

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
```

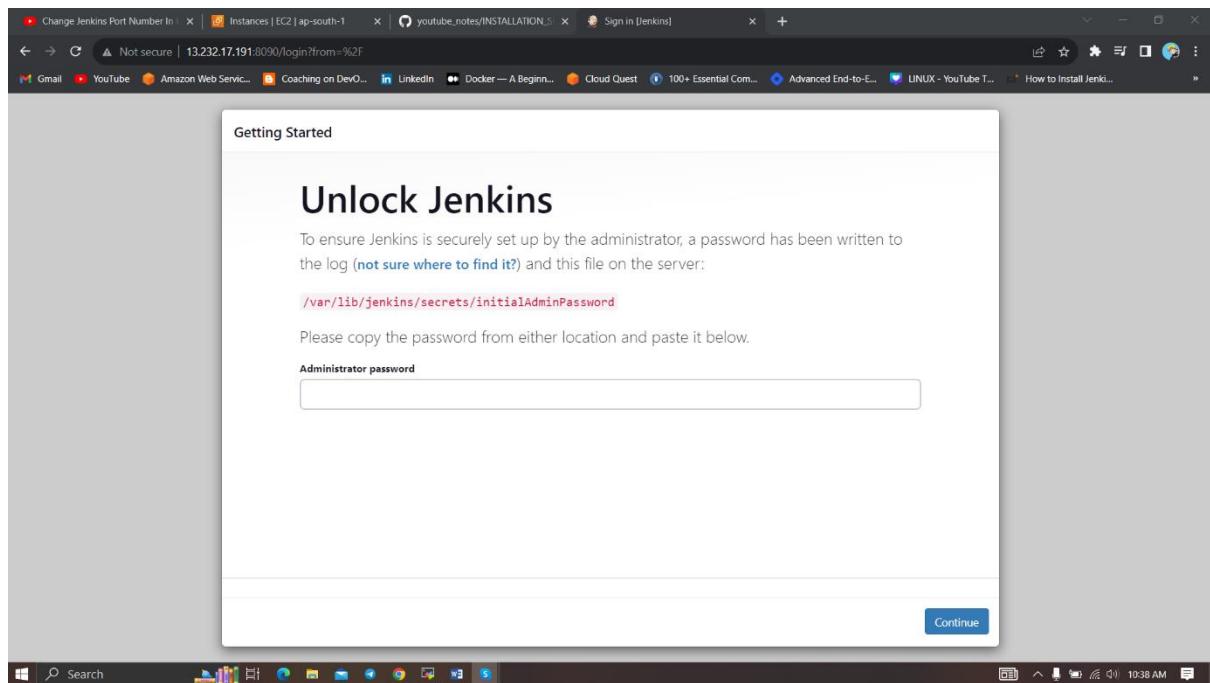
```
sudo systemctl status jenkins  
sudo chmod 777 jenkins.sh  
.jenkins.sh # this will install jenkins
```

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.

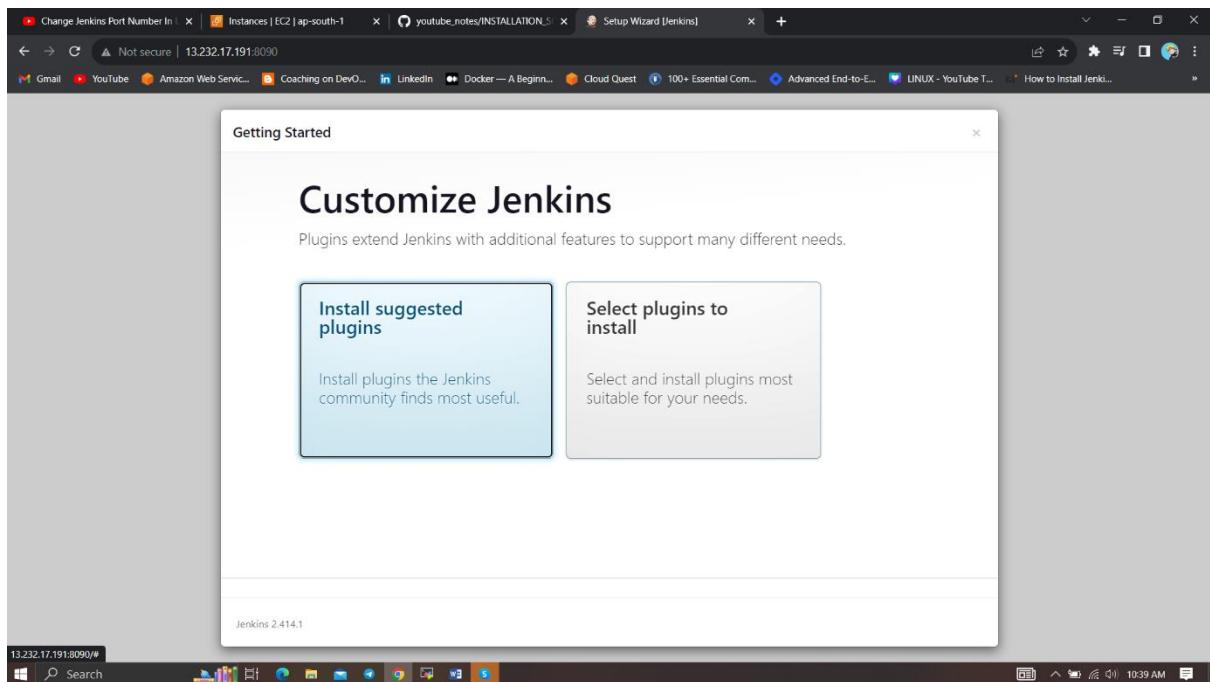
Now, grab your Public IP Address

<EC2 Public IP Address:8080>

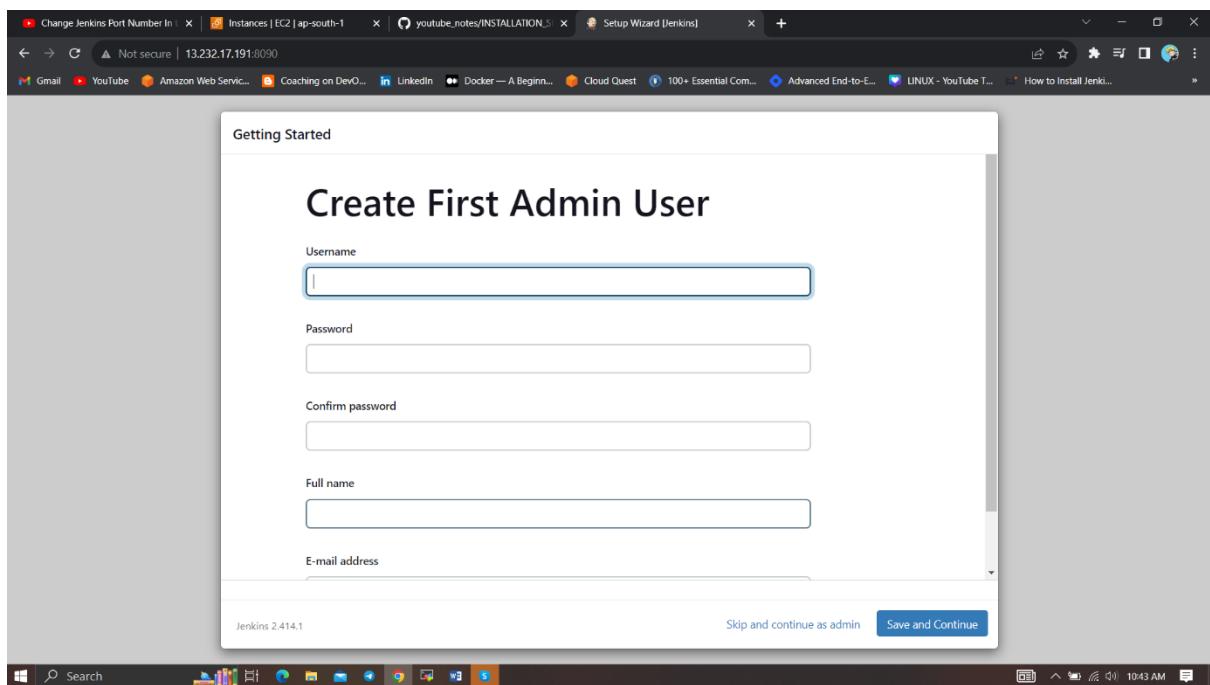
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



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`

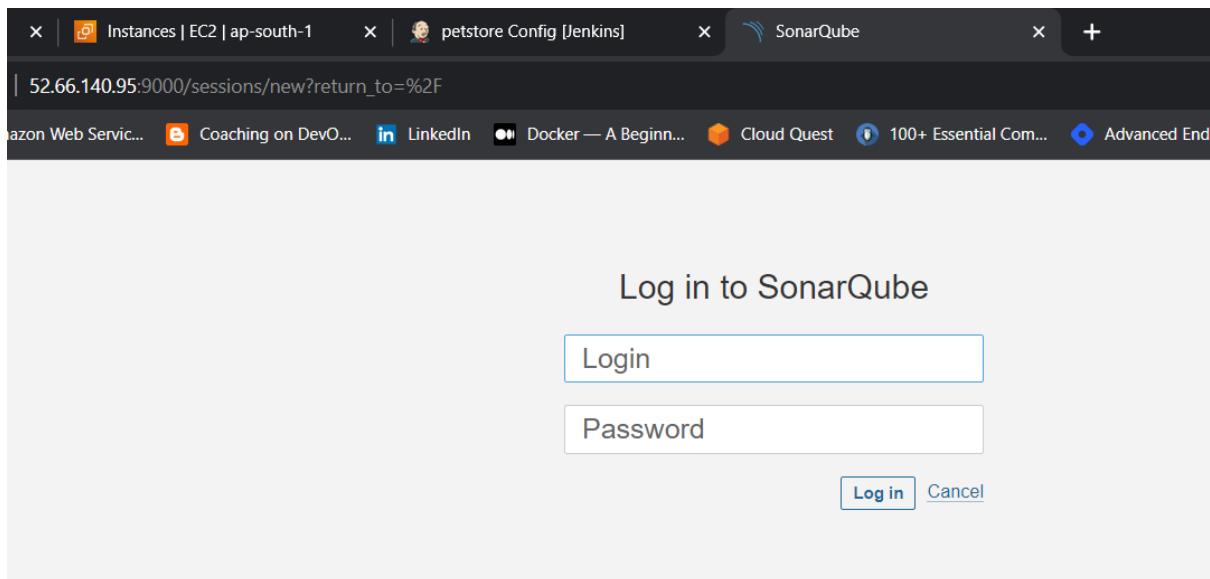
`sudo chmod 777 /var/run/docker.sock`

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
44bab92f8eb7: Pull complete
2cbeec57a24e: Pull complete
20481384b6a: Pull complete
bf7b177a74f8: Pull complete
38617faac714: Pull complete
706f20f5f85e: Pull complete
65a2956b2c57: Pull complete
Digest: sha256:1a118f8ab969dfc3d4ea8b4455a5a6560654511c88a6816f1603f764d5dcc77c
Status: Downloaded newer image for sonarqube:lts-community
4b60c96bf9ad3d62289436af7f752fdb84993092d0ca3065e2f2e32301b50139
ubuntu@ip-172-31-42-253:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS             NAMES
4b60c96bf9ad        sonarqubelts-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:~$
```

