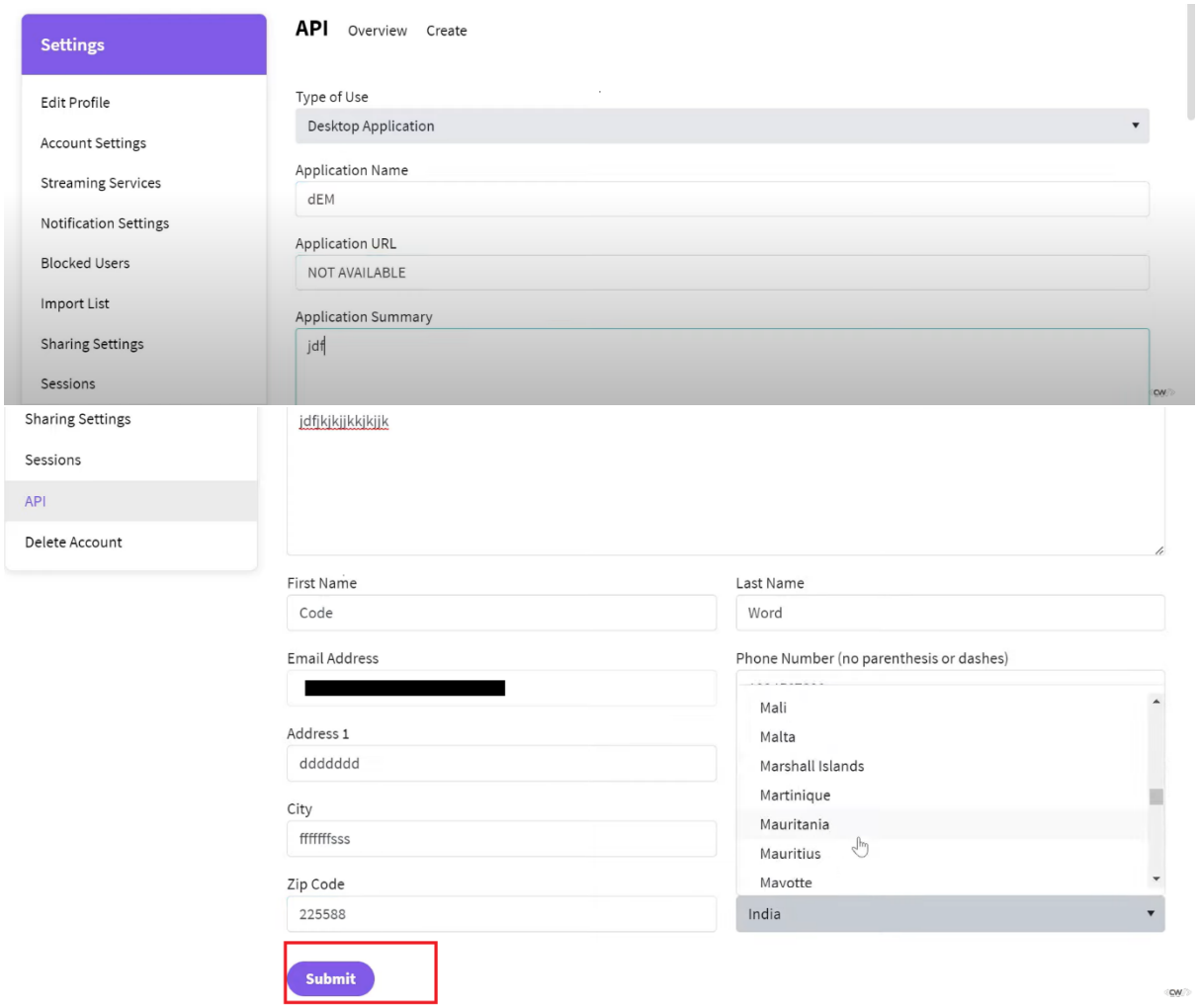
Click on Developer



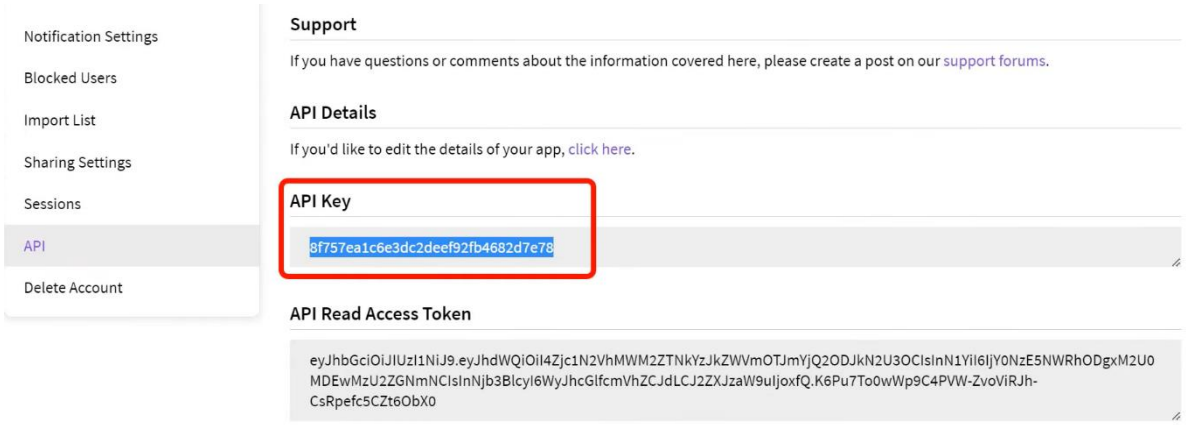
Now you have to accept the terms and conditions.



Provide basic details



Click on submit and you will get your API key.



Step 4 — Install Prometheus and Grafana On the new Server

First of all, let's create a dedicated Linux user sometimes called a system account for Prometheus. Having individual users for each service serves two main purposes:

It is a security measure to reduce the impact in case of an incident with the service.

It simplifies administration as it becomes easier to track down what resources belong to which service.

To create a system user or system account, run the following command:

sudo useradd \

--system \

--no-create-home \

--shell /bin/false prometheus

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo useradd \
    --system \
    --no-create-home \
    --shell /bin/false prometheus
ubuntu@ip-172-31-38-156:~$
```

–system – Will create a system account.
–no-create-home – We don't need a home directory for Prometheus or any other system accounts in our case.
–shell /bin/false – It prevents logging in as a Prometheus user.
Prometheus – Will create a Prometheus user and a group with the same name.

Let's check the latest version of Prometheus from the **download page**.

You can use the curl or wget command to download Prometheus.

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

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ wget https://github.com/prometheus/prometheus/releases/download/v2.47.1/prometheus-2.47.1.linux-amd64.tar.gz
--2023-10-06 08:51:59--  https://github.com/prometheus/prometheus/releases/download/v2.47.1/prometheus-2.47.1.linux-amd64.tar.gz
Resolving github.com (github.com)... 20.207.73.82
Connecting to github.com (github.com)|20.207.73.82|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/2f9b7b37-63a0-428b-adb5-0294482fd743?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=
IAIWNJYAX4CSVEH53A%2F20231006%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20231006T085159Z&X-Amz-Expires=300&X-Amz-Signature=6962d584d4eaad7d1a082e0285936e53a7a6976aa6f71b5d51765dbfe
6a1c&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=6838921&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.47.1.linux-amd64.tar.gz&response-content-type=
lication%2Foctet-stream [following]
--2023-10-06 08:51:59--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/2f9b7b37-63a0-428b-adb5-0294482fd743?X-Amz-Algorithm=AWS4-HMAC-SHA256&X
z-Credential=AKIAIWNJYAX4CSVEH53A%2F20231006%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20231006T085159Z&X-Amz-Expires=300&X-Amz-Signature=6962d584d4eaad7d1a082e0285936e53a7a6976aa6
b5d51765dbfe36e6a1c&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=6838921&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.47.1.linux-amd64.tar.gz&response
ontent-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 95713066 (91M) [application/octet-stream]
Saving to: 'prometheus-2.47.1.linux-amd64.tar.gz'

prometheus-2.47.1.linux-amd64.tar.gz          100%[=========================================================================================>]  91.28M  4.01MB/s    in 88s

2023-10-06 08:53:28 (1.03 MB/s) - 'prometheus-2.47.1.linux-amd64.tar.gz' saved [95713066/95713066]

ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ ls
prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-38-156:~$
```

Then, we need to extract all Prometheus files from the archive.

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

```
ubuntu@ip-172-31-38-156:~$ ls
prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ tar -xvf prometheus-2.47.1.linux-amd64.tar.gz
prometheus-2.47.1.linux-amd64/
prometheus-2.47.1.linux-amd64/LICENSE
prometheus-2.47.1.linux-amd64/NOTICE
prometheus-2.47.1.linux-amd64/prometheus.yml
prometheus-2.47.1.linux-amd64/consoles/
prometheus-2.47.1.linux-amd64/consoles/prometheus.html
prometheus-2.47.1.linux-amd64/consoles/prometheus-overview.html
prometheus-2.47.1.linux-amd64/consoles/node-cpu.html
prometheus-2.47.1.linux-amd64/consoles/index.html.example
prometheus-2.47.1.linux-amd64/consoles/node.html
prometheus-2.47.1.linux-amd64/consoles/node-disk.html
prometheus-2.47.1.linux-amd64/consoles/node-overview.html
prometheus-2.47.1.linux-amd64/promtool
prometheus-2.47.1.linux-amd64/console_libraries/
prometheus-2.47.1.linux-amd64/console_libraries/prom.lib
prometheus-2.47.1.linux-amd64/console_libraries/menu.lib
prometheus-2.47.1.linux-amd64/prometheus
ubuntu@ip-172-31-38-156:~$
```

Usually, you would have a disk mounted to the data directory. For this tutorial, I will simply create a /data directory. Also, you need a folder for Prometheus configuration files.

sudo mkdir -p /data /etc/prometheus

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo mkdir -p /data /etc/prometheus
ubuntu@ip-172-31-38-156:~$
```

Now, let's change the directory to Prometheus and move some files.

cd prometheus-2.47.1.linux-amd64/

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ cd prometheus-2.47.1.linux-amd64/
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$ ls -l
total 236852
-rw-r--r-- 1 ubuntu ubuntu     11357 Oct  4 11:05 LICENSE
-rw-r--r-- 1 ubuntu ubuntu      3773 Oct  4 11:05 NOTICE
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct  4 11:05 console_libraries
drwxr-xr-x 2 ubuntu ubuntu      4096 Oct  4 11:05 consoles
-rwxr-xr-x 1 ubuntu ubuntu 124158156 Oct  4 10:35 prometheus
-rw-r--r-- 1 ubuntu ubuntu       934 Oct  4 11:05 prometheus.yml
-rwxr-xr-x 1 ubuntu ubuntu 118343283 Oct  4 10:38 promtool
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
```

First of all, let's move the Prometheus binary and a promtool to the /usr/local/bin/. promtool is used to check configuration files and Prometheus rules.

sudo mv prometheus promtool /usr/local/bin/

```
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$ sudo mv prometheus promtool /usr/local/bin/
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
```

Optionally, we can move console libraries to the Prometheus configuration directory. Console templates allow for the creation of arbitrary consoles using the Go templating language. You don't need to worry about it if you're just getting started.

sudo mv consoles/ console_libraries/ /etc/prometheus/

```
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$ sudo mv consoles/ console_libraries/ /etc/prometheus/
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
```

Finally, let's move the example of the main Prometheus configuration file.

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

```
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$ sudo mv prometheus.yml /etc/prometheus/prometheus.yml
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
```

To avoid permission issues, you need to set the correct ownership for the /etc/prometheus/ and data directory.

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

```
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$ sudo chown -R prometheus:prometheus /etc/prometheus/ /data/
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
```

You can delete the archive and a Prometheus folder when you are done.

cd

rm -rf prometheus-2.47.1.linux-amd64.tar.gz

```
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$
ubuntu@ip-172-31-38-156:~/prometheus-2.47.1.linux-amd64$ cd ..
ubuntu@ip-172-31-38-156:~$ ll
total 93516
drwxr-x--- 5 ubuntu ubuntu     4096 Oct  6 08:54 ./
drwxr-xr-x 3 root   root       4096 Oct  6 08:49 ../
-rw------- 1 ubuntu ubuntu       62 Oct  6 08:50 .Xauthority
-rw-r--r-- 1 ubuntu ubuntu      220 Jan  6  2022 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu     3771 Jan  6  2022 .bashrc
drwx------ 2 ubuntu ubuntu     4096 Oct  6 08:50 .cache/
-rw-r--r-- 1 ubuntu ubuntu      807 Jan  6  2022 .profile
drwx------ 2 ubuntu ubuntu     4096 Oct  6 08:49 .ssh/
-rw-r--r-- 1 ubuntu ubuntu        0 Oct  6 08:50 .sudo_as_admin_successful
-rw-rw-r-- 1 ubuntu ubuntu      165 Oct  6 08:53 .wget-hsts
drwxr-xr-x 2 ubuntu ubuntu     4096 Oct  6 08:56 prometheus-2.47.1.linux-amd64/
-rw-rw-r-- 1 ubuntu ubuntu 95713066 Oct  4 11:15 prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ rm -rf prometheus-2.47.1.linux-amd64.tar.gz
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ ls
prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-38-156:~$
```

Verify that you can execute the Prometheus binary by running the following command:

prometheus --version

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ prometheus --version
prometheus, version 2.47.1 (branch: HEAD, revision: c4d1a8beff37cc004f1dc4ab9d2e73193f51aaeb)
  build user:       root@4829330363be
  build date:       20231004-10:31:16
  go version:       go1.21.1
  platform:         linux/amd64
  tags:             netgo,builtinassets,stringlabels
ubuntu@ip-172-31-38-156:~$
```

To get more information and configuration options, run Prometheus Help.

prometheus --help

We're going to use some of these options in the service definition.

We're going to use Systemd, which is a system and service manager for Linux operating systems. For that, we need to create a Systemd unit configuration file.

sudo vim /etc/systemd/system/prometheus.service

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo vim /etc/systemd/system/prometheus.service
```

Prometheus.service

[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

```
[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
```

Let's go over a few of the most important options related to Systemd and Prometheus. Restart – Configures whether the service shall be restarted when the service process exits, is killed, or a timeout is reached.
RestartSec – Configures the time to sleep before restarting a service.
User and Group – Are Linux user and a group to start a Prometheus process.
–config.file=/etc/prometheus/prometheus.yml – Path to the main Prometheus configuration file.
–storage.tsdb.path=/data – Location to store Prometheus data.
–web.listen-address=0.0.0.0:9090 – Configure to listen on all network interfaces. In some situations, you may have a proxy such as nginx to redirect requests to Prometheus. In that case, you would configure Prometheus to listen only on **localhost**.
–web.enable-lifecycle — Allows to manage Prometheus, for example, to reload configuration without restarting the service.

To automatically start the Prometheus after reboot, run enable.

sudo systemctl enable prometheus

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl enable prometheus
Created symlink /etc/systemd/system/multi-user.target.wants/prometheus.service → /etc/systemd/system/prometheus.service.
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
```

Then just start the Prometheus.

sudo systemctl start prometheus

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl start prometheus
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
```

To check the status of Prometheus run the following command:

sudo systemctl status prometheus

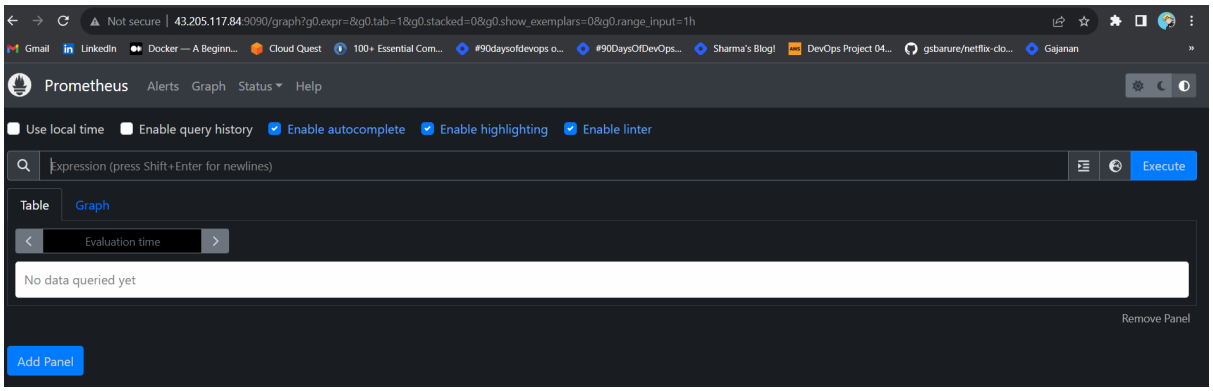
```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl status prometheus
● prometheus.service - Prometheus
     Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-10-06 09:00:39 UTC; 13s ago
   Main PID: 1941 (prometheus)
      Tasks: 6 (limit: 1141)
     Memory: 16.2M
        CPU: 64ms
     CGroup: /system.slice/prometheus.service
             └─1941 /usr/local/bin/prometheus --config.file=/etc/prometheus/prometheus.yml --storage.tsdb.path=/data --web.console.templates=/etc/prometheus/consoles --web.console.lib

Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.798Z caller=tls_config.go:274 level=info component=web msg="Listening on" address=[::]:9090
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.798Z caller=tls_config.go:277 level=info component=web msg="TLS is disabled." http2=false address=[::]:9090
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.798Z caller=head.go:760 level=info component=tsdb msg="WAL segment loaded" segment=0 maxSegment=0
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.798Z caller=head.go:797 level=info component=tsdb msg="WAL replay completed" checkpoint_replay_duration=42.05
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.801Z caller=main.go:1045 level=info fs_type=EXT4_SUPER_MAGIC
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.801Z caller=main.go:1048 level=info msg="TSDB started"
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.801Z caller=main.go:1229 level=info msg="Loading configuration file" filename=/etc/prometheus/prometheus.yml
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.805Z caller=main.go:1266 level=info msg="Completed loading of configuration file" filename=/etc/prometheus/pr
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.805Z caller=main.go:1009 level=info msg="Server is ready to receive web requests."
Oct 06 09:00:39 ip-172-31-38-156 prometheus[1941]: ts=2023-10-06T09:00:39.805Z caller=manager.go:1009 level=info component="rule manager" msg="Starting rule manager..."
lines 1-20/20 (END)
```

Suppose you encounter any issues with Prometheus or are unable to start it. The easiest way to find the problem is to use the journalctl command and search for errors.

journalctl -u prometheus -f --no-pager

Now we can try to access it via the browser. I'm going to be using the IP address of the Ubuntu server. You need to append port 9090 to the IP.

<public-ip:9090>



If you go to targets, you should see only one – Prometheus target. It scrapes itself every 15 seconds by default.

Install Node Exporter on Ubuntu 22.04

Next, we're going to set up and configure Node Exporter to collect Linux system metrics like CPU load and disk I/O. Node Exporter will expose these as Prometheus-style metrics. Since the installation process is very similar, I'm not going to cover as deep as Prometheus.

