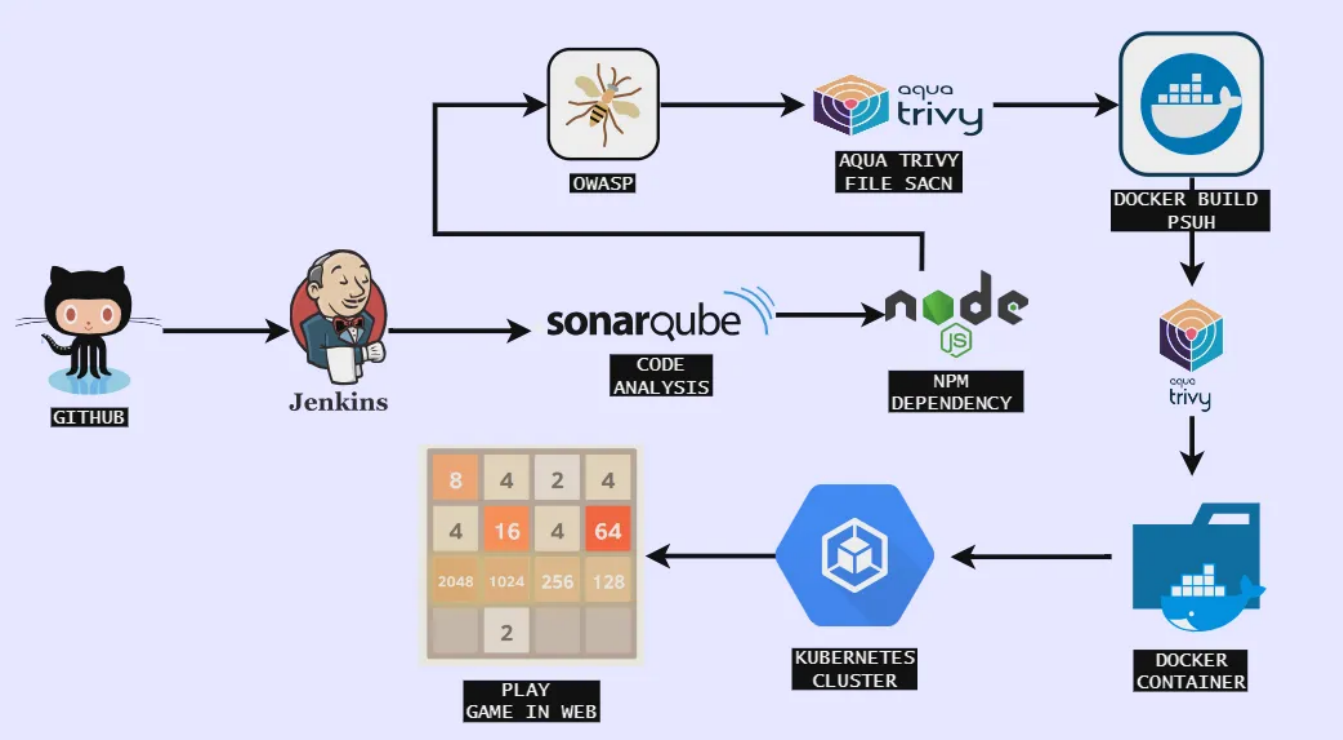
DevSecOps: Deploying the 2048 Game on Docker and Kubernetes with Jenkins CI/CD



Hello friends, we will be deploying a React Js 2048 Game. We will be using Jenkins as a CICD tool and deploying our application on a Docker container and Kubernetes Cluster. I Hope this detailed blog is useful.

Youtube video : https://youtu.be/21Z-u8Fd_Mk?si=Ux4WabhFAxtYH9-g

GitHub Repo : https://github.com/Aj7Ay/2048-React-CICD.git

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

Step 4 — Create a Pipeline Project in Jenkins using a Declarative Pipeline

Step 5 — Install OWASP Dependency Check Plugins

Step 6 — Docker Image Build and Push

Step 7 — Deploy the image using Docker

Step 8 — Kubernetes master and slave setup on Ubuntu (20.04)

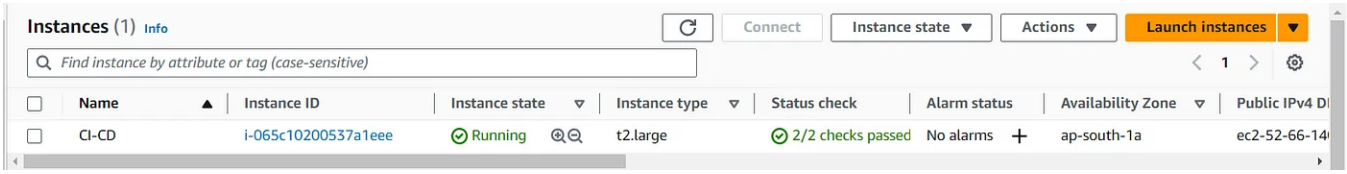
Step 9 — Access the Game on Browser.