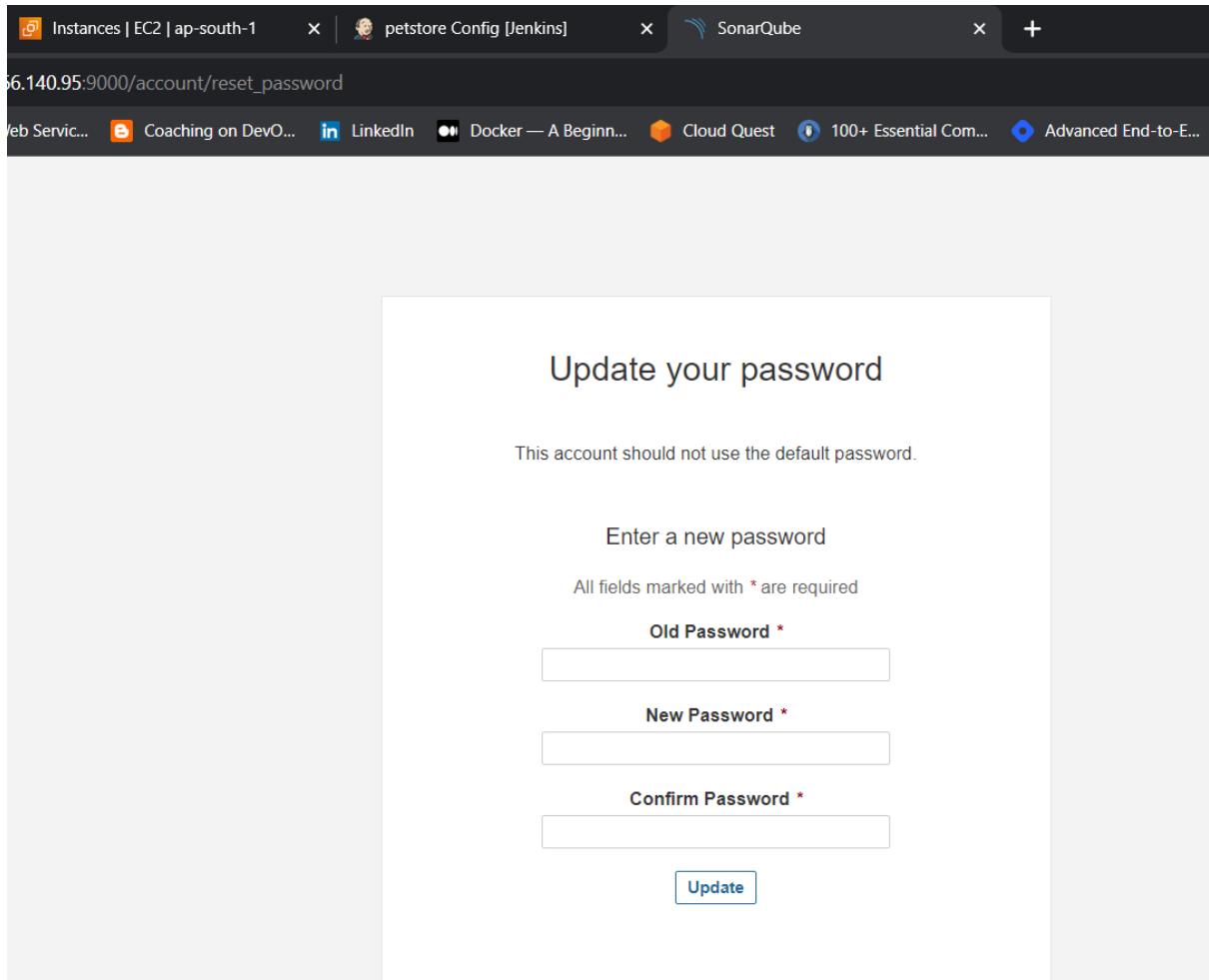
Now our sonarqube is up and running



Enter username and password, click on login and change password

username admin

password admin



Update New password, This is Sonar Dashboard.

The screenshot shows the SonarQube interface for creating a new project. At the top, there's a navigation bar with links like Gmail, YouTube, Amazon Web Servic..., Coaching on DevO..., LinkedIn, Docker — A Beginner's Guide, Cloud Quest, 100+ Essential Com..., Advanced End-to-End..., LINUX - YouTube T..., and How to Install Jenkins. Below the navigation is a header with 'sonarcube' and tabs for Projects, Issues, Rules, Quality Profiles, Quality Gates, Administration, and a search bar. The main content area has a heading 'How do you want to create your project?'. It says 'Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform. First, you need to set up a DevOps platform configuration.' Below this are five cards with icons and labels: 'From Azure DevOps' (blue icon), 'From Bitbucket Server' (blue icon), 'From Bitbucket Cloud' (blue icon), 'From GitHub' (GitHub logo icon), and 'From GitLab' (GitLab logo icon). Each card has a 'Set up global configuration' link below it.

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

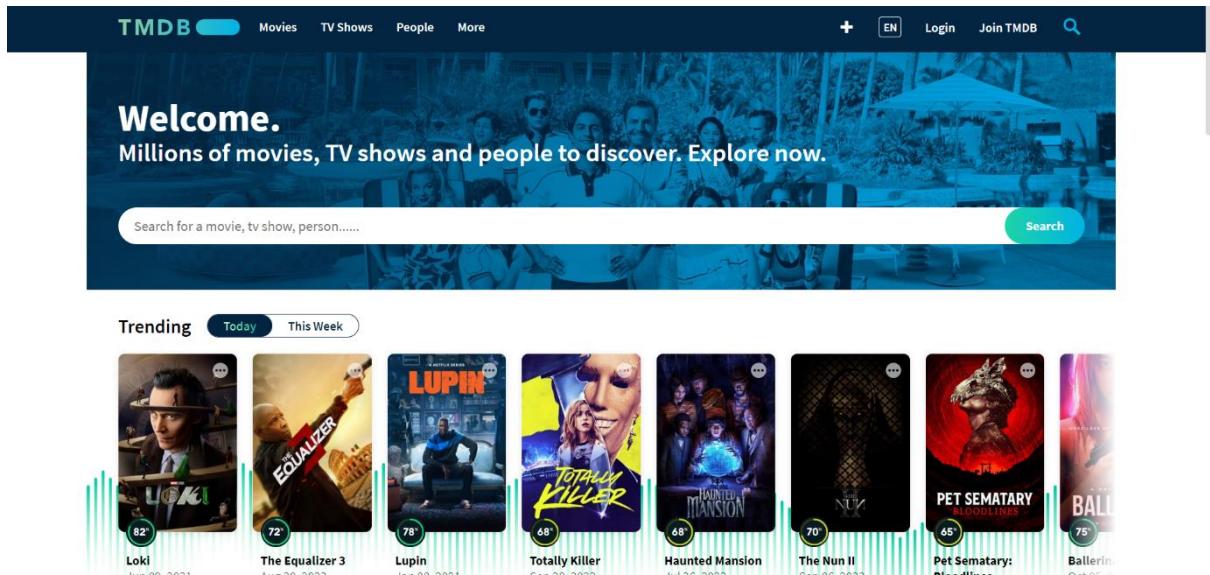
Step 3: Create a TMDB API Key

Next, we will create a TMDB API key

Open a new tab in the Browser and search for TMDB

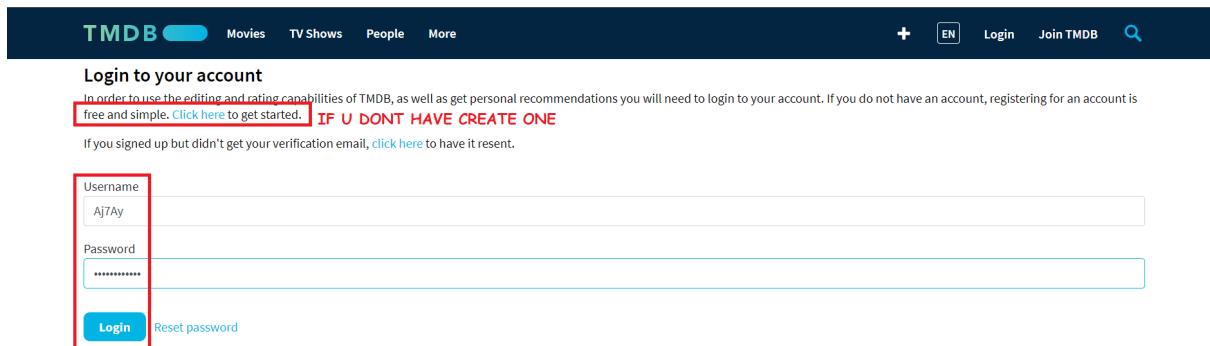
The screenshot shows a Google search results page for the query 'TMDB'. The first result is 'The Movie Database (TMDB)' with the URL <https://www.themoviedb.org>. The snippet describes TMDB as a popular, user-editable database for movies and TV shows. Below the snippet are links for 'API Reference', 'API key', 'Login to your account', and 'Popular Movies'. To the right of the snippet, there's a sidebar with the TMDB logo, a 'Laravel TMDB' badge, and a 'Results from themoviedb.org' section. Further down, there's a 'The Movie Database' card with details: Website (themoviedb.org), Category: film database, Date launched: 2008, and Owner: Fan TV.