First, let's create a system user for Node Exporter by running the following command:

sudo useradd \

--system \

--no-create-home \

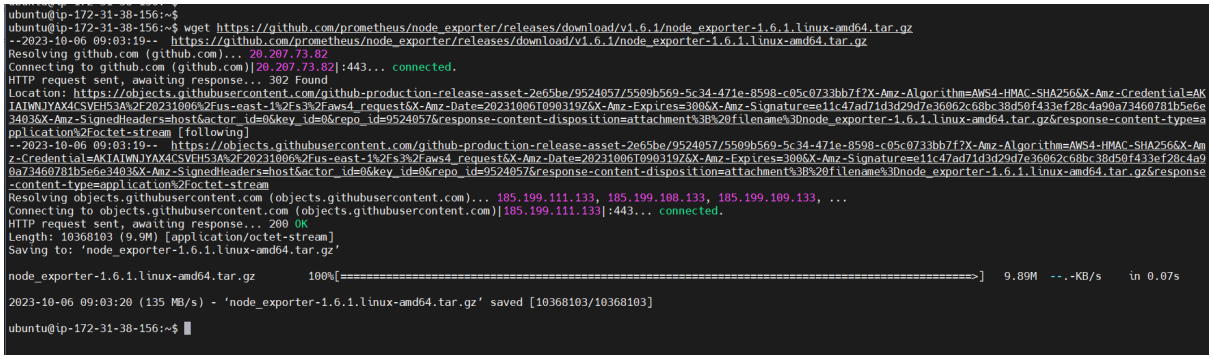
--shell /bin/false node_exporter

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo useradd \
    --system \
    --no-create-home \
    --shell /bin/false node_exporter
ubuntu@ip-172-31-38-156:~$
```

You can **download Node Exporter** from the same page.

Use the wget command to download the binary.

wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz



Extract the node exporter from the archive.

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

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ ls
node_exporter-1.6.1.linux-amd64.tar.gz  prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ tar -xvf node_exporter-1.6.1.linux-amd64.tar.gz
node_exporter-1.6.1.linux-amd64/
node_exporter-1.6.1.linux-amd64/NOTICE
node_exporter-1.6.1.linux-amd64/node_exporter
node_exporter-1.6.1.linux-amd64/LICENSE
ubuntu@ip-172-31-38-156:~$
```

Move binary to the /usr/local/bin.

sudo mv \

node_exporter-1.6.1.linux-amd64/node_exporter \

/usr/local/bin/

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo mv \
  node_exporter-1.6.1.linux-amd64/node_exporter \
  /usr/local/bin/
ubuntu@ip-172-31-38-156:~$
```

Clean up, and delete node_exporter archive and a folder.

rm -rf node_exporter*

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ ls
node_exporter-1.6.1.linux-amd64  node_exporter-1.6.1.linux-amd64.tar.gz  prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ rm -rf node_exporter*
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ ls
prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-38-156:~$
```

Verify that you can run the binary.

node_exporter --version

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ node_exporter --version
node_exporter, version 1.6.1 (branch: HEAD, revision: 4a1b77600c1873a8233f3ffb55afcedbb63b8d84)
  build user:       root@586879db11e5
  build date:       20230717-12:10:52
  go version:       go1.20.6
  platform:         linux/amd64
  tags:             netgo osusergo static_build
ubuntu@ip-172-31-38-156:~$
```

Node Exporter has a lot of plugins that we can enable. If you run Node Exporter help you will get all the options.

node_exporter --help

–collector.logind We’re going to enable the login controller, just for the demo.

Next, create a similar systemd unit file.

sudo vim /etc/systemd/system/node_exporter.service

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo vim /etc/systemd/system/node_exporter.service
```

node_exporter.service

[Unit]

Description=Node Exporter

Wants=network-online.target

After=network-online.target

StartLimitIntervalSec=500

StartLimitBurst=5

[Service]

User=node_exporter

Group=node_exporter

Type=simple

Restart=on-failure

RestartSec=5s

ExecStart=/usr/local/bin/node_exporter \

  --collector.logind

[Install]

WantedBy=multi-user.target

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

StartLimitIntervalSec=500
StartLimitBurst=5

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

[Install]
WantedBy=multi-user.target
```

Replace Prometheus user and group to node_exporter, and update the ExecStart command.

To automatically start the Node Exporter after reboot, enable the service.

sudo systemctl enable node_exporter

Then start the Node Exporter.

sudo systemctl start node_exporter

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl enable node_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node_exporter.service → /etc/systemd/system/node_exporter.service.
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl start node_exporter
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
```

Check the status of Node Exporter with the following command:

sudo systemctl status node_exporter

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ 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 Fri 2023-10-06 09:07:39 UTC; 8s ago
   Main PID: 2030 (node_exporter)
      Tasks: 3 (limit: 1141)
     Memory: 2.0M
        CPU: 9ms
     CGroup: /system.slice/node_exporter.service
             └─2030 /usr/local/bin/node_exporter --collector.logind

Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=thermal_zone
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=time
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=timex
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=udp_queues
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=uname
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=vmstat
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=xfs
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=zfs
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.415Z caller=tls_config.go:274 level=info msg="Listening on" address=[::]:9100
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.415Z caller=tls_config.go:277 level=info msg="TLS is disabled." http2=false address=[::]:9100
ubuntu@ip-172-31-38-156:~$
```

If you have any issues, check logs with journalctl

journalctl -u node_exporter -f --no-pager

At this point, we have only a single target in our Prometheus. There are many different service discovery mechanisms built into Prometheus. For example, Prometheus can dynamically discover targets in AWS, GCP, and other clouds based on the labels. In the following tutorials, I'll give you a few examples of deploying Prometheus in a cloud-specific environment. For this tutorial, let's keep it simple and keep adding static targets. Also, I have a lesson on how to deploy and manage Prometheus in the Kubernetes cluster.

To create a static target, you need to add job_name with static_configs.

sudo vim /etc/prometheus/prometheus.yml

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo vim /etc/prometheus/prometheus.yml
ubuntu@ip-172-31-38-156:~$
```

prometheus.yml

- job_name: node_export

  static_configs:

   - targets: ["localhost:9100"]

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

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

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

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

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

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: node_export
    static_configs:
      - targets: ["localhost:9100"]

~
```

By default, Node Exporter will be exposed on port 9100.

Since we enabled lifecycle management via API calls, we can reload the Prometheus config without restarting the service and causing downtime.

Before, restarting check if the config is valid.

promtool check config /etc/prometheus/prometheus.yml

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ promtool check config /etc/prometheus/prometheus.yml
Checking /etc/prometheus/prometheus.yml
 SUCCESS: /etc/prometheus/prometheus.yml is valid prometheus config file syntax

ubuntu@ip-172-31-38-156:~$
```

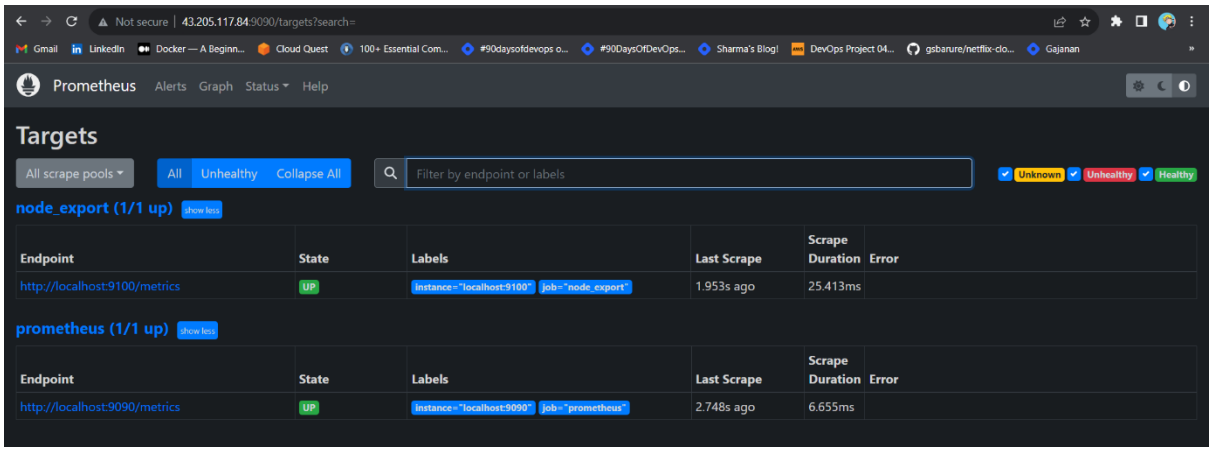
Then, you can use a POST request to reload the config.

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

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ curl -X POST http://localhost:9090/-/reload
ubuntu@ip-172-31-38-156:~$
```

Check the targets section

http://<ip>:9090/targets



Install Grafana on Ubuntu 22.04

To visualize metrics we can use Grafana. There are many different data sources that Grafana supports, one of them is Prometheus.

First, let's make sure that all the dependencies are installed.

sudo apt-get install -y apt-transport-https software-properties-common

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo apt-get install -y apt-transport-https software-properties-common
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  python3-software-properties
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  python3-software-properties software-properties-common
2 upgraded, 1 newly installed, 0 to remove and 127 not upgraded.
Need to get 44.4 kB of archives.
After this operation, 169 kB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.10 [1510 B]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 software-properties-common all 0.99.22.7 [14.1 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-software-properties all 0.99.22.7 [28.8 kB]
Fetched 44.4 kB in 0s (2002 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 64295 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.4.10_all.deb ...
Unpacking apt-transport-https (2.4.10) ...
Preparing to unpack .../software-properties-common_0.99.22.7_all.deb ...
Unpacking software-properties-common (0.99.22.7) over (0.99.22.6) ...
Preparing to unpack .../python3-software-properties_0.99.22.7_all.deb ...
Unpacking python3-software-properties (0.99.22.7) over (0.99.22.6) ...
Setting up apt-transport-https (2.4.10) ...
Setting up python3-software-properties (0.99.22.7) ...
Setting up software-properties-common (0.99.22.7) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.
```

Next, add the GPG key.

wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).
OK
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
```

Add this repository for stable releases.

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

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list
deb https://packages.grafana.com/oss/deb stable main
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
```

After you add the repository, update and install Garafana.

sudo apt-get update

sudo apt-get -y install grafana

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo apt-get -y install grafana
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core libfontconfig1 musl
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core grafana libfontconfig1 musl
0 upgraded, 5 newly installed, 0 to remove and 127 not upgraded.
Need to get 104 MB of archives.
After this operation, 379 MB of additional disk space will be used.
Get:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 fonts-dejavu-core all 2.37-2build1 [1041 kB]
Get:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 fontconfig-config all 2.13.1-4.2ubuntu5 [29.1 kB]
Get:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libfontconfig1 amd64 2.13.1-4.2ubuntu5 [131 kB]
Get:4 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 musl amd64 1.2.2-4 [407 kB]
Get:5 https://packages.grafana.com/oss/deb stable/main amd64 grafana amd64 10.1.4 [102 MB]
17% [5 grafana 0 B/102 MB 0%]
```

To automatically start the Grafana after reboot, enable the service.

sudo systemctl enable grafana-server

Then start the Grafana.

sudo systemctl start grafana-server

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl enable grafana-server
Synchronizing state of grafana-server.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable grafana-server
Created symlink /etc/systemd/system/multi-user.target.wants/grafana-server.service → /lib/systemd/system/grafana-server.service.
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl start grafana-server
ubuntu@ip-172-31-38-156:~$
```

To check the status of Grafana, run the following command:

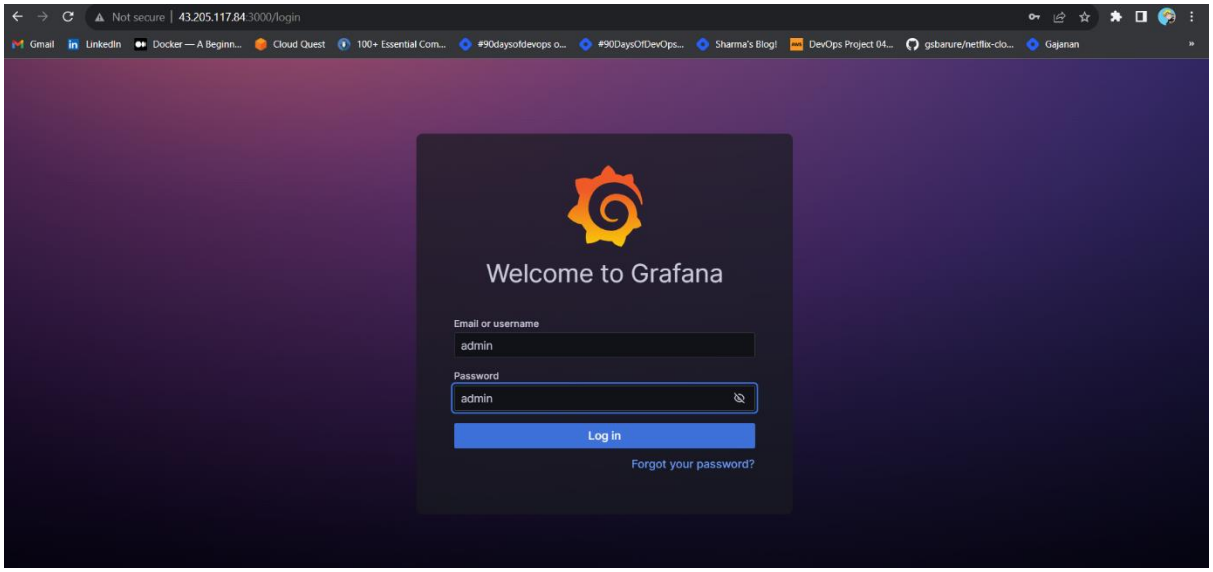
sudo systemctl status grafana-server

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ 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 Fri 2023-10-06 09:14:01 UTC; 7s ago
       Docs: http://docs.grafana.org
   Main PID: 3263 (grafana)
      Tasks: 5 (limit: 1141)
     Memory: 175.9M
        CPU: 4.330s
     CGroup: /system.slice/grafana-server.service
             └─3263 /usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:default.paths.logs=/var/log/g

Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=migrator t=2023-10-06T09:14:06.750063027Z level=info msg="Executing migration" id="Add unique index for folder.title and folder.
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=migrator t=2023-10-06T09:14:06.757199101Z level=info msg="migrations completed" performed=497 skipped=0 duration=4.148747644s
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=sqlstore t=2023-10-06T09:14:06.774714527Z level=info msg="Created default admin" user=admin
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=sqlstore t=2023-10-06T09:14:06.775163061Z level=info msg="Created default organization"
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=secrets t=2023-10-06T09:14:06.782069114Z level=info msg="Envelope encryption state" enabled=true currentprovider=secretKey.v1
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=plugin.loader t=2023-10-06T09:14:06.883193007Z level=warn msg="Plugin missing module.js" pluginID=input warning="Missing module.
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=plugin.loader t=2023-10-06T09:14:06.88352314Z level=info msg="Plugin registered" pluginID=input
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=local.finder t=2023-10-06T09:14:06.883678317Z level=warn msg="Skipping finding plugins as directory does not exist" path=/var/li
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=query_data t=2023-10-06T09:14:06.890278741Z level=info msg="Query Service initialization"
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=live.push_http t=2023-10-06T09:14:06.897367677Z level=info msg="Live Push Gateway initialization"
lines 1-21/21 (END)
```

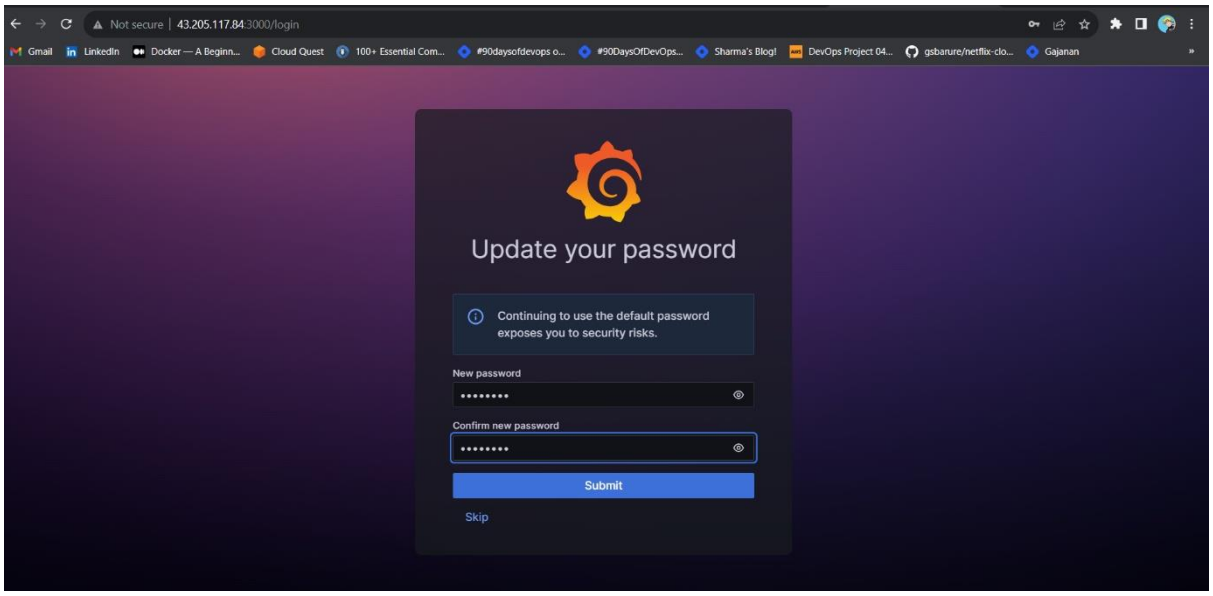
Go to http://<ip>:3000 and log in to the Grafana using default credentials. The username is admin, and the password is admin as well.

username admin

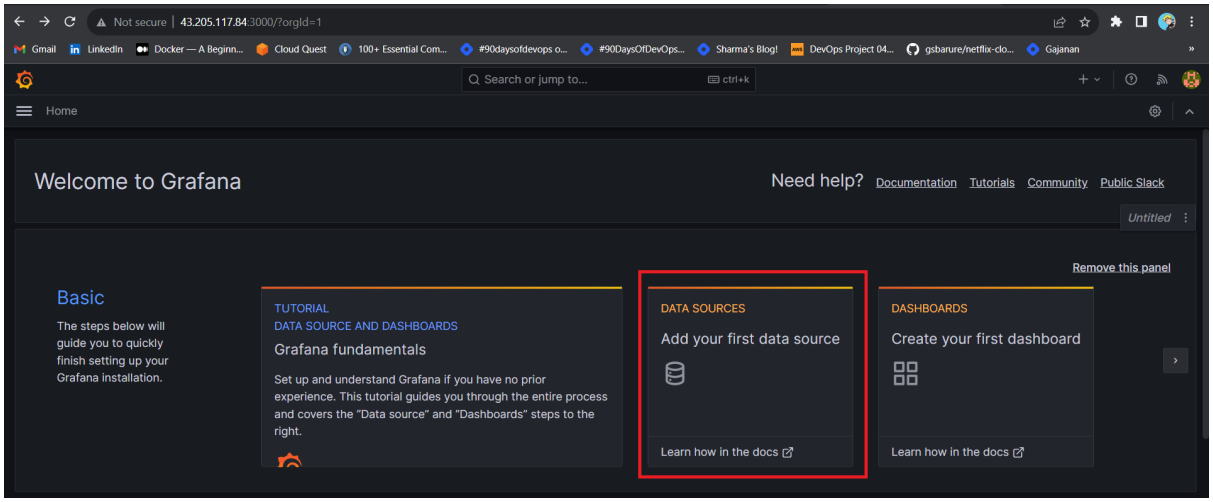
password admin



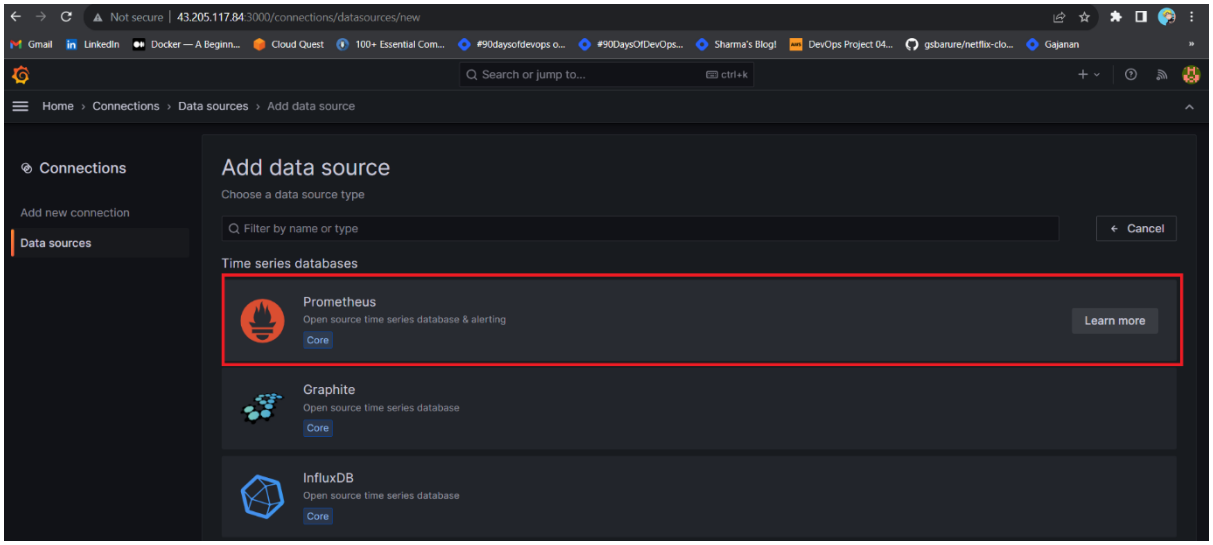
When you log in for the first time, you get the option to change the password.



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

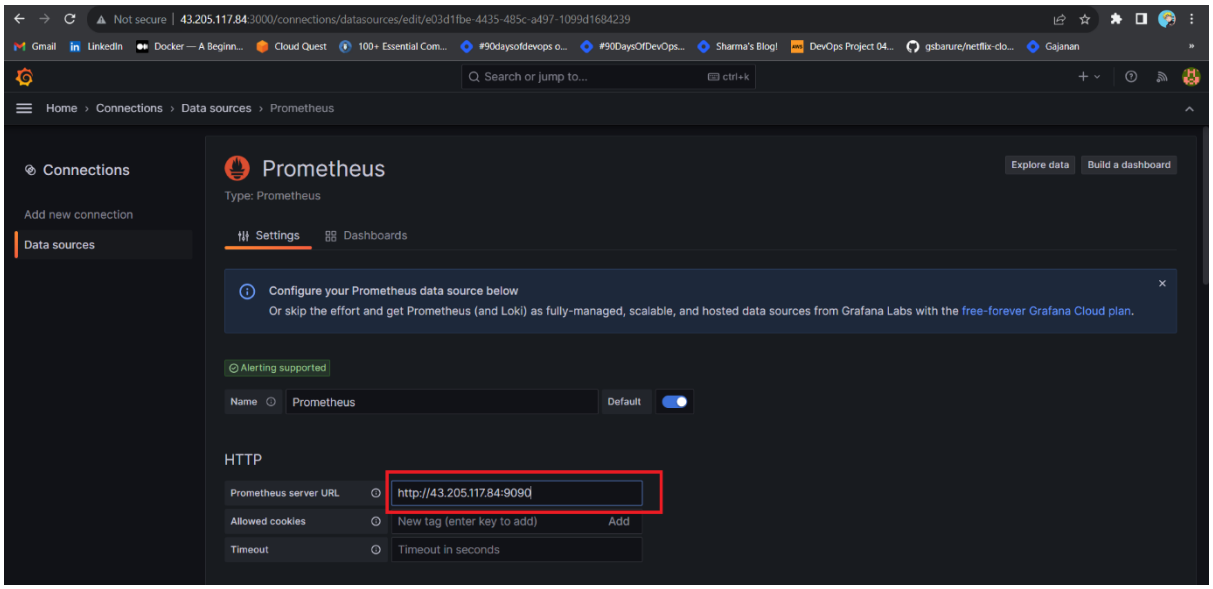


Click Add data source and select Prometheus.

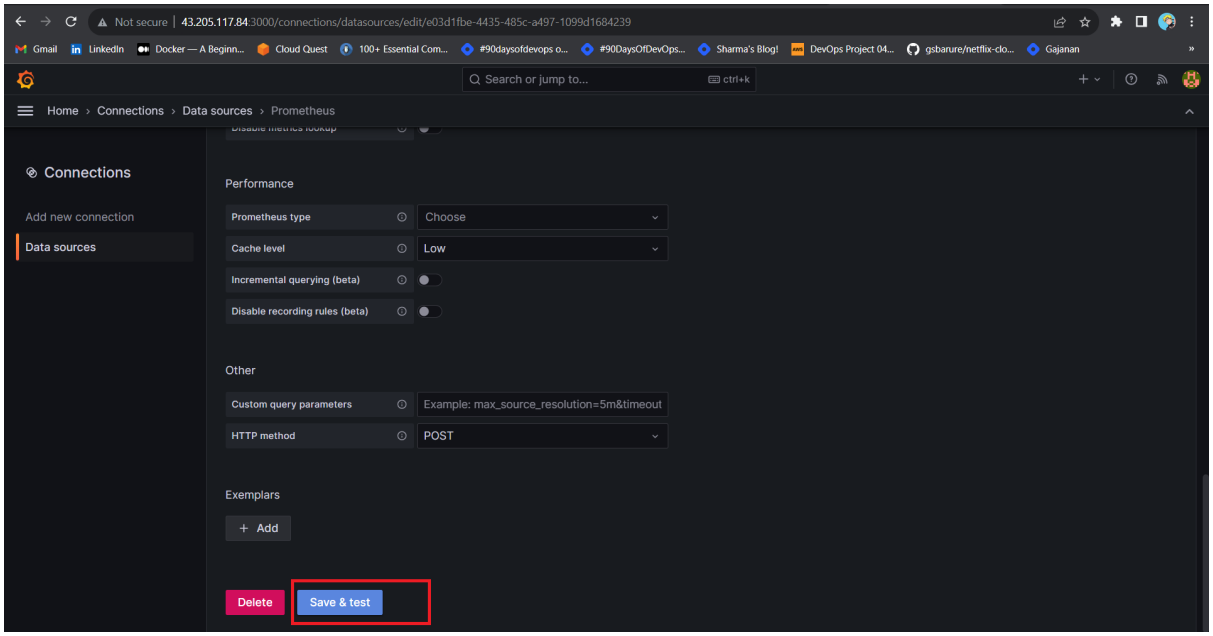


For the URL, enter localhost:9090 and click Save and test. You can see Data source is working.

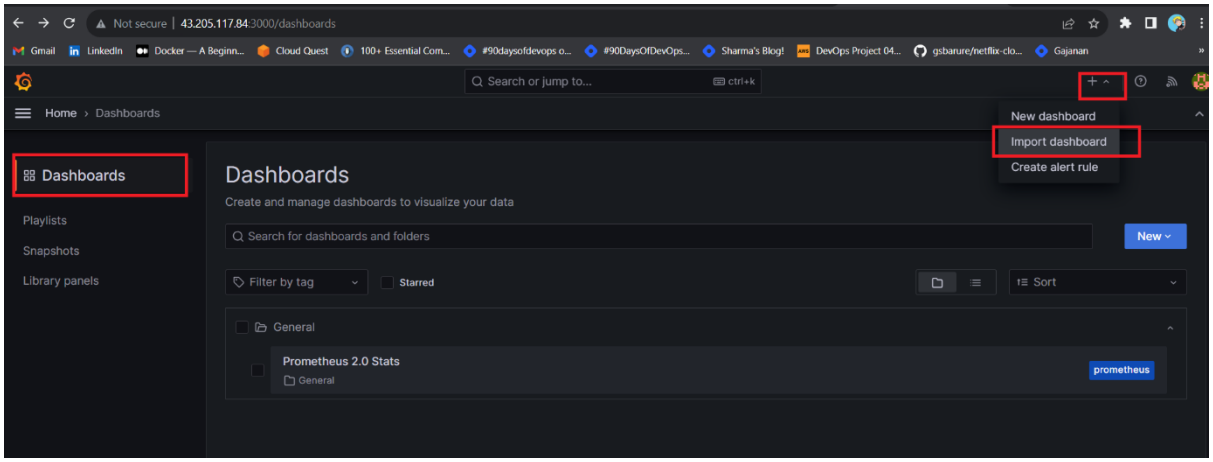
<public-ip:9090>



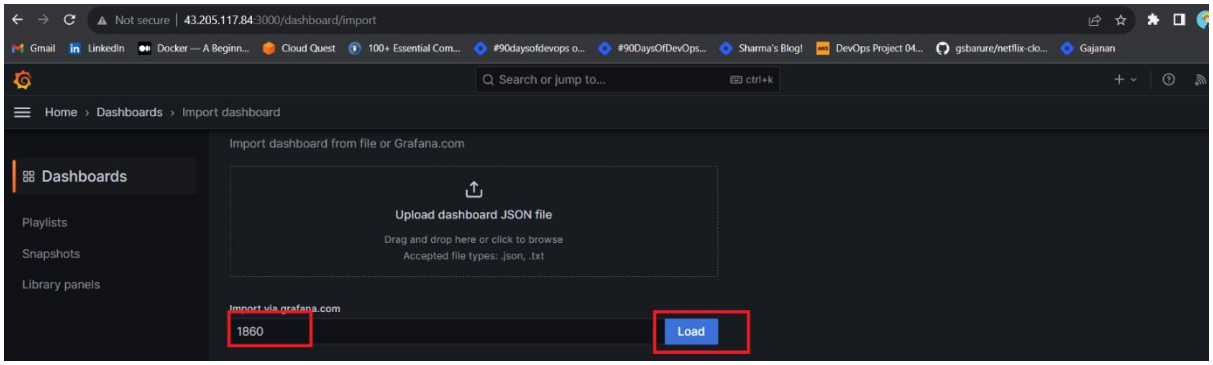
Click on Save and Test.



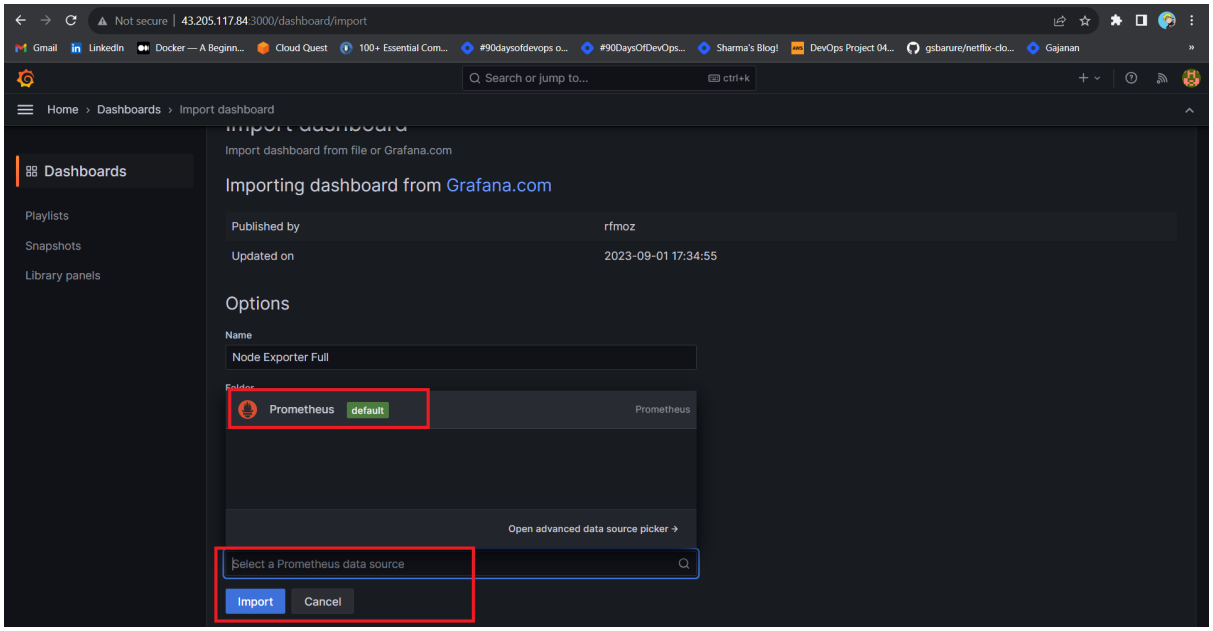
Let’s add Dashboard for a better view



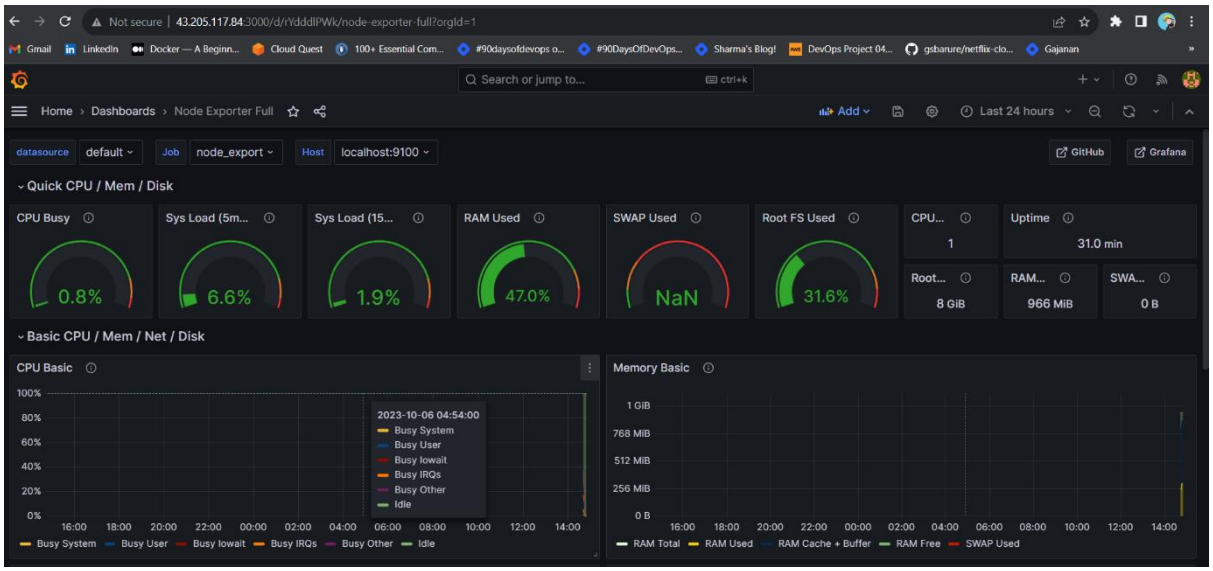
Click on Import Dashboard paste this code 1860 and click on load



Select the Datasource and click on Import



You will see this output



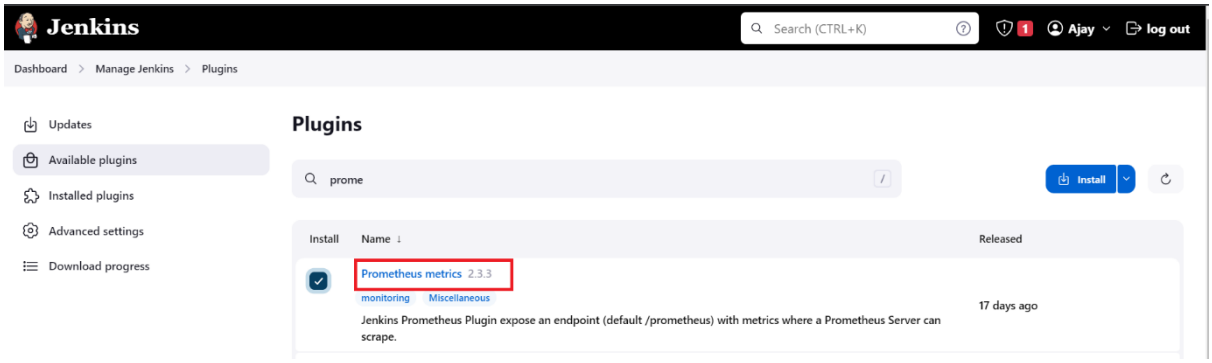
Step 5 — Install the Prometheus Plugin and Integrate it with the Prometheus server

Let's Monitor JENKINS SYSTEM

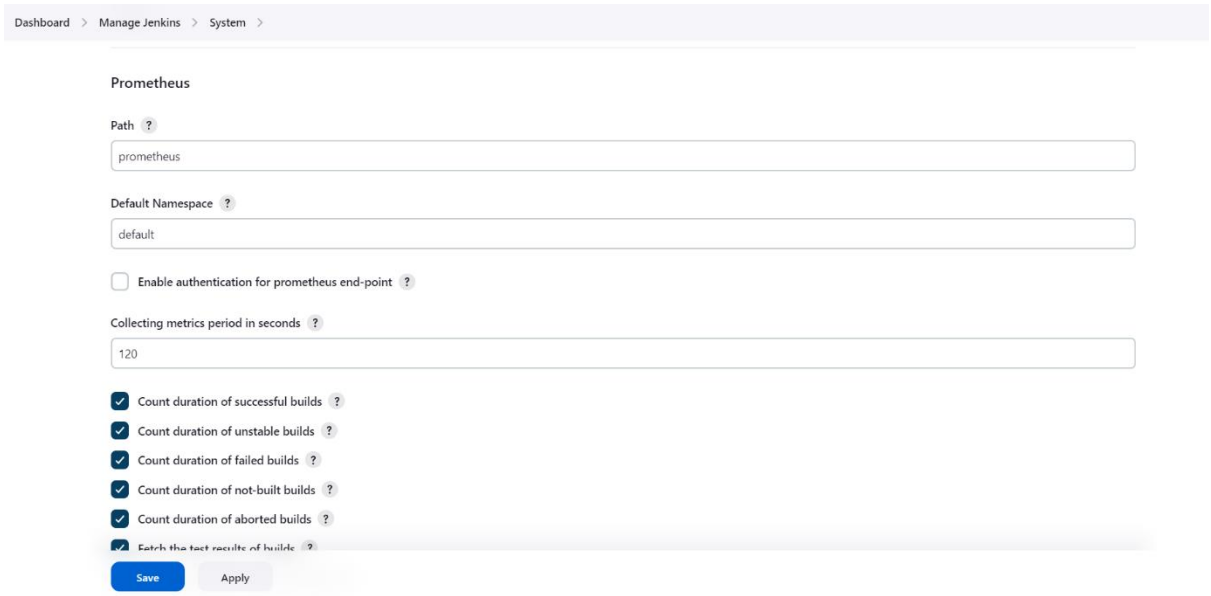
Need Jenkins up and running machine

Goto Manage Jenkins –> Plugins –> Available Plugins

Search for Prometheus and install it



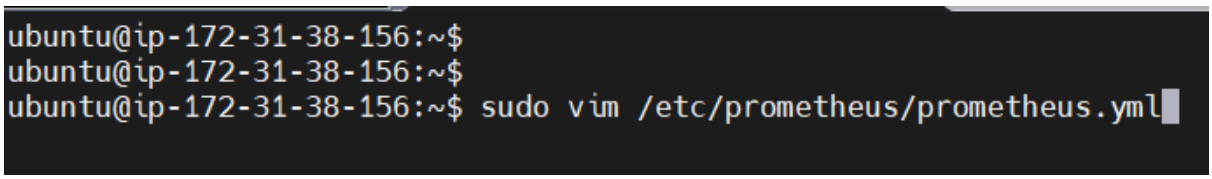
Once that is done you will Prometheus is set to /Prometheus path in system configurations



Nothing to change click on apply and save

To create a static target, you need to add job_name with static_configs. go to Prometheus server

sudo vim /etc/prometheus/prometheus.yml



Paste below code

```
- job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['<jenkins-ip>:8080']
```

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

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

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

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

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

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: node_export
    static_configs:
      - targets: ["localhost:9100"]

  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['3.111.170.92:8080']
```

Before, restarting check if the config is valid.

promtool check config /etc/prometheus/prometheus.yml

Then, you can use a POST request to reload the config.

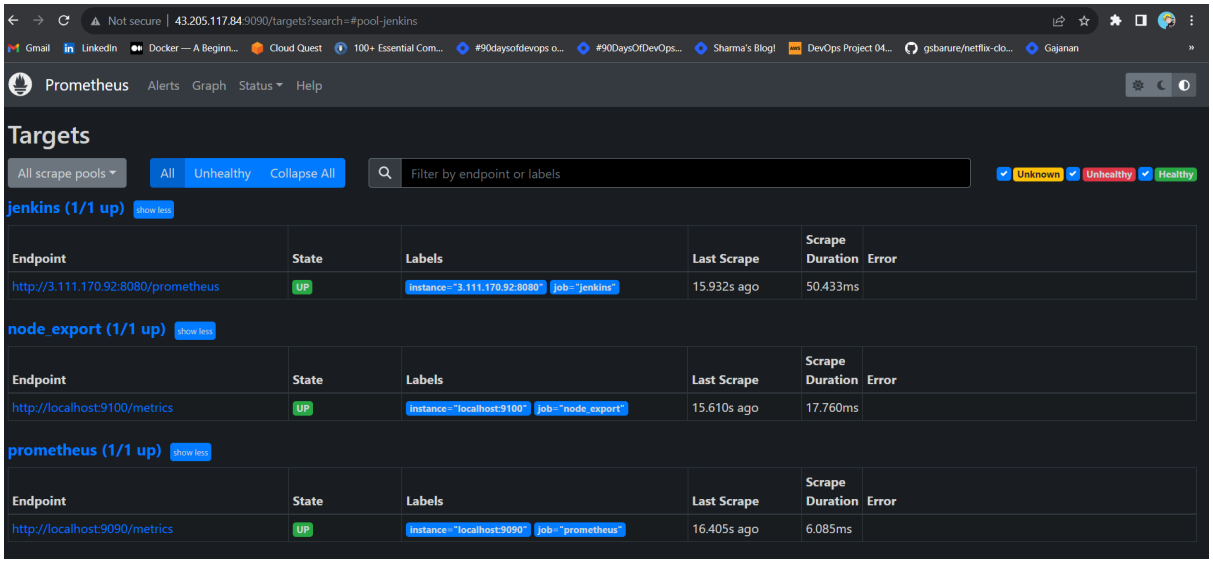
curl -X POST http://localhost:9090/-/reload

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ promtool check config /etc/prometheus/prometheus.yml
Checking /etc/prometheus/prometheus.yml
 SUCCESS: /etc/prometheus/prometheus.yml is valid prometheus config file syntax

ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ curl -X POST http://localhost:9090/-/reload
ubuntu@ip-172-31-38-156:~$
```

Check the targets section

http://<ip>:9090/targets

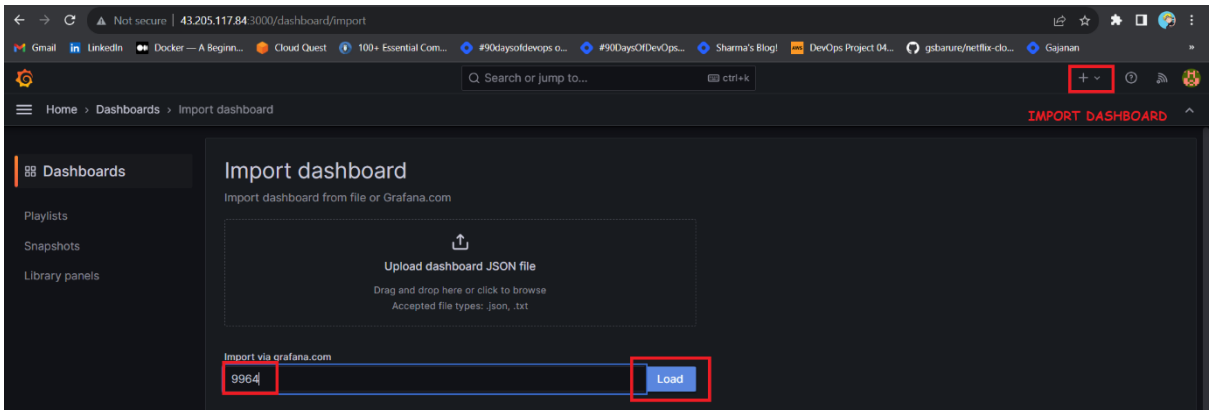
You will see Jenkins is added to it



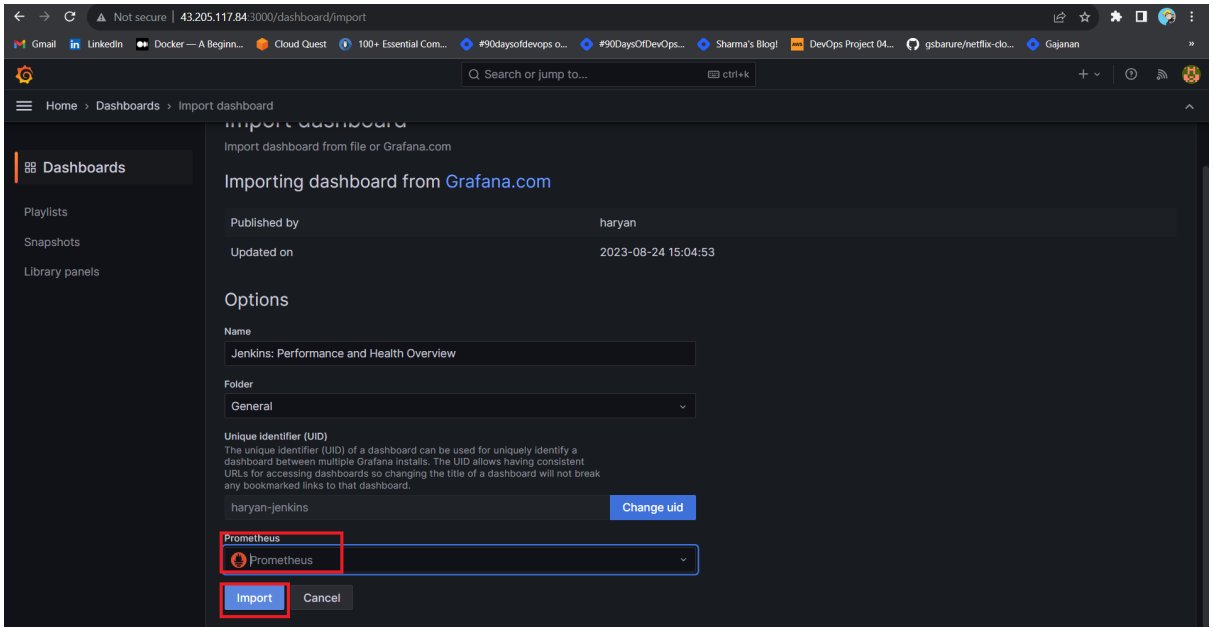
Let’s add Dashboard for a better view in Grafana

Click On Dashboard –> + symbol –> Import Dashboard

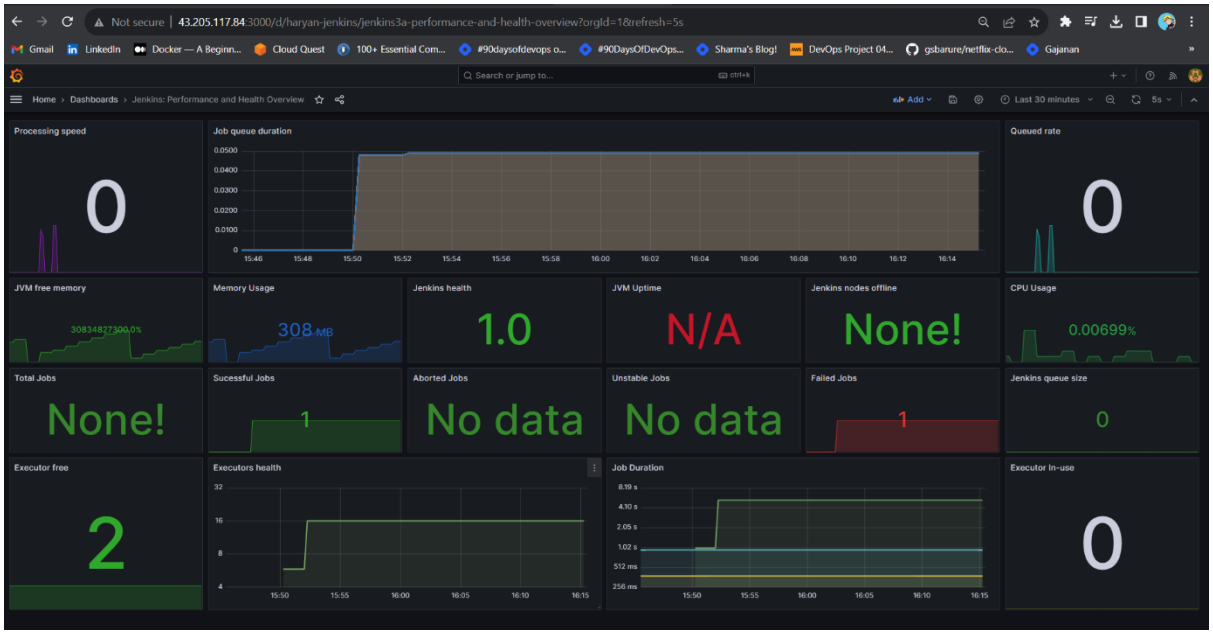
Use Id 9964 and click on load



Select the data source and click on Import

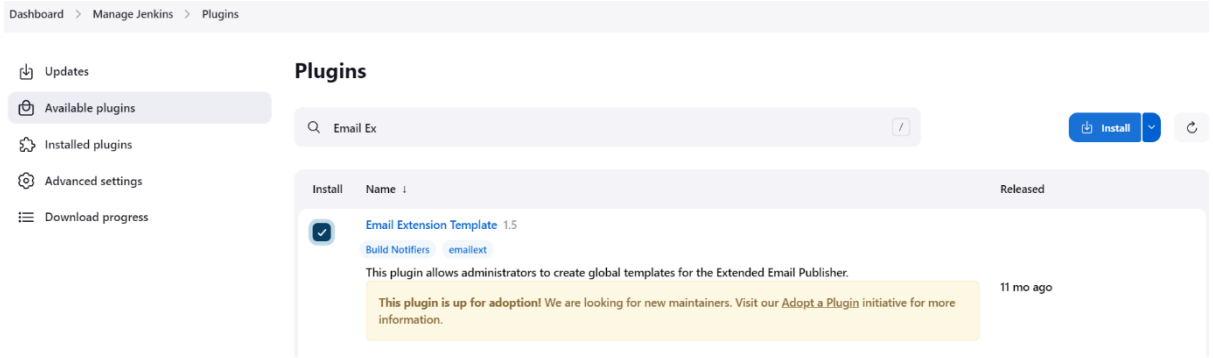


Now you will see the Detailed overview of Jenkins



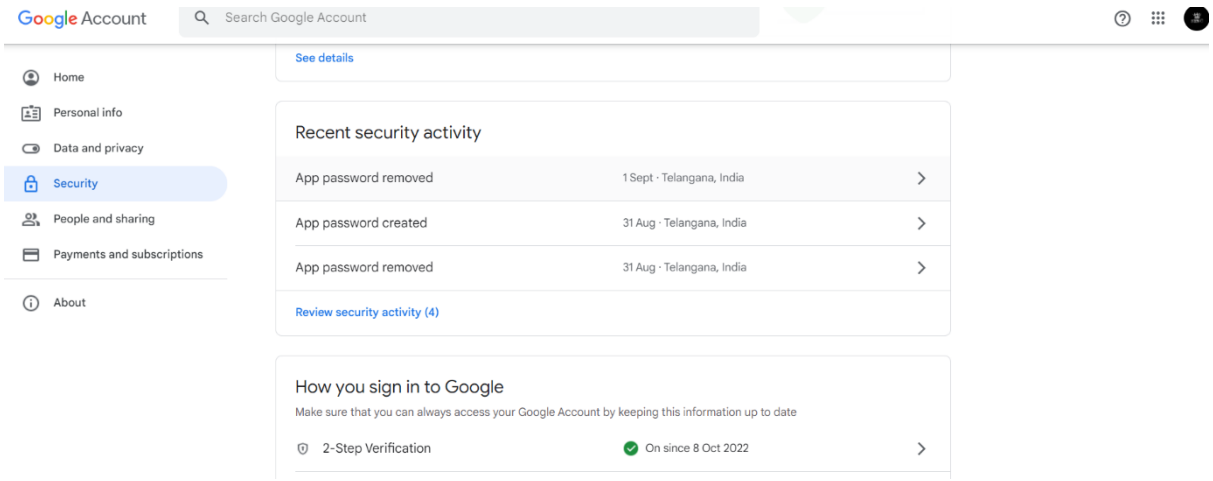
Step 6 — Email Integration With Jenkins and Plugin Setup

Install Email Extension Plugin in Jenkins



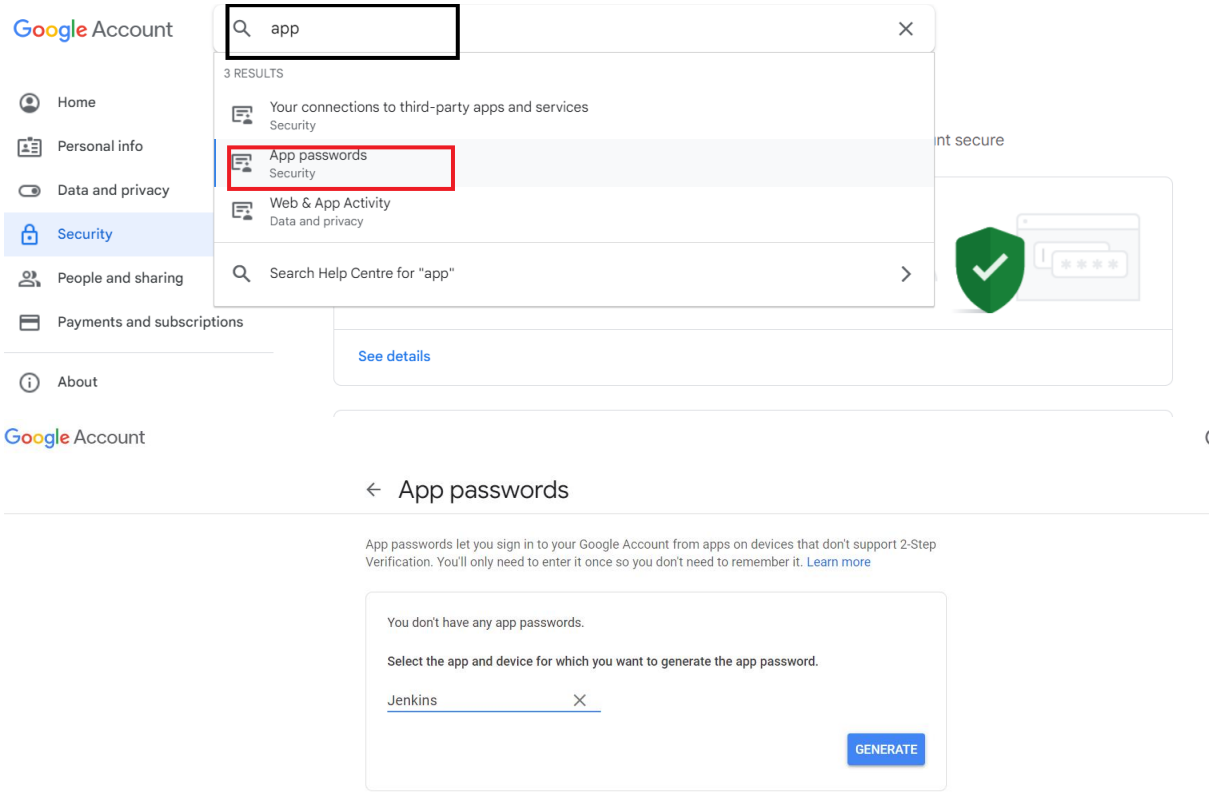
Go to your Gmail and click on your profile

Then click on Manage Your Google Account –> click on the security tab on the left side panel you will get this page(provide mail password).

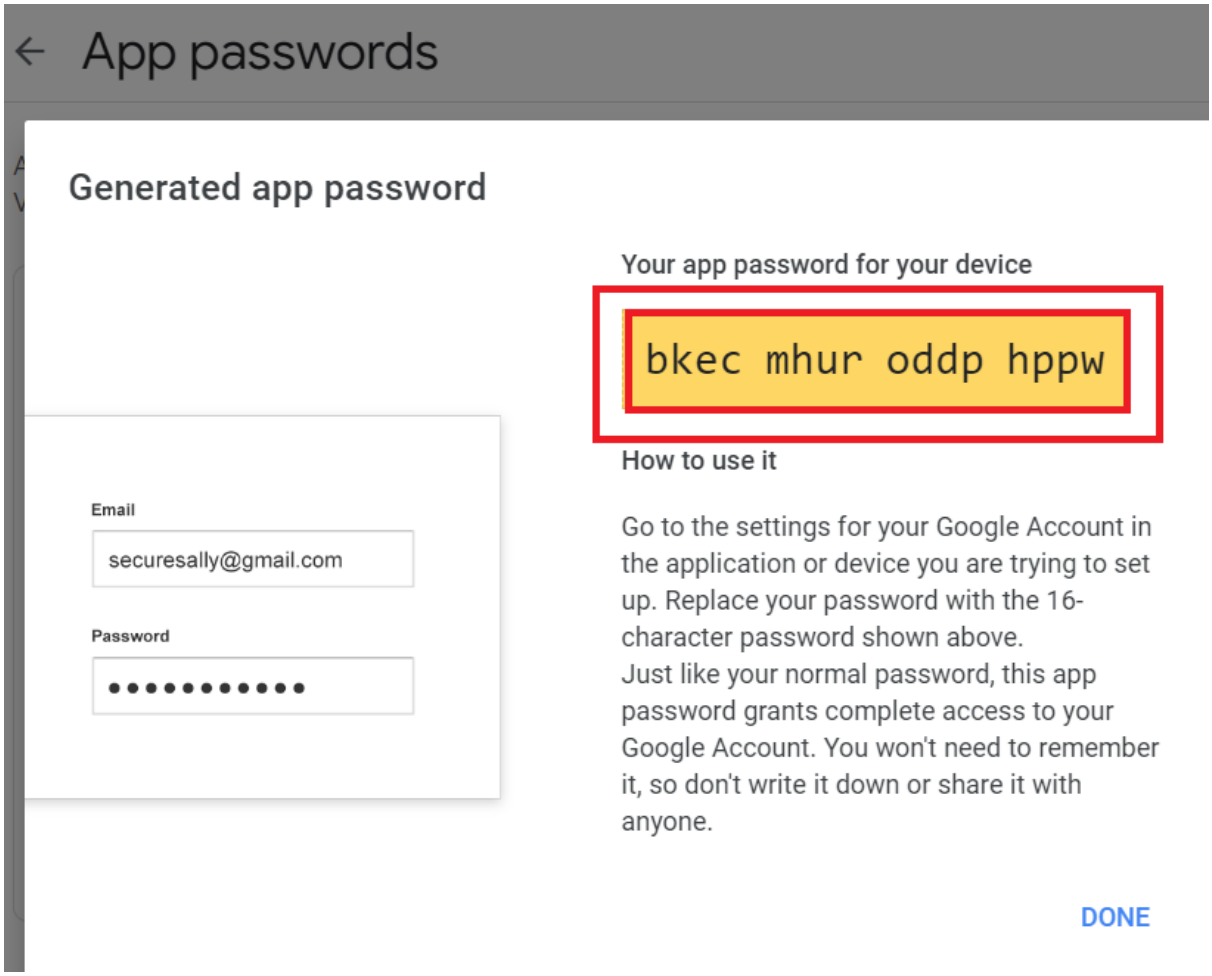


2-step verification should be enabled.

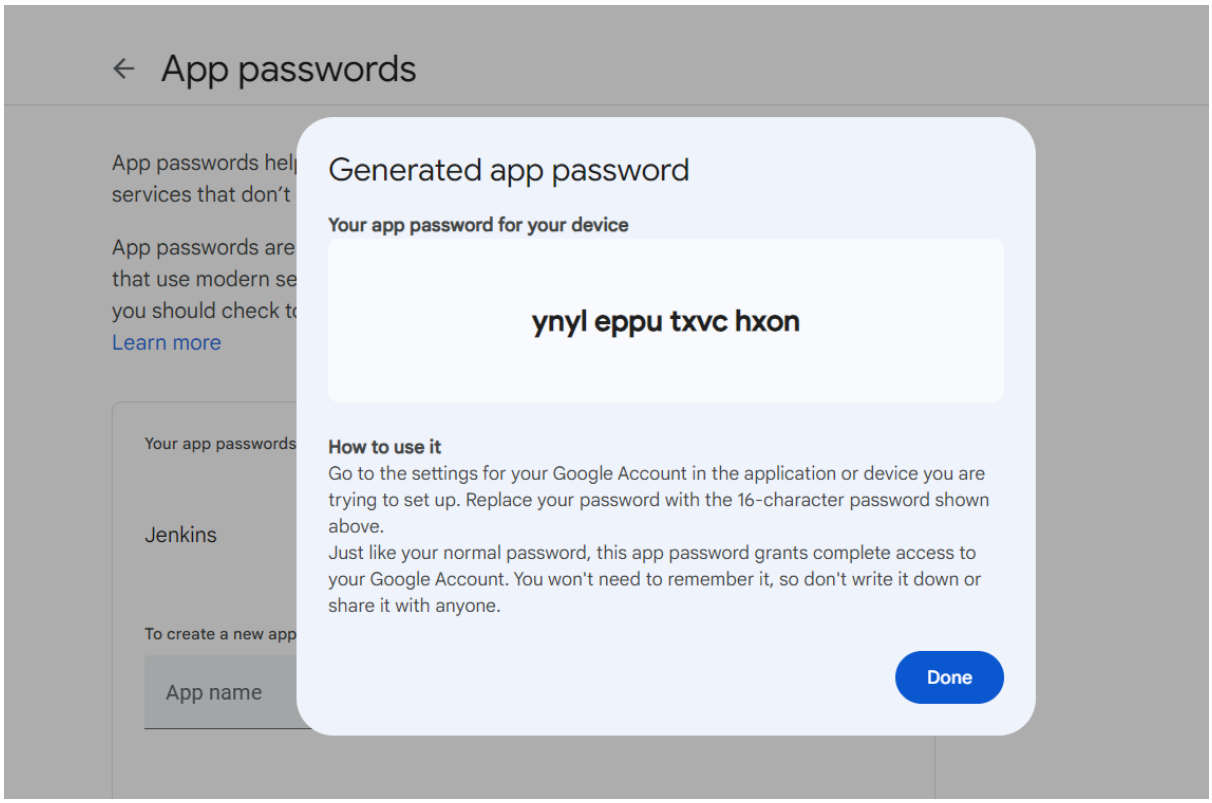
Search for the app in the search bar you will get app passwords like the below image



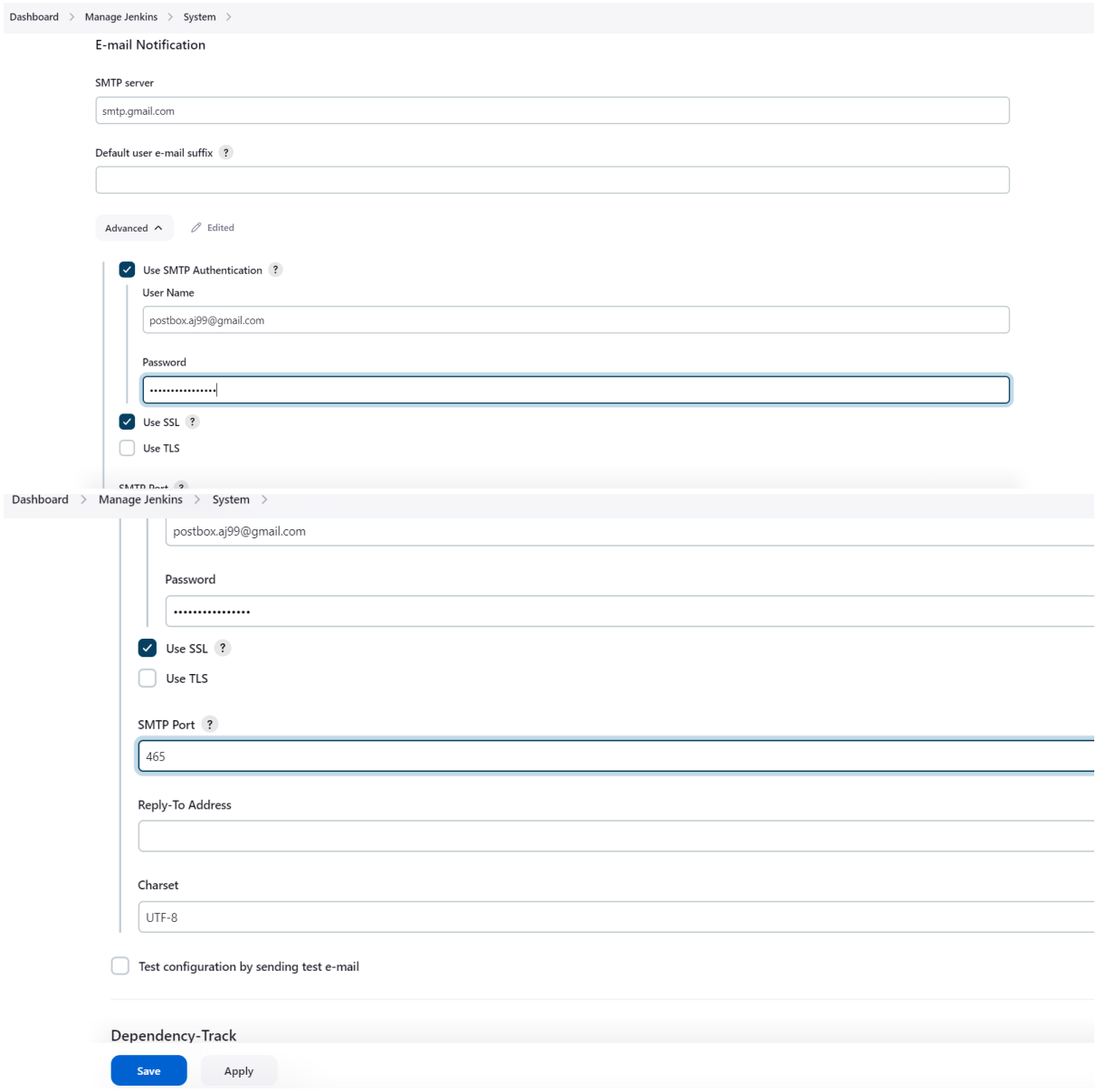
Click on other and provide your name and click on Generate and copy the password



In the new update, you will get a password like this

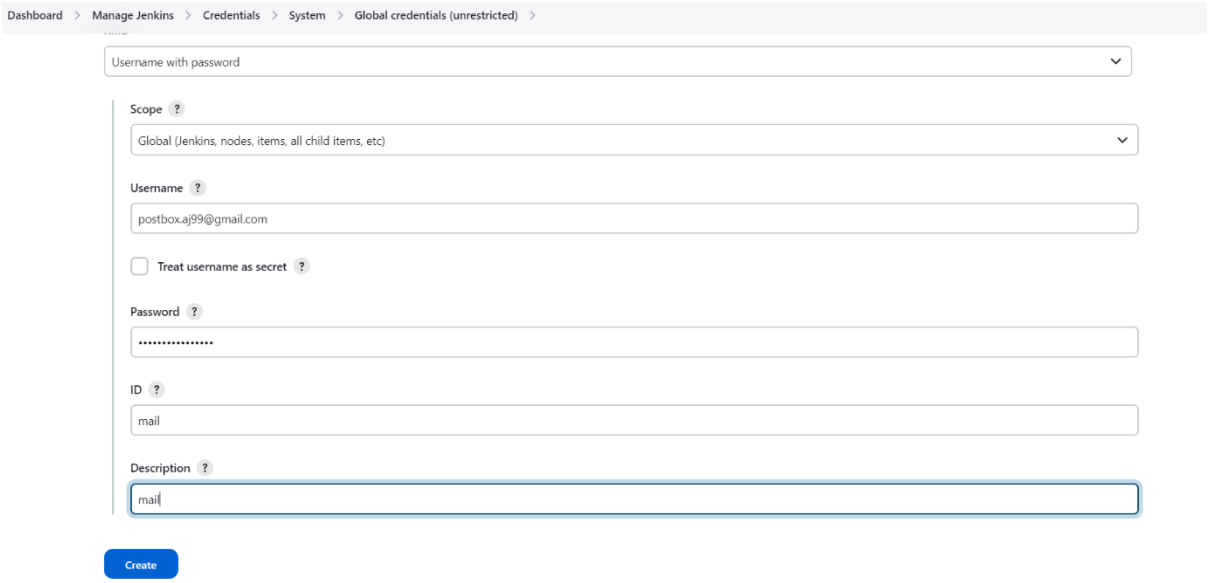


Once the plugin is installed in Jenkins, click on manage Jenkins –> configure system there under the E-mail Notification section configure the details as shown in the below image



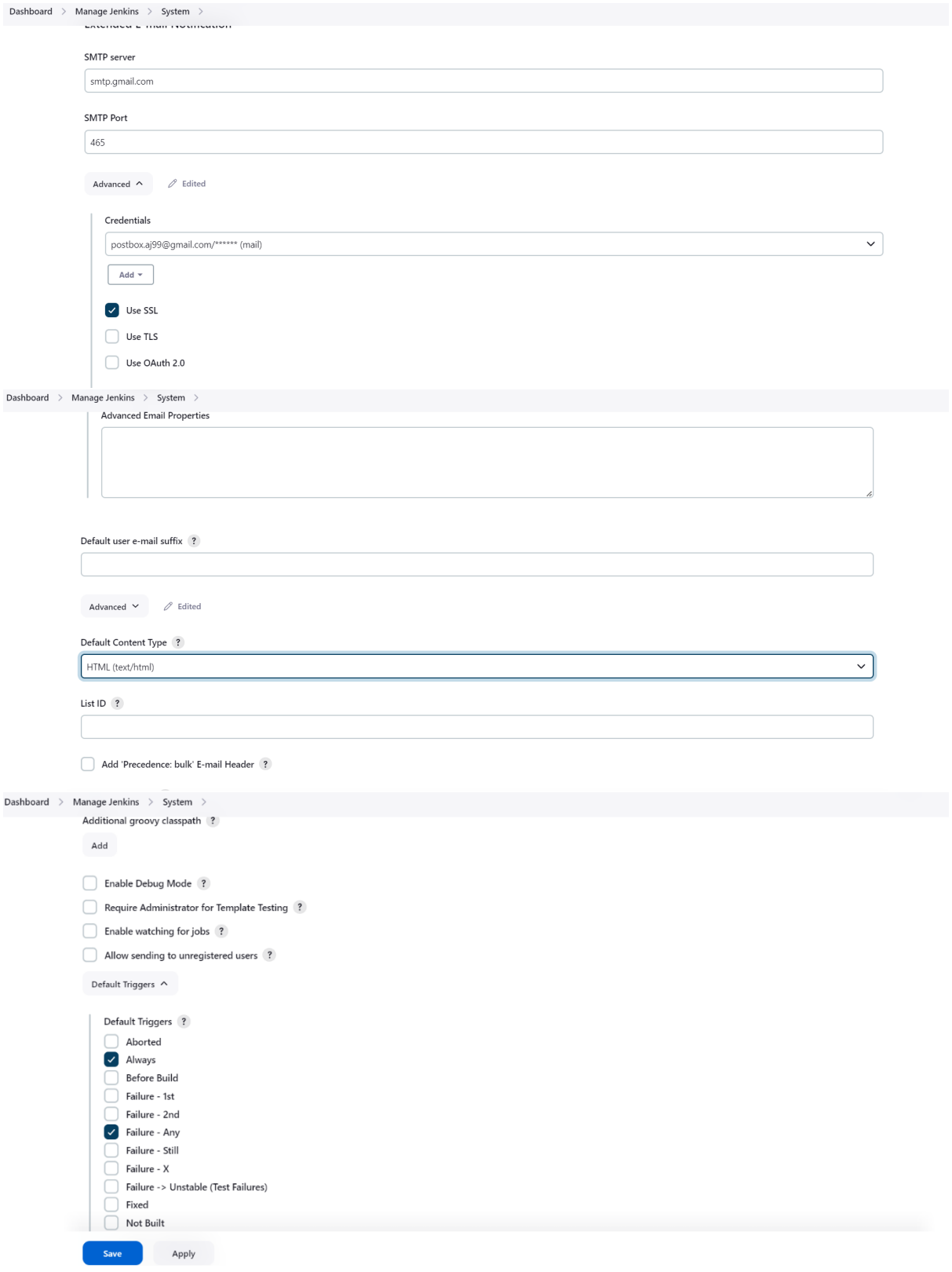
Click on Apply and save.

Click on Manage Jenkins–> credentials and add your mail username and generated password



This is to just verify the mail configuration

Now under the Extended E-mail Notification section configure the details as shown in the below images



Click on Apply and save.