Step 10 — 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
```

```
#!/bin/bash
sudo apt update -y
#sudo apt upgrade -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /e
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.
sudo apt update -y
sudo apt install temurin-17-jdk -y
/usr/bin/java --version
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee
```

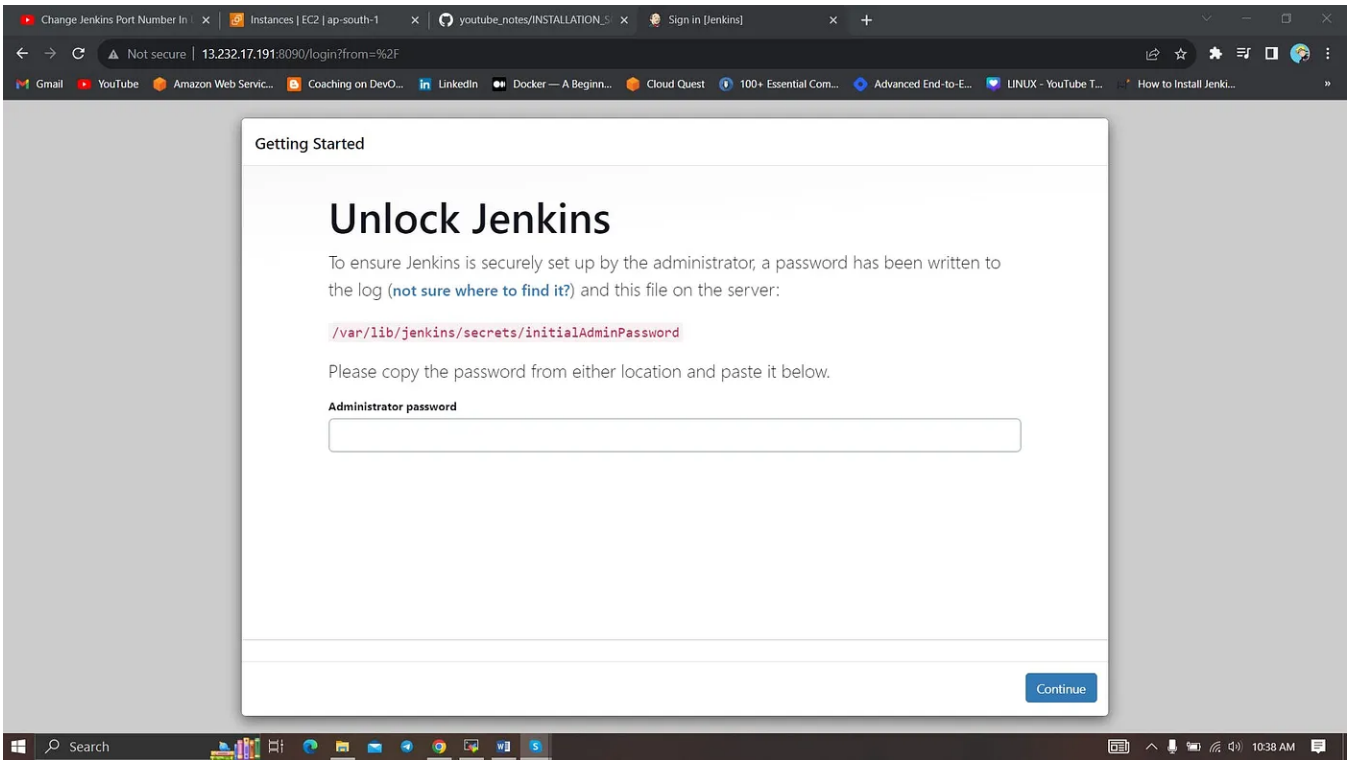
```
                    /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
                    https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
                                /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update -y
sudo apt-get install jenkins -y
sudo systemctl start jenkins
sudo systemctl status jenkins
```