Click on the first result, you will see this page

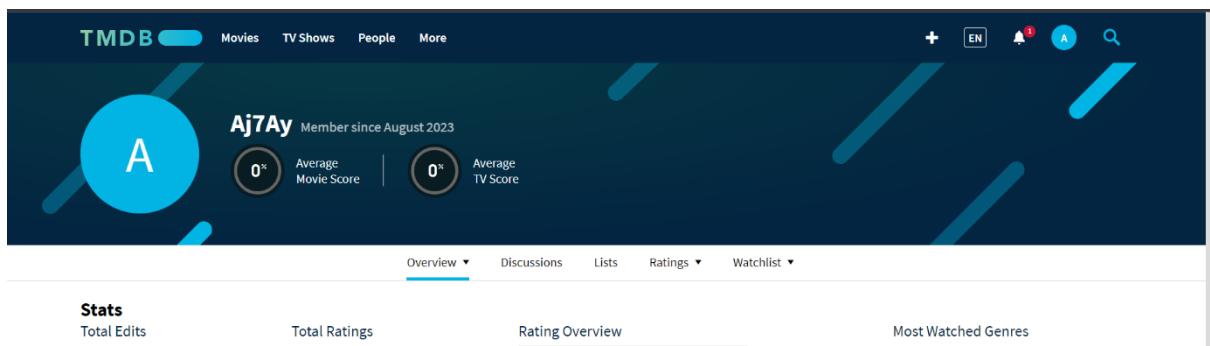


Click on the Login on the top right. You will get this page.

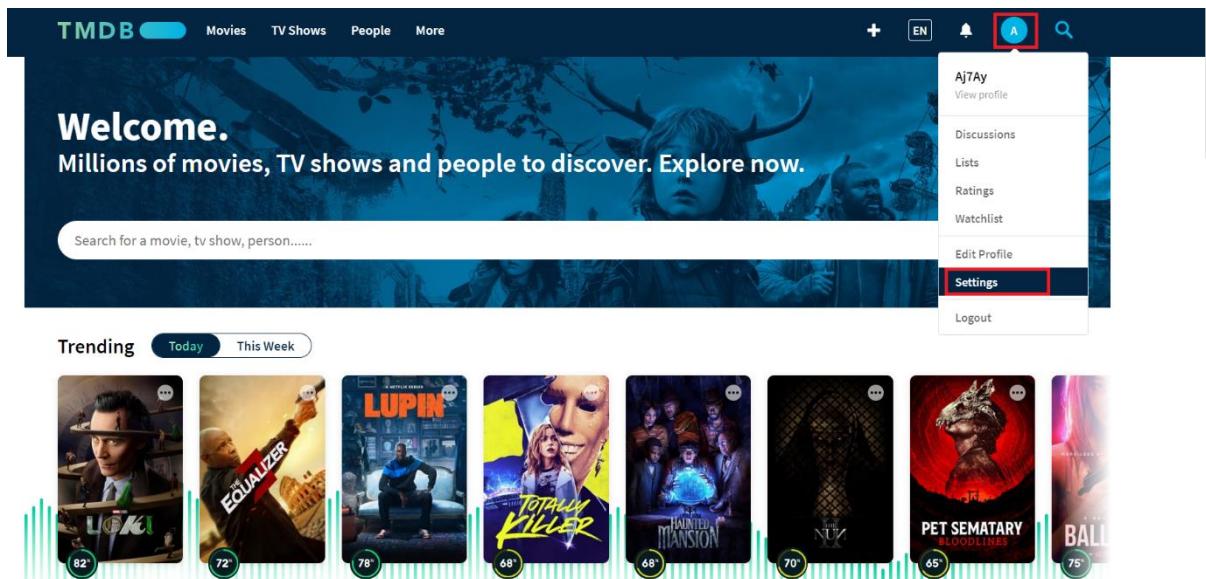
You need to create an account here. click on click here. I have account that's why i added my details there.



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.

A screenshot of the TMDB account settings page for 'Aj7Ay'. On the left, a sidebar menu has 'API' highlighted with a red box. The main content area is titled 'Account Settings' and includes fields for 'Default Language' (set to English (en-US)), 'Fallback Language' (set to None (Don't Fallback)), 'Country' (set to India), 'Timezone - Auto detect?' (set to Asia - Kabul), 'Include Adult Items in Search?' (set to No), and 'Filter Profanity?' (set to Yes). A red box highlights the 'API' link in the sidebar.

Now click on create

A screenshot of the TMDB API documentation page. On the left, a sidebar menu has 'API' highlighted with a red box. The main content area has tabs for 'API', 'Overview', and 'Create', with 'Create' highlighted by a red box. The 'Create' tab contains text about TMDB's API service and a link to find attribution logos. It also includes sections for 'Documentation' (link to developer.themoviedb.org) and 'Support' (link to support forums). A red box highlights the 'Create' tab in the navigation bar.

Click on Developer

The screenshot shows the TMDB API settings interface. On the left, a sidebar titled 'Settings' contains links for 'Edit Profile', 'Account Settings', 'Streaming Services', 'Notification Settings', 'Blocked Users', and 'Import List'. The main area is titled 'API' with sub-links 'Overview' and 'Create'. A question 'What type of API key do you wish to register?' is displayed. Two options are shown: 'Developer' (selected, highlighted with a red box) and 'Professional'. The 'Developer' option includes a list of bullet points: 'You are an individual', 'Your project is still in development', 'Your project is non profit', and 'Your project is ad supported'. The 'Professional' option includes a list of bullet points: 'You represent a company', 'Your project is for profit (not ad supported)', and 'You are an OEM or hardware vendor'.

Now you have to accept the terms and conditions.

12. General Terms

1. **Relationship of the Parties.** Notwithstanding any provision hereof, for all purposes of the Terms of Use, you and TMDB shall be and act independently and not as partner, joint venturer, agent, employee or employer of the other. You shall not have any authority to assume or create any obligation for or on behalf of TMDB, express or implied, and you shall not attempt to bind TMDB to any contract.
2. **Invalidity of Specific Terms.** If any provision of the Terms of Use is found by a court of competent jurisdiction to be invalid, the parties nevertheless agree that the other provisions of this agreement will remain in full force and effect and the court should endeavor to give effect to the parties' intentions as reflected in the invalid provision.
3. **Location of Lawsuit and Choice of Law.** THE TERMS OF USE AND THE RELATIONSHIP BETWEEN YOU AND TMDB SHALL BE GOVERNED BY THE LAWS OF THE STATE OF CALIFORNIA WITHOUT REGARD TO ITS CONFLICT OF LAW PROVISIONS. YOU AND TMDB AGREE TO SUBMIT TO THE PERSONAL JURISDICTION OF THE COURTS LOCATED WITHIN THE COUNTY OF SAN MATEO, CALIFORNIA.
4. **No Waiver of Rights by TMDB.** TMDB's failure to exercise or enforce any right or provision of the Terms of Use shall not constitute a waiver of such right or provision.
5. **No Transfer.** The rights and obligations of these Terms of Use is personal to you and may not be transferred by you, either voluntarily or by operation of law.
6. **Notice.** Any notice to be sent to you under these Terms of Use may be sent via email, post, or any other reasonable means, at the contact information provided by you to TMDB from time to time. It is your obligation to insure that this information is current.

Miscellaneous. The section headings and subheadings contained in this agreement are included for convenience only, and shall not limit or otherwise affect the terms of the Terms of Use. Any construction or interpretation to be made of the Terms of Use shall not be construed against the drafter. The Terms of Use constitute the entire agreement between TMDB and you with respect to the subject matter hereof.

This Agreement was last updated on: July 28, 2014.

Cancel

Accept

CW

Provide basic details

API Overview Create

Type of Use
Desktop Application

Application Name
dEM

Application URL
NOT AVAILABLE

Application Summary
jdf

Sharing Settings
jdfjkjkjjkkjkjk

API

Delete Account

First Name
Code

Last Name
Word

Email Address
[REDACTED]

Address 1
ddddd

City
ffffffss

Zip Code
225588

Phone Number (no parenthesis or dashes)

- Mali
- Malta
- Marshall Islands
- Martinique
- Mauritania
- Mauritius
- Mavotte
- India

Submit

Click on submit and you will get your API key.

Notification Settings

Blocked Users

Import List

Sharing Settings

Sessions

API

Delete Account

Support

If you have questions or comments about the information covered here, please create a post on our support forums.

API Details

If you'd like to edit the details of your app, [click here](#).

API Key

8f757ea1c6e3dc2deef92fb4682d7e78

API Read Access Token

eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiI4Zjc1N2VhMWM2ZTNKYZJkZWVmOTJmYjQ2ODJkN2U3OCIsInN1YiI6IjY0NzE5NWRhODgxM2U0MDEwMzU2ZGNmNCIsInJhb3BlcyI6WyJhcGlfcmVhZCJdLCJ2ZXJzaW9uljoxfQ.K6Pu7To0wWp9C4PVW-ZvoViRJh-CsRpefc5Czt6ObX0

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/6838921/2f9b737-63a0-428b-adb5-0294482fd743?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=IAIWNYJAX4CSVEH53A%2F20231006%2fus-east-1%2F%2Faws4_request%2Famz-Date=20231006T085150Z&X-Amz-Expires=300&X-Amz-Signature=6962d5844eaad7d1a082e0285936e53a7a6976aaef71b5d51765dbfe6a61c&X-Amz-SignedHeaders=host&actor_id=6838921&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.47.1.linux-amd64.tar.gz&response-content-type=application/octet-stream [following]
--2023-10-06 08:51:59  https://objects.githubusercontent.com/github-production-release-asset-2e65be/6838921/2f9b737-63a0-428b-adb5-0294482fd743?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=IAIWNYJAX4CSVEH53A%2F20231006%2fus-east-1%2F%2Faws4_request%2Famz-Date=20231006T085150Z&X-Amz-Expires=300&X-Amz-Signature=6962d5844eaad7d1a082e0285936e53a7a6976aaef71b5d51765dbfe6a61c&X-Amz-SignedHeaders=host&actor_id=6838921&response-content-disposition=attachment%3B%20filename%3Dprometheus-2.47.1.linux-amd64.tar.gz&response-content-type=application/octet-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: 9513066 (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]
```

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:~$ 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$ 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$ 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  
drwxr-xr-x  2 ubuntu ubuntu 4096 Oct  6 08:50 .cache/  
-rw-r--r--  1 ubuntu ubuntu   807 Jan  6 2022 .profile  
drwxr----- 2 ubuntu ubuntu 4096 Oct  6 08:49 .ssh/  
-rw-r--r--  1 ubuntu ubuntu      0 Oct  6 08:50 .sudo_as_admin_successful  
-rv-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/  
-rv-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:~$ 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:~$ 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:~$
```

