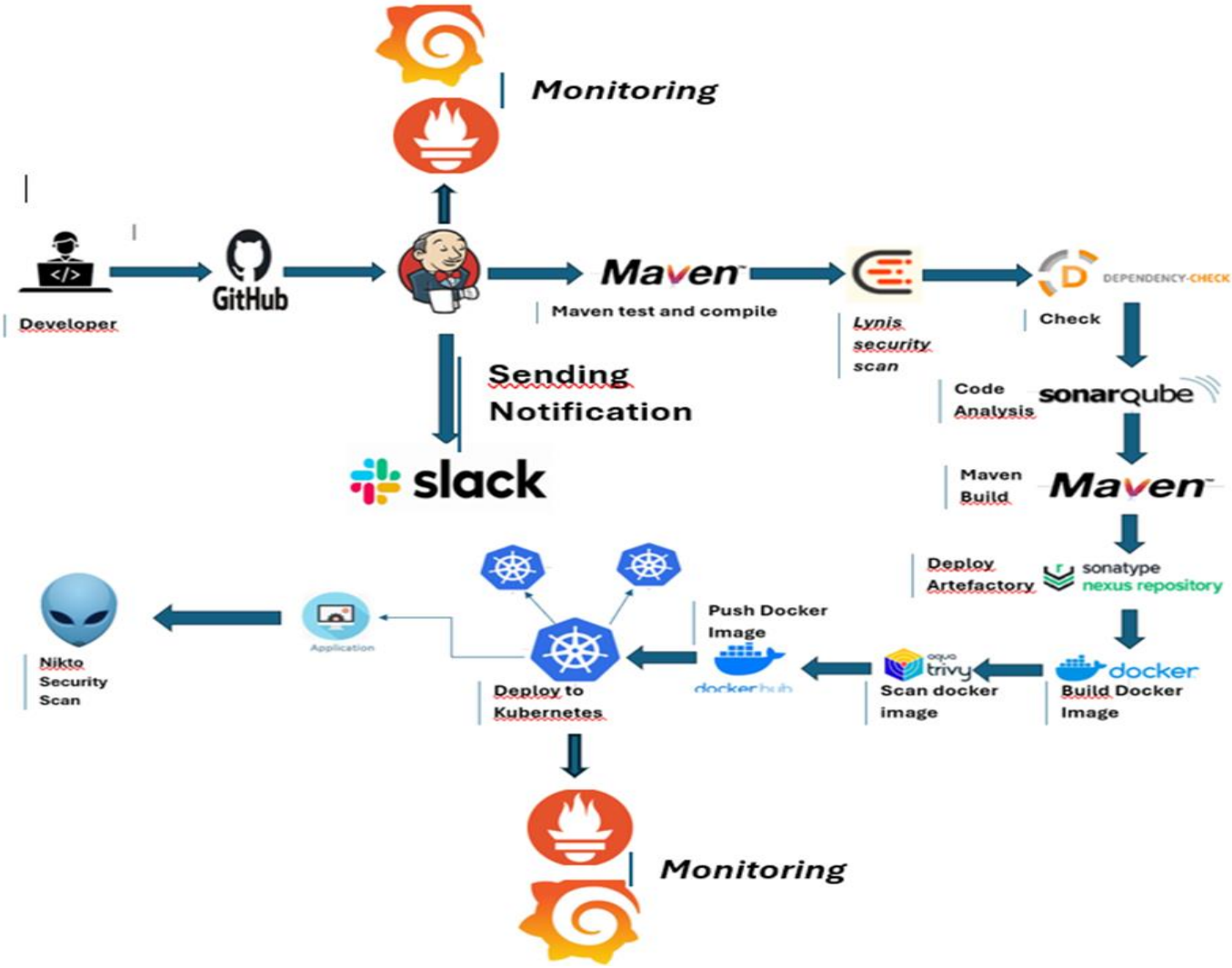
COMPLETE CI/CD PROJECT DEVSECOPS



Project GitHub Repo- :
https://github.com/eyaboubaker/Devscops.git

We need four servers for our today's Project

Jenkins Server- On this Server, Jenkins will be installed with some other tools such as sonarqube,sonatype nexus,Lynis,trivy, ,Nikto, Prometheus, Node Exporter, and Grafana.

Kubernetes Master Server- This Server will be used as the Kubernetes Master Cluster Node which will deploy the applications on worker nodes.

Kubernetes Worker1 Server- This Server will be used as the Kubernetes Worker1 Node on which the application will be deployed by the master node.

Kubernetes Worker2 Server- This Server will be used as the Kubernetes Worker2 Node on which the application will be deployed by the master node.

Let's create the following instances.

Jenkins Server

Ubuntu OS 22.04 version.

8GB ram

30GB Disk

Kubernetes Master & Worker Node

We have to create two Kubernetes Nodes which need at least 2 CPUs. Log in to the Jenkins Server(Devscops)

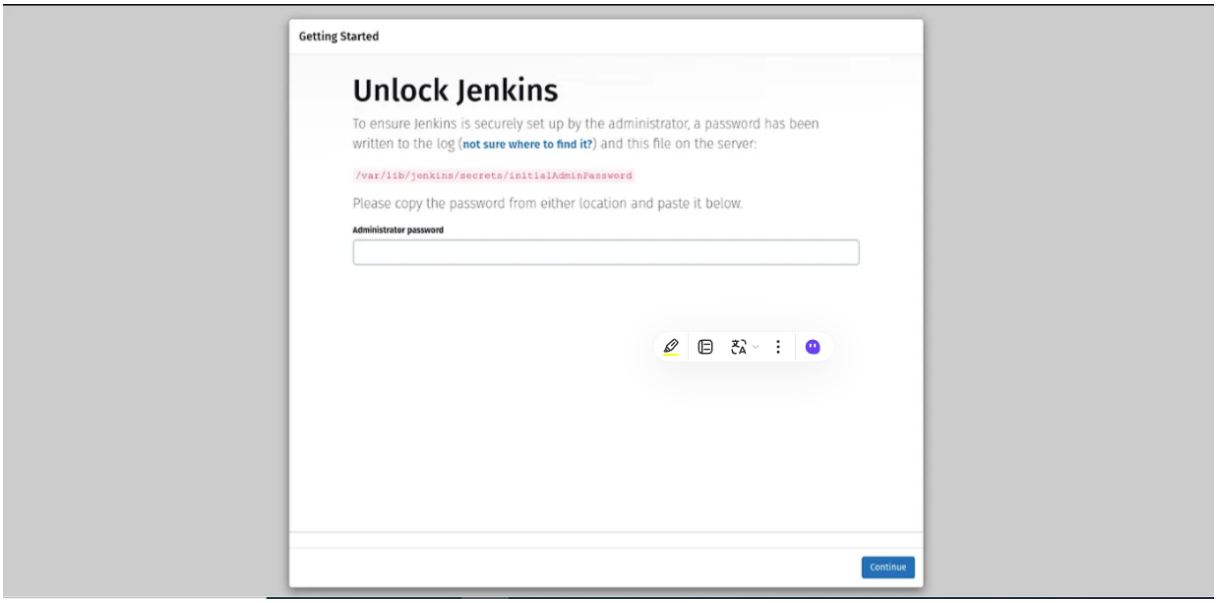
Install jenkins

```
# Installing Java

sudo apt update -y

sudo apt install fontconfig openjdk-17-jre -y

java -version

# Installing Jenkins

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

   https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

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 update

sudo apt install jenkins -y

sudo systemctl enable jenkins

sudo ufw allow 8080/tcp

sudo ufw reload
```

Check the status of the Jenkins server

```
devscops@devscops-virtual-machine:~$ sudo systemctl status jenkins.service
[sudo] password for devscops:
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-05-31 12:54:11 CET; 6 days ago
   Main PID: 5436 (java)
      Tasks: 62 (limit: 7117)
     Memory: 1.1G
        CPU: 1h 11min 46.560s
     CGroup: /system.slice/jenkins.service
             └─5436 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.actor.AbstractActor.aroundReceive(AbstractActor.scala:227)
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.actor.ActorCell.receiveMessage(ActorCell.scala:612)
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.actor.ActorCell.invoke(ActorCell.scala:581)
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.dispatch.Mailbox.processMailbox(Mailbox.scala:268)
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.dispatch.Mailbox.run(Mailbox.scala:229)
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.dispatch.Mailbox.exec(Mailbox.scala:241)
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.dispatch.forkjoin.ForkJoinTask.doExec(ForkJoinTask.java:260)
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.dispatch.forkjoin.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.dispatch.forkjoin.ForkJoinPool.runWorker(ForkJoinPool.java:1979)
23:22:48 06 جوان devscops-virtual-machine jenkins[5436]:         at akka.dispatch.forkjoin.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java
lines 1-20/20 (END)
```

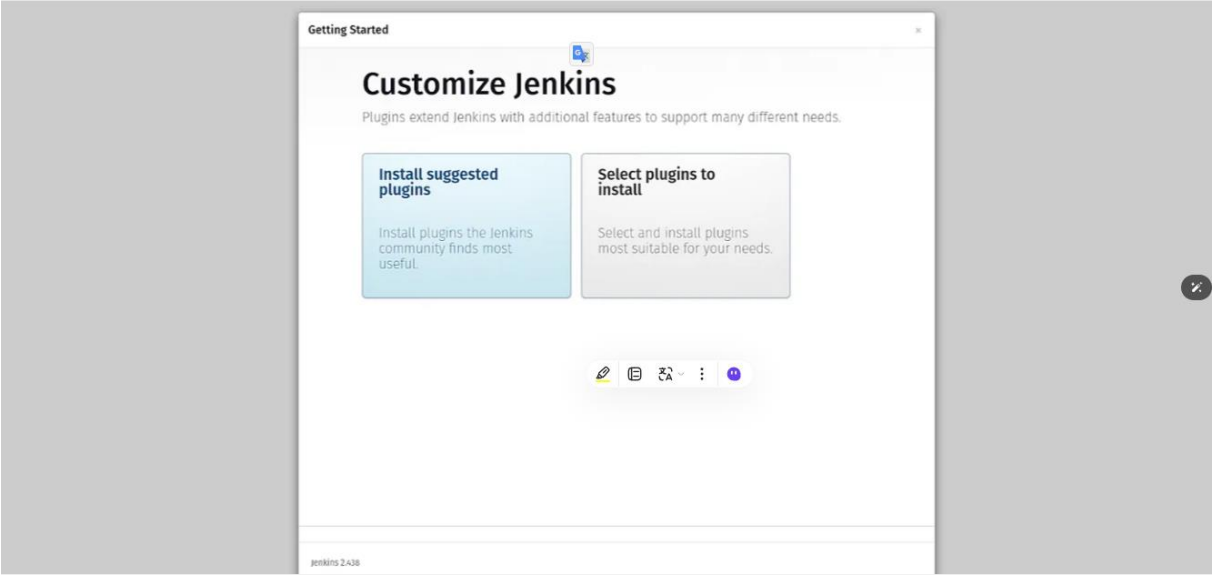
Copy your Jenkins Server Public IP and paste it into your favorite browser with port number 8080.