```
sudo chmod 777 jenkins.sh
./jenkins.sh    # this will installl jenkins
```

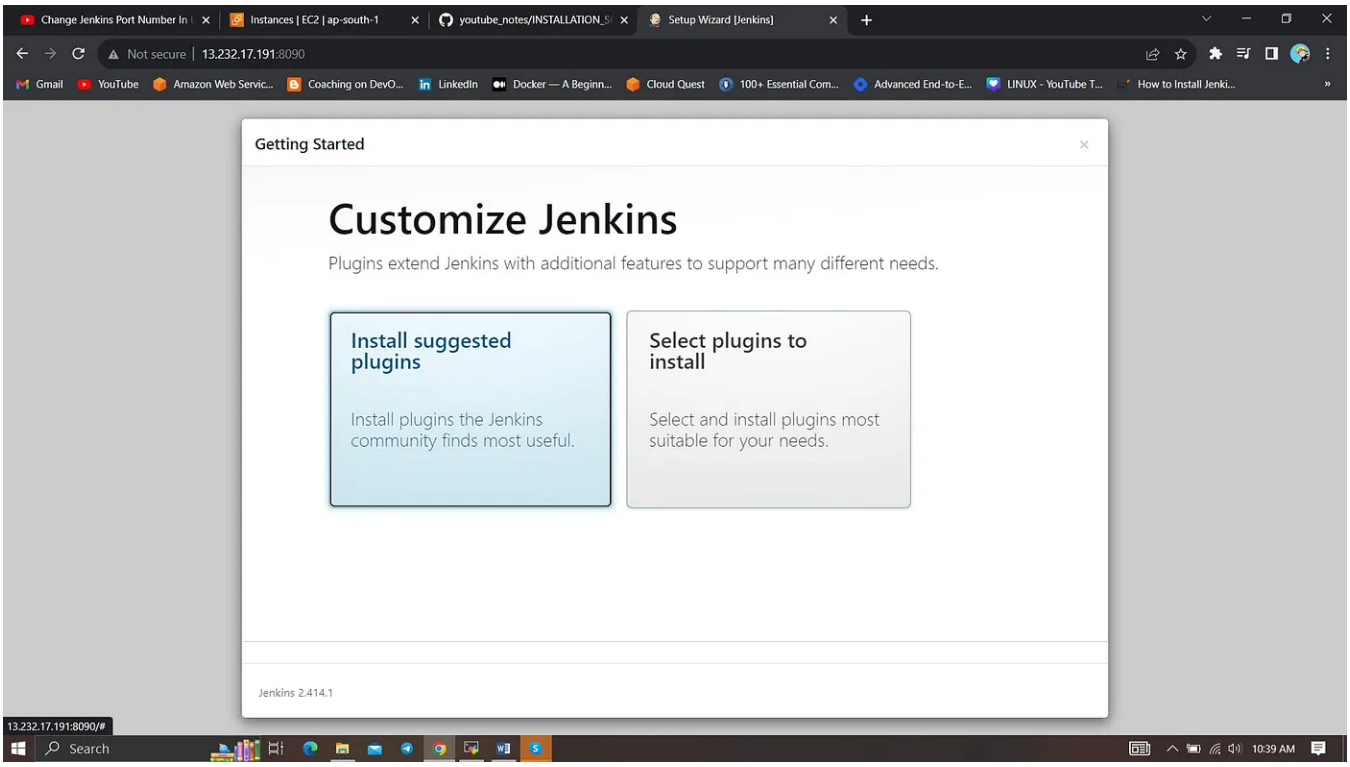
Once Jenkins is installed, you will need to go to your AWS EC2 Security Group and open Inbound Port 8080, since Jenkins works on Port 8080.

Now, grab your Public IP Address

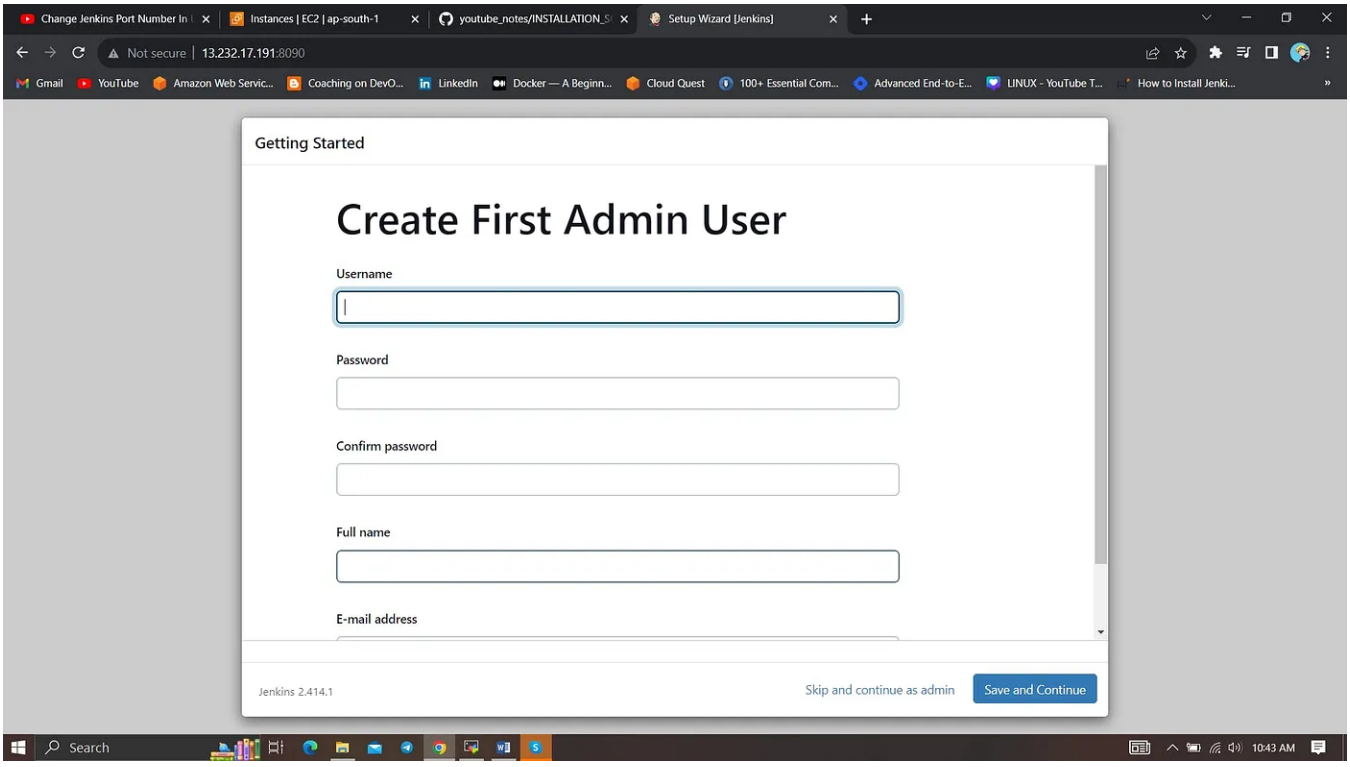
```
<EC2 Public IP Address:8080>
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