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
```

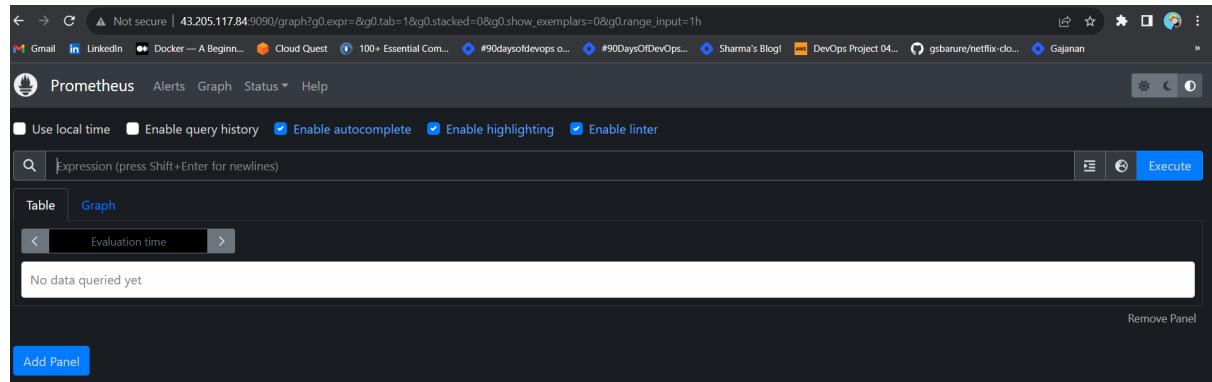
```
ubuntu@ip-172-31-38-156:~$ 
ubuntu@ip-172-31-38-156:~$ sudo systemctl status prometheus
● prometheus.service - Prometheus
   Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-10-06 09:00:39 UTC; 13s ago
       Main 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.libl
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=tls_config.go:761 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=tls_config.go:777 level=info component=tsdb msg="WAL replay completed" checkpoint_replay_duration=42.05s
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:1266 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: 1-28 (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  
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@586879db1e5  
  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:~$ 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:~$ sudo systemctl start node_exporter
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:~$ 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=tme
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=tmemex
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:~$ 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:~$ curl -X POST http://localhost:9090/-/reload
ubuntu@ip-172-31-38-156:~$
```

Check the targets section

<http://<ip>:9090/targets>

The screenshot shows the Prometheus Targets page. At the top, there are tabs for 'All', 'Unhealthy', and 'Collapse All'. A search bar is followed by filter checkboxes for 'Unknown', 'Unhealthy', and 'Healthy'. Below this, there are two sections: 'node_export (1/1 up)' and 'prometheus (1/1 up)'. Each section has a table with columns: Endpoint, State, Labels, Last Scrape, Scrape Duration, and Error. In the 'node_export' section, the endpoint is 'http://localhost:9100/metrics', state is 'UP', labels include 'instance="localhost:9100"', 'job="node_export"', last scrape was 1.953s ago, and scrape duration was 25.413ms. In the 'prometheus' section, the endpoint is 'http://localhost:9090/metrics', state is 'UP', labels include 'instance="localhost:9090"', 'job="prometheus"', last scrape was 2.748s ago, and scrape duration was 6.655ms.

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9100/metrics	UP	instance="localhost:9100", job="node_export"	1.953s ago	25.413ms	

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090", job="prometheus"	2.748s ago	6.655ms	

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:~$ 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:~$ 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:~$
```

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:~$ 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:~$
```

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 kB]
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:~$ 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:~$ 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 Process: 3263 (/usr/share/grafana/bin/grafana server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana.pid --packaging=deb cfg:default.paths.logs=/var/log/grafana)
   Tasks: 2 (limit: 1141)
   Memory: 175.0M
      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.pid --packaging=deb cfg:default.paths.logs=/var/log/grafana

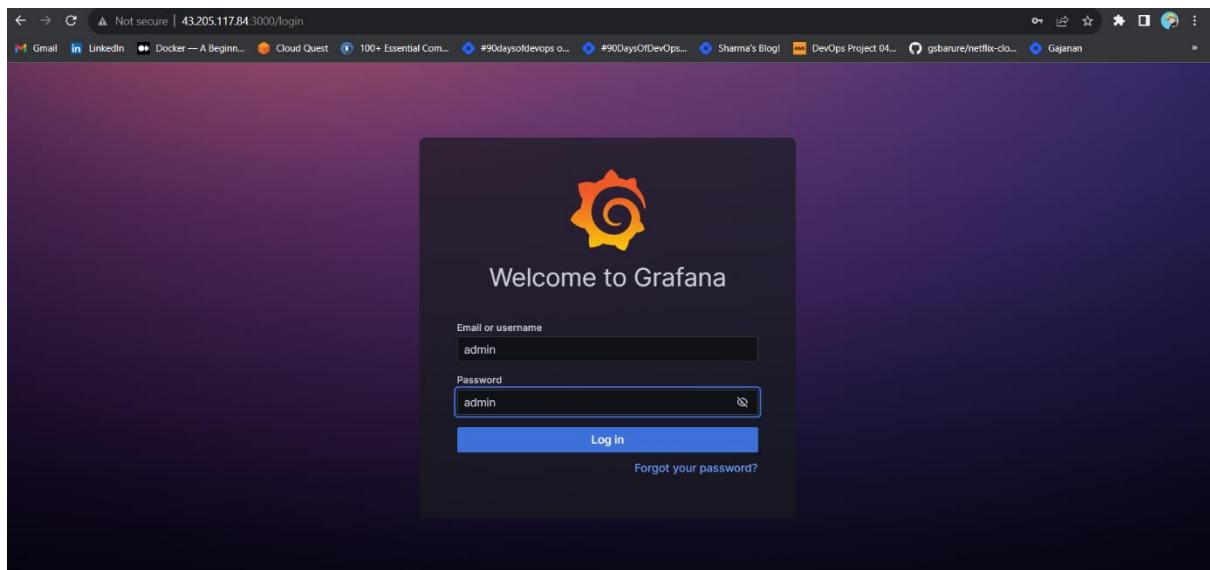
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=migrator t=2023-10-06T09:14:06.750063027Z level=info msg="Executing migration" id="Add unique index for folder.title and folder.description"
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=migrator t=2023-10-06T09:14:06.757199101Z level=info 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.js"
Oct 06 09:14:06 ip-172-31-38-156 grafana[3263]: logger=plugin.loader t=2023-10-06T09:14:06.883523142 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 funding plugins as directory does not exist" path=/var/lib/grafana/plugins
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"

```

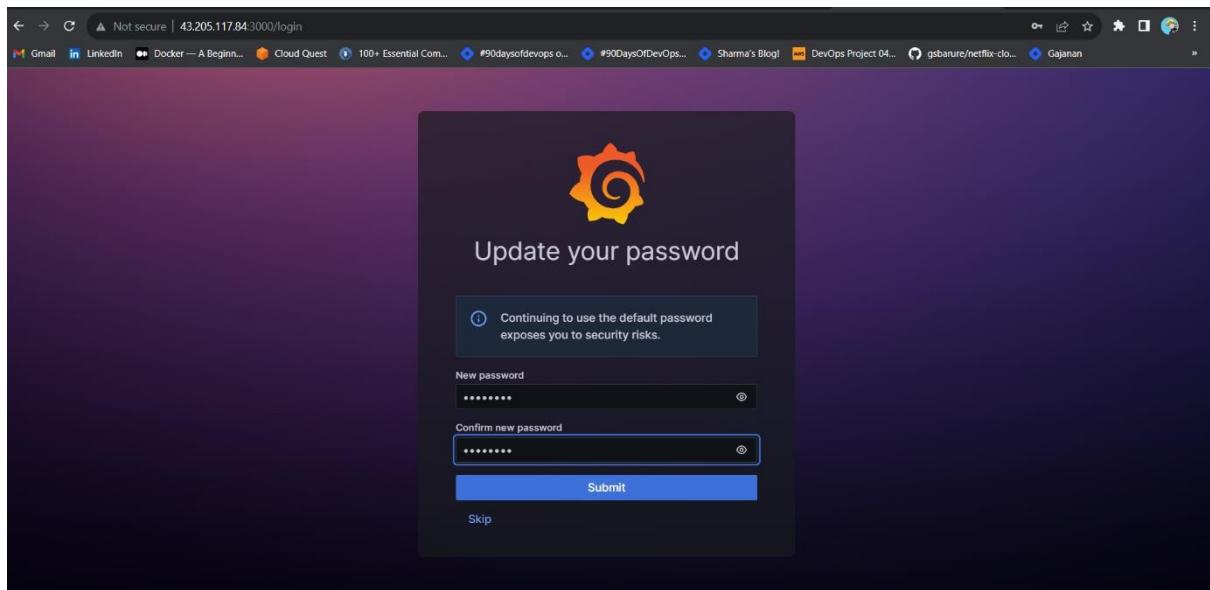
Go to <http://<ip>:3000> and log in to the Grafana using default credentials. The username is admin, and the password is admin as well.

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.

Welcome to Grafana

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

TUTORIAL DATA SOURCE AND DASHBOARDS

Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

DATA SOURCES

Add your first data source

DASHBOARDS

Create your first dashboard

Need help? Documentation Tutorials Community Public Slack

Remove this panel

Learn how in the docs

Learn how in the docs

Click Add data source and select Prometheus.

The screenshot shows the 'Add data source' interface in Grafana. Under the 'Time series databases' heading, the 'Prometheus' option is selected and highlighted with a red box. The 'Prometheus' entry includes an icon of a hand holding a torch, the text 'Open source time series database & alerting', and a 'Core' status indicator. Other options like 'Graphite' and 'InfluxDB' are also listed below it.

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

<public-ip:9090>

The screenshot shows the configuration page for the 'Prometheus' data source. The 'Settings' tab is active. A callout box points to the 'Prometheus server URL' input field, which contains the value 'http://43.205.117.84:9090'. Other fields in the 'HTTP' section include 'Allowed cookies' and 'Timeout'.

Click on Save and Test.