```
post {
    always {
        emailext attachLog: true,
```

```
        subject: "'${currentBuild.result}'",
        body: "Project: ${env.JOB_NAME}<br/>" +
          "Build Number: ${env.BUILD_NUMBER}<br/>" +
          "URL: ${env.BUILD_URL}<br/>",
        to: 'postbox.aj99@gmail.com',  #change Your mail
        attachmentsPattern: 'trivyfs.txt,trivyimage.txt'
    }
}
```

Next, we will log in to Jenkins and start to configure our Pipeline in Jenkins

Step 7 — Install Plugins like JDK, Sonarqube Scanner, NodeJs, OWASP Dependency Check

7A — Install Plugin

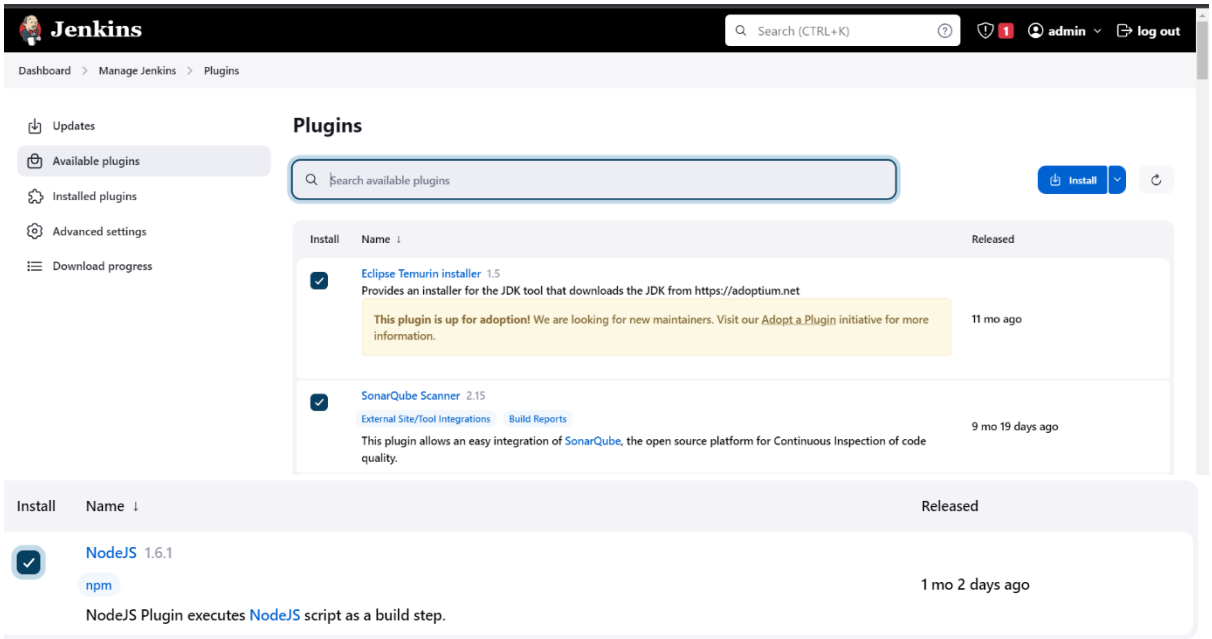
Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

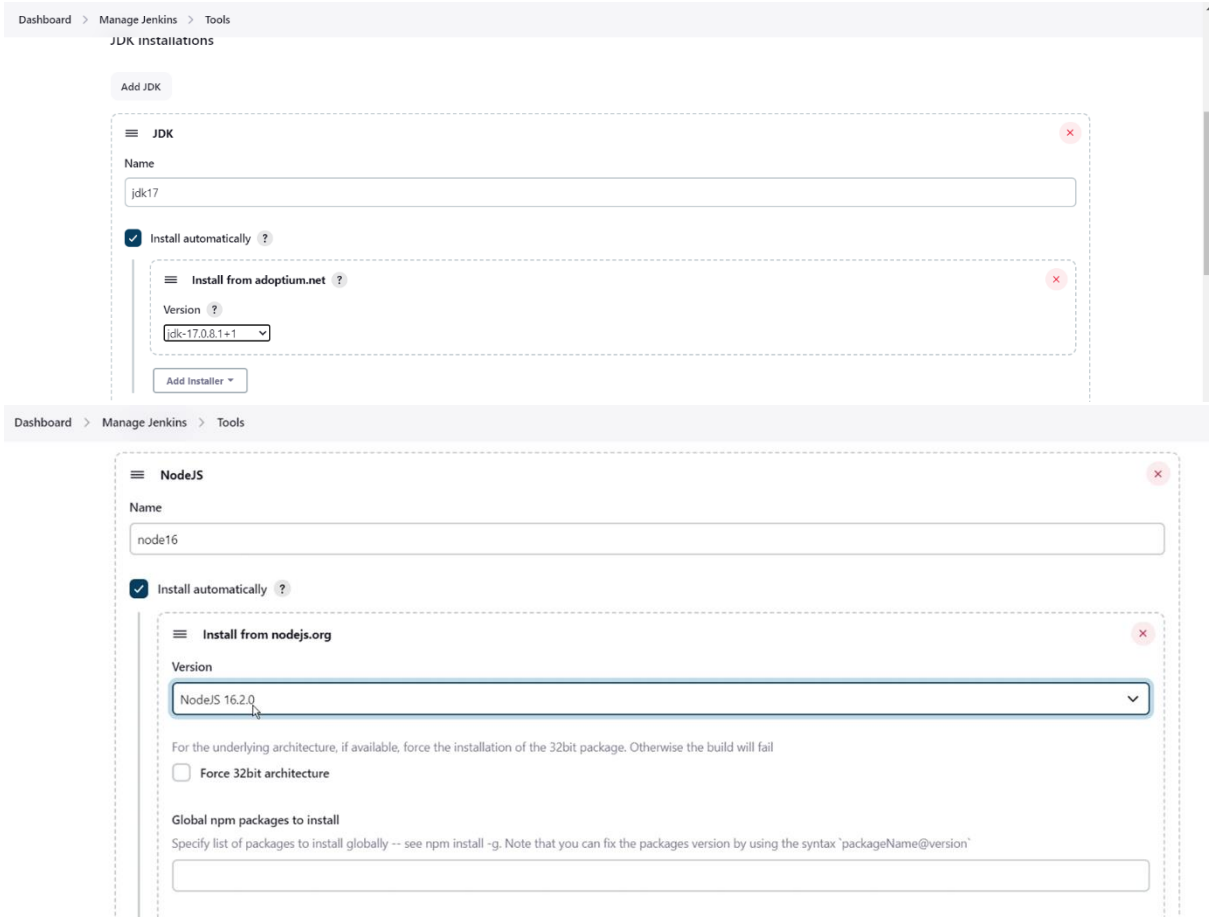
2 → SonarQube Scanner (Install without restart)

3 → NodeJs Plugin (Install Without restart)



7B — Configure Java and Nodejs in Global Tool Configuration

Goto Manage Jenkins → Tools → Install JDK(17) and NodeJs(16)→ Click on Apply and Save

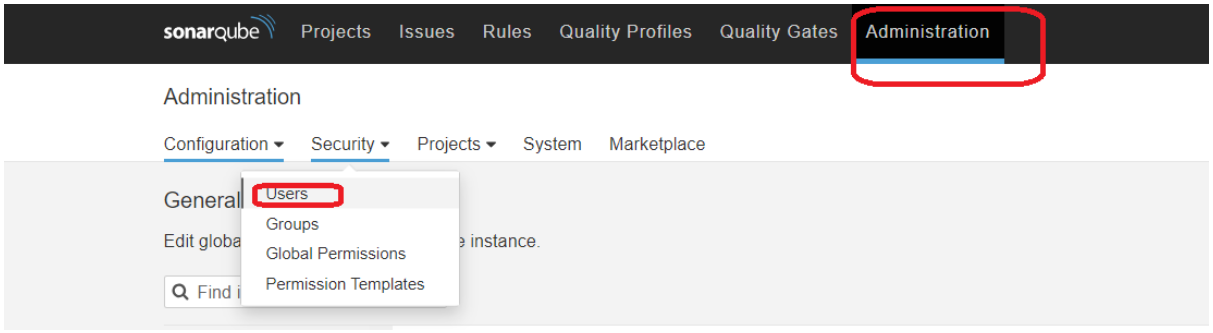


7C — Create a Job

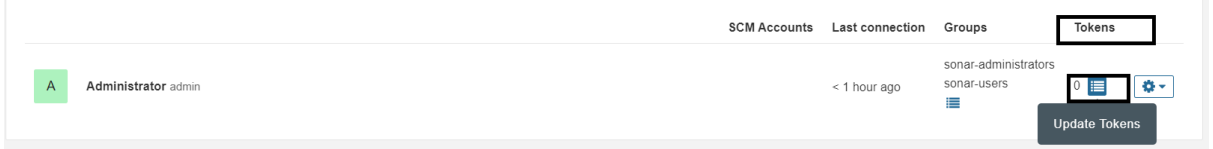
create a job as Netflix Name, select pipeline and click on ok.

Step 8 — Configure Sonar Server in Manage Jenkins

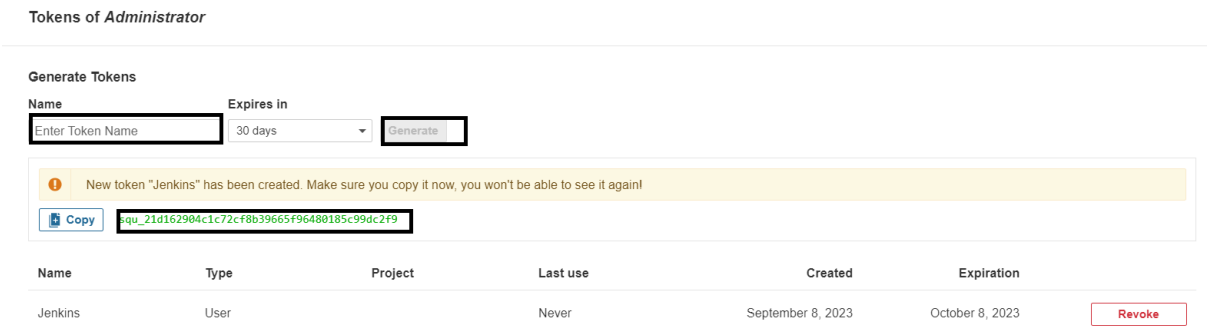
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

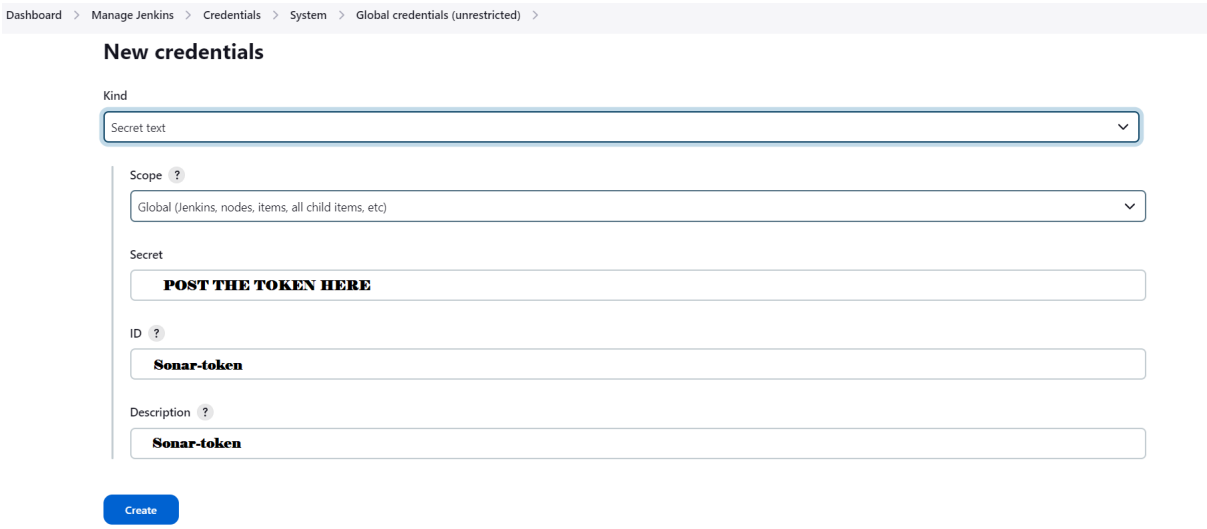


Create a token with a name and generate

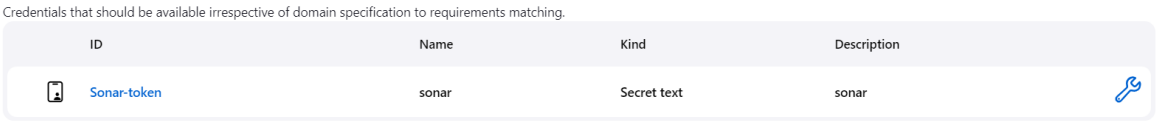


copy Token

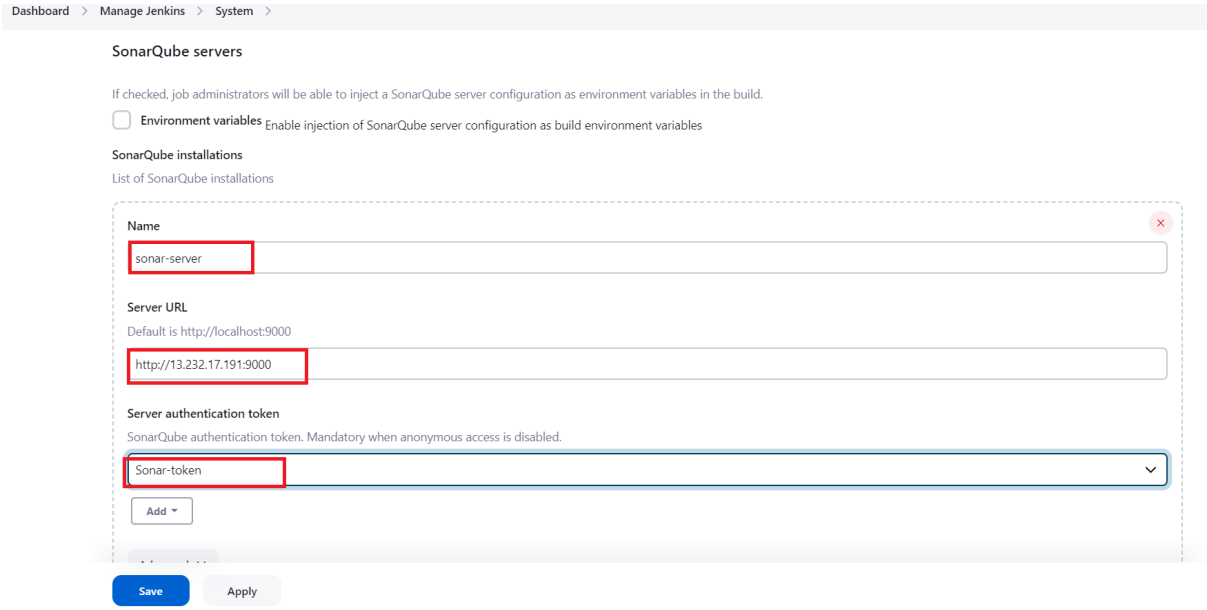
Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text. It should look like this



You will this page once you click on create



Now, go to Dashboard → Manage Jenkins → System and Add like the below image.

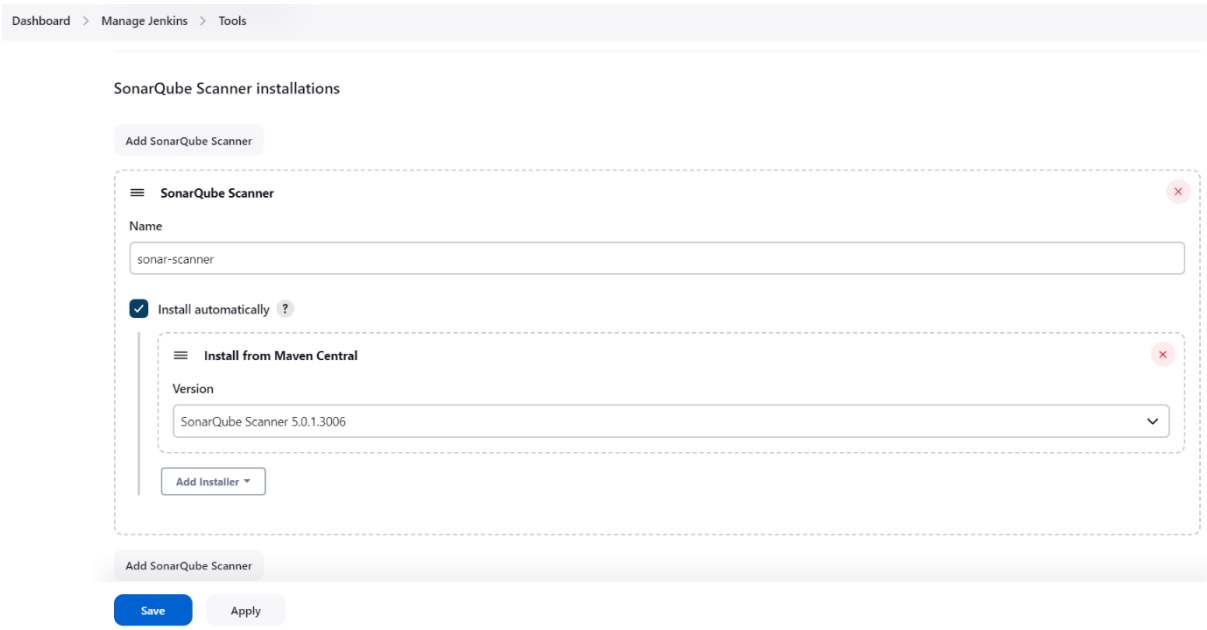


Click on Apply and Save

The Configure System option is used in Jenkins to configure different server

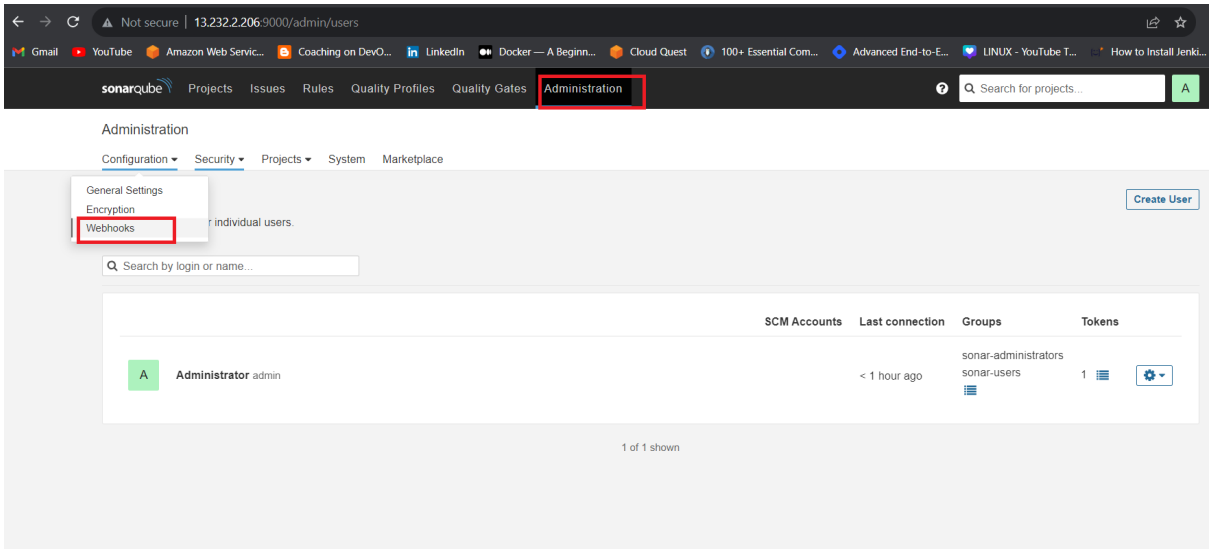
Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

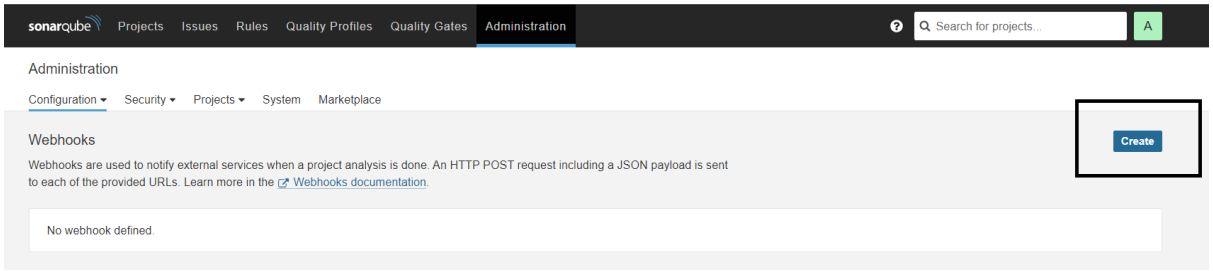


In the Sonarqube Dashboard add a quality gate also