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

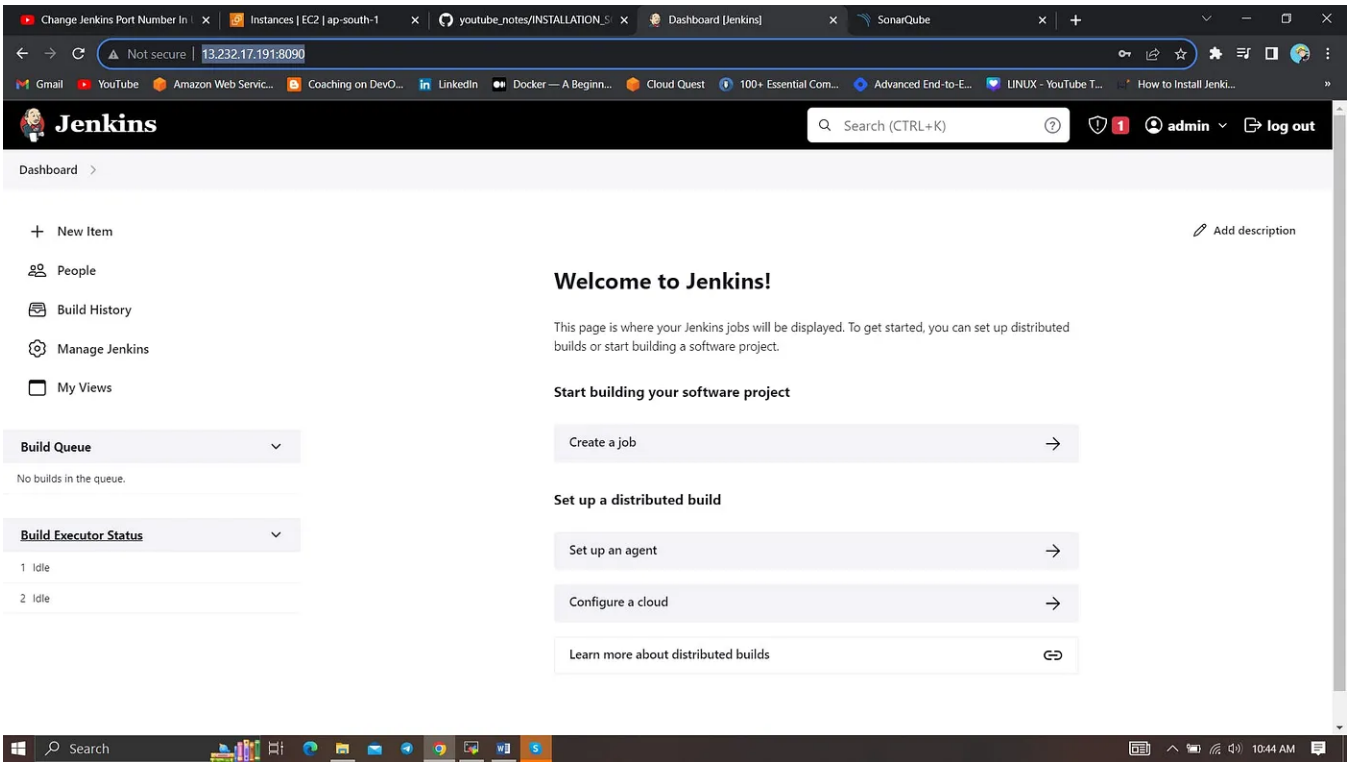


Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

Jenkins Getting Started Screen.



2B — Install Docker

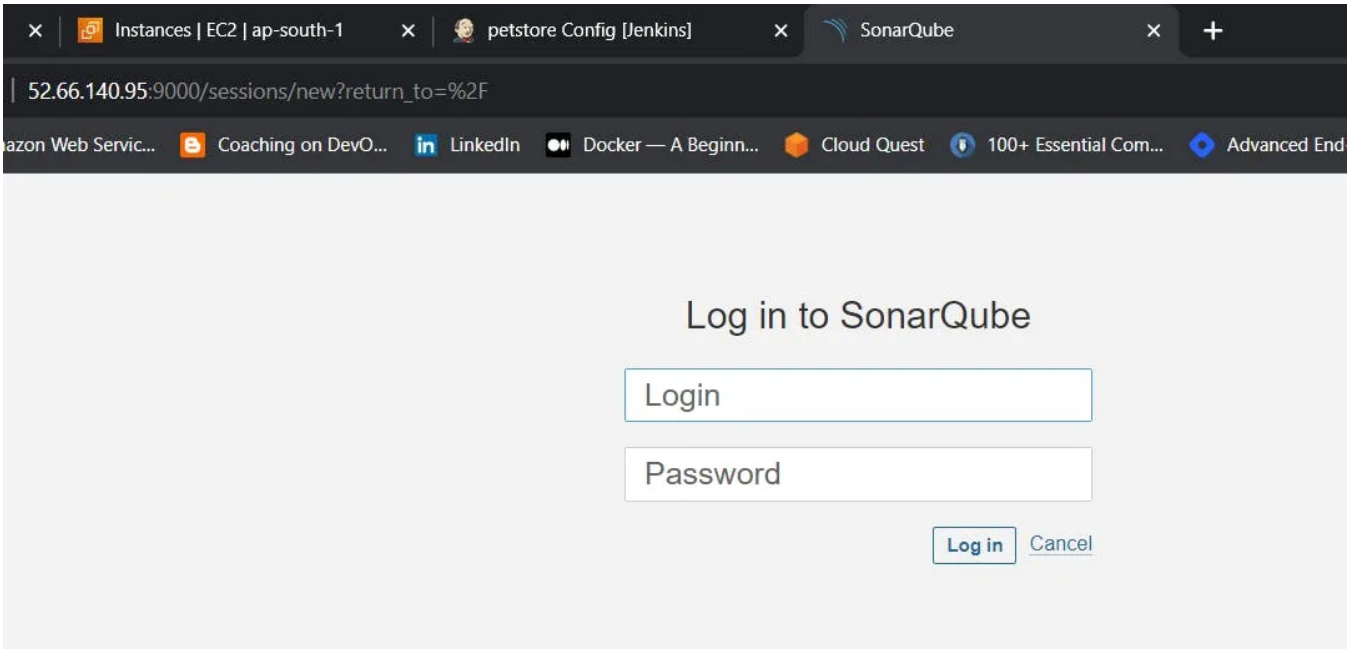
```
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER   #my case is ubuntu
newgrp docker
sudo chmod 777 /var/run/docker.sock
```