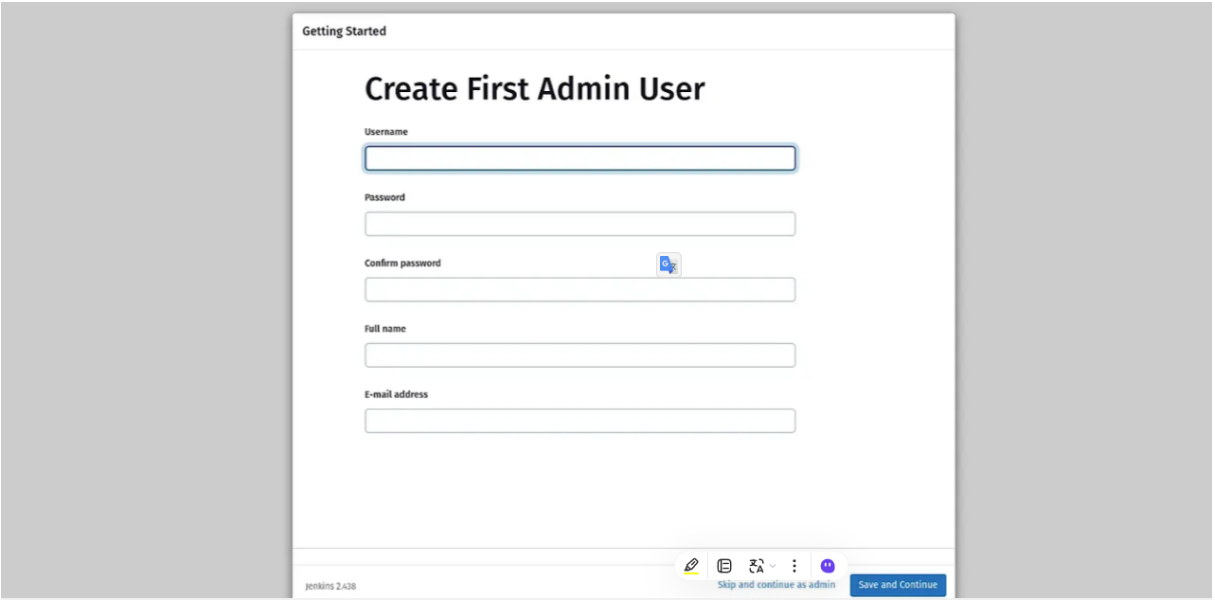
Run the command on your Jenkins server

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

Click on the **Install suggested plugins**



Click on the **Skip and continue as admin**



Install Docker and configure on the **Jenkins Server**

```
1- sudo apt install apt-transport-https ca-certificates curl software-properties-common


2- curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg


3- echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
4- sudo apt update
5-  apt-cache policy docker-ce
6-  sudo apt install docker-ce
7-  sudo systemctl status docker
8- sudo usermod -aG docker jenkins
9-  sudo chmod 666 /var/run/docker.sock
```

Install git

```
1- sudo apt install git-all
2- git –version
```

Install Trivy

```
1- sudo apt-get install wget apt-transport-https gnupg lsb-release

2- wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -

3- echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | sudo tee -a /etc/apt/sources.list.d/trivy.list

4- sudo apt-get update

5- sudo apt-get install trivy

6- sudo apt install mlocate

7- sudo apt install colorized-logs
```

Install Lynis

```
1- echo "deb https://packages.cisofy.com/community/lynis/deb/ stable main" | sudo tee /etc/apt/sources.list.d/cisofy-lynis.list
```

```
2- wget -O - https://packages.cisofy.com/keys/cisofy-software-public.key |
sudo apt-key add -

3- sudo apt update

4- sudo apt install lynis

5- lynis show version

6- sudo lynis audit system
```

Install Nikto

```
1- sudo apt install nikto

2- nikto -h example.com
```

Install Sonarqube on your **Jenkins Server**

```
1- sudo sysctl -w vm.max_map_count=262144

2-  sudo sysctl -w fs.file-max=65536

3-ulimit -n 65536

4-ulimit -u 4096

5- sudo apt-get update

6-  sudo apt-get install openjdk-17-jdk -y

7-sudo apt-get install openjdk-17-jre -y

8- sudo apt install dirmngr ca-certificates software-properties-common
apt-transport-https lsb-release curl -y

9-curl -fSsL https://www.postgresql.org/media/keys/ACCC4CF8.asc | gpg --
dearmor | sudo tee /usr/share/keyrings/postgresql.gpg > /dev/null

10-sudo apt install postgresql-client-15 postgresql-15

11-systemctl status postgresql

12-sudo systemctl enable postgresql

13-sudo passwd postgres

14-su - postgres
```

15-psql

16-ALTER USER sonar WITH ENCRYPTED password 'sonar';

17-CREATE DATABASE sonarqube OWNER sonar;

18-grant all privileges on DATABASE sonarqube to sonar;

19-\q

20-sudo wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-9.9.4.87374.zip

21-sudo unzip sonarqube-9.9.4.87374.zip -d /opt

22-sudo mv /opt/sonarqube-9.9.4.87374 /opt/sonarqube

23-cd /opt/sonarqube

24-sudo groupadd sonar

25-sudo useradd -c "user to run SonarQube" -d /opt/sonarqube -g sonar sonar

26-sudo chown sonar:sonar /opt/sonarqube -R

27-sudo nano /opt/sonarqube/conf/sonar.properties

sonar.jdbc.username=sonar

sonar.jdbc.password=sonar

sonar.jdbc.url=jdbc:postgresql://localhost:5432/sonarqube

28- sudo nano /opt/sonarqube/bin/linux-x86-64/sonar.sh

RUN_AS_USER=sonar

29-sudo nano /etc/systemd/system/sonar.service

```
[Unit]

Description=SonarQube service

After=syslog.target network.target

[Service]

Type=forking
```

```
ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop
User=ddsonar
Group=ddsonar
Restart=always
LimitNOFILE=65536
LimitNPROC=4096
[Install]
WantedBy=multi-user.target
```

```
30- sudo systemctl enable sonar
31- sudo systemctl start sonar
32- sudo systemctl status sonar
```

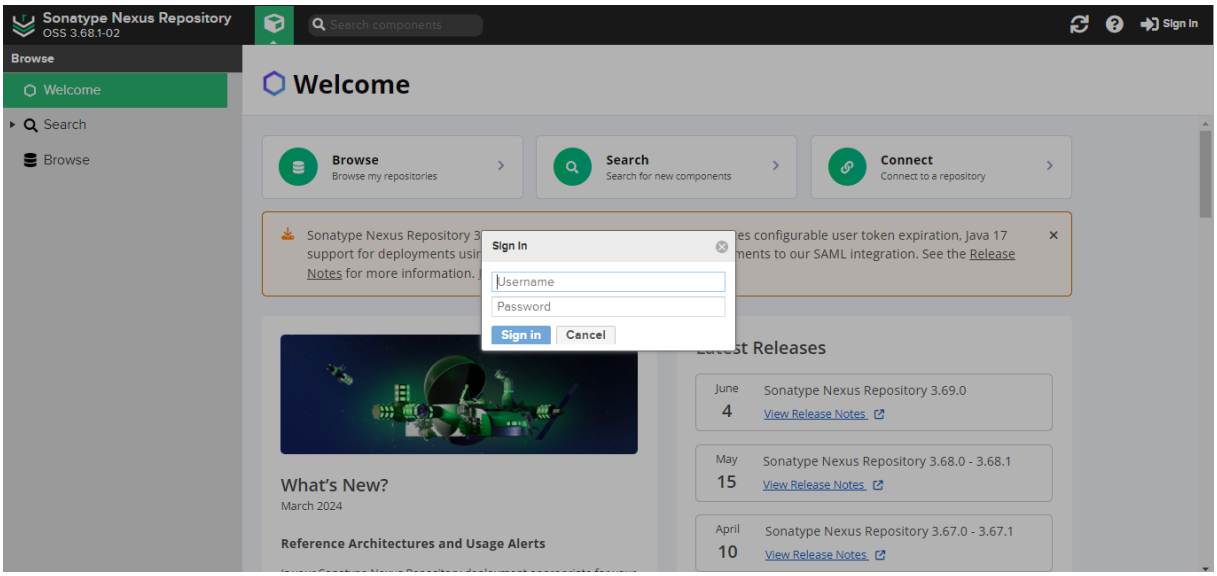
Now, copy your Public IP of Jenkins Server and add 9000 Port on your browser. The username and password will be admi

Install Nexus on your **Jenkins Server**

```
1- sudo apt-get update
2-sudo apt install openjdk-8-jre-headless
```

```
3-cd /opt

4-sudo wget https://download.sonatype.com/nexus/3/latest-unix.tar.gz

5-sudo tar -zxvf latest-unix.tar.gz

6-sudo mv /opt/nexus-3.30.1-01 /opt/nexus

7-sudo adduser nexus

8-sudo visudo

9-nexus ALL=(ALL) NOPASSWD: ALL

10-sudo chown -R nexus:nexus /opt/nexus

11-sudo chown -R nexus:nexus /opt/sonatype-work

12-sudo nano /opt/nexus/bin/nexus.rc

13-run_as_user="nexus"

14-sudo nano /etc/systemd/system/nexus.service

[Unit]

Description=nexus service

After=network.target

[Service]

Type=forking

LimitNOFILE=65536

ExecStart=/opt/nexus/bin/nexus start

ExecStop=/opt/nexus/bin/nexus stop

User=nexus

Restart=on-abort

[Install]

WantedBy=multi-user.target

15-sudo systemctl start nexus
```

```
16-sudo systemctl enable nexus

17-sudo systemctl status nexus

18- sudo ufw allow 8081/tcp

19-cat /opt/sonatype-work/nexus3/admin.password
```



Install and Configure the Prometheus, Node Exporter, and Grafana

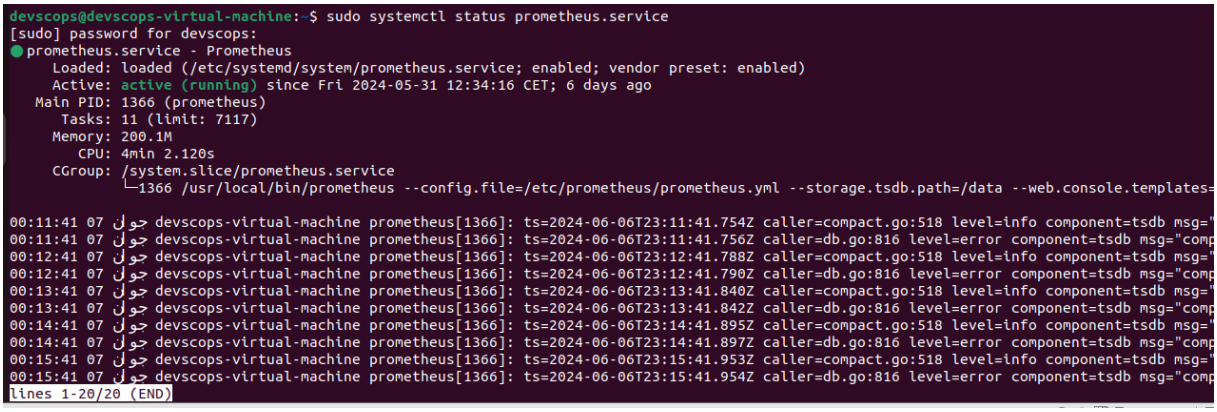
- Prometheus

```
1- sudo useradd \

    --system \

    --no-create-home \

    --shell /bin/false prometheus


2- sudo wget https://github.com/prometheus/prometheus/releases/download/v2.32.1/prometheus-2.32.1.linux-amd64.tar.gz

3- sudo tar -xvf prometheus-2.32.1.linux-amd64.tar.gz

4- sudo mkdir -p /data /etc/prometheus

5- cd prometheus-2.32.1.linux-amd64
```

```
6- sudo mv prometheus promtool /usr/local/bin/

7- sudo mv consoles/ console_libraries/ /etc/prometheus/

8- sudo mv prometheus.yml /etc/prometheus/prometheus.yml

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

10- cd

      sudo rm -rf prometheus*

11- prometheus --version

12- prometheus --help

13- sudo nano /etc/systemd/system/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
```

```
14- sudo systemctl enable prometheus

15- sudo systemctl start prometheus

16- sudo systemctl status prometheus
```



Once the Prometheus service is up and running then, copy the public IP of your **machine** and paste it into your favorite browser with a 9090 port.

Now, we have to install a node exporter to visualize the machine or hardware level data such as CPU, RAM, etc on our Grafana dashboard.