Administration–> Configuration–>Webhooks



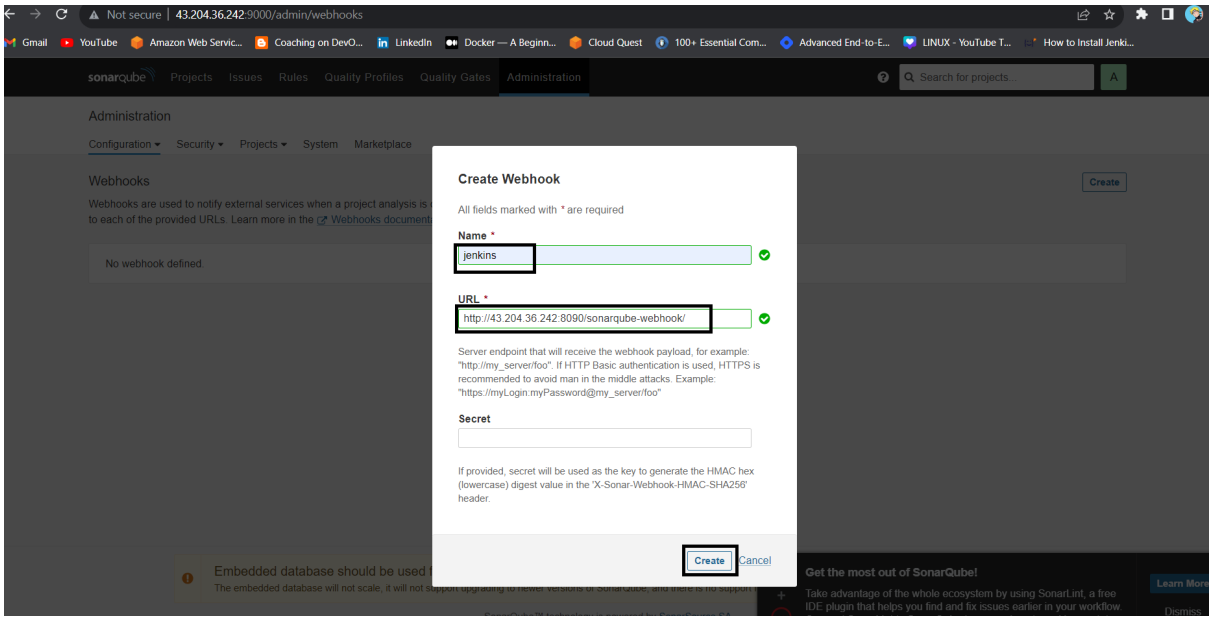
Click on Create



Add details

#in url section of quality gate

<http://jenkins-public-ip:8080>/sonarqube-webhook/



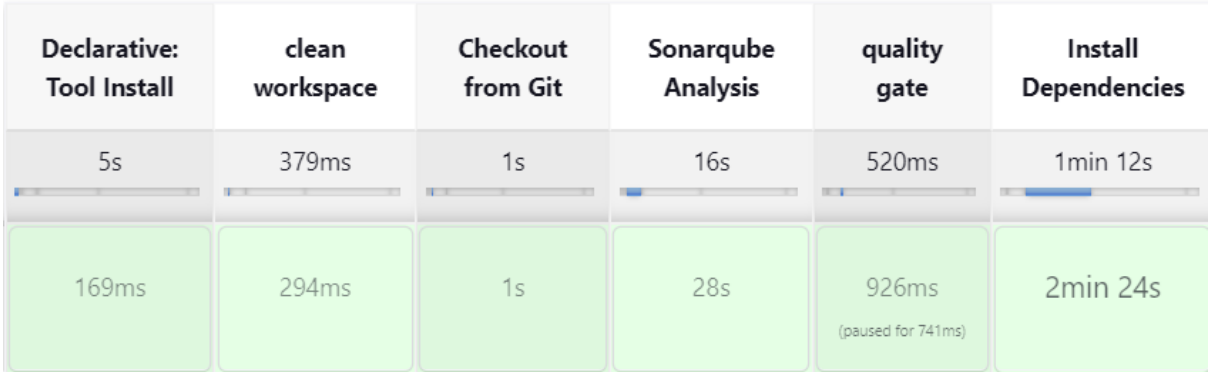
Let's go to our Pipeline and add the script in our Pipeline Script.

pipeline{

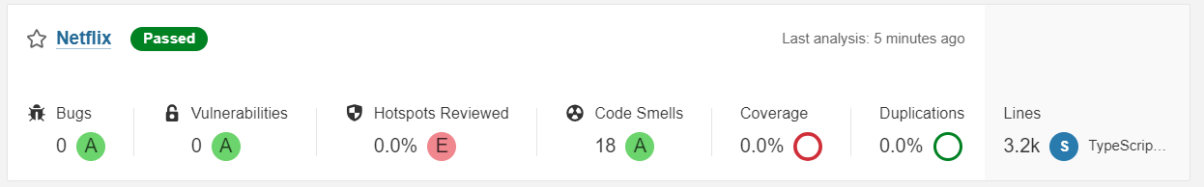
```
agent any
tools{
   jdk 'jdk17'
   nodejs 'node16'
}
environment {
   SCANNER_HOME=tool 'sonar-scanner'
}
stages {
   stage('clean workspace'){
      steps{
         cleanWs()
      }
   }
   stage('Checkout from Git'){
      steps{
         git branch: 'main', url: 'https://github.com/Aj7Ay/Netflix-clone.git'
      }
   }
   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"
        }
    }
  }
  post {
   always {
    emailext attachLog: true,
        subject: "'${currentBuild.result}'",
        body: "Project: ${env.JOB_NAME}<br/>" +
            "Build Number: ${env.BUILD_NUMBER}<br/>" +
            "URL: ${env.BUILD_URL}<br/>",
        to: 'postbox.aj99@gmail.com',
        attachmentsPattern: 'trivyfs.txt,trivyimage.txt'
    }
  }
}
```

Click on Build now, you will see the stage view like this



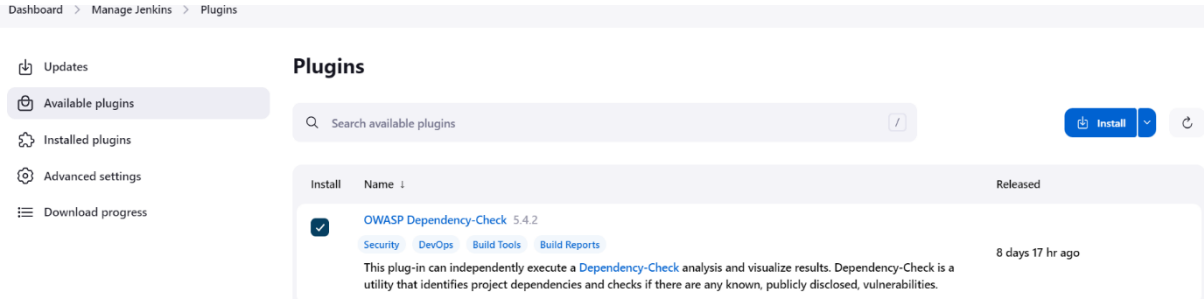
To see the report, you can go to Sonarqube Server and go to Projects.



You can see the report has been generated and the status shows as passed. You can see that there are 3.2k lines it scanned. To see a detailed report, you can go to issues.

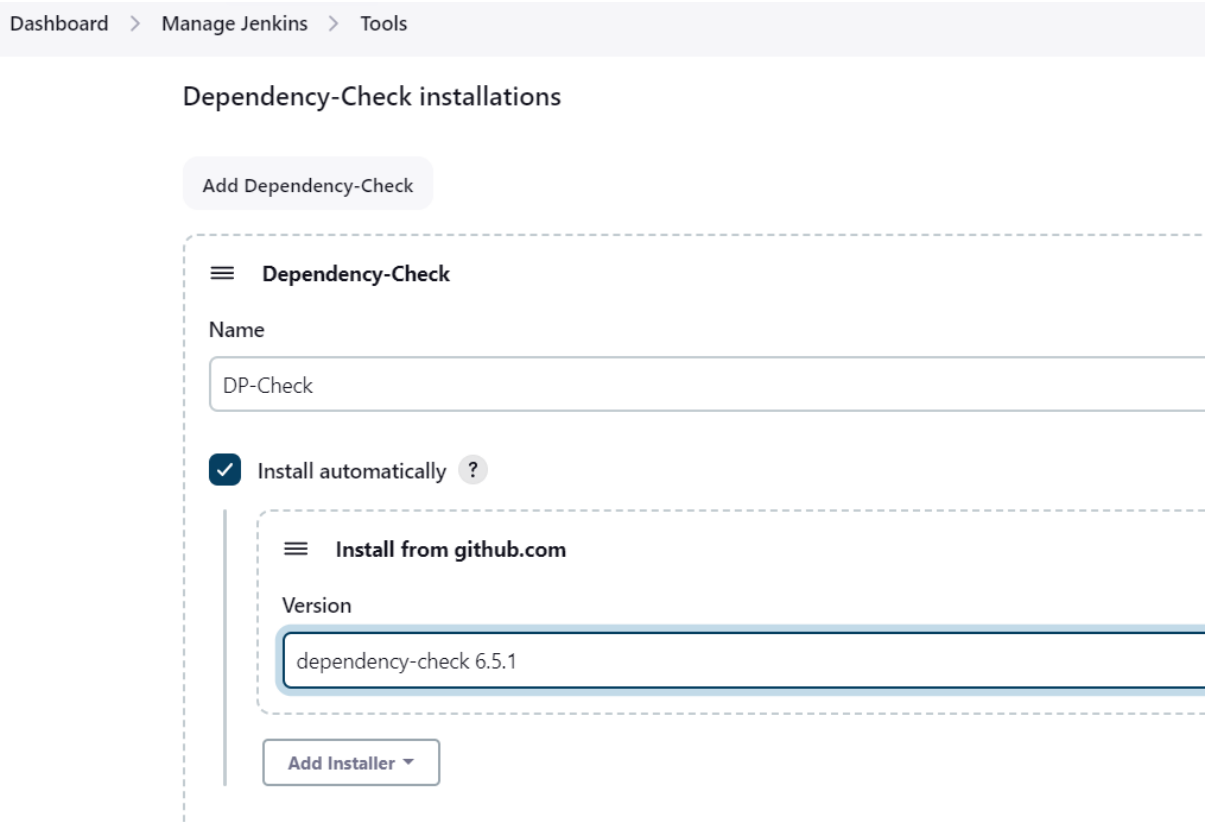
Step 9 — Install OWASP Dependency Check Plugins

GotoDashboard → Manage Jenkins → Plugins → OWASP Dependency-Check. Click on it and install it without restart.



First, we configured the Plugin and next, we had to configure the Tool

Goto Dashboard → Manage Jenkins → Tools →

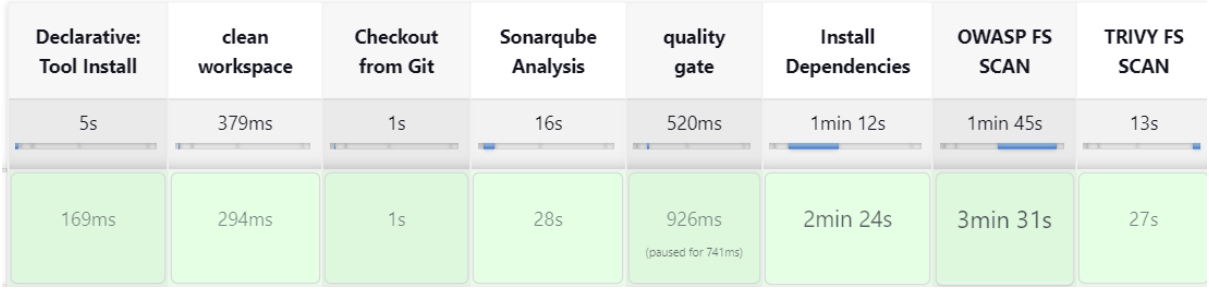


Click on Apply and Save here.

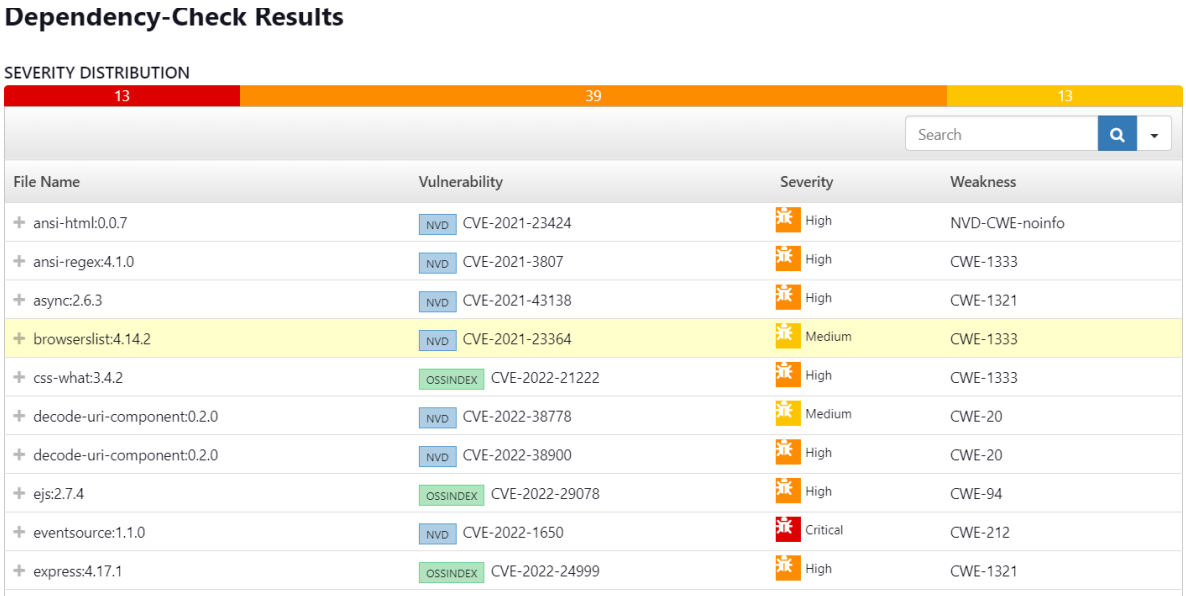
Now go configure → Pipeline and add this stage to your pipeline and build.