The screenshot shows the 'Data sources' configuration page for a Prometheus connection. The 'Prometheus type' dropdown is set to 'Choose'. The 'Cache level' is set to 'Low'. Under 'Other', there are fields for 'Custom query parameters' (with an example: max_source_resolution=5m&timeout) and 'HTTP method' (set to 'POST'). A 'Save & test' button at the bottom is highlighted with a red box.

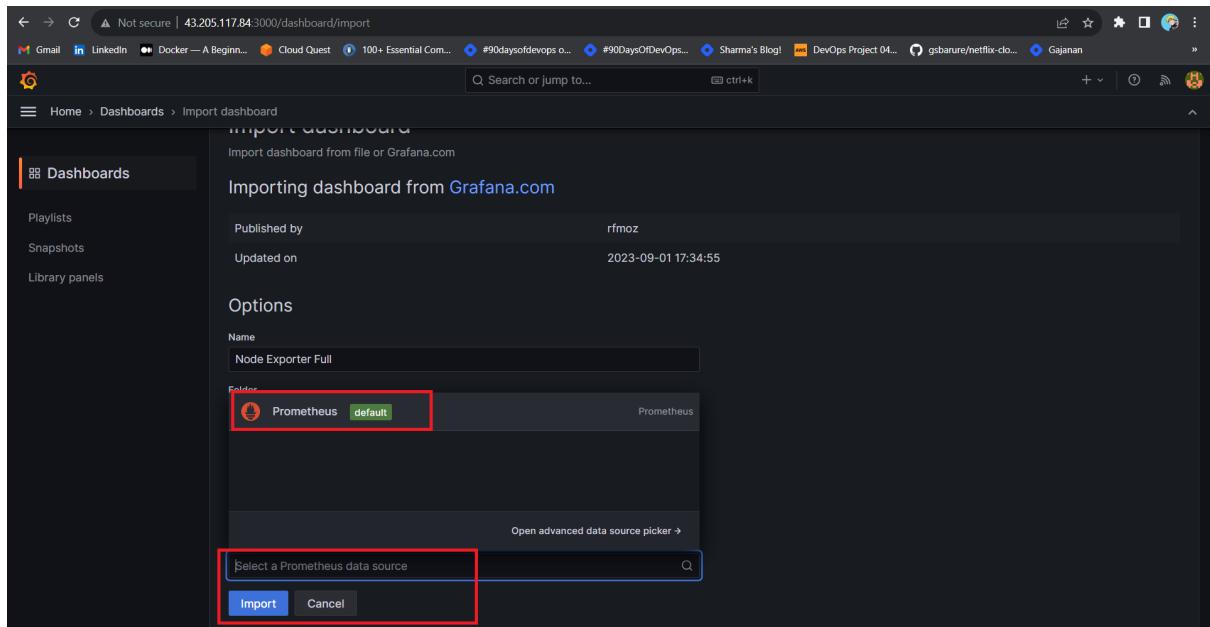
Let's add Dashboard for a better view

The screenshot shows the 'Dashboards' list page. The 'Dashboards' section is highlighted with a red box. On the right side, there are buttons for 'New dashboard', 'Import dashboard' (which is highlighted with a red box), and 'Create alert rule'.

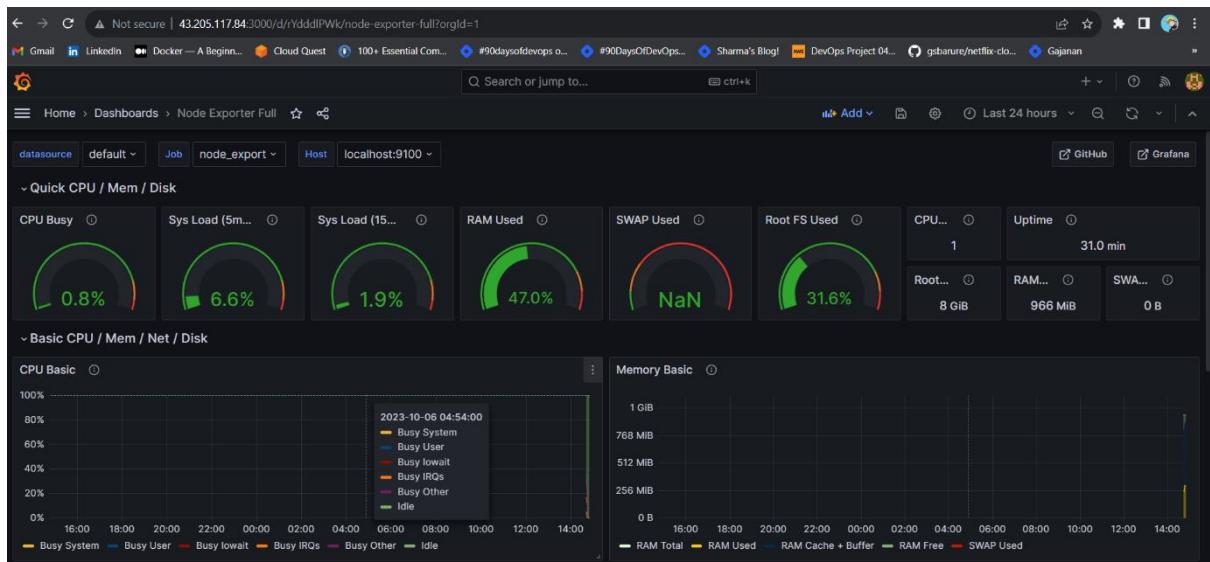
Click on Import Dashboard paste this code 1860 and click on load

The screenshot shows the 'Import dashboard' screen. The 'Import via grafana.com' section is highlighted with a red box. Below it, the code '1860' is pasted into the input field. The 'Load' button at the bottom is highlighted with a red box.

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

The screenshot shows the Jenkins Plugins page. In the top right corner, there is a search bar with the placeholder 'Search (CTRL+K)' and a user icon labeled 'Ajay'. Below the search bar are several small icons. The main area is titled 'Plugins' and contains a sidebar with links: 'Updates', 'Available plugins' (which is selected and highlighted in grey), 'Installed plugins', 'Advanced settings', and 'Download progress'. A search bar at the top has the text 'prome'. Below it, a table lists a single plugin: 'Prometheus metrics 2.3.3'. The table columns are 'Install', 'Name', and 'Released'. The 'Install' button is greyed out. The 'Name' column shows 'Prometheus metrics 2.3.3' with a red box around it. The 'Released' column shows '17 days ago'. Below the table, a note states: 'Jenkins Prometheus Plugin expose an endpoint (default /prometheus) with metrics where a Prometheus Server can scrape.' At the bottom right of the table area is a blue 'Install' button.

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

The screenshot shows the Jenkins System configuration page under the 'Prometheus' section. The URL is 'Dashboard > Manage Jenkins > System > Prometheus'. The page has several input fields: 'Path' (set to 'prometheus'), 'Default Namespace' (set to 'default'), and a checkbox for 'Enable authentication for prometheus end-point' which is unchecked. There is also a field for 'Collecting metrics period in seconds' set to '120'. Below these fields is a list of checkboxes for collecting build durations: 'Count duration of successful builds', 'Count duration of unstable builds', 'Count duration of failed builds', 'Count duration of not-built builds', 'Count duration of aborted builds', and 'Fetch the last results of builds'. At the bottom are two buttons: a blue 'Save' button and a grey 'Apply' button.

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

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

The screenshot shows the Prometheus Targets page. It lists three healthy targets:

- jenkins (1/1 up)**: Endpoint <http://3.111.170.92:8080/prometheus>, State UP, Labels `instance="3.111.170.92:8080"`, `job="jenkins"`, Last Scrape 15.932s ago, Scrape Duration 50.433ms.
- node_export (1/1 up)**: Endpoint <http://localhost:9100/metrics>, State UP, Labels `instance="localhost:9100"`, `job="node_export"`, Last Scrape 15.610s ago, Scrape Duration 17.760ms.
- prometheus (1/1 up)**: Endpoint <http://localhost:9090/metrics>, State UP, Labels `instance="localhost:9090"`, `job="prometheus"`, Last Scrape 16.405s ago, Scrape Duration 6.085ms.

Let's add Dashboard for a better view in Grafana

Click On Dashboard → + symbol → Import Dashboard

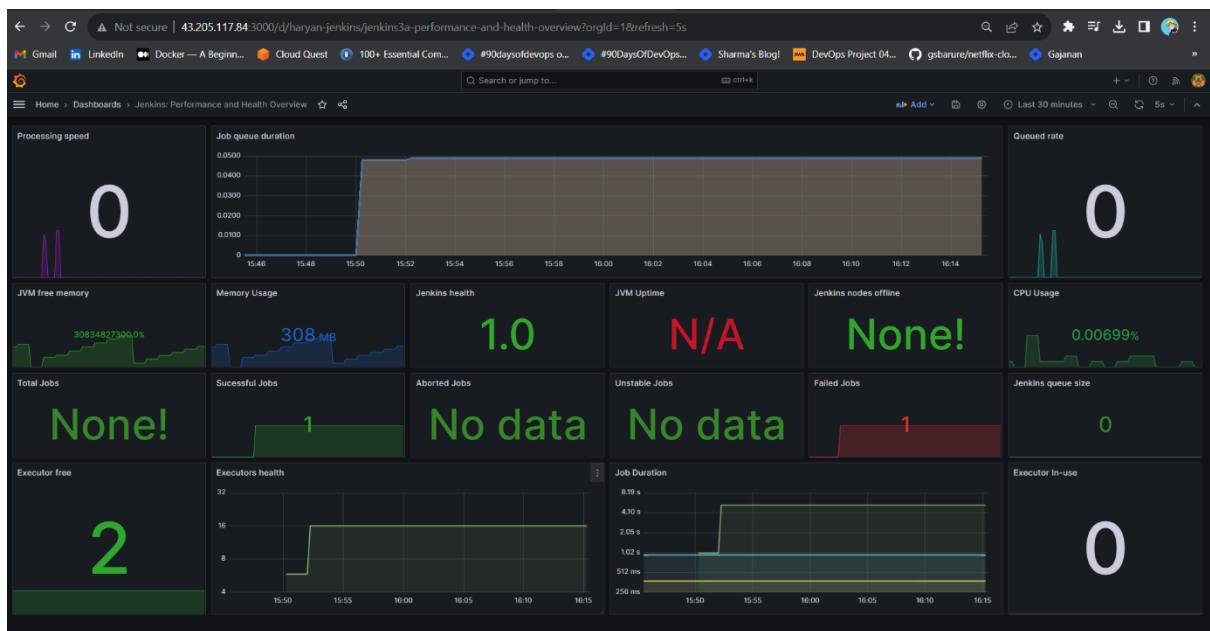
Use Id 9964 and click on load

The screenshot shows the Grafana Import dashboard page. The URL is <http://43.205.117.84:3000/dashboard/import>. The 'Import via grafana.com' input field contains the value `9964`, which is highlighted with a red box. The 'Load' button next to it is also highlighted with a red box.