Node Exporter

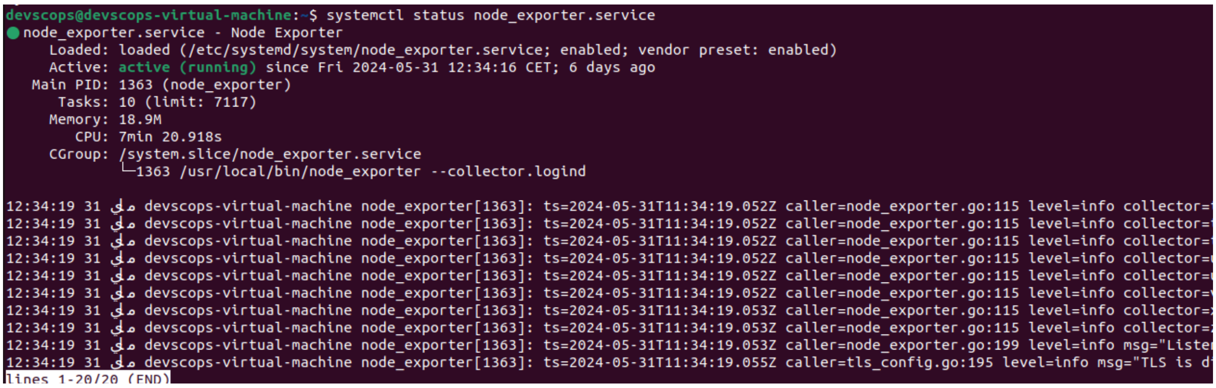
```
1- sudo useradd \

    --system \

    --no-create-home \

    --shell /bin/false node_exporter
```

```
2- sudo wget
https://github.com/prometheus/node_exporter/releases/download/v1.3.1/node_
exporter-1.3.1.linux-amd64.tar.gz

3- sudo tar -xvf node_exporter-1.3.1.linux-amd64.tar.gz

4- sudo mv \

  node_exporter-1.3.1.linux-amd64/node_exporter \

  /usr/local/bin/

5- rm -rf node_exporter*

6- node_exporter --version

7- sudo nano /etc/systemd/system/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
```

```
8- sudo systemctl enable node_exporter
```

```
9- sudo systemctl start node_exporter
```

```
   10-      sudo systemctl status node_exporter
```

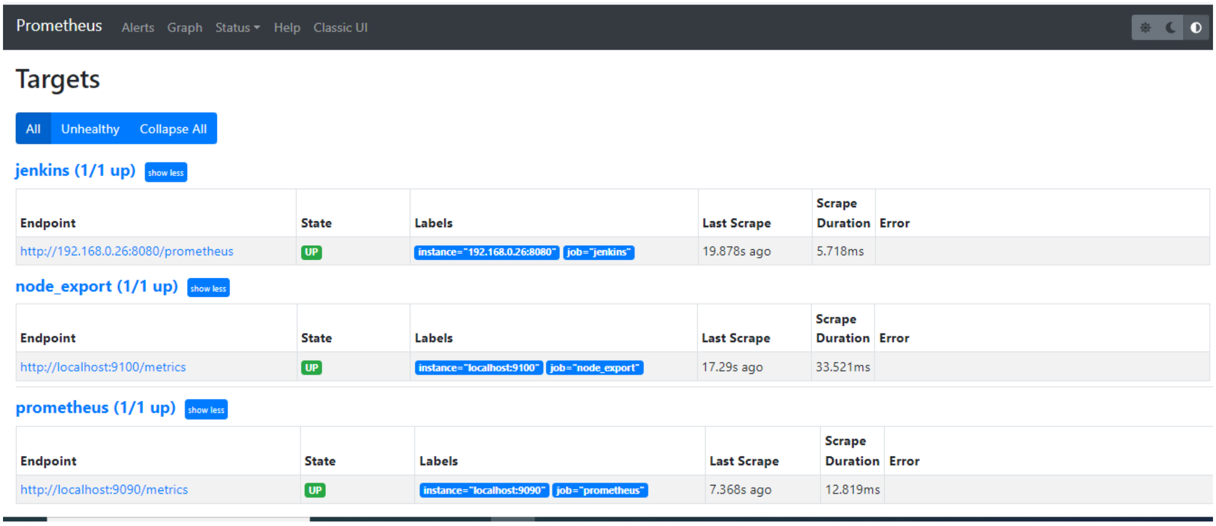


```
devscops@devscops-virtual-machine:~$ systemctl status node_exporter.service
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-05-31 12:34:16 CET; 6 days ago
   Main PID: 1363 (node_exporter)
      Tasks: 10 (limit: 7117)
     Memory: 18.9M
        CPU: 7min 20.918s
     CGroup: /system.slice/node_exporter.service
             └─1363 /usr/local/bin/node_exporter --collector.logind

12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.052Z caller=node_exporter.go:115 level=info collector=
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.052Z caller=node_exporter.go:115 level=info collector=
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.052Z caller=node_exporter.go:115 level=info collector=
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.052Z caller=node_exporter.go:115 level=info collector=
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.052Z caller=node_exporter.go:115 level=info collector=
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.052Z caller=node_exporter.go:115 level=info collector=
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.053Z caller=node_exporter.go:115 level=info collector=
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.053Z caller=node_exporter.go:115 level=info collector=
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.053Z caller=node_exporter.go:199 level=info msg="Liste
12:34:19 31 مي devscops-virtual-machine node_exporter[1363]: ts=2024-05-31T11:34:19.055Z caller=tls_config.go:195 level=info msg="TLS is d
lines 1-20/20 (END)
```

```
11- sudo nano /etc/prometheus/prometheus.yml
```

```
      - job_name: "node_export"

    static_configs:

        - targets: ["localhost:9100"]
```

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



Targets

jenkins (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.0.26:8080/prometheus	UP	instance="192.168.0.26:8080" job="jenkins"	19.878s ago	5.718ms	

node_export (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9100/metrics	UP	instance="localhost:9100" job="node_export"	17.29s ago	33.521ms	

prometheus (1/1 up)

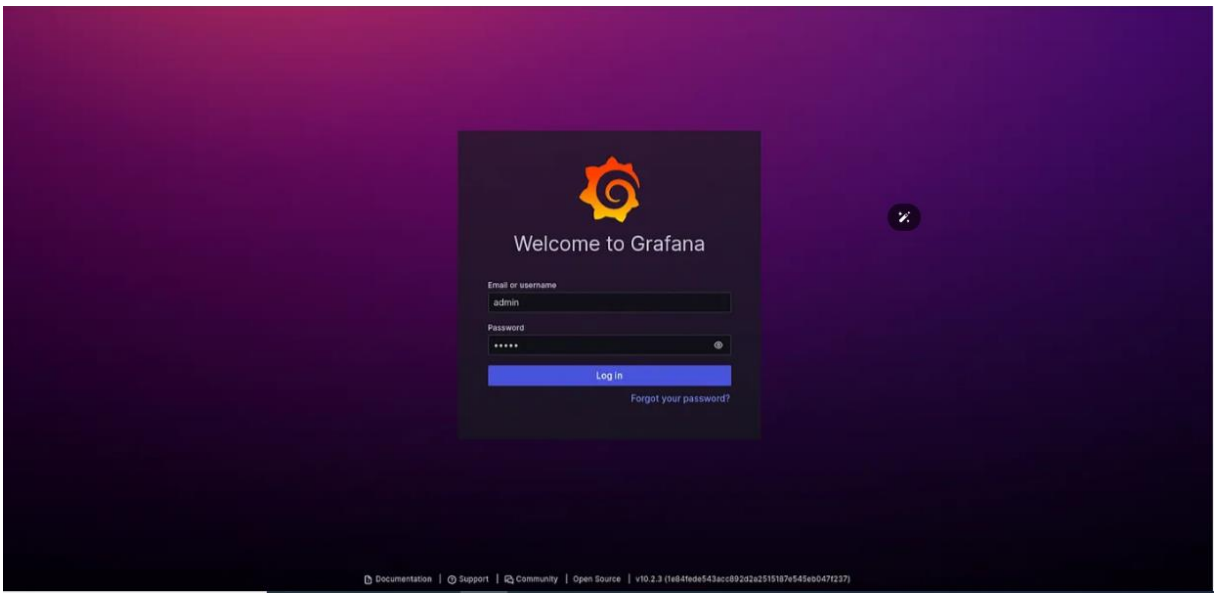
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	7.368s ago	12.819ms	

Now, install the Grafana tool to visualize all the data that is coming with the help of Prometheus.