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

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

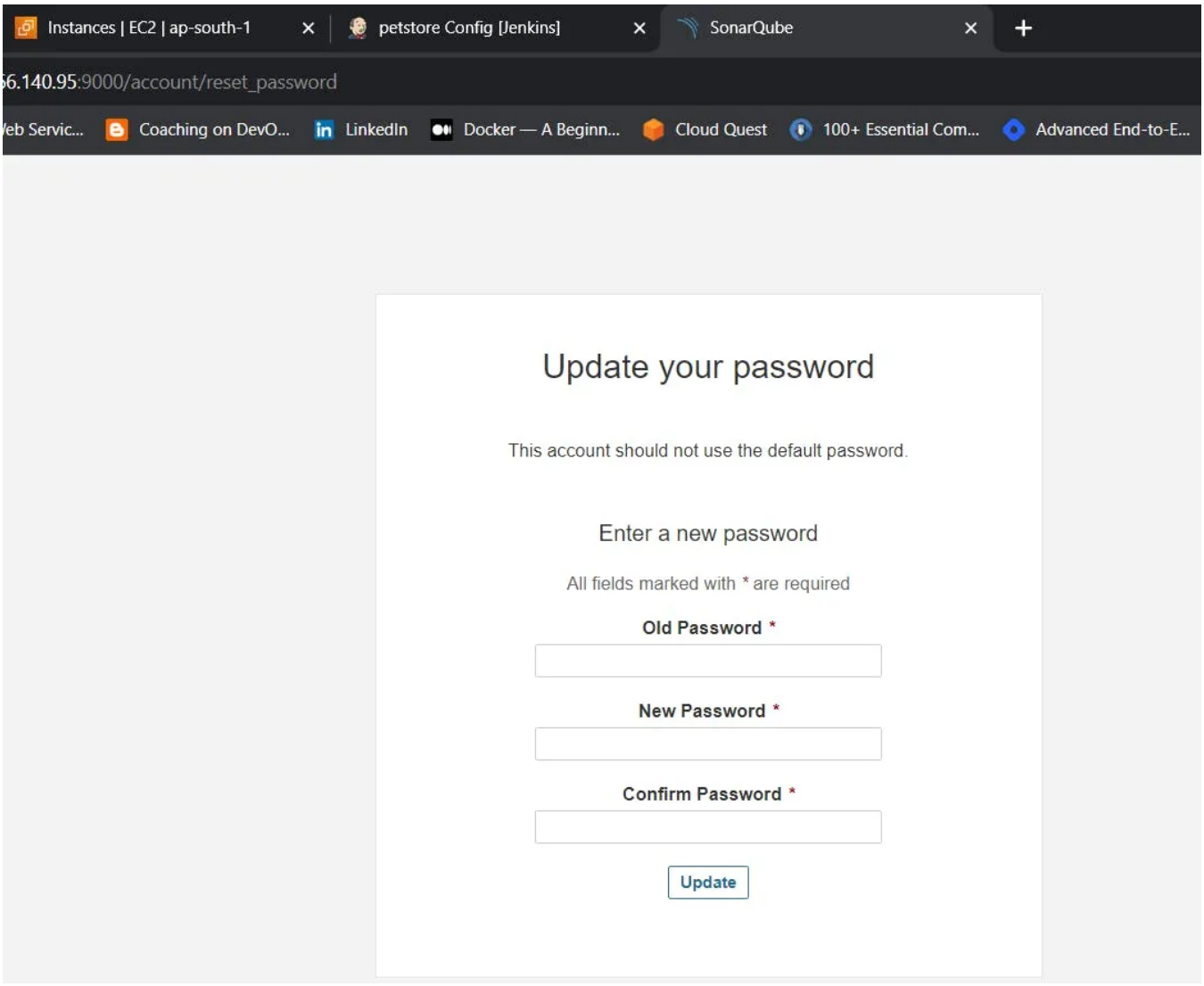
```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882f8eb: Pull complete
2cabec57fa36: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

Now our Sonarqube is up and running

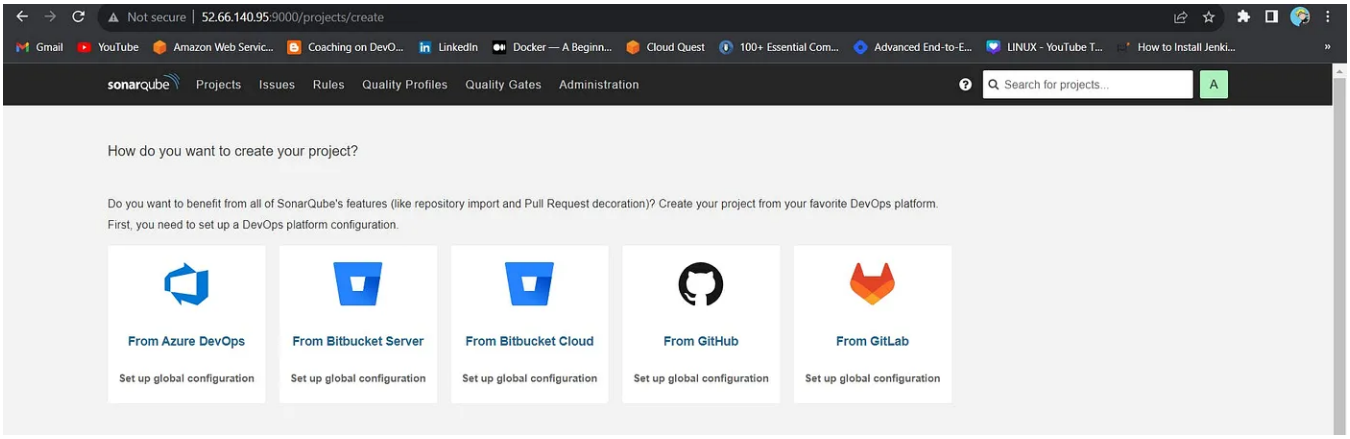


Enter username and password, click on login and change password