Select the data source and click on Import

The screenshot shows the 'Import from Grafana.com' interface. It includes fields for 'Name' (Jenkins: Performance and Health Overview), 'Folder' (General), and a 'Unique identifier (UID)' dropdown set to 'Prometheus'. A prominent red box highlights the 'Import' button at the bottom.

Now you will see the Detailed overview of Jenkins



Step 6 — Email Integration With Jenkins and Plugin Setup

Install Email Extension Plugin in Jenkins

The Jenkins Manage Jenkins > Plugins page shows the 'Available plugins' tab. A search bar is set to 'Email Ex'. The 'Email Extension Template' plugin by 'emailext' is listed with a checked checkbox under the 'Install' column. A red box highlights the 'Install' button.

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).

The screenshot shows the 'Security' section of the Google Account settings. On the left, a sidebar lists options like Home, Personal info, Data and privacy, Security (which is selected and highlighted in blue), People and sharing, Payments and subscriptions, and About. The main content area displays 'Recent security activity' with three entries: 'App password removed' (1 Sept - Telangana, India), 'App password created' (31 Aug - Telangana, India), and 'App password removed' (31 Aug - Telangana, India). Below this is a section titled 'How you sign in to Google' with a note to keep information up-to-date. It shows '2-Step Verification' is enabled 'On since 8 Oct 2022'.

2-step verification should be enabled.

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

The screenshot shows the 'App passwords' section of the Google Account settings. The sidebar is identical to the previous screenshot. The main content area has a search bar with 'app' typed in, showing '3 RESULTS'. The first result, 'App passwords' under 'Security', is highlighted with a red box. To the right, there's a green shield icon with a checkmark and a small diagram of a computer screen with a password field. Below the search bar is a link to 'See details'.

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

← App passwords

Generated app password

Your app password for your device

bkec mhur oddp hppw

How to use it

Email

securesally@gmail.com

Password

••••••••••••

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above. Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

DONE

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

← App passwords

Generated app password

Your app password for your device

yndl eppu txvc hxon

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above. Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

Done

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

The screenshot shows the 'E-mail Notification' configuration page in Jenkins. The 'SMTP server' field contains 'smtp.gmail.com'. The 'Default user e-mail suffix' field is empty. Under the 'Advanced' tab, 'Use SMTP Authentication' is checked, and the 'User Name' is 'postbox.aj99@gmail.com' with a matching password. 'Use SSL' is checked, while 'Use TLS' is unchecked. The 'SMTP Port' is set to 465. Other fields include 'Reply-To Address' (empty), 'Charset' (UTF-8), and a 'Test configuration by sending test e-mail' checkbox which is unchecked. At the bottom, there are 'Save' and 'Apply' buttons.

Dashboard > Manage Jenkins > System >

E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix ?

Advanced ▾ Edited

Use SMTP Authentication ?

User Name

postbox.aj99@gmail.com

Password

.....

Use SSL ?

Use TLS

SMTP Port ?

465

Reply-To Address

Charset

UTF-8

Test configuration by sending test e-mail

Dependency-Track

Save Apply

Click on Apply and save.

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

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
postbox.aj99@gmail.com

Treat username as secret ?

Password ?
.....

ID ?
mail

Description ?
mail

Create

This screenshot shows the Jenkins 'Global credentials (unrestricted)' configuration page. A new credential is being created with the type 'Username with password'. The 'Scope' is set to 'Global'. The 'Username' is 'postbox.aj99@gmail.com', and the 'Password' field contains several dots. The 'ID' is 'mail', and the 'Description' is also 'mail'. A 'Create' button is visible at the bottom.

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

Dashboard > Manage Jenkins > System > **Advanced e-mail configuration**

SMTP server: smtp.gmail.com

SMTP Port: 465

Advanced ^ Edited

Credentials: postbox.aj99@gmail.com/***** (mail)

Add

Use SSL
 Use TLS
 Use OAuth 2.0

Dashboard > Manage Jenkins > System > **Advanced Email Properties**

Default user e-mail suffix: ?

Advanced ^ Edited

Default Content Type: HTML (text/html)

List ID: ?

Add 'Precedence: bulk' E-mail Header ?

Dashboard > Manage Jenkins > System > **Additional groovy classpath** ?

Add

Enable Debug Mode ?
 Require Administrator for Template Testing ?
 Enable watching for jobs ?
 Allow sending to unregistered users ?

Default Triggers ^

Default Triggers ?

- Aborted
- Always
- Before Build
- Failure - 1st
- Failure - 2nd
- Failure - Any
- Failure - Still
- Failure - X
- Failure -> Unstable (Test Failures)
- Fixed
- Not Built

Save **Apply**

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)

The screenshot shows the Jenkins management interface for plugin installation. The top navigation bar includes 'Dashboard', 'Manage Jenkins', 'Plugins', 'Search (CTRL+K)', and user information ('admin'). The main content area is titled 'Plugins' and displays the 'Available plugins' tab. A search bar at the top right allows searching for available plugins. Below the search bar, a table lists three plugins:

- Eclipse Temurin installer** 1.5: Described as providing an installer for the JDK tool that downloads the JDK from <https://adoptium.net>. It is marked as up for adoption and was released 11 months ago.
- SonarQube Scanner** 2.15: Described as allowing an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality. It was released 9 months and 19 days ago.
- NodeJS** 1.6.1: Described as executing NodeJS script as a build step. It was released 1 month and 2 days ago.

7B — Configure Java and Nodejs in Global Tool Configuration

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

The screenshot shows the Jenkins 'Manage Jenkins' interface under the 'Tools' section. It displays two configurations:

- JDK Installations:** A configuration for 'jdk17'. It has an 'Install automatically' checkbox checked, and a dropdown menu for 'Install from adoptium.net' set to 'jdk-17.0.8.1+1'. There is also an 'Add installer' button.
- NodeJS:** A configuration for 'node16'. It has an 'Install automatically' checkbox checked, and a dropdown menu for 'Install from nodejs.org' set to 'NodeJS 16.2.0'. It includes a note about forcing 32-bit architecture and a field for 'Global npm packages to install'.

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

The screenshot shows the Sonarqube Administration interface. The top navigation bar has tabs for sonarqube, Projects, Issues, Rules, Quality Profiles, Quality Gates, and Administration. The Administration tab is highlighted with a red box. Below the navigation, there is a sub-navigation for Configuration, Security, Projects, System, and Marketplace. A dropdown menu for 'Users' is open, with other options like General, Groups, Global Permissions, and Permission Templates. The 'Users' option is highlighted with a red box.

click on update Token

The screenshot shows the Sonarqube User profile for 'Administrator admin'. At the top, there are tabs for SCM Accounts, Last connection, Groups, and Tokens. The 'Tokens' tab is highlighted with a red box. Below the tabs, it shows a table with one row for 'sonar-administrators'. To the right of the table is a 'Tokens' button, which is also highlighted with a red box. The 'Update Tokens' button is visible at the bottom right of the tokens section.

Create a token with a name and generate

Tokens of **Administrator**

Generate Tokens

Name	Expires in
Enter Token Name	30 days

New token "Jenkins" has been created. Make sure you copy it now, you won't be able to see it again!

Copy `sq_u_21d162904c1c72cf8b39665f96480185c99dc2f9`

Name	Type	Project	Last use	Created	Expiration
Jenkins	User		Never	September 8, 2023	October 8, 2023

Revoke

copy Token

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

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: **POST THE TOKEN HERE**

ID: **Sonar-token**

Description: **Sonar-token**

Create

You will this page once you click on create

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
 Sonar-token	sonar	Secret text	sonar

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

Dashboard > Manage Jenkins > System >

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

Environment variables Enable injection of SonarQube server configuration as build environment variables

SonarQube installations

List of SonarQube installations

Name	<input type="text" value="sonar-server"/>	X
Server URL	<input type="text" value="http://13.232.17.191:9000"/>	
Server authentication token	<input type="text" value="Sonar-token"/>	
	<input type="button" value="Add"/>	
	<input type="button" value="Save"/>	<input type="button" value="Apply"/>

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.

Dashboard > Manage Jenkins > Tools

SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner		X
Name	<input type="text" value="sonar-scanner"/>	
<input checked="" type="checkbox"/> Install automatically ?		
Install from Maven Central		
Version	<input type="text" value="SonarQube Scanner 5.0.1.3006"/>	
<input type="button" value="Add Installer"/>		
<input type="button" value="Add SonarQube Scanner"/>		
<input type="button" value="Save"/>		<input type="button" value="Apply"/>

In the Sonarqube Dashboard add a quality gate also

Administration-> Configuration->Webhooks

The screenshot shows the SonarQube Administration interface. The 'Administration' tab is selected. Under 'Configuration', the 'Webhooks' option is highlighted with a red box. A search bar at the top says 'Search for projects...'. On the right, there's a 'Create User' button and a user profile for 'Administrator admin'. Below the search bar, there's a table with columns: SCM Accounts, Last connection, Groups, and Tokens. One row is shown for 'Administrator admin'.

Click on Create

The screenshot shows the SonarQube Administration interface under the 'Webhooks' section. A large red box highlights the 'Create' button on the right side of the page.

Add details

#in url section of quality gate

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

The screenshot shows the 'Create Webhook' dialog box. It has fields for 'Name' (set to 'jenkins'), 'URL' (set to 'http://43.204.36.242:8090/sonarqube-webhook/'), and 'Secret' (empty). A note at the bottom says 'Embedded database should be used'. A 'Create' button is highlighted with a red box. The background shows the SonarQube Administration interface.