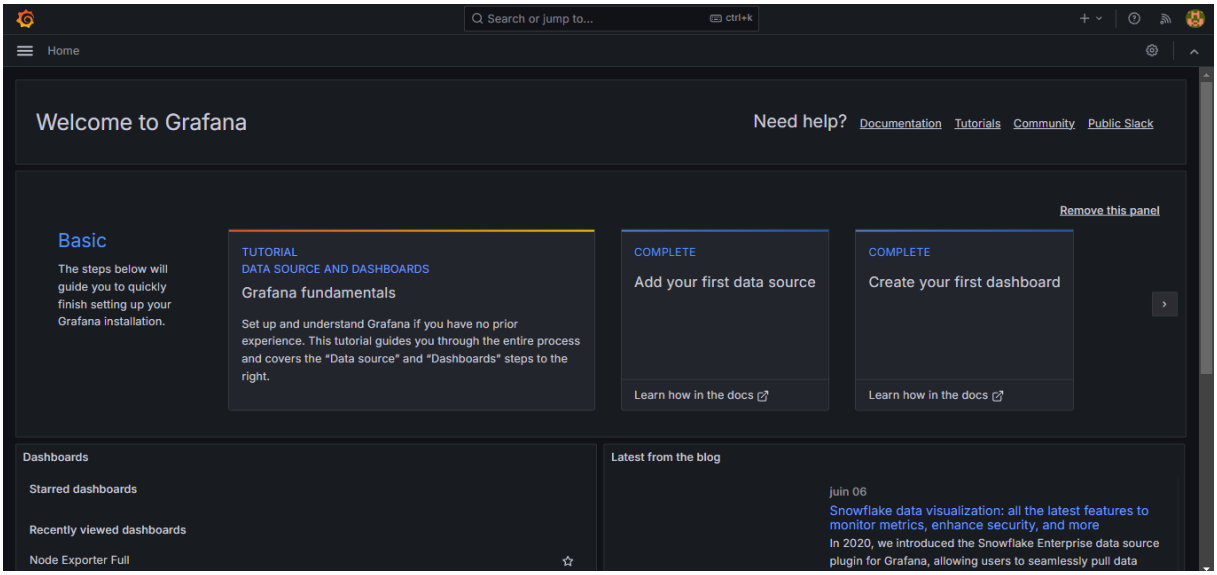
- Grafana

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

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

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

4- sudo apt-get update

5- sudo apt-get -y install grafana

6- sudo systemctl enable grafana-server

7- sudo systemctl start grafana-server

8- sudo systemctl status grafana-server
```

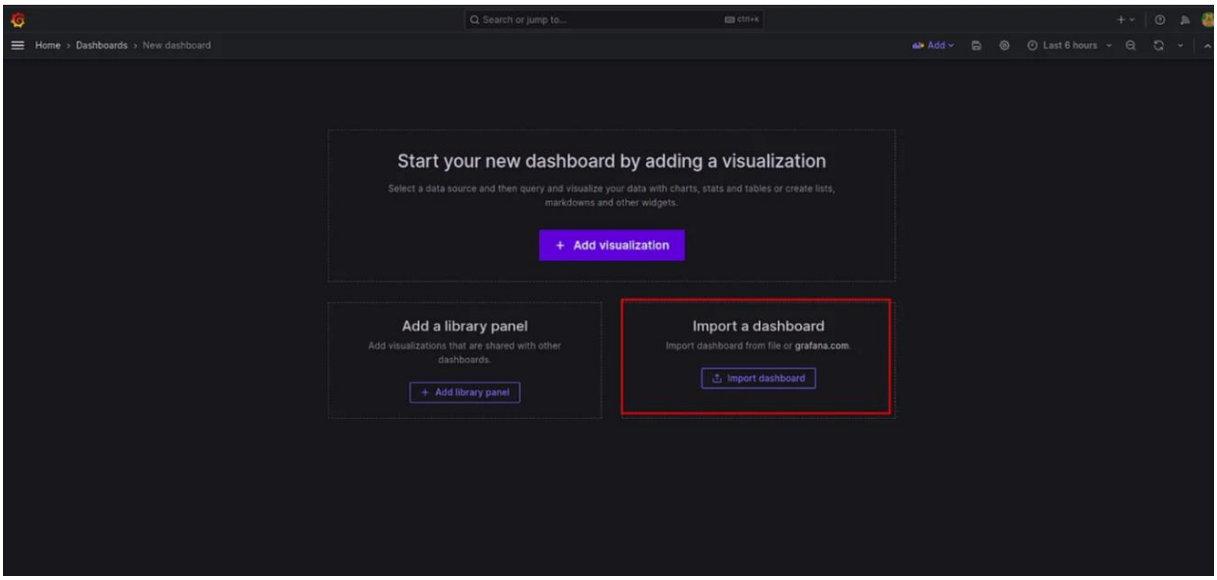
To access the Grafana dashboard, copy the public IP address of the **machine** and paste it into your favorite browser with port 3000



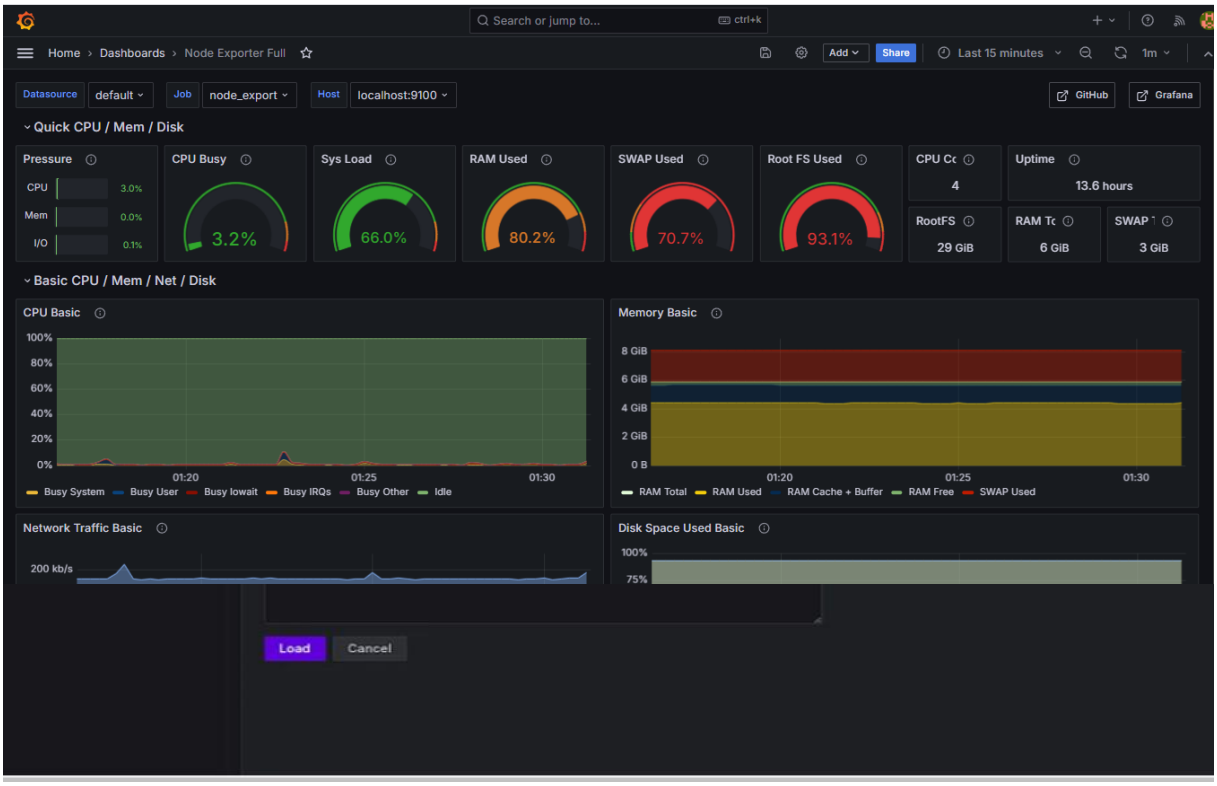
username and password will be admin



Go to the dashboard section of Grafana and click on the Import **dashboard.**



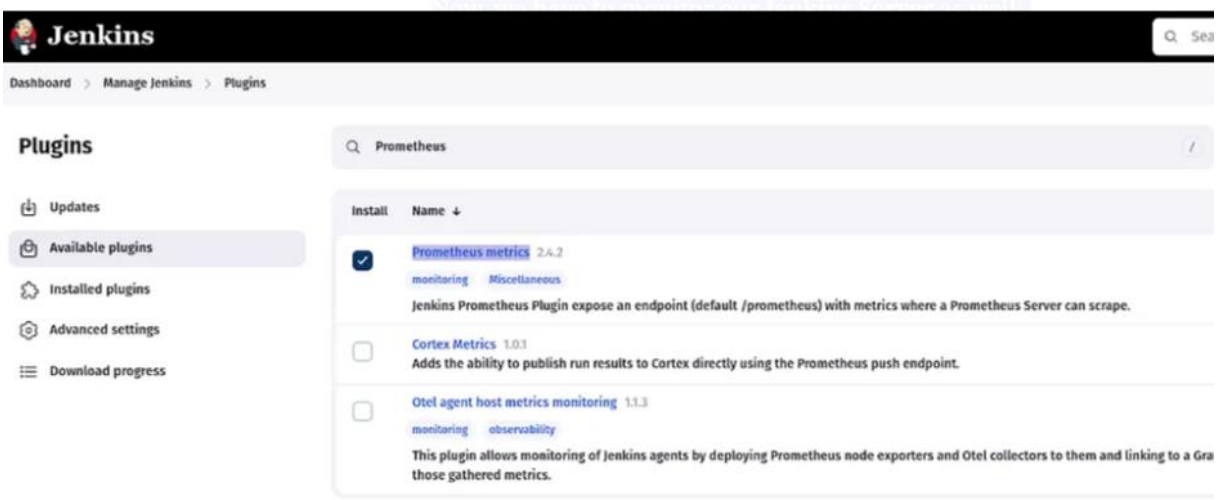
Add 1860 for the node exporter .



Now, we have to monitor our **Jenkins Server** as well.

For that, we need to install the Prometheus metric plugin on our Jenkins.

Go to **Manage Jenkins** -> Plugin search for Prometheus metrics install it and restart your Jenkins.



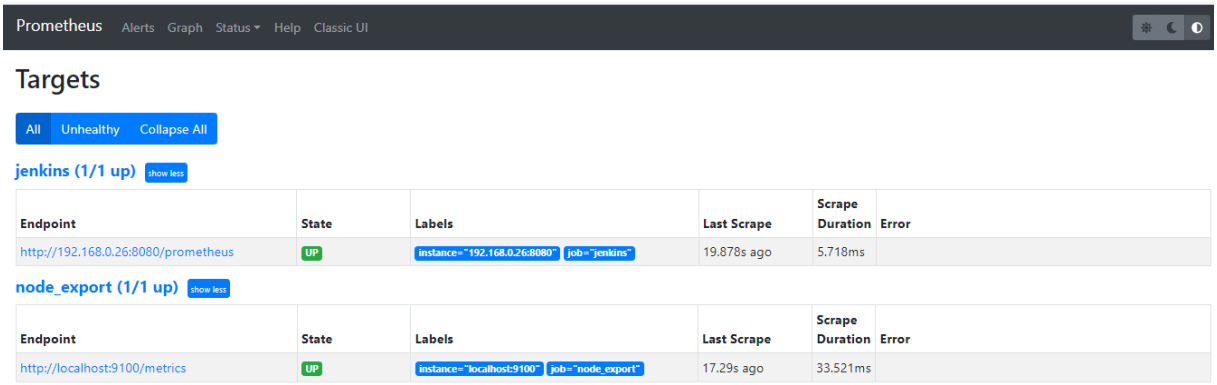
Edit the /etc/prometheus/prometheus.yml file