```
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"
        }
    }
```

The stage view would look like this,



You will see that in status, a graph will also be generated and Vulnerabilities.



Step 10 — Docker Image Build and Push

We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

Docker

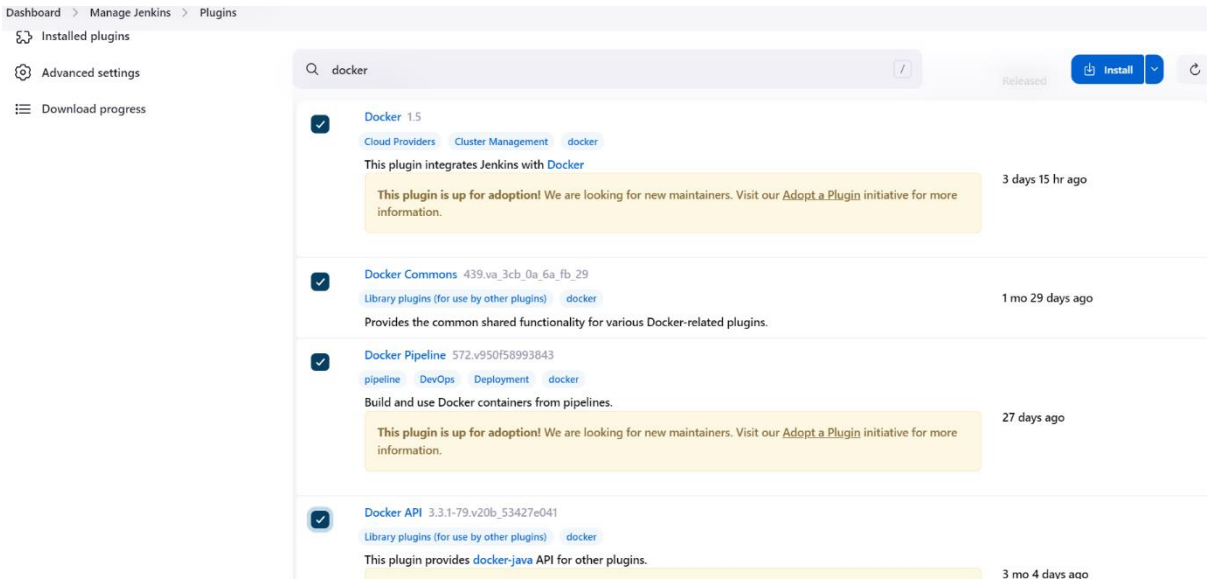
Docker Commons

Docker Pipeline

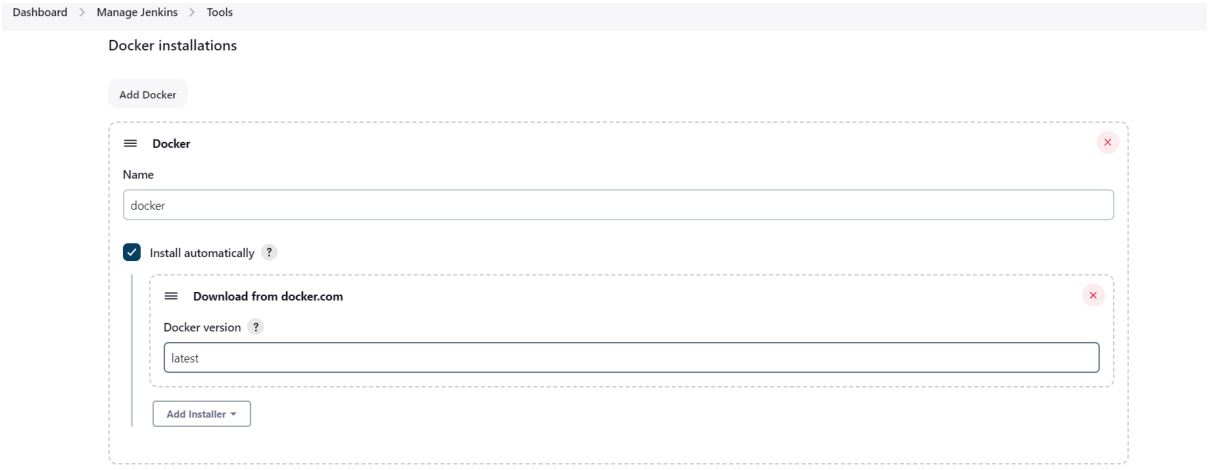
Docker API

docker-build-step

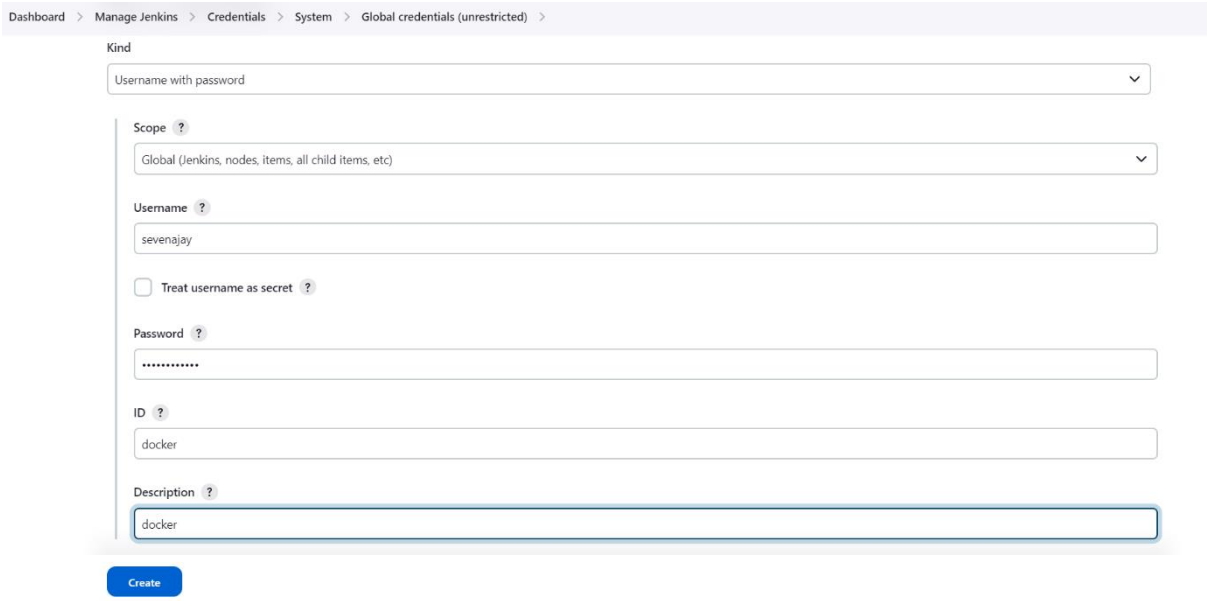
and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



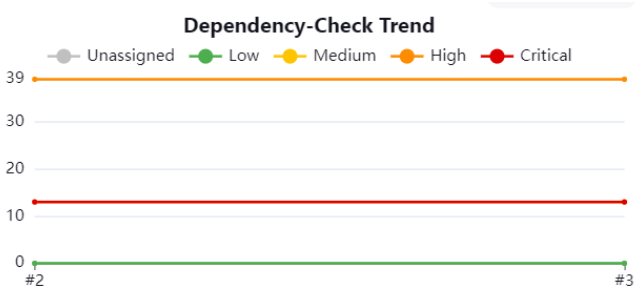
Add DockerHub Username and Password under Global Credentials



Add this stage to Pipeline Script

```
stage("Docker Build & Push"){
        steps{
           script{
              withDockerRegistry(credentialsId: 'docker', toolName: 'docker'){
                 sh "docker build --build-arg TMDB_V3_API_KEY=Aj7ay86fe14eca3e76869b92 -t netflix ."
                 sh "docker tag netflix sevenajay/netflix:latest "
                 sh "docker push sevenajay/netflix:latest "
              }
           }
        }
    }
    stage("TRIVY"){
        steps{
           sh "trivy image sevenajay/netflix:latest > trivyimage.txt"
        }
    }
```

You will see the output below, with a dependency trend.



Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY
3s	366ms	1s	19s	451ms	1min 20s	2min 1s	16s	3min 9s	4s
154ms	341ms	1s	25s	315ms	1min 36s	2min 31s	23s	3min 9s	4s

When you log in to Dockerhub, you will see a new image is created

Now Run the container to see if the game coming up or not by adding the below stage

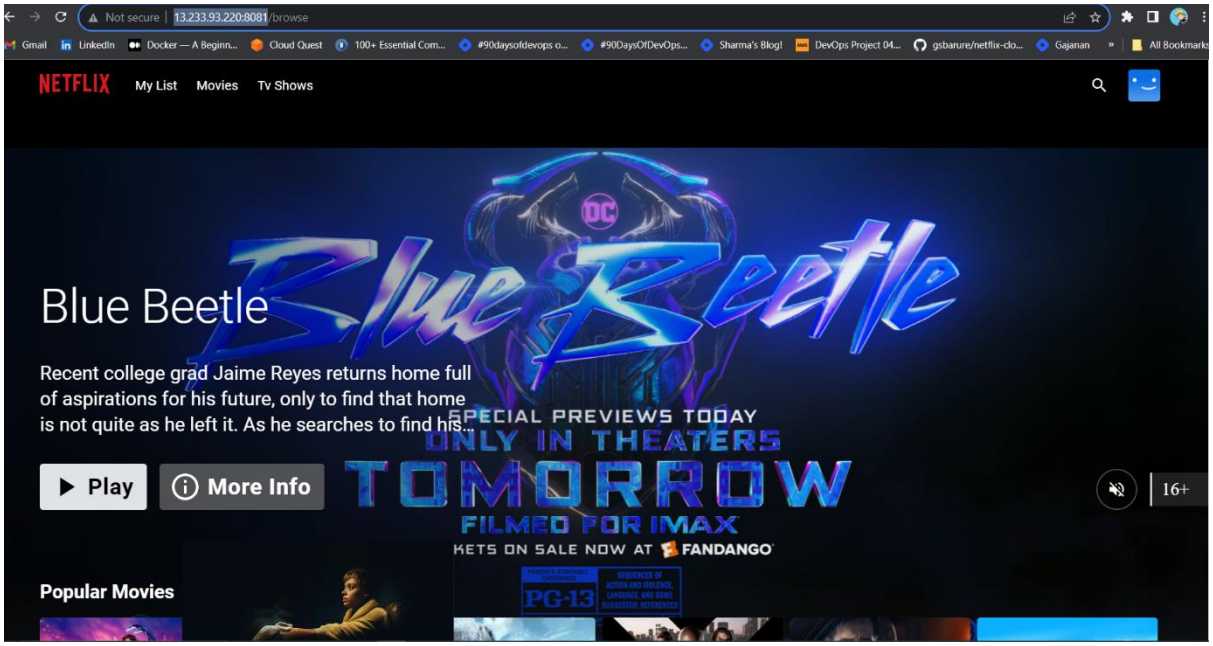
```
stage('Deploy to container'){
        steps{
           sh 'docker run -d --name netflix -p 8081:80 sevenajay/netflix:latest'
        }
    }
```

stage view

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY	Deploy to container
144ms	284ms	1s	25s	410ms	1min 47s	2min 43s	23s	2min 7s	36s	789ms
146ms	251ms	1s	26s	305ms	1min 36s	2min 35s	23s	1min 50s	2min 8s	1s

<Jenkins-public-ip:8081>

You will get this output



Step 11 — Kuberenetes Setup

Connect your machines to Putty or Mobaxtreme

Take-Two Ubuntu 20.04 instances one for k8s master and the other one for worker.

Install Kubectl on Jenkins machine also.

Kubectl is to be installed on Jenkins also

sudo apt update

sudo apt install curl

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

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version --client

Part 1 ———-Master Node————

sudo hostnamectl set-hostname K8s-Master

———-Worker Node————

sudo hostnamectl set-hostname K8s-Worker

Part 2 ————Both Master & Node ————

sudo apt-get update

sudo apt-get install -y docker.io

sudo usermod –aG docker Ubuntu

newgrp docker

sudo chmod 777 /var/run/docker.sock

Part 3 ————— Master —————

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl gpg

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

————-Worker Node—————

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

in case your in root exit from it and run below commands

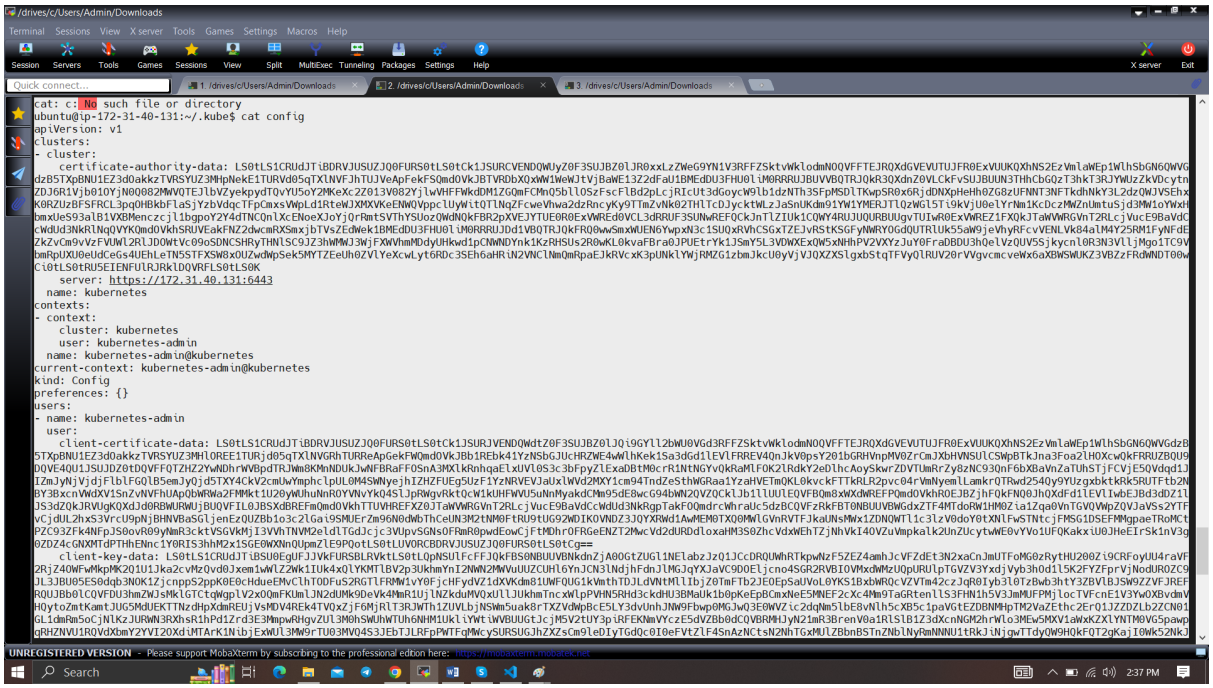
mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

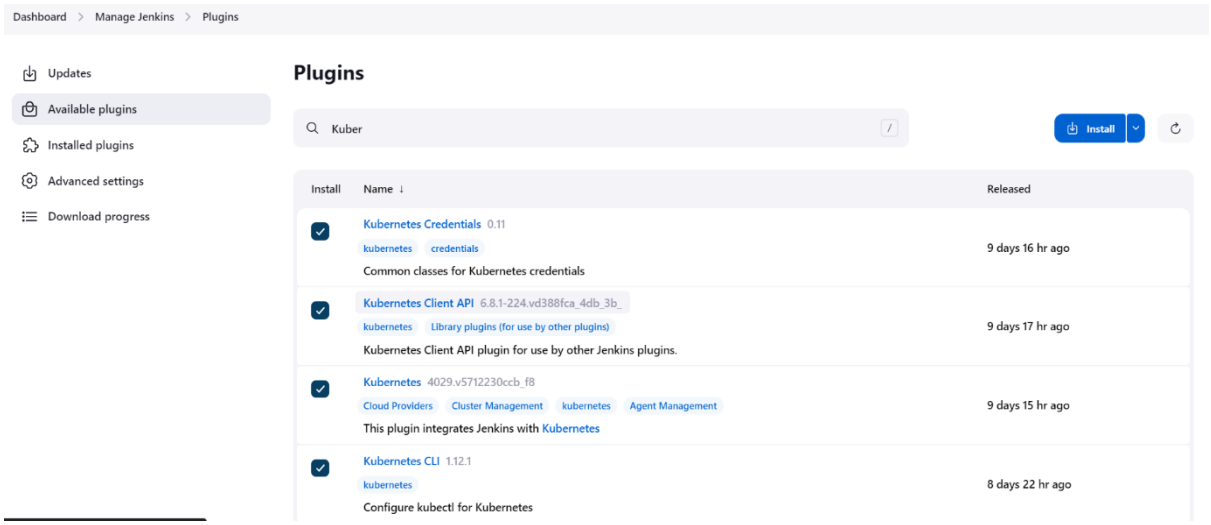
Copy the config file to Jenkins master or the local file manager and save it



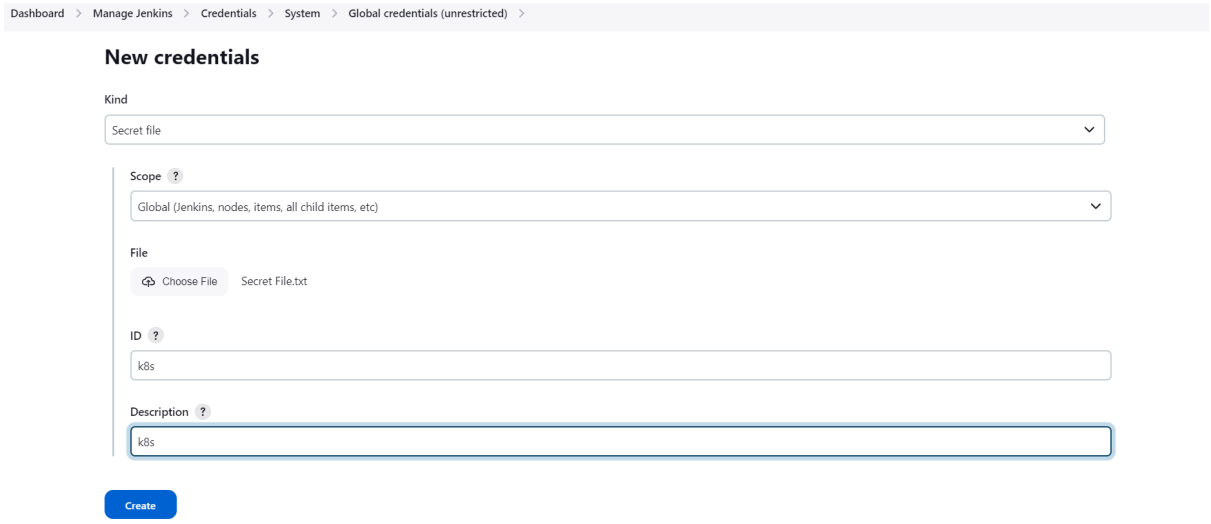
copy it and save it in documents or another folder save it as secret-file.txt

Note: create a secret-file.txt in your file explorer save the config in it and use this at the kubernetes credential section.

Install Kubernetes Plugin, Once it's installed successfully



goto manage Jenkins –> manage credentials –> Click on Jenkins global –> add credentials



Install Node_exporter on both master and worker

Let's add Node_exporter on Master and Worker to monitor the metrics

First, let's create a system user for Node Exporter by running the following command:

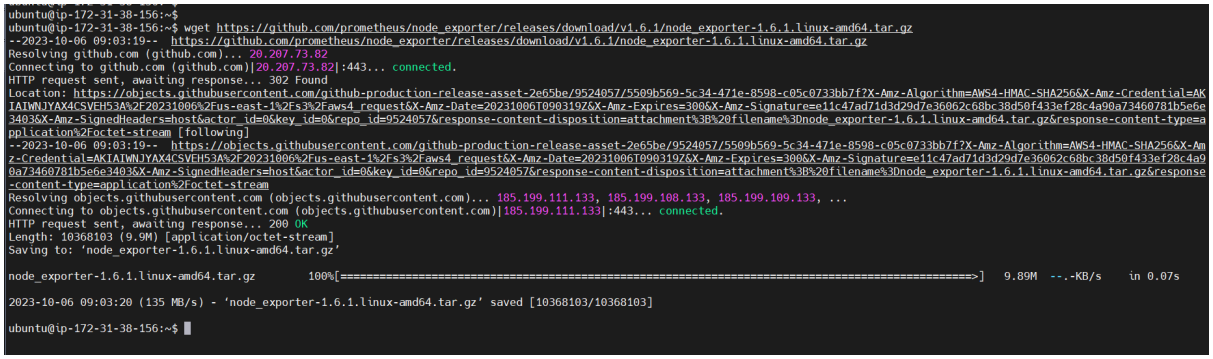
```
sudo useradd \
    --system \
    --no-create-home \
    --shell /bin/false node_exporter
```

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo useradd \
    --system \
    --no-create-home \
    --shell /bin/false node_exporter
ubuntu@ip-172-31-38-156:~$
```

You can **download Node Exporter** from the same page.