```
username admin
password admin
```



Update New password, This is Sonar Dashboard.



2C — Install Trivy

```
vi trivy.sh
```

```
sudo apt-get install wget apt-transport-https gnupg lsb-release -y
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dea
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github
sudo apt-get update
sudo apt-get install trivy -y
```

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

Step 3 — Install Plugins like JDK, Sonarqube Scanner, NodeJS, OWASP Dependency Check

3A — Install Plugin

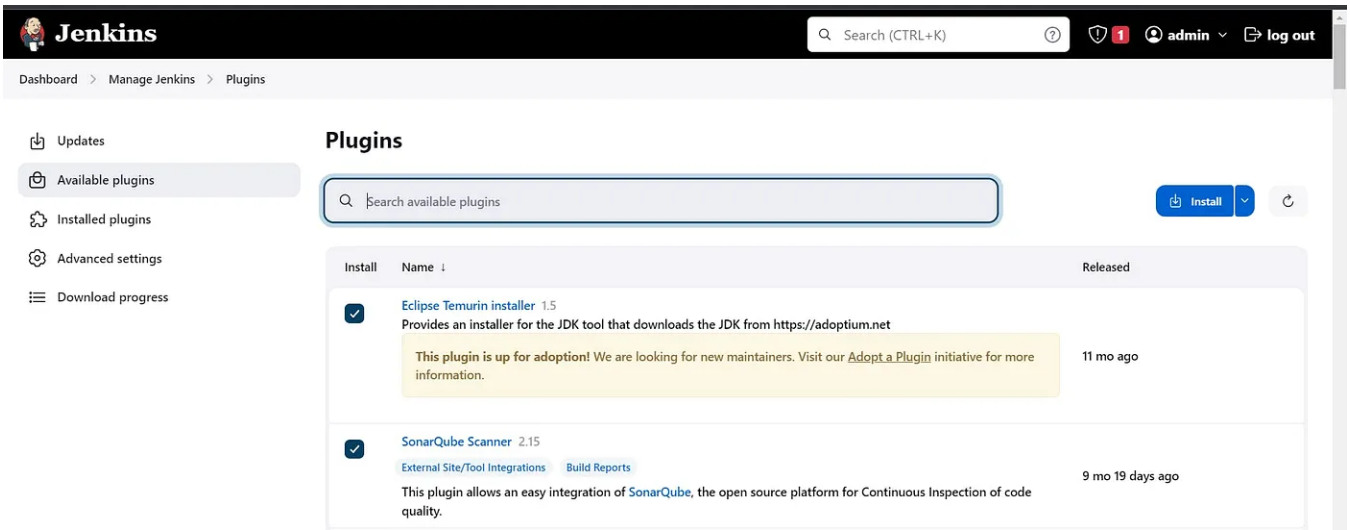
Go to Manage Jenkins →Plugins → Available Plugins →

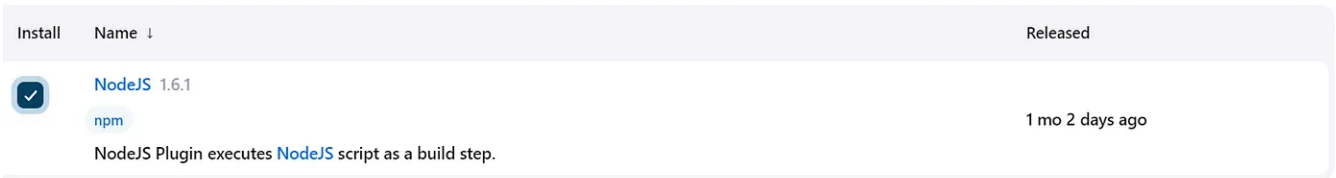
Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

2 → Sonarqube Scanner (Install without restart)

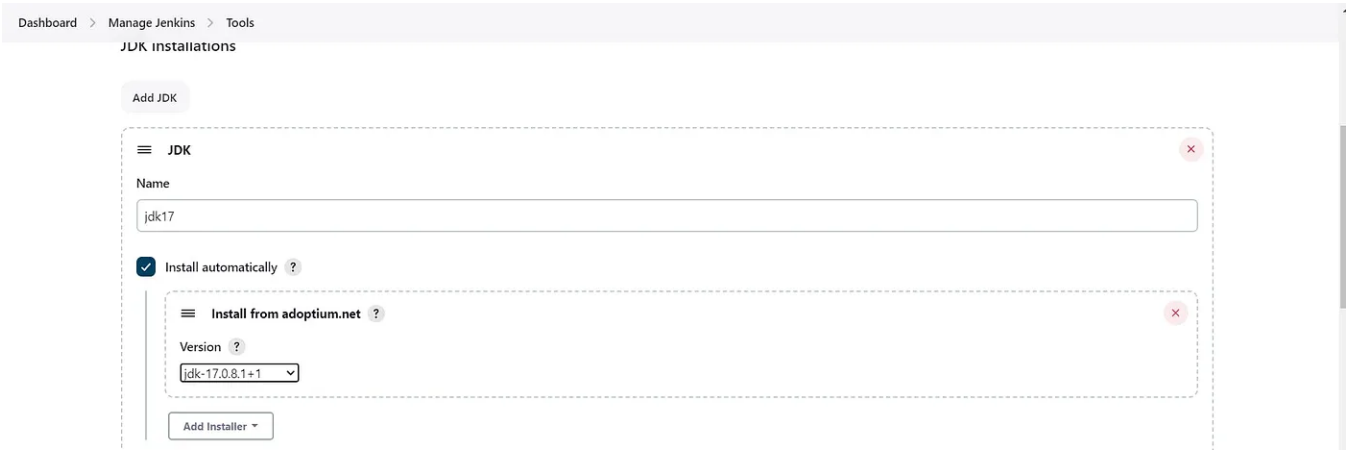
3 → NodeJS Plugin (Install Without restart)

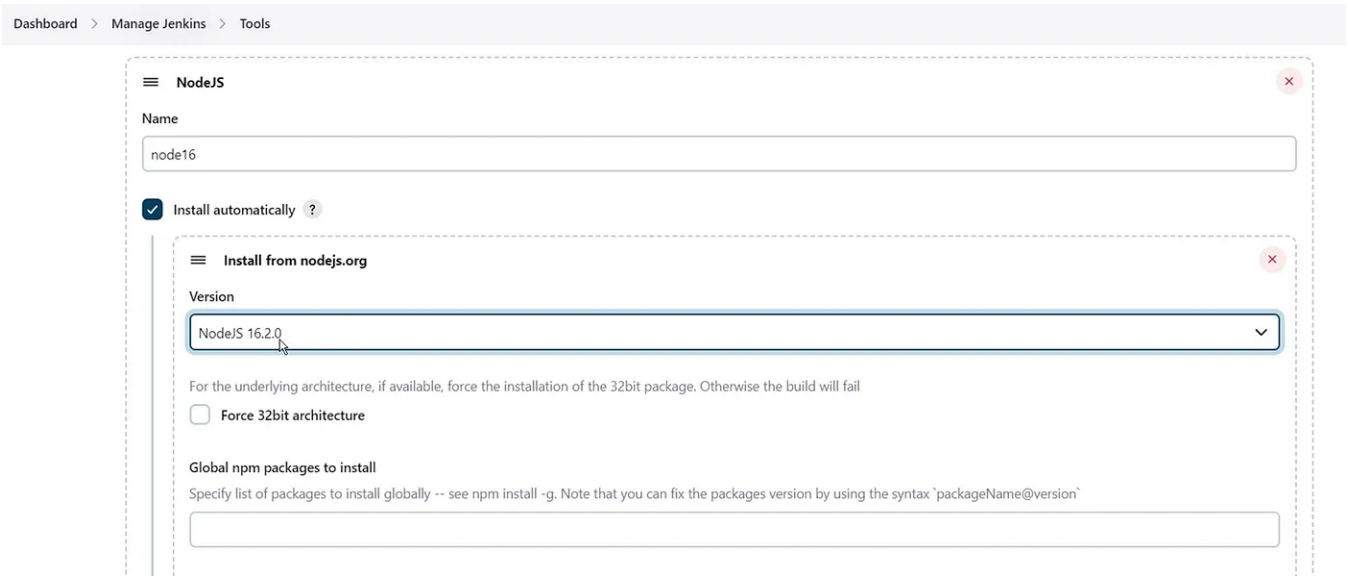




3B — Configure Java and Nodejs in Global Tool Configuration