```
sudo vim /etc/prometheus/prometheus.yml
```

```
- job_name: "jenkins"
    static_configs:
      - targets: ["<jenkins-server-public-ip>:8080"
```

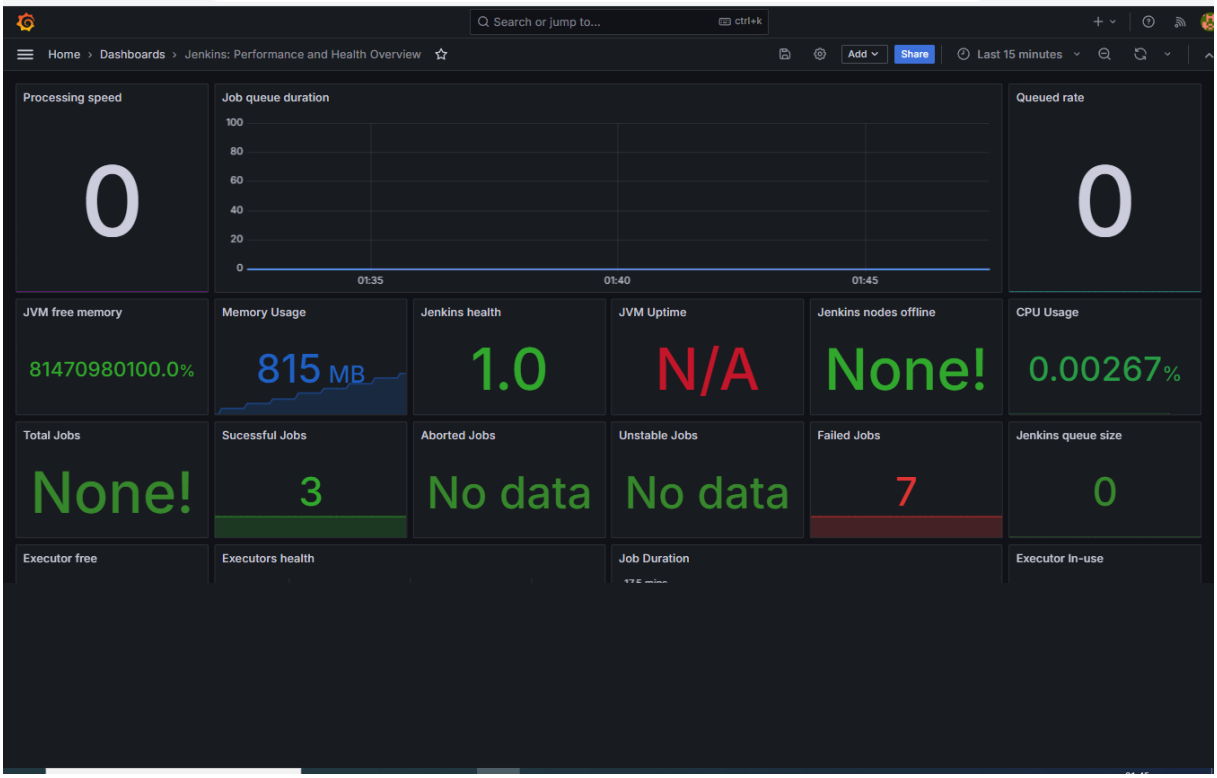
```
promtool check config /etc/prometheus/prometheus.yml
```

You will see the targets that you have added in the /etc/prometheus/prometheus.yml file.



To add the Jenkins Dashboard on your Grafana server.
Provide the 9964 to Load the dashboard.

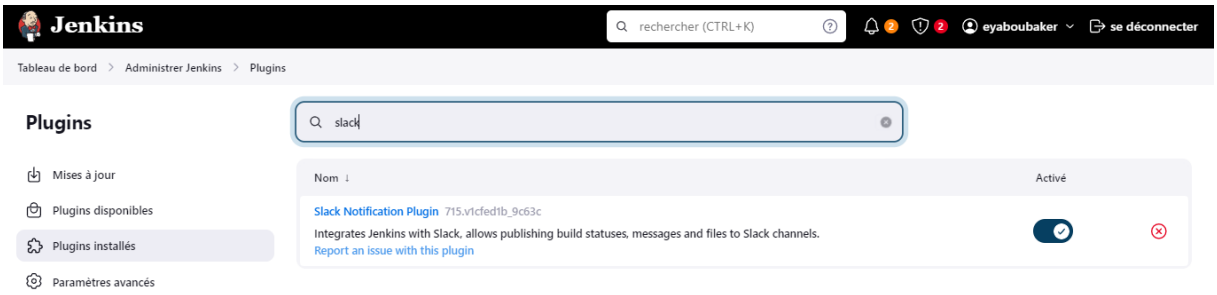
You will see your Jenkins Monitoring dashboard in the below



snippet.

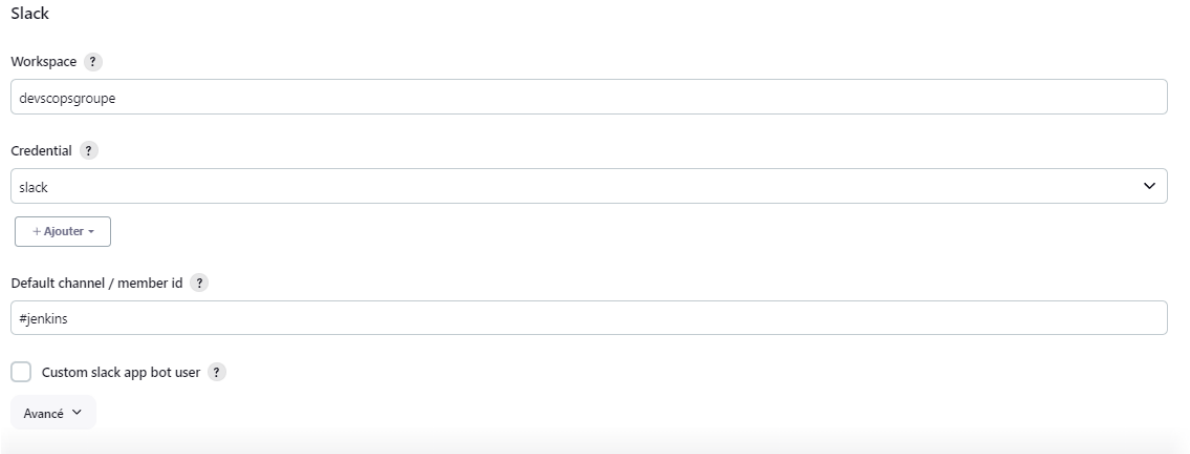
Now, we have to configure slack for the alerts.

Go to Jenkins -> **Manage Jenkins** -> **System**

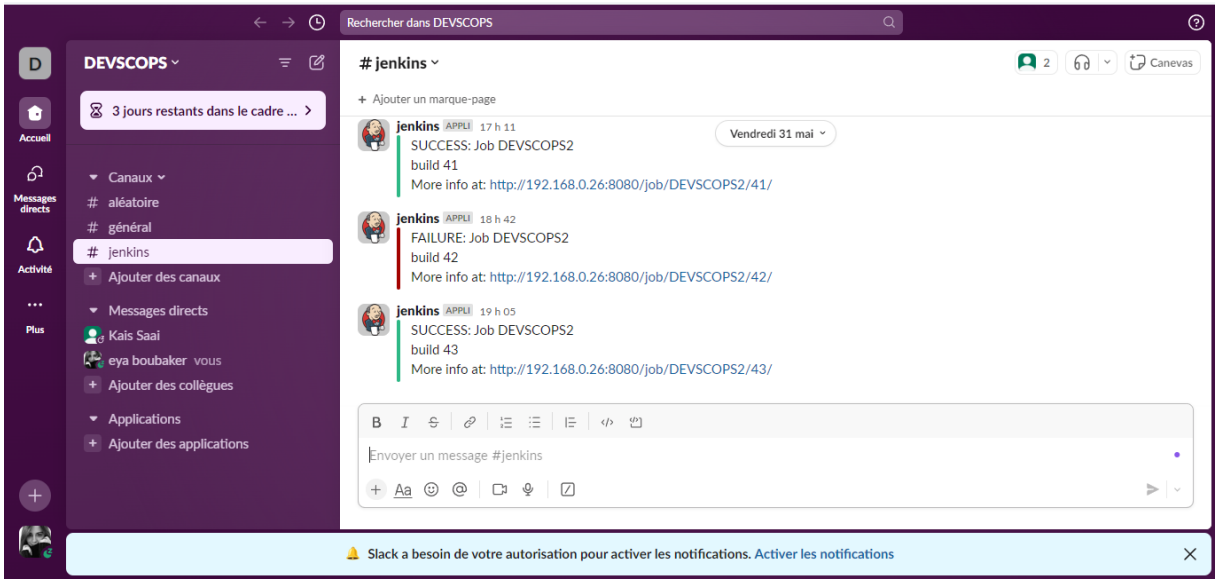


Search for Slack.

Provide the devscopsgroupe in the **Workspace** and the credential slack and jenkins in the default channel.



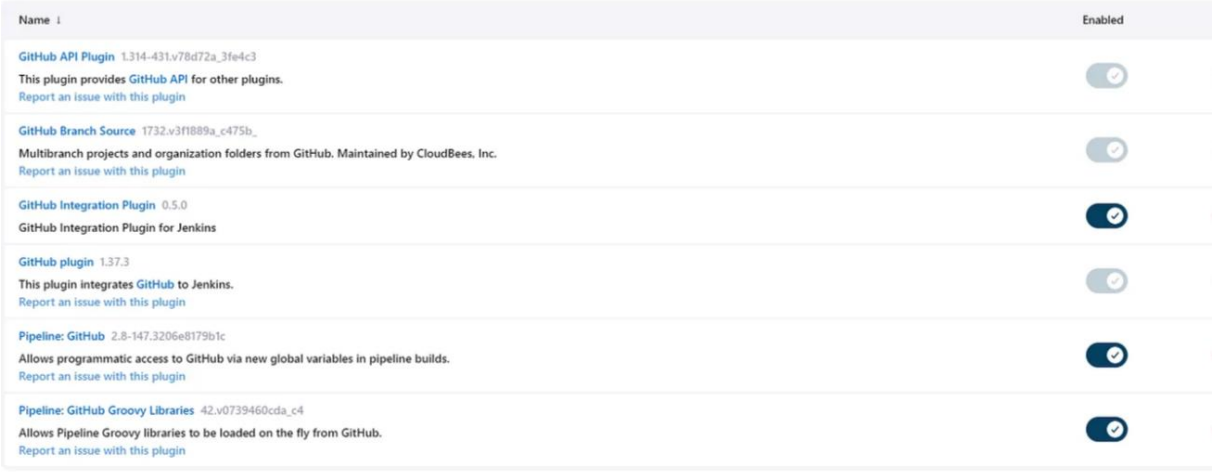
You can see below for the reference.



Now, we will set up our Jenkins Pipeline. But there are some plugins required to work with them.

Download the following plugins

Name ↓	Enabled
CloudBees Docker Build and Publish plugin 1.4.0 This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry. Report an issue with this plugin	✓
CloudBees Docker Custom Build Environment Plugin 1.7.3 Run builds inside a docker container, defined by a Docker image or Dockerfile stored in project SCM. Report an issue with this plugin	✓
CloudBees Docker Hub/Registry Notification 2.7.0 Integrates Jenkins with DockerHub and Docker Registry Report an issue with this plugin	✓
CloudBees Docker Traceability 1.2 Provides an ability to trace server deployments via fingerprints Report an issue with this plugin	✓
Docker 1.4 This plugin integrates Jenkins with Docker Report an issue with this plugin	✓
Docker API Plugin 3.3.1-79.v20b_53427e041 This plugin provides docker-java API for other plugins. Report an issue with this plugin **This plugin is up for adoption!** We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	
Docker Pipeline 572.v950f58993843 Build and use Docker containers from pipelines. Report an issue with this plugin	✓
Docker Slaves Plugin 1.0.7 Uses Docker containers to run Jenkins build agents. Report an issue with this plugin	✓
Docker Swarm Plugin 1.11 Docker swarm cloud plugin. Report an issue with this plugin Warning: The currently installed plugin version may not be safe to use. Please review the following security notices: • Stored XSS vulnerability	✓
docker-build-step 2.9 This plugin allows to add various docker commands to your job as build steps. Report an issue with this plugin	✓
OWASP Dependency-Check 5.4.2 This plug-in can independently execute a Dependency-Check analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities. Report an issue with this plugin	✓
Eclipse Temurin installer Plugin 1.5	
Name ↓	Enabled
SonarQube Scanner for Jenkins 2.15 This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality. Report an issue with this plugin	✓



Now, configure the plugins

Go to **Manage Jenkins -> Tools**

Click on Add JDK and provide the following things below

Installations JDK

Installations JDK Edited

Ajouter JDK

JDK

Nom

jdk17

Install automatically

Install from adoptium.net

Version

jdk-17.0.9+9

Ajouter un installateur

Installations SonarQube Scanner

Installations SonarQube Scanner Edited

Ajouter SonarQube Scanner

SonarQube Scanner

Name

sonar-scanner

Install automatically

Installer depuis Maven Central

Version

SonarQube Scanner 5.0.1.3006

Ajouter un installateur

Installations Maven

Installations Maven Edited

Ajouter Maven

Maven

Nom

maven3

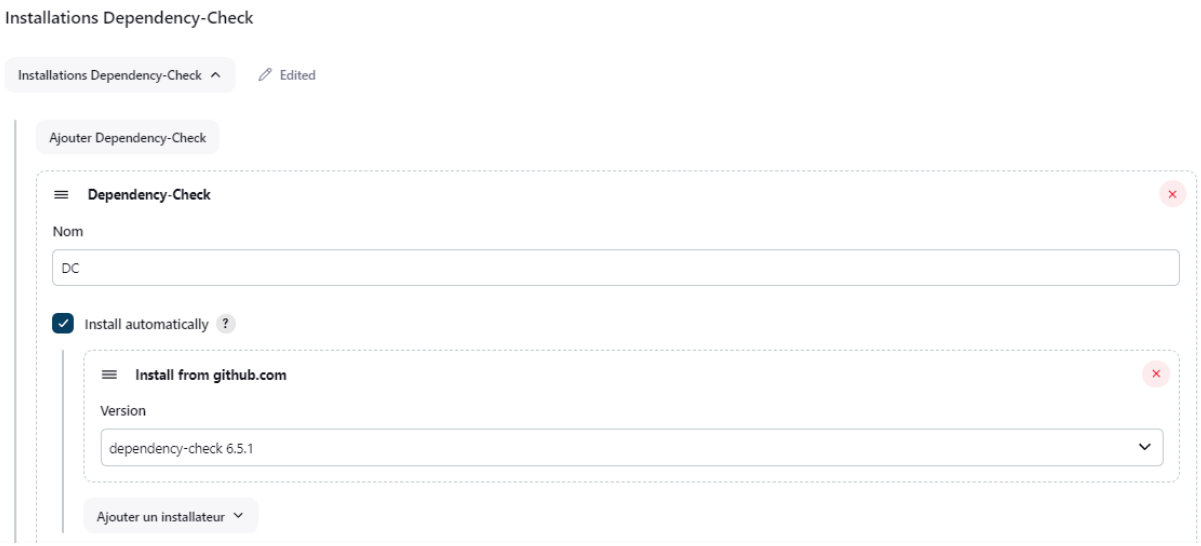
Install automatically

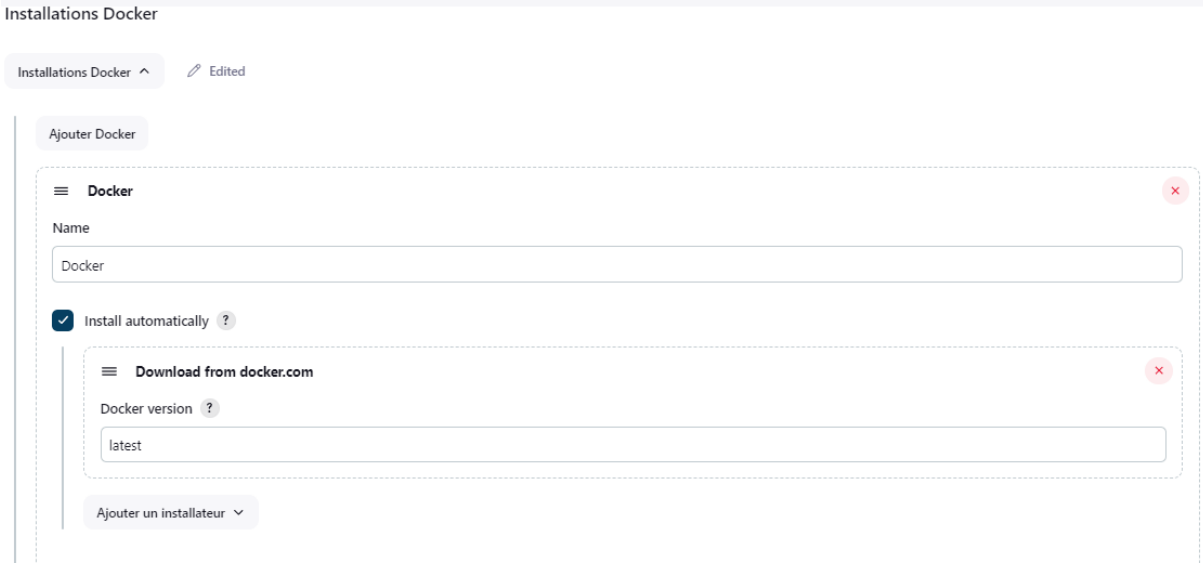
Install from Apache

Version

3.6.3

Ajouter un installateur

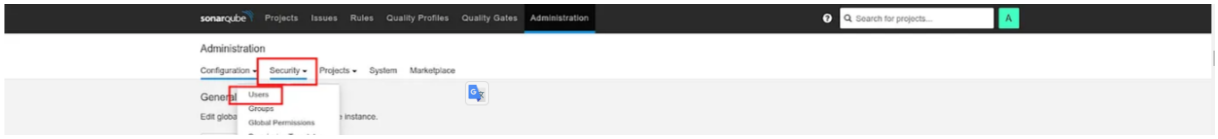




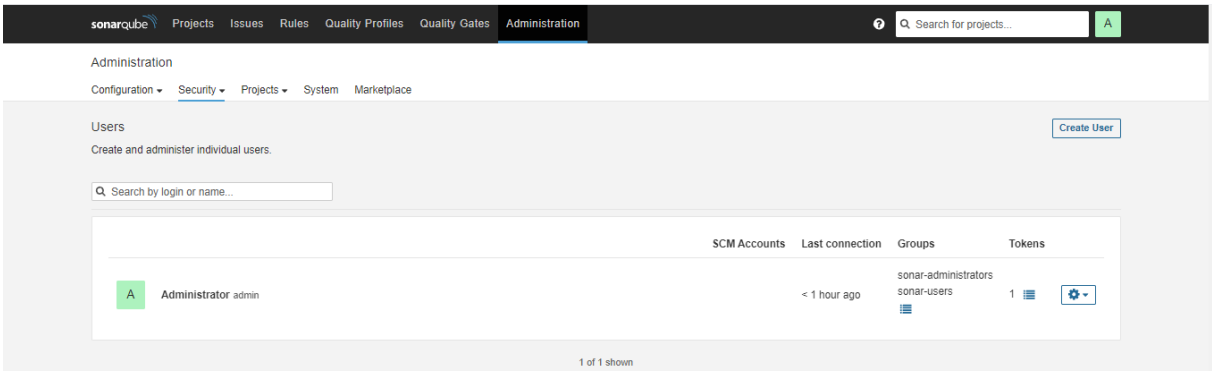
Now, we will configure Sonarqube

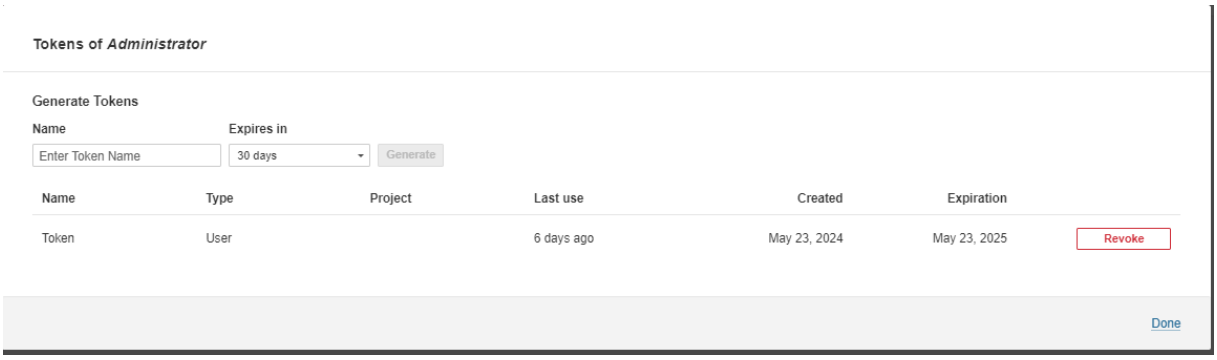
To access the sonarqube, copy the machine public IP with port number 9000

Then, click **Security** and click on **Users.**



Click on the highlighted blue box on the right to generate the token.

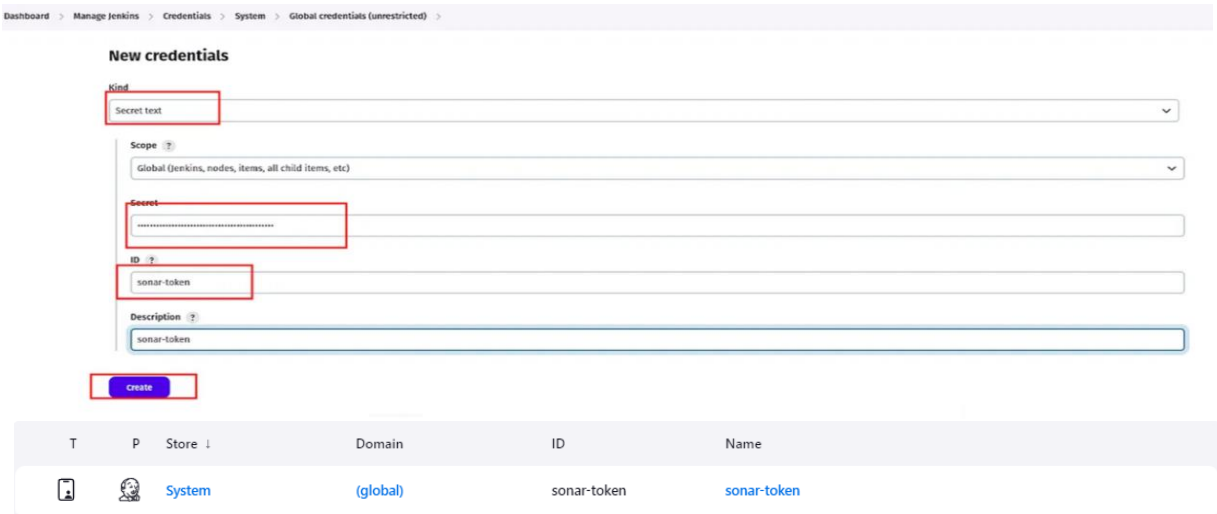




Now, add the token to your Jenkins credentials

Go to **Manage Jenkins -> Credentials.**

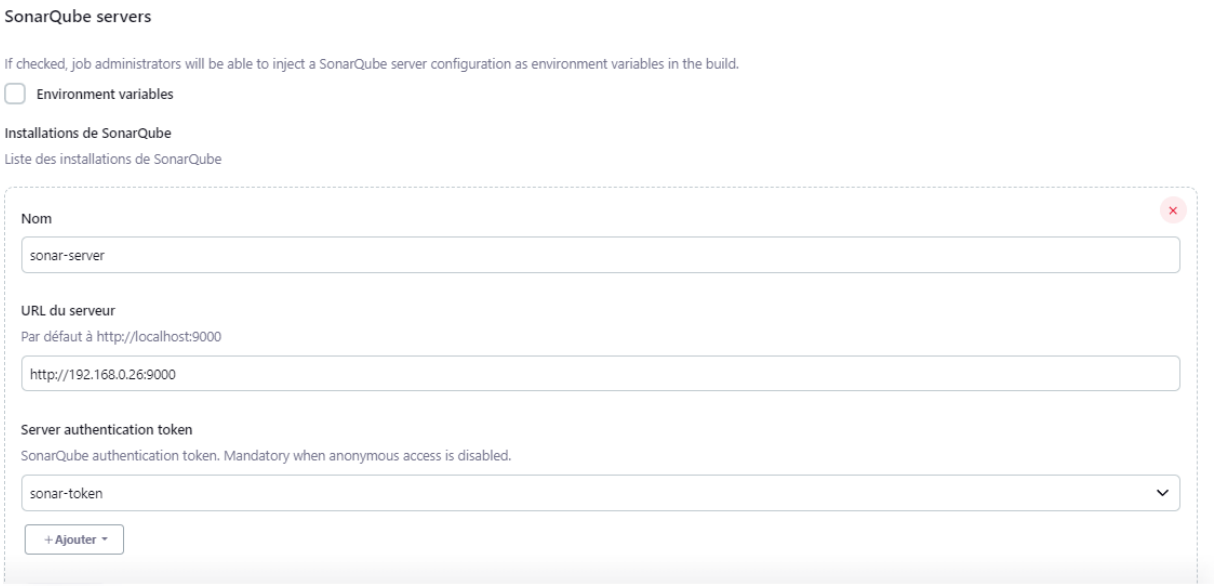
Provide your token then provide the ID as sonar-token to call the credentials.



Go to **Manage Jenkins -> System**

Click on **Add Sonarqube**

Provide the name sonar-server with the Server URL and select the credentials that we have added.

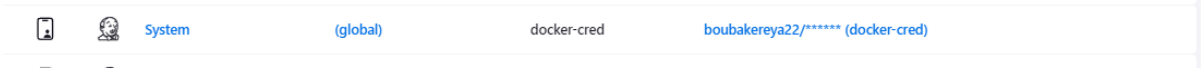


Now, we have to build our Docker Image and push it to DockerHub

To do that, we need to configure the following things.

Go to **Manage Jenkins -> Credentials**

Add Docker Credentials to your JenkinsNow, we will create the Jenkins Pipeline



Click on **Create item.**

Provide the name of your Jenkins Pipeline and select **Pipeline.**