Use the wget command to download the binary.

wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz



Extract the node exporter from the archive.

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

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ ls
node_exporter-1.6.1.linux-amd64.tar.gz  prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ tar -xvf node_exporter-1.6.1.linux-amd64.tar.gz
node_exporter-1.6.1.linux-amd64/
node_exporter-1.6.1.linux-amd64/NOTICE
node_exporter-1.6.1.linux-amd64/node_exporter
node_exporter-1.6.1.linux-amd64/LICENSE
ubuntu@ip-172-31-38-156:~$
```

Move binary to the /usr/local/bin.

sudo mv \

node_exporter-1.6.1.linux-amd64/node_exporter \

/usr/local/bin/

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo mv \
  node_exporter-1.6.1.linux-amd64/node_exporter \
  /usr/local/bin/
ubuntu@ip-172-31-38-156:~$
```

Clean up, and delete node_exporter archive and a folder.

rm -rf node_exporter*

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ ls
node_exporter-1.6.1.linux-amd64  node_exporter-1.6.1.linux-amd64.tar.gz  prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ rm -rf node_exporter*
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ ls
prometheus-2.47.1.linux-amd64
ubuntu@ip-172-31-38-156:~$
```

Verify that you can run the binary.

node_exporter --version

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ node_exporter --version
node_exporter, version 1.6.1 (branch: HEAD, revision: 4a1b77600c1873a8233f3ffb55afcedbb63b8d84)
  build user:       root@586879db11e5
  build date:       20230717-12:10:52
  go version:       go1.20.6
  platform:         linux/amd64
  tags:             netgo osusergo static_build
ubuntu@ip-172-31-38-156:~$
```

Node Exporter has a lot of plugins that we can enable. If you run Node Exporter help you will get all the options.

node_exporter --help

–collector.logind We’re going to enable the login controller, just for the demo.

Next, create a similar systemd unit file.

sudo vim /etc/systemd/system/node_exporter.service

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo vim /etc/systemd/system/node_exporter.service
```

node_exporter.service

[Unit]

Description=Node Exporter

Wants=network-online.target

After=network-online.target

StartLimitIntervalSec=500

StartLimitBurst=5

[Service]

User=node_exporter

Group=node_exporter

Type=simple

Restart=on-failure

RestartSec=5s

ExecStart=/usr/local/bin/node_exporter \

  --collector.logind

[Install]

WantedBy=multi-user.target

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

StartLimitIntervalSec=500
StartLimitBurst=5

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

[Install]
WantedBy=multi-user.target
```

Replace Prometheus user and group to node_exporter, and update the ExecStart command.

To automatically start the Node Exporter after reboot, enable the service.

sudo systemctl enable node_exporter

Then start the Node Exporter.

sudo systemctl start node_exporter

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl enable node_exporter
Created symlink /etc/systemd/system/multi-user.target.wants/node_exporter.service → /etc/systemd/system/node_exporter.service.
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo systemctl start node_exporter
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
```

Check the status of Node Exporter with the following command:

sudo systemctl status node_exporter

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ 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 Fri 2023-10-06 09:07:39 UTC; 8s ago
   Main PID: 2030 (node_exporter)
      Tasks: 3 (limit: 1141)
     Memory: 2.0M
        CPU: 9ms
     CGroup: /system.slice/node_exporter.service
             └─2030 /usr/local/bin/node_exporter --collector.logind

Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=thermal_zone
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=time
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=timex
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=udp_queues
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=uname
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=vmstat
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=xfs
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.414Z caller=node_exporter.go:117 level=info collector=zfs
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.415Z caller=tls_config.go:274 level=info msg="Listening on" address=[::]:9100
Oct 06 09:07:39 ip-172-31-38-156 node_exporter[2030]: ts=2023-10-06T09:07:39.415Z caller=tls_config.go:277 level=info msg="TLS is disabled." http2=false address=[::]:9100
ubuntu@ip-172-31-38-156:~$
```

If you have any issues, check logs with journalctl

journalctl -u node_exporter -f --no-pager

At this point, we have only a single target in our Prometheus. There are many different service discovery mechanisms built into Prometheus. For example, Prometheus can dynamically discover targets in AWS, GCP, and other clouds based on the labels. In the following tutorials, I'll give you a few examples of deploying Prometheus in a cloud-specific environment. For this tutorial, let's keep it simple and keep adding static targets. Also, I have a lesson on how to deploy and manage Prometheus in the Kubernetes cluster.

To create a static target, you need to add job_name with static_configs. Go to Prometheus server

sudo vim /etc/prometheus/prometheus.yml

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ sudo vim /etc/prometheus/prometheus.yml
ubuntu@ip-172-31-38-156:~$
```

prometheus.yml

- job_name: node_export_masterk8s

  static_configs:

   - targets: ["<master-ip>:9100"]

 - job_name: node_export_workerk8s

  static_configs:

   - targets: ["<worker-ip>:9100"]

By default, Node Exporter will be exposed on port 9100.

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
          # - alertmanager:9093

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

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

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

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: node_export
    static_configs:
      - targets: ["localhost:9100"]

  - job_name: 'jenkins'
    metrics_path: '/prometheus'
    static_configs:
      - targets: ['13.234.114.77:8080']

  - job_name: node_export_workerk8s
    static_configs:
      - targets: ["65.0.45.118:9100"]

  - job_name: node_export_master__k8s
    static_configs:
      - targets: ["43.204.100.15:9100"]
```

Since we enabled lifecycle management via API calls, we can reload the Prometheus config without restarting the service and causing downtime.

Before, restarting check if the config is valid.

promtool check config /etc/prometheus/prometheus.yml

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ promtool check config /etc/prometheus/prometheus.yml
Checking /etc/prometheus/prometheus.yml
 SUCCESS: /etc/prometheus/prometheus.yml is valid prometheus config file syntax

ubuntu@ip-172-31-38-156:~$
```

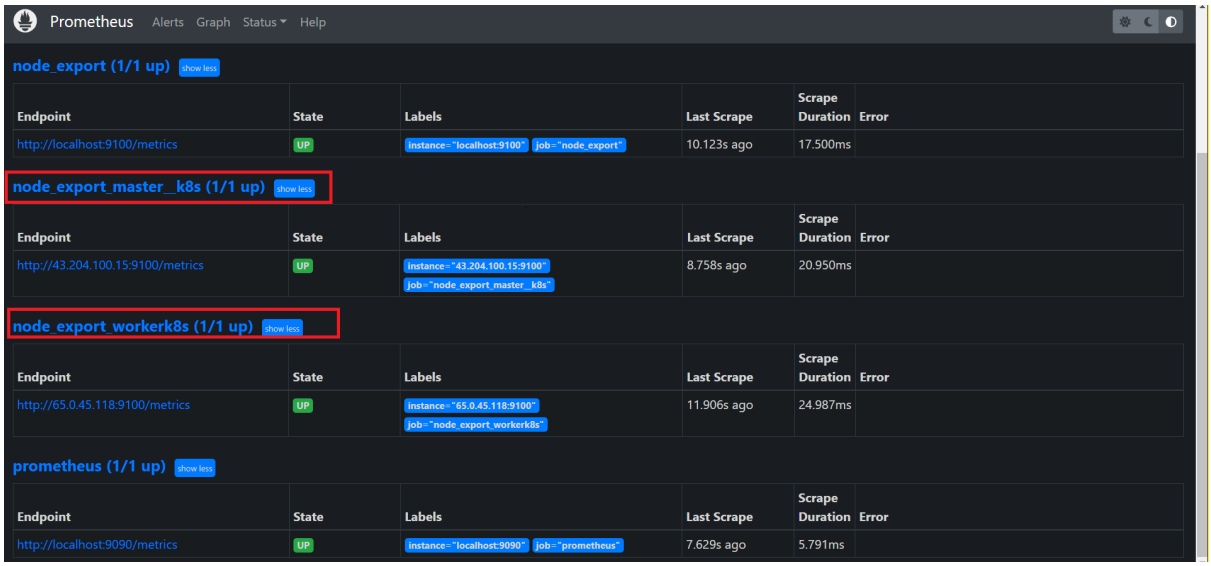
Then, you can use a POST request to reload the config.

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

```
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$
ubuntu@ip-172-31-38-156:~$ curl -X POST http://localhost:9090/-/reload
ubuntu@ip-172-31-38-156:~$
```

Check the targets section

http://<ip>:9090/targets

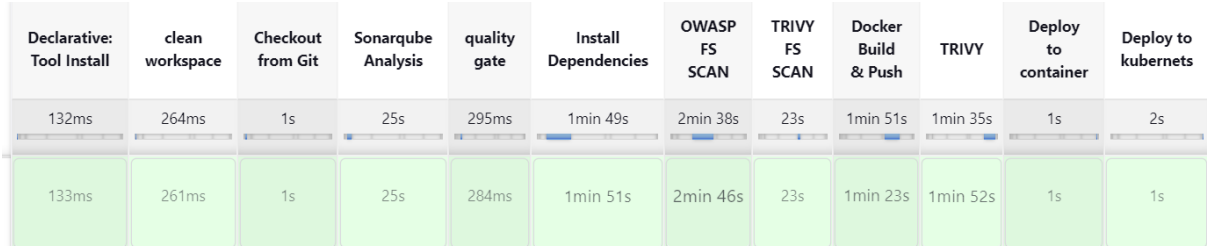


final step to deploy on the Kubernetes cluster

```
stage('Deploy to kubernets'){
        steps{
            script{
                dir('Kubernetes') {
                    withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {
                        sh 'kubectl apply -f deployment.yml'
                        sh 'kubectl apply -f service.yml'
```

```
            }
        }
    }
}
}
```

stage view



Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN	Docker Build & Push	TRIVY	Deploy to container	Deploy to kubernets
132ms	264ms	1s	25s	295ms	1min 49s	2min 38s	23s	1min 51s	1min 35s	1s	2s
133ms	261ms	1s	25s	284ms	1min 51s	2min 46s	23s	1min 23s	1min 52s	1s	1s

In the Kubernetes cluster(master) give this command

kubectl get all

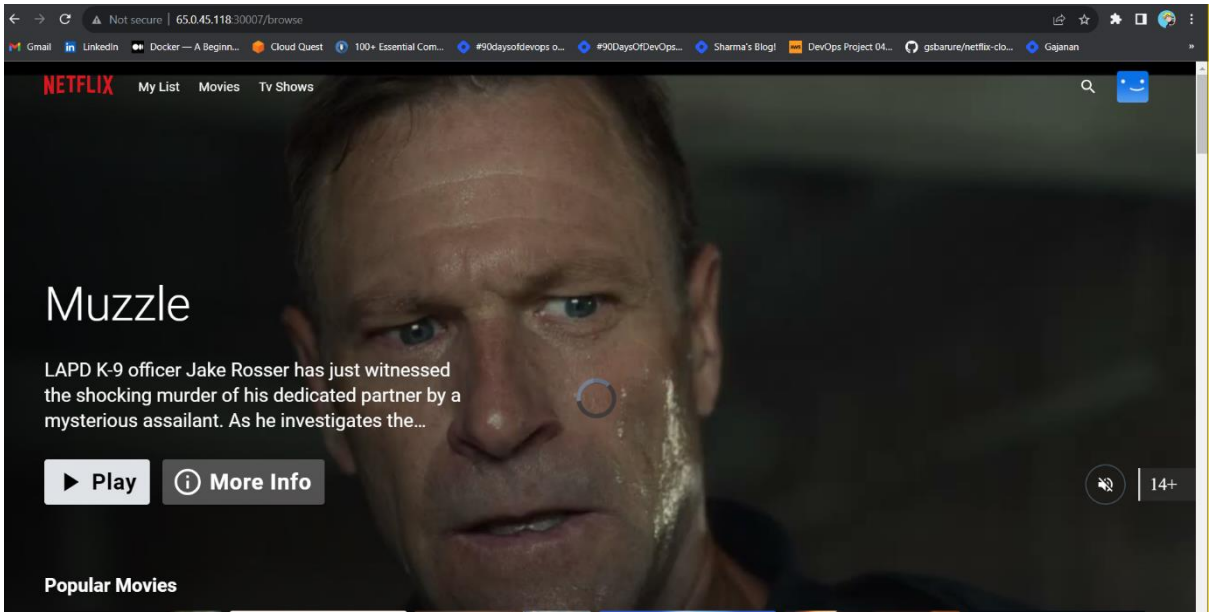
kubectl get svc

```
ubuntu@ip-172-31-40-131:~$ kubectl get all
NAME                             READY   STATUS    RESTARTS   AGE
pod/petshop-768578655f-kzcd9     1/1     Running   0          43s

NAME                 TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
service/kubernetes   ClusterIP      10.96.0.1        <none>        443/TCP        58m
service/petshop      LoadBalancer   10.104.122.152   <pending>     80:30699/TCP   21m

NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/petshop    1/1     1            1           43s

NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.apps/petshop-768578655f    1         1         1       43s
ubuntu@ip-172-31-40-131:~$
```

STEP 12:Access from a Web browser with

<public-ip-of-slave:service port>

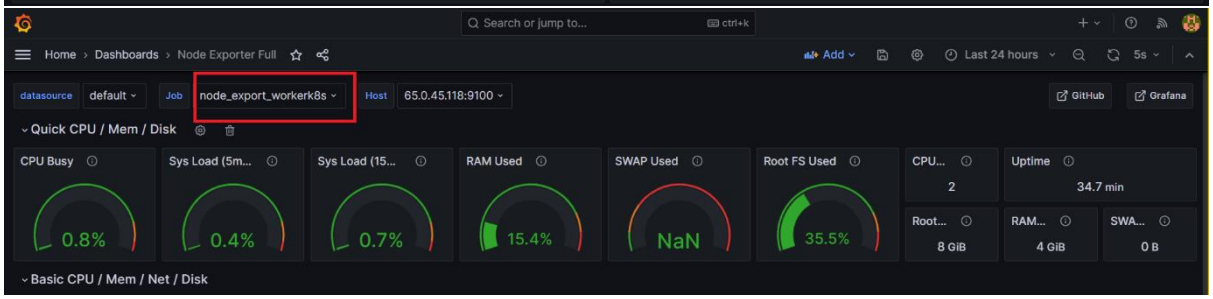
output:


Not secure | 65.0.45.118:30007/browse
Gmail
LinkedIn
Docker — A Beginn...
Cloud Quest
100+ Essential Com...
#90daysofdevops o...
#90DaysOfDevOps...
Sharma's Blog!
DevOps Project 04...
gsbarure/netflix-clo...
Gajanan
NETFLIX
My List
Movies
Tv Shows
Muzzle
LAPD K-9 officer Jake Rosser has just witnessed the shocking murder of his dedicated partner by a mysterious assailant. As he investigates the...
Play
More Info
14+
Popular Movies

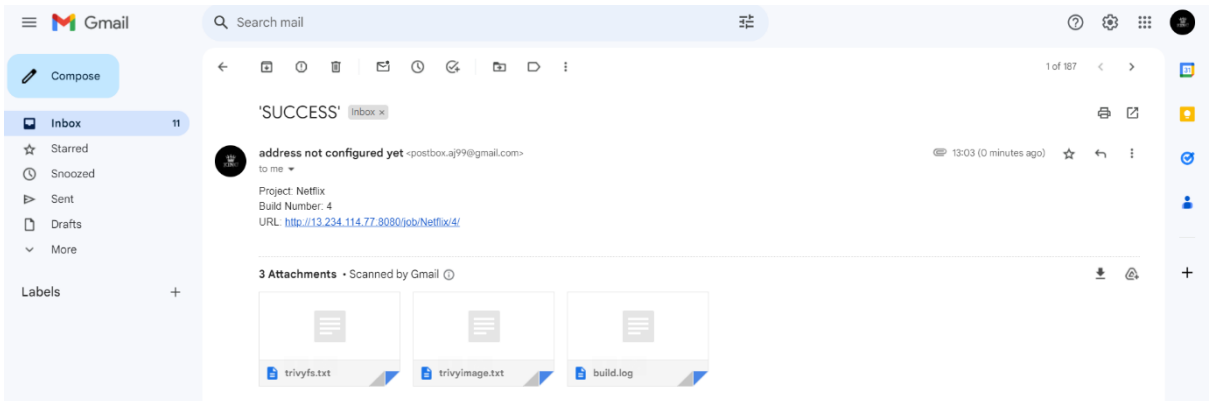
Monitoring


Home › Dashboards › Jenkins: Performance and Health Overview
Processing speed
0
Job queue duration
Queued rate
0
JVM free memory
Memory Usage
497 MB
Jenkins health
1.0
JVM Uptime
N/A
Jenkins nodes offline
None!
CPU Usage
0.00112%
Total Jobs
None!
Sucessful Jobs
4
Aborted Jobs
No data
Unstable Jobs
No data
Failed Jobs
None!
Jenkins queue size
0
Executor free
2
Executors health
Job Duration
Executor In-use
0


Home › Dashboards › Node Exporter Full
Add
Last 24 hours
5s
datasource
default
Job
node_export_master__k8s
Host
43.204.100.15:9100
GitHub
Grafana
Quick CPU / Mem / Disk
CPU Busy
3.2%
Sys Load (5m...
5.4%
Sys Load (15...
6.5%
RAM Used
25.5%
SWAP Used
NaN
Root FS Used
47.6%
CPU...
2
Uptime
34.5 min
Root...
8 GiB
RAM...
4 GiB
SWA...
0 B
Basic CPU / Mem / Net / Disk
CPU Basic
Memory Basic
Busy System
Busy User
Busy Iowait
Busy IRQs
Busy Other
Idle
RAM Total
RAM Used
RAM Cache + Buffer
RAM Free
SWAP Used


Home › Dashboards › Node Exporter Full
Add
Last 24 hours
5s
datasource
default
Job
node_export_workerk8s
Host
65.0.45.118:9100
GitHub
Grafana
Quick CPU / Mem / Disk
CPU Busy
0.8%
Sys Load (5m...
0.4%
Sys Load (15...
0.7%
RAM Used
15.4%
SWAP Used
NaN
Root FS Used
35.5%
CPU...
2
Uptime
34.7 min
Root...
8 GiB
RAM...
4 GiB
SWA...
0 B
Basic CPU / Mem / Net / Disk

Mail


Gmail
Search mail
Compose
Inbox 11
Starred
Snoozed
Sent
Drafts
More
Labels
'SUCCESS'
address not configured yet
to me
Project: Netflix
Build Number: 4
URL: http://13.234.114.77:8080/job/Netflix/4/
3 Attachments • Scanned by Gmail
trivyfs.txt
trivyimage.txt
build.log

Step 13: Terminate instances.

Complete Pipeline

```
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/Aj7Ay/Netflix-clone.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=AJ7AYe14eca3e76864yah319b92 -t
netflix ."
                    sh "docker tag netflix sevenajay/netflix:latest "
                    sh "docker push sevenajay/netflix:latest "
                }
```

```
            }
        }
    }
    stage("TRIVY"){
        steps{
            sh "trivy image sevenajay/netflix:latest > trivyimage.txt"
        }
    }
    stage('Deploy to container'){
        steps{
            sh 'docker run -d --name netflix -p 8081:80 sevenajay/netflix:latest'
        }
    }
    stage('Deploy to kubernets'){
        steps{
            script{
                dir('Kubernetes') {
                    withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'k8s',
namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {
                        sh 'kubectl apply -f deployment.yml'
                        sh 'kubectl apply -f service.yml'
                    }
                }
            }
        }
    }

}
post {
 always {
    emailext attachLog: true,
```

```
        subject: "'${currentBuild.result}'",
        body: "Project: ${env.JOB_NAME}<br/>" +
          "Build Number: ${env.BUILD_NUMBER}<br/>" +
          "URL: ${env.BUILD_URL}<br/>",
        to: 'postbox.aj99@gmail.com',
        attachmentsPattern: 'trivyfs.txt,trivyimage.txt'
    }
}
```