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 {

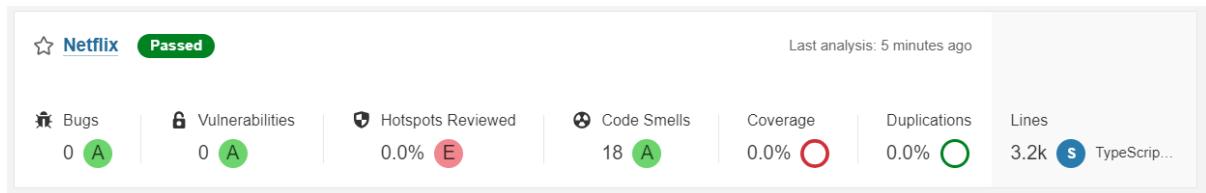
        emailext attachLog: true,
        subject: "${currentBuild.result}",
        body: "Project: ${env.JOB_NAME}<br/>" +
            "Build Number: ${env.BUILD_NUMBER}<br/>" +
            "URL: ${env.BUILD_URL}<br/>",
        to: 'postbox.aj99@gmail.com',
        attachmentsPattern: 'trivyfs.txt,trivyimage.txt'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

```

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

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies
5s	379ms	1s	16s	520ms	1min 12s
169ms	294ms	1s	28s	926ms (paused for 741ms)	2min 24s

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,

Declarative: Tool Install	clean workspace	Checkout from Git	Sonarqube Analysis	quality gate	Install Dependencies	OWASP FS SCAN	TRIVY FS SCAN
5s	379ms	1s	16s	520ms	1min 12s	1min 45s	13s
169ms	294ms	1s	28s	926ms (paused for 741ms)	2min 24s	3min 31s	27s

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

Dependency-Check Results

SEVERITY DISTRIBUTION			
13	39	13	
File Name	Vulnerability	Severity	Weakness
+ ansi-html:0.0.7	[NVD] CVE-2021-23424	High	NVD-CWE-noinfo
+ ansi-regex:4.1.0	[NVD] CVE-2021-3807	High	CWE-1333
+ async:2.6.3	[NVD] CVE-2021-43138	High	CWE-1321
+ browserslist:4.14.2	[NVD] CVE-2021-23364	Medium	CWE-1333
+ css-what:3.4.2	[OSSINDEX] CVE-2022-21222	High	CWE-1333
+ decode-uri-component:0.2.0	[NVD] CVE-2022-38778	Medium	CWE-20
+ decode-uri-component:0.2.0	[NVD] CVE-2022-38900	High	CWE-20
+ ejss:2.7.4	[OSSINDEX] CVE-2022-29078	High	CWE-94
+ eventsource:1.1.0	[NVD] CVE-2022-1650	Critical	CWE-212
+ express:4.17.1	[OSSINDEX] CVE-2022-24999	High	CWE-1321

Step 10 — Docker Image Build and Push

We need to install the Docker tool in our system, Goto Dashboard → Manage Jenkins → 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

The screenshot shows the Jenkins 'Manage Jenkins' section under 'Plugins'. A search bar at the top contains the text 'docker'. Below it, a list of four Docker-related plugins is displayed:

- Docker 1.5** (Released 3 days 15 hr ago) - This plugin integrates Jenkins with Docker. It is up for adoption.
- Docker Commons** (439.va_3cb_0a_6a_fb_29) (Released 1 mo 29 days ago) - Provides the common shared functionality for various Docker-related plugins.
- Docker Pipeline** (572.v950f58993843) (Released 27 days ago) - Build and use Docker containers from pipelines. It is up for adoption.
- Docker API** (3.3.1-79.v20b_53427e041) (Released 3 mo 4 days ago) - This plugin provides docker-java API for other plugins.

Each plugin entry includes its name, version, release date, and a brief description. The 'Docker 1.5' entry is highlighted with a yellow background.

Now, goto Dashboard → Manage Jenkins → Tools →

The screenshot shows the Jenkins 'Manage Jenkins' section under 'Tools'. Under 'Docker installations', there is a configuration for a new Docker instance:

- Name:** docker
- Install automatically?** (checked)
- Download from docker.com**
- Docker version:** latest
- Add Installer** button

Add DockerHub Username and Password under Global Credentials

Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Kind

Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
sevenajay

Treat username as secret ?

Password ?
.....

ID ?
docker

Description ?
docker

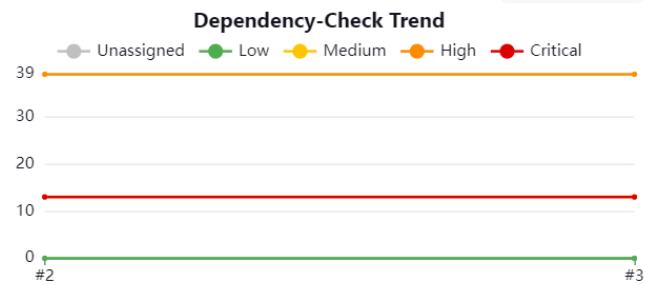
Create

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

 **sevenajay / netflix**

Description
This repository does not have a description 

Docker commands
To push a new tag to this repository:

```
docker push sevenajay/netflix:tagname
```

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

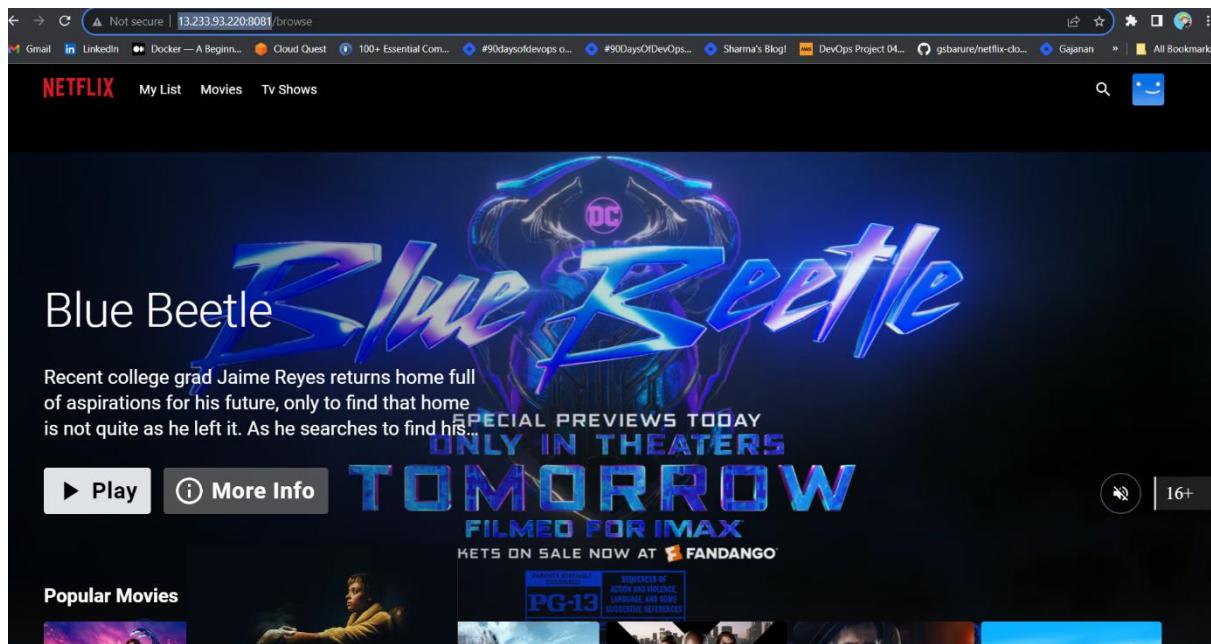
```
stage('Deploy to container'){
    steps{
        sh 'docker run -d --name netflix -p 8081:80 sevenajay/netflix:latest'
    }
}
```

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
```

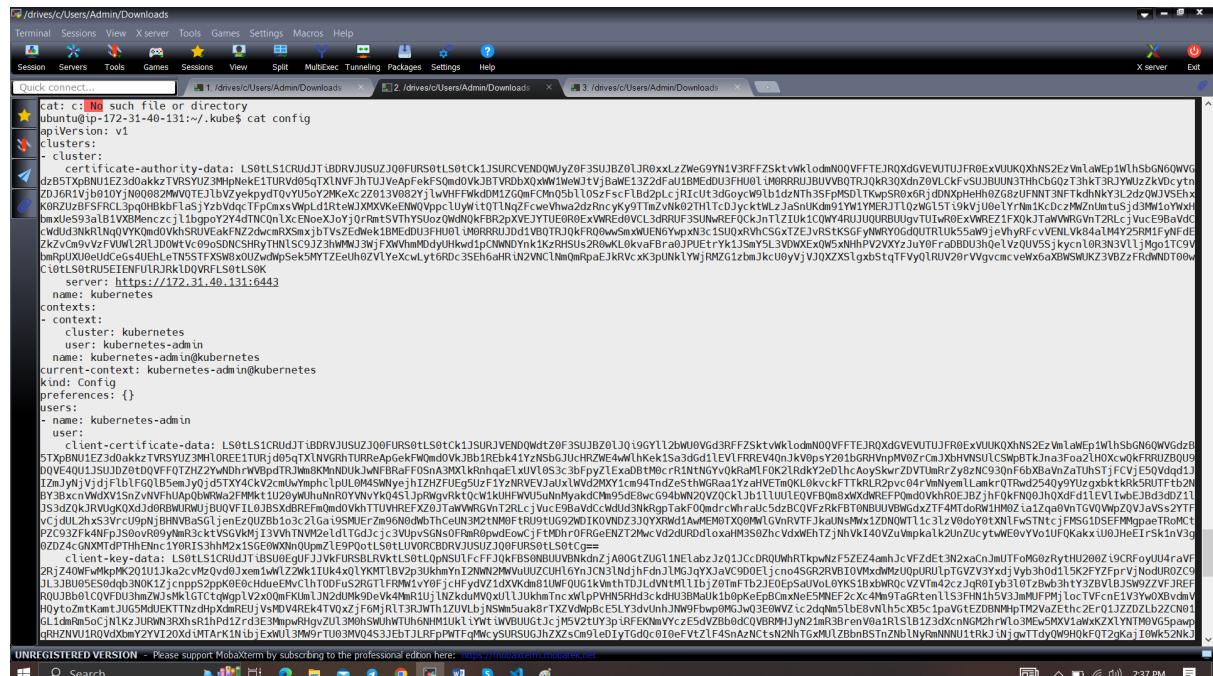
```
mkdir -p $HOME/.kube
```

```
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