```
def COLOR_MAP = [
    'FAILURE' : 'danger',
    'SUCCESS' : 'good'
]
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        maven 'maven3'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('Checkout From Git'){
            steps{
                git branch: 'master', url:
'https://github.com/eyaboubaker/Devscops.git'
            }
```

```
        }
        stage('mvn compile'){
            steps{
                sh 'mvn clean compile'
            }
        }
        stage('mvn test'){
            steps{
                sh 'mvn test -DskipTests=true'
            }
        }
    stage('Lynis Security Scan') {
        steps {
            script {
        // Exécutez le balayage de sécurité Lynis et convertissez la
sortie en HTML
        sh 'lynis audit system | ansi2html > lynis-report.html'



       // Affiche le chemin absolu du fichier de rapport dans la
console de sortie Jenkins
```

```
                def reportPath = "${WORKSPACE}/lynis-
report.html"

                echo "Chemin du rapport Lynis : ${reportPath}"


// Archive le fichier de rapport pour qu'il soit accessible après
la construction

                archiveArtifacts artifacts: 'lynis-report.html'

   }

}

}

stage('OWASP FS SCAN') {

          steps {

             dependencyCheck additionalArguments: '--scan ./' ,
odcInstallation: 'DC'

             dependencyCheckPublisher pattern: '**/dependency-
check-report.xml'

          }

       }


       stage("Sonarqube Analysis "){

          steps{

             withSonarQubeEnv('sonar-server') {
```

```
                sh ''' $SCANNER_HOME/bin/sonar-scanner -
Dsonar.projectName=EKART \

                -Dsonar.java.binaries=. \

                -Dsonar.projectKey=EKART '''

            }

        }

    }

    stage('Build'){

        steps{

            sh "mvn package -DskipTests=true "

    }

    }

    stage('Publish To Nexus') {

        steps {

            withMaven(globalMavenSettingsConfig: 'global-
maven', jdk: 'jdk17', maven: 'maven3', mavenSettingsConfig: '',
traceability: true) {

                sh "mvn deploy -DskipTests=true"

    }

    }

}

    stage('Build & Tag Docker Image') {
```

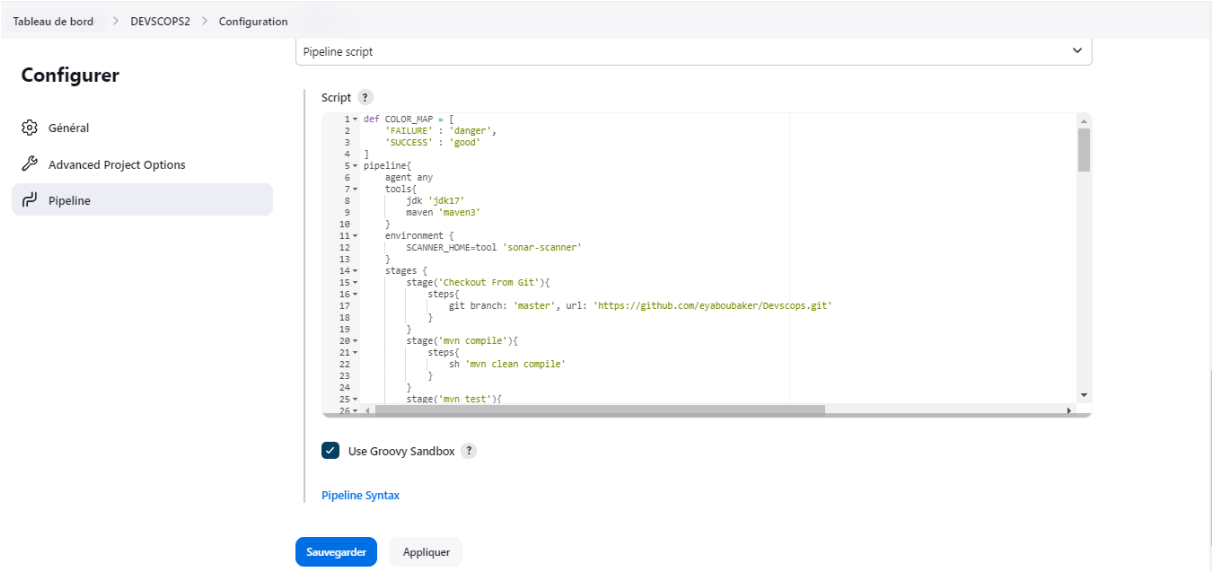
```
        steps {
          script {
            withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'Docker') {
                sh "docker build -t boubakereya22/ekart:latest -
f docker/Dockerfile ."
            }
          }
        }
      }
    stage('TRIVY') {
        steps {
          script {
            // Effectue le balayage de sécurité de l'image et écrit
la sortie dans un fichier HTML
            sh 'trivy image --format table --timeout 5m -o trivy-
image-report.html boubakereya22/ekart:latest'


            // Affiche le chemin absolu du fichier de rapport
dans la console de sortie Jenkins
            def reportPath = "${WORKSPACE}/trivy-image-
report.html"
            echo "Chemin du rapport Trivy : ${reportPath}"
```

```
                // Archive le fichier de rapport pour qu'il soit
accessible après la construction

                archiveArtifacts artifacts: 'trivy-image-report.html'

            }

        }

    }

    stage('Push Docker Image') {

        steps {

          script {

              withDockerRegistry(credentialsId: 'docker-cred',
toolName: 'Docker') {

                    sh "docker push boubakereya22/ekart:latest"

                }

          }

        }

    }

    stage('Deploy To Kubernetes') {

        steps {

          withKubeConfig(caCertificate: '', clusterName: '',
contextName: '', credentialsId: 'k3s-cred', namespace:
'webapps', restrictKubeConfigAccess: false, serverUrl:
'https://192.168.0.8:6443') {
```

```
                sh "kubectl apply -f deploymentservice.yml"


            }

        }

    }


    stage('Verify the Deployment') {

        steps {

            withKubeConfig(caCertificate: '', clusterName: '',
contextName: '', credentialsId: 'k3s-cred', namespace:
'webapps', restrictKubeConfigAccess: false, serverUrl:
'https://192.168.0.8:6443') {

                sh "kubectl get pods -n webapps"

                sh "kubectl get svc -n webapps"

            }

        }

    }

    stage('Nikto Security Scan') {

        steps {

            script {

            // Exécuter Nikto et enregistrer la sortie dans nikto-
report.html dans le répertoire de travail
```

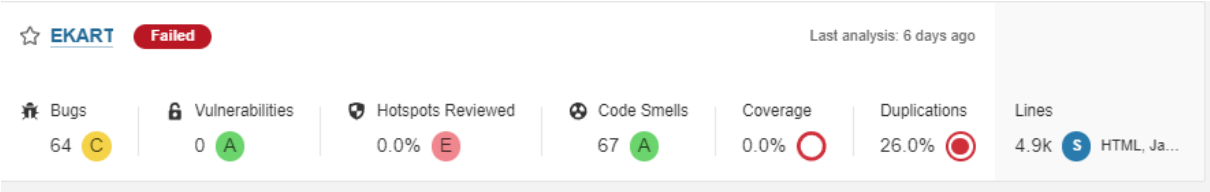
```
            sh 'nikto -h https://192.168.0.9:8070 -o Nikto-report.html'

            // Affiche le chemin absolu du fichier de rapport dans la console de sortie Jenkins

                def reportPath = "${WORKSPACE}/Nikto-report.html"

                echo "Chemin du rapport Nikto : ${reportPath}"



            // Archive le fichier de rapport pour qu'il soit accessible après la construction

                archiveArtifacts artifacts:'Nikto-report.html'

        }

    }

     }

}

  post {

  always {

    echo 'Slack Notifications'

    slackSend (

      channel: '#jenkins',

      color: COLOR_MAP[currentBuild.currentResult],
```

```
            message: "${currentBuild.currentResult}: Job ${env.JOB_NAME} \n build ${env.BUILD_NUMBER} \n More info at: ${env.BUILD_URL}"

        )

    }

}

}
```



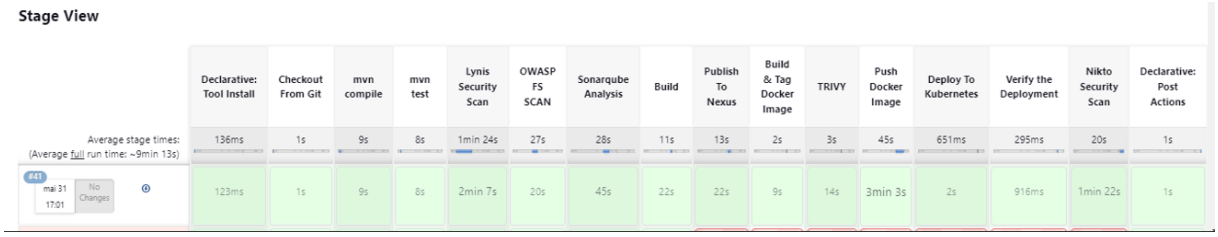
Click on build pipeline and after getting the success of the pipeline.

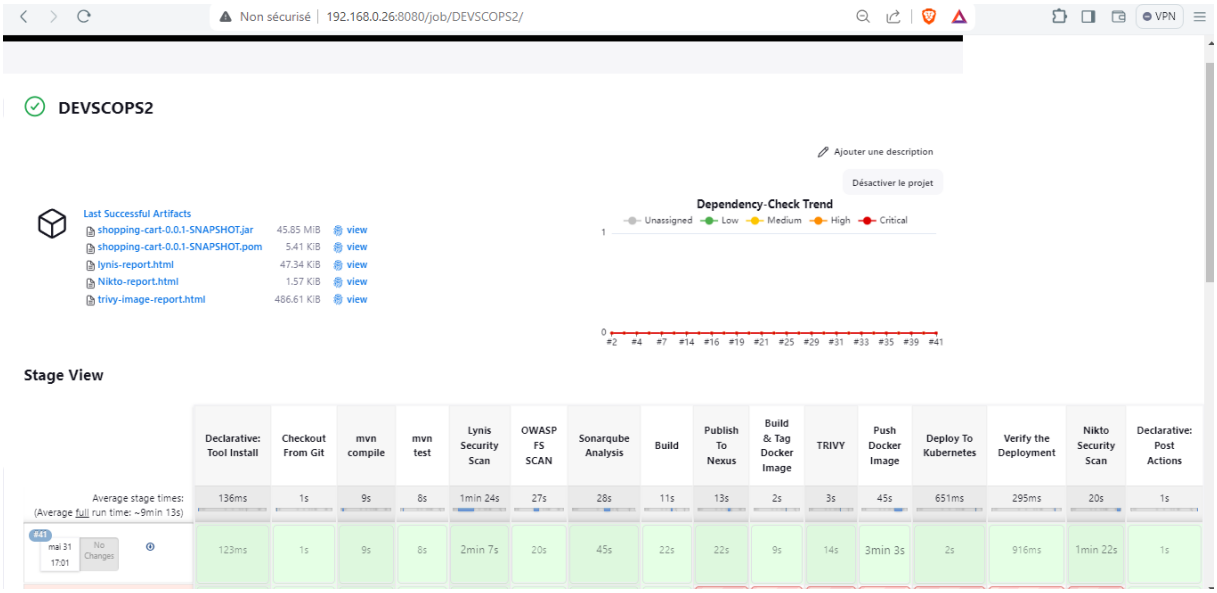
You will see the Sonarqube code quality analysis which will look like the below snippet.



Now, click on **Build Now.**

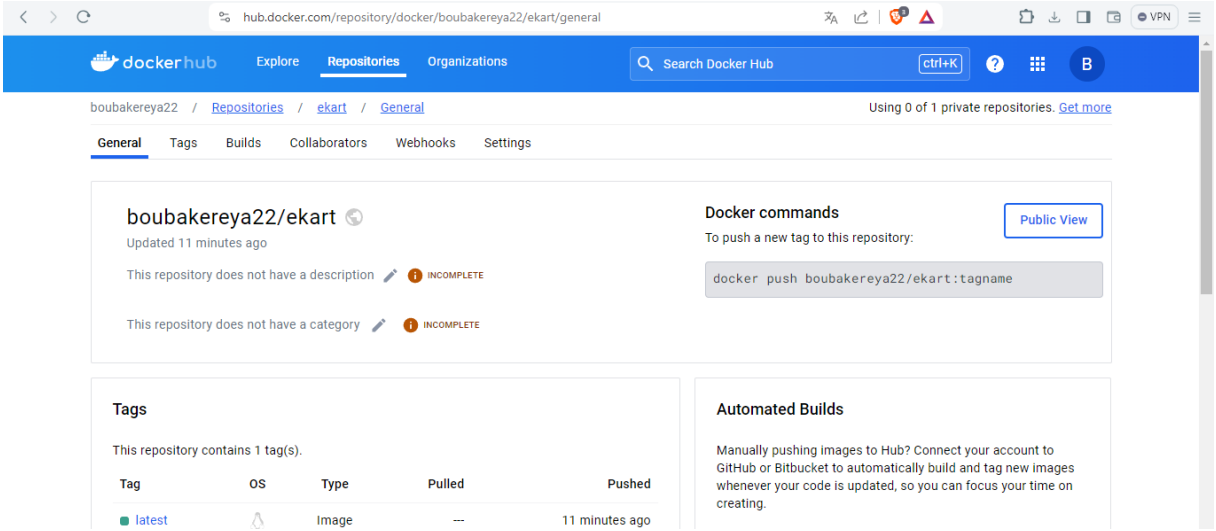
As you can see Our Pipeline is successful.





you will all the artefacts, Now, validate whether the docker image has been pushed to DockerHub or not.

As you can see in the below screenshot, Our Docker image is present on Docker Hub.



Now, we have to deploy our application using Kubernetes.

As you know, we have three Kubernetes Nodes of which one is the Master and the other are the Worker Node.

Login to your both Kubernetes Master and Worker Nodes

Master + worker

```
1- sudo apt update

2- sudo apt upgrade -y

3- sudo reboot

4-

      addresses:

                  - @IP/24

              nameservers:

                  addresses:
```

```
                    - @IP
              routes:
                  - to: default
                    via: @IP_Gateway
```

Master

```
5- sudo hostnamectl set-hostname "k8s-master.placeholder.tn"
```

worker

```
6- sudo hostnamectl set-hostname "k8s-worker1.placeholder.tn"

7- sudo hostnamectl set-hostname "k8s-worker2.placeholder.tn"
```

Master + worker

```
8- exec bash

9- sudo nano /etc/hosts

      192.168.1.10   k8s-master.placeholder.tn k8s-master

      192.168.1.11   k8s-worker1.placeholder.tn k8s-worker1

      192.168.1.12   k8s-worker2.placeholder.tn k8s-worker2

10- sudo swapoff -a

11- sudo nano /etc/fstab

      #/swap.img       none       swap       sw       0       0

12- sudo mount -a

13- free -h

14 - sudo tee /etc/modules-load.d/containerd.conf <<EOF

overlay

br_netfilter
```

```
EOF

15- sudo modprobe overlay

16- sudo modprobe br_netfilter

17- sudo tee /etc/sysctl.d/kubernetes.conf <<EOF

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

net.ipv4.ip_forward = 1

EOF

18- sudo apt install -y curl wget

/// Master

19- curl -sfL https://get.k3s.io | sh -

20- sudo systemctl status k3s

21- mkdir ~/.kube

22- sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config && sudo chown
$USER ~/.kube/config

23- sudo chmod 600 ~/.kube/config && export KUBECONFIG=~/.kube/config

24- curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="--flannel-
backend=none --disable-network-policy --cluster-cidr=10.10.0.0/16" sh
-
```

Master

```
25- sudo cat /var/lib/rancher/k3s/server/node-token
```

Worker

```
26- curl -sfL https://get.k3s.io | K3S_URL=https://serverip:6443
K3S_TOKEN=mytoken sh -
```

Master

```
27- kubectl create -f
https://raw.githubusercontent.com/projectcalico/calico/v3.28.0/manife
sts/tigera-operator.yaml
```

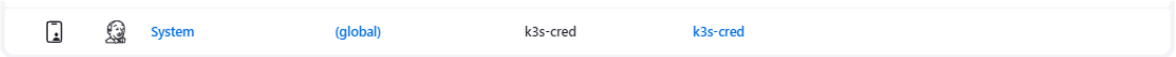
Both nodes are ready.