```
cat: config: No such file or directory
ubuntu@ip-172-31-40-131:~/.kube$ cat config
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUJ0OFURS0tLS0tCk1JSURJVEND0wdtZ0F3JSUJBz01J0i9GYll2bwU0gD3RFZSktrwklodnN00VFTEJR0XgDVETUJFJ0ExVUJK0XhNS2EzWmlwEp1WlhbGN60WVGdzb5KtRN01Ez3doakkzTbRSYUZ3MhpNeKE1TURRd0pqTXNVRHTRURApgekFW0md0VkjBjB1REpk41YzNsbgJuChRzWE4wLlhkek1sa3dg1lEvLFRRER40nJkv0psY201bGRHnvpMw0zCrJkbHvnsULCSwpBTkJna3f0a2lHOxcm0kFRRIuzBQJ9ZDj6W1jbj01OYiN60082MwVOTE1blVzhexkpdyt0vUy5o2MKcXe27013Vb82Yj1wVHFwdkM0m0FcM051l05FscF1Bd2plc_jjRtCU3dGoy9W1b1dzNt03FpmSd1TkuPsr0xRjzdnXpHeHh0zGz8UFNNT3NF1TkdhKy3L2dr0WV1ShhxK08RzUzBFSFRC13p0h0Bkbf1asjzbVdqcfPcmxsWp1z2NcYy9tMzNk02ThtCdjcyktMzl2jaSnukdm91Yw1LYMErJt02wG15t19Kvju0e1YrNu1c0zczM2wzNlntus; d3M1o1YwXhbmJx93aLB1VkBmencjz1lbgpy2Y24tDNCtEhNEoXj0yRnetsVtHySu0z0wN0kfBR2pVExJYtUE0R0ExwMREdgVCL3drRUF33UMwREF0CKJnTz1Uk1C0W4RnU0JU0RBU0qgvTu1t08ExWREZ1FX0k1Jx0vWmRGvnt2RlcjVucE9BaVdcwJUd3NkR1NgY0Km0dVkrhVnRvEakfNZ2cmR0xSmjxTvsZEd0tewJmXKwehNw0z2d2ncky9tMzNk02ThtCdjcyktMzl2jaSnukdm91Yw1LYMErJt02wG15t19Kvju0e1YrNu1c0zczM2wzNlntus; d3M1o1YwXhbmJx93aLB1VkBmencjz1lbgpy2Y24tDNCtEhNEoXj0yRnetsVtHySu0z0wN0kfBR2pVExJYtUE0
```

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

The screenshot shows the Jenkins 'Plugins' page. On the left, there is a sidebar with links: 'Updates', 'Available plugins' (which is selected), 'Installed plugins', 'Advanced settings', and 'Download progress'. The main area has a search bar with 'Kuber' typed in. Below the search bar, there is a 'Install' button and a refresh icon. The results table has columns: 'Install', 'Name', and 'Released'. There are four entries:

Install	Name	Released
<input checked="" type="checkbox"/>	Kubernetes Credentials 0.11 kubernetes credentials	9 days 16 hr ago
<input checked="" type="checkbox"/>	Kubernetes Client API 6.8.1-224.vd388fca_4db_3b kubernetes Library plugins (for use by other plugins)	9 days 17 hr ago
<input checked="" type="checkbox"/>	Kubernetes 4029.v5712230ccb_f8 Cloud Providers Cluster Management kubernetes Agent Management	9 days 15 hr ago
<input checked="" type="checkbox"/>	Kubernetes CLI 1.12.1 kubernetes	8 days 22 hr ago

Common classes for Kubernetes credentials
Kubernetes Client API plugin for use by other Jenkins plugins.
This plugin integrates Jenkins with Kubernetes
Configure kubectl for Kubernetes

goto manage Jenkins → manage credentials → Click on Jenkins global → add credentials

The screenshot shows the 'New credentials' page. The 'Kind' dropdown is set to 'Secret file'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc)'. The 'File' section shows a 'Choose File' button with 'Secret File.txt' selected. The 'ID' field contains 'k8s'. The 'Description' field contains 'k8s'. At the bottom, there is a 'Create' button.

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
```

```
ubuntu@ip-172-31-38-156:~$ 
ubuntu@ip-172-31-38-156:~$ wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz
--2023-10-06 09:03:19-- https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.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/9524057/5500b569-5c34-471e-8598-c05c0733bb7f?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAWNUYAX4CSEH53A/2F202310062fus-east-1%2F53%2Faw4_requestX-Amz-Date=20231006T09031978X-Amz-Expires=3085X-Amz-Signature=e1c47ad71d3d29d7e360b2c68bc38d50f1433e128c4a90a73460781b5e6e34035X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=9524057&response-content-disposition=attachment%3B%20filename%3Dnode_exporter-1.6.1.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream [following]
--2023-10-06 09:03:19-- https://objects.githubusercontent.com/github-production-release-asset-2e65be/9524057/5500b569-5c34-471e-8598-c05c0733bb7f?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAWNUYAX4CSEH53A/2F202310062fus-east-1%2F53%2Faw4_requestX-Amz-Date=20231006T09031978X-Amz-Expires=3085X-Amz-Signature=e1c47ad71d3d29d7e360b2c68bc38d50f1433e128c4a90a73460781b5e6e34035X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=9524057&response-content-disposition=attachment%3B%20filename%3Dnode_exporter-1.6.1.linux-amd64.tar.gz&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.109.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10368103 (9.9M) [application/octet-stream]
Saving to: 'node_exporter-1.6.1.linux-amd64.tar.gz'

node_exporter-1.6.1.linux-amd64.tar.gz    100%[=====] 9.89M --.-KB/s   in 0.07s

2023-10-06 09:03:20 (135 MB/s) - 'node_exporter-1.6.1.linux-amd64.tar.gz' saved [10368103/10368103]

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

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:~$ 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:~$ sudo systemctl start node_exporter
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:~$ 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=tme
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=tmem
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:~$ 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:~$ 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:~$ curl -X POST http://localhost:9090/-/reload
ubuntu@ip-172-31-38-156:~$
```

Check the targets section

<http://<ip>:9090/targets>

node_export (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9100/metrics	UP	instance="localhost:9100" job="node_export"	10.123s ago	17.500ms	

node_export_master_k8s (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://43.204.100.15:9100/metrics	UP	instance="43.204.100.15:9100" job="node_export master k8s"	8.758s ago	20.950ms	

node_export_workerk8s (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://65.0.45.118:9100/metrics	UP	instance="65.0.45.118:9100" job="node_export_workerk8s"	11.906s ago	24.987ms	

prometheus (1/1 up)					
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	7.629s ago	5.791ms	

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 kubernetes
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

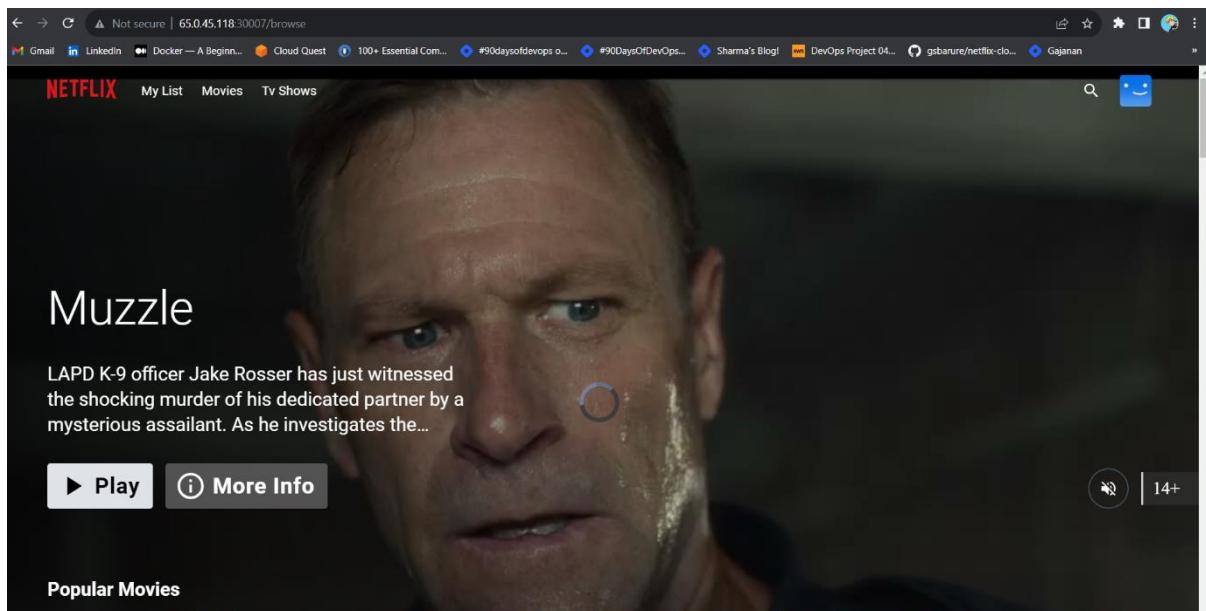
NAME                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/petshop 1/1      1           1           43s

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>

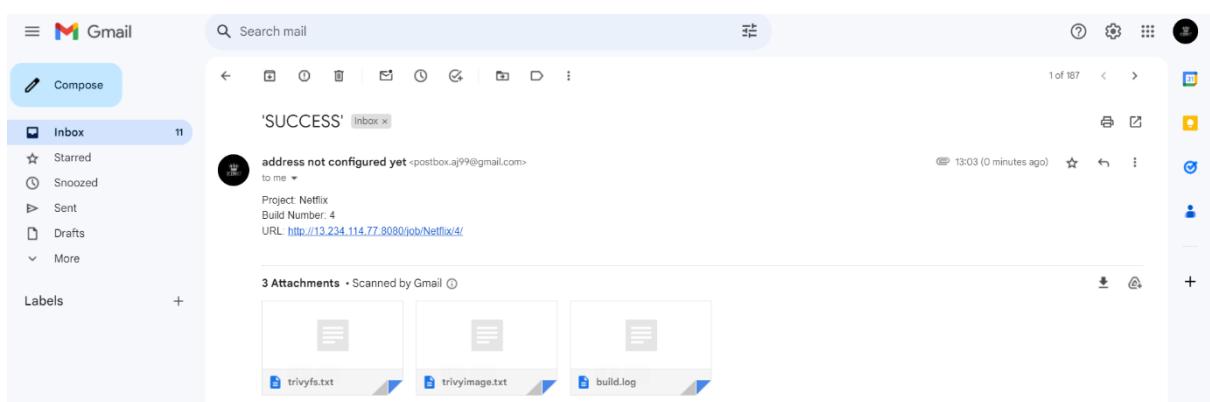
output:



Monitoring



Mail



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'  
}  
}
```