```
eya@k3smaster:~$ sudo kubectl get nodes
NAME                     STATUS   ROLES                  AGE   VERSION
k3smaster.example.net    Ready    control-plane,master   3d    v1.29.5+k3s1
k3sworker2.example.net   Ready    <none>                 3d    v1.29.5+k3s1
k3sworker1.example.net   Ready    <none>                 3d    v1.29.5+k3s1
```

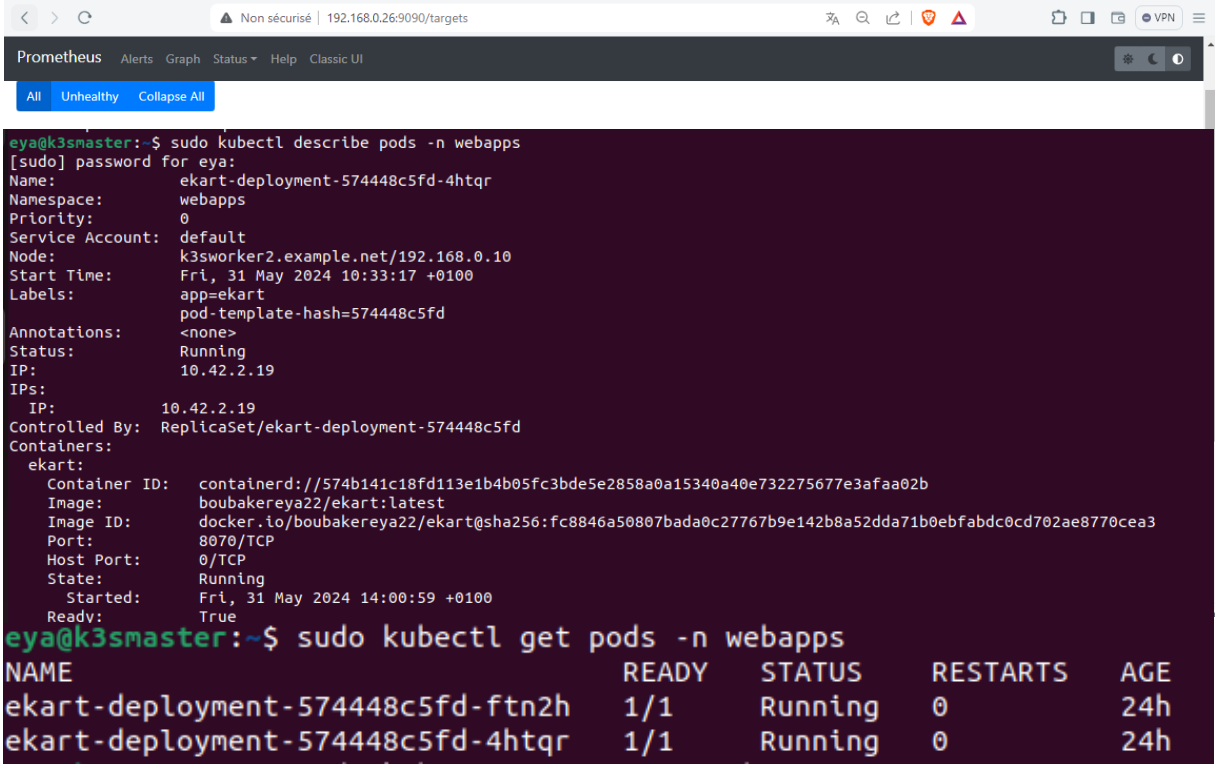
Now, we will set Kubernetes Monitoring for both Master and worker Nodes

```
GNU nano 6.2                    /etc/prometheus/prometheus.yml

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

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

- job_name: 'jenkins'
  metrics_path: '/prometheus'
  static_configs:
    - targets: ["192.168.0.26:8080"]

- job_name: node_export_masterk3s
  static_configs:
    - targets: ["192.168.0.8:9100"]

- job_name: node_export_worker1k3s
  static_configs:
    - targets: ["192.168.0.9:9100"]

- job_name: node_export_worker2k3s
  static_configs:
    - targets: ["192.168.0.10:9100"]
```

Click on **Add credentials.**



You can validate whether your pods are running or not from your Kubernetes master node.



Also, you can check the Console logs for the earlier results.

We got a notification in slack that our pipeline was successful.

```
+ kubectl get pods -n webapps
NAME                              READY   STATUS    RESTARTS   AGE
ekart-deployment-574448c5fd-ftn2h   1/1     Running   0          24h
ekart-deployment-574448c5fd-4htqr   1/1     Running   0          24h
[Pipeline] sh
+ kubectl get svc -n webapps
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP                          PORT(S)          AGE
ekart-ssvc   LoadBalancer   10.43.73.239   192.168.0.10,192.168.0.8,192.168.0.9   8070:30419/TCP   2d
[Pipeline] }
```

jenkins APPLI 19 h 05

SUCCESS: Job DEVSCOPS2
build 43
More info at: http://192.168.0.26:8080/job/DEVSCOPS2/43/

Build Artifacts

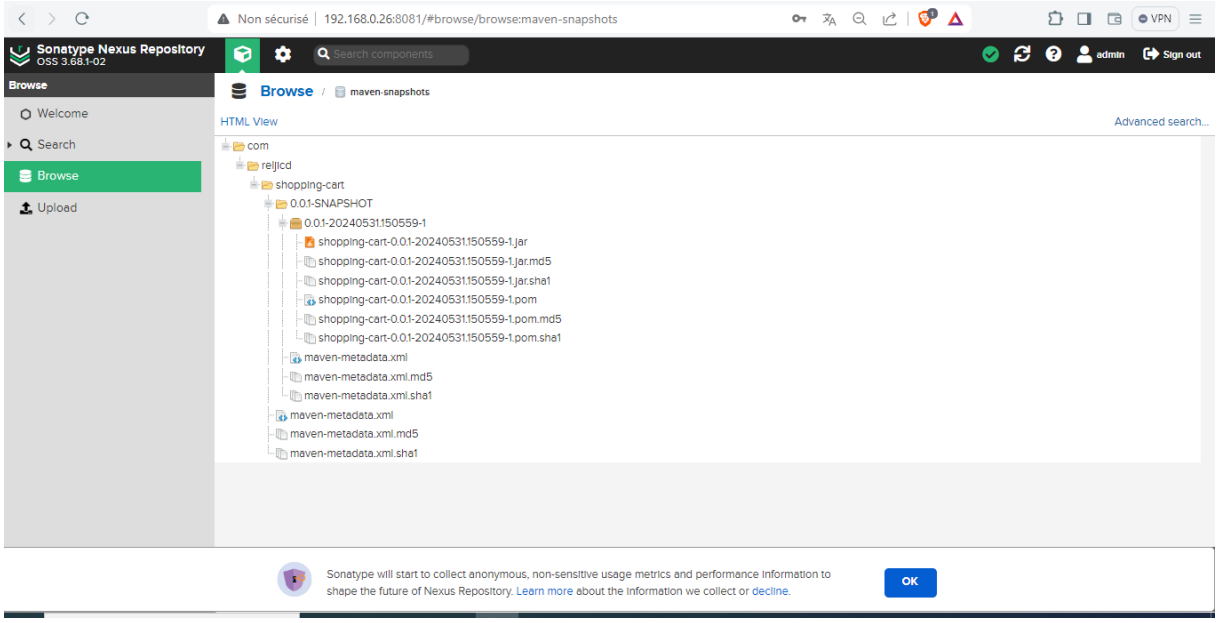
shopping-cart-0.0.1-SNAPSHOT.jar	45.85 MiB	view
shopping-cart-0.0.1-SNAPSHOT.pom	5.41 KiB	view
lynis-report.html	47.34 KiB	view
Nikto-report.html	1.57 KiB	view
trivy-image-report.html	486.61 KiB	view

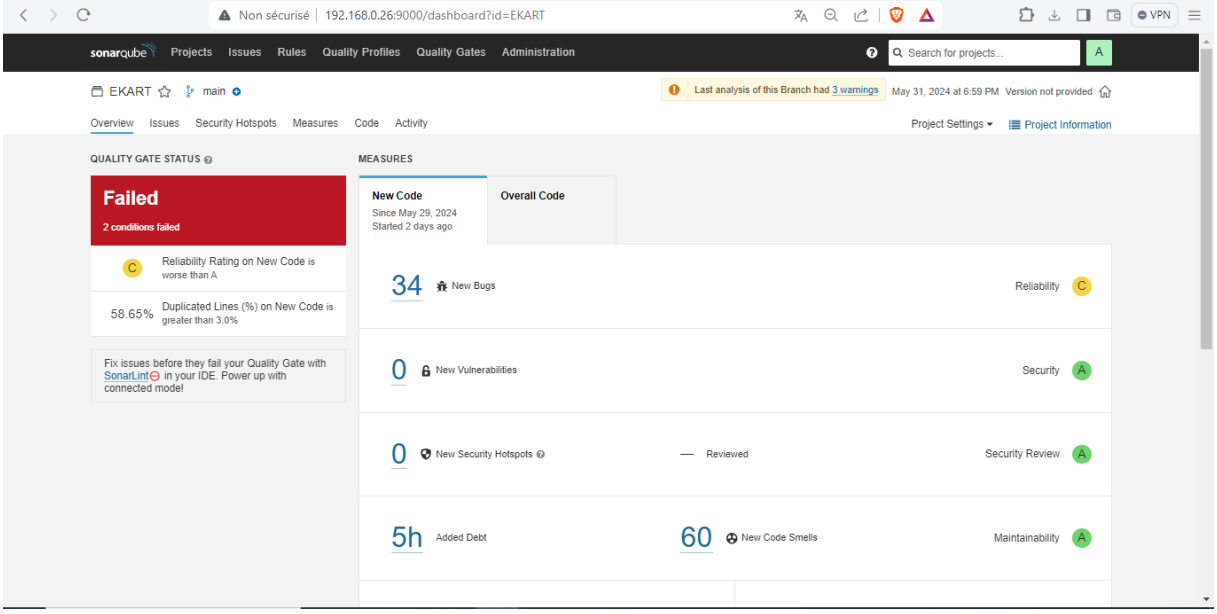
we got the vulnerabilities for our Docker Image.

/trivy-image-report.html

2k 62.5% 0 0 0 0

```
boubakereya22/ekart:latest (alpine 3.7.0)
=========================================
Total: 338 (UNKNOWN: 0, LOW: 140, MEDIUM: 140, HIGH: 43, CRITICAL: 15)

┌──────────────────┬────────────────┬──────────┬────────┬───────────────┬───────────────┬──────────────────────────────────────────────────────────┐
│     Library      │ Vulnerability  │ Severity │ Status │ Installed Version │ Fixed Version │                          Title                           │
├──────────────────┼────────────────┼──────────┼────────┼───────────────┼───────────────┼──────────────────────────────────────────────────────────┤
│ freetype         │ CVE-2018-6942  │ MEDIUM   │ fixed  │ 2.8.1-r2      │ 2.8.1-r3      │ freetype: NULL pointer dereference in the Ins_GETVARIATION() │
│                  │                │          │        │               │               │ function                                                 │
│                  │                │          │        │               │               │ https://avd.aquasec.com/nvd/cve-2018-6942                │
├──────────────────┼────────────────┼──────────┼        ├───────────────┼───────────────┼──────────────────────────────────────────────────────────┤
│ krb5-libs        │ CVE-2017-15088 │ CRITICAL │        │ 1.15.2-r1     │ 1.15.3-r0     │ krb5: Buffer overflow in get_matching_data()             │
│                  │                │          │        │               │               │ https://avd.aquasec.com/nvd/cve-2017-15088               │
│                  ├────────────────┼──────────┤        │               │               ├──────────────────────────────────────────────────────────┤
│                  │ CVE-2018-5709  │ HIGH     │        │               │               │ krb5: integer overflow in dbentry->n_key_data in         │
│                  │                │          │        │               │               │ kadmin/dbutil/dump.c                                     │
│                  │                │          │        │               │               │ https://avd.aquasec.com/nvd/cve-2018-5709                │
│                  ├────────────────┼──────────┤        │               ├───────────────┼──────────────────────────────────────────────────────────┤
│                  │ CVE-2018-20217 │ MEDIUM   │        │               │ 1.15.4-r0     │ krb5: Reachable assertion in the KDC using S4U2Self requests │
│                  │                │          │        │               │               │ https://avd.aquasec.com/nvd/cve-2018-20217               │
│                  ├────────────────┤          │        │               ├───────────────┼──────────────────────────────────────────────────────────┤
│                  │ CVE-2018-5710  │          │        │               │ 1.15.3-r0     │ krb5: null pointer deference in strlen function in       │
│                  │                │          │        │               │               │ plugins/kdb/ldap/libkdb_ldap/ldap_principal2.c           │
│                  │                │          │        │               │               │ https://avd.aquasec.com/nvd/cve-2018-5710                │
├──────────────────┼────────────────┼──────────┼        ├───────────────┼───────────────┼──────────────────────────────────────────────────────────┤
```





Go to the Grafana Dashboard and select Node Exporter.

You will see the real-time hardware specs of your Kubernetes master node.

You will see the real-time hardware specs of your Kubernetes worker node.

Copy the Public IP of Worker Node and paste it on your favorite browser with port 8304 and see the magic.