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



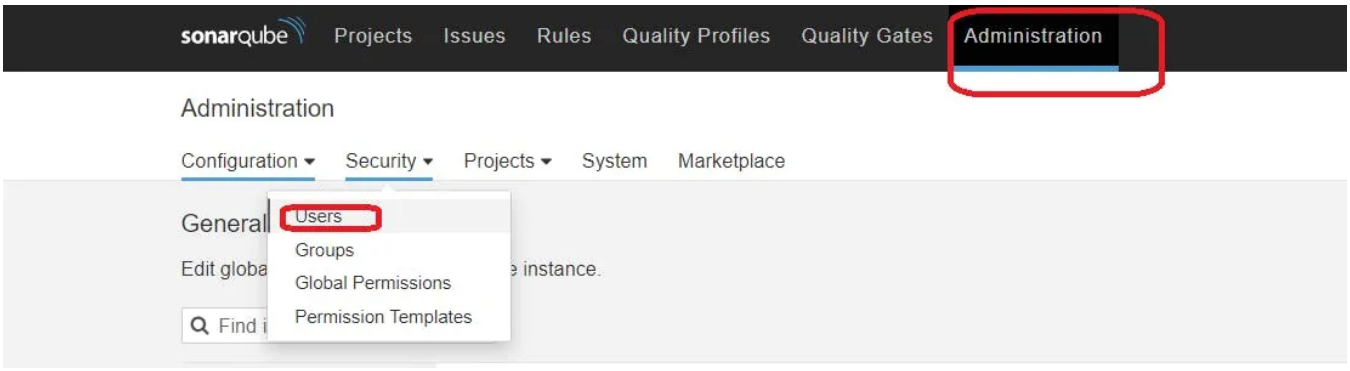


3C — Create a Job

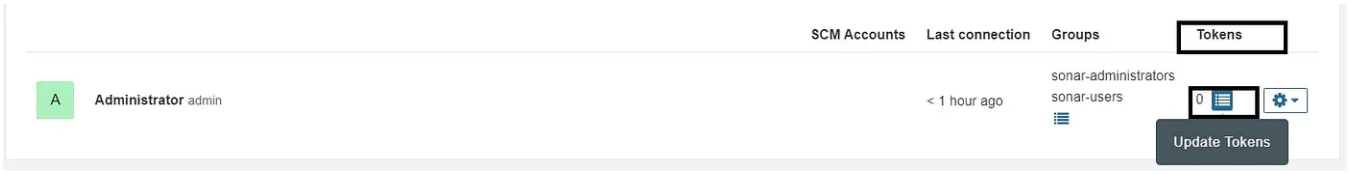
create a job as 2048 Name, select pipeline and click on ok.

Step 4 — Configure Sonar Server in Manage Jenkins

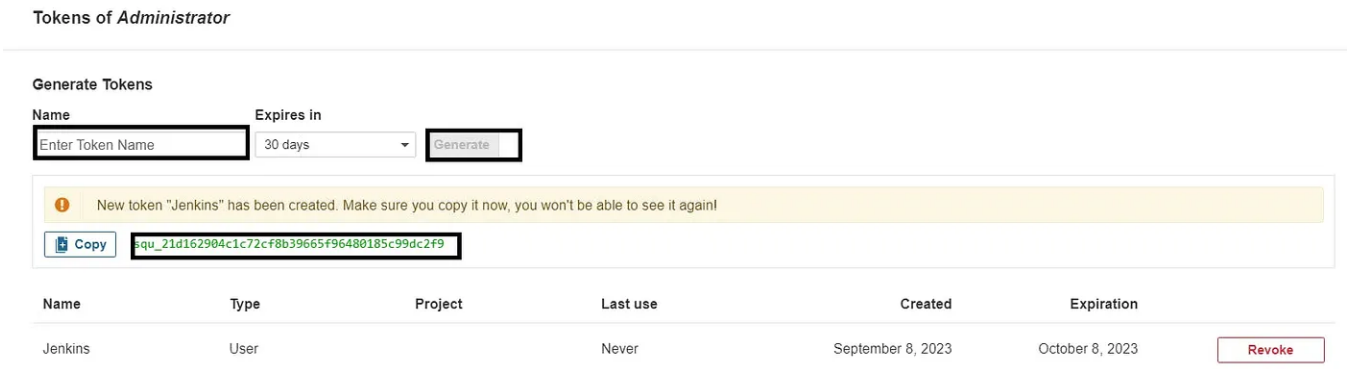
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

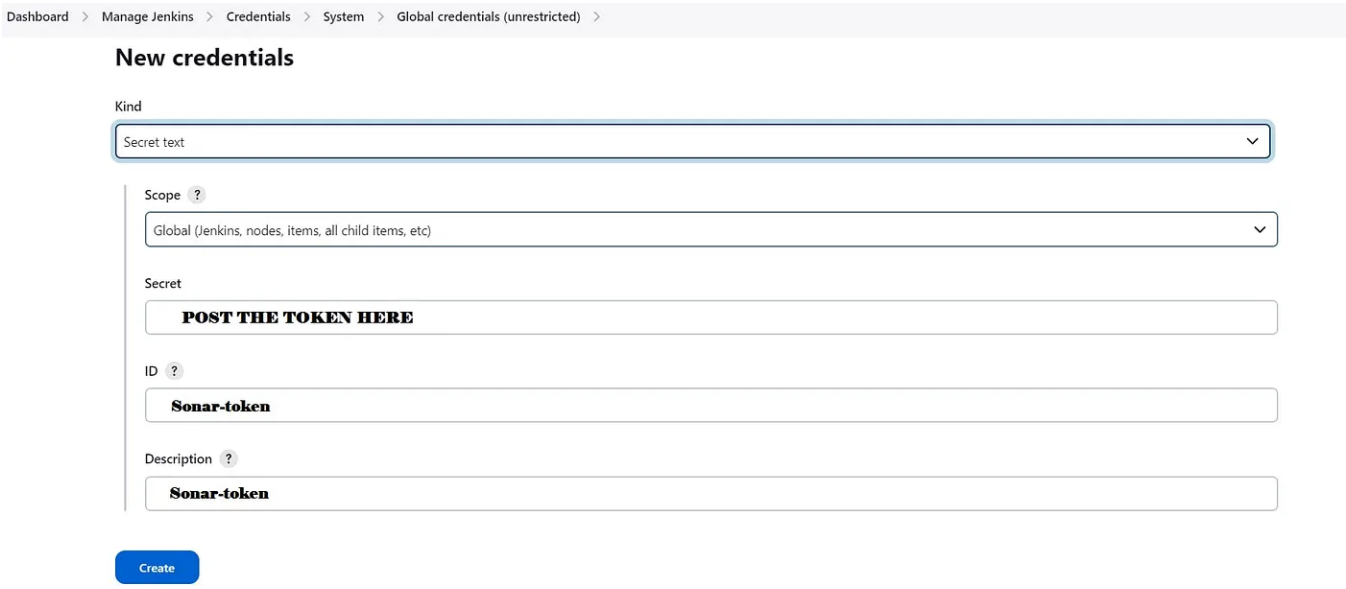


Create a token with a name and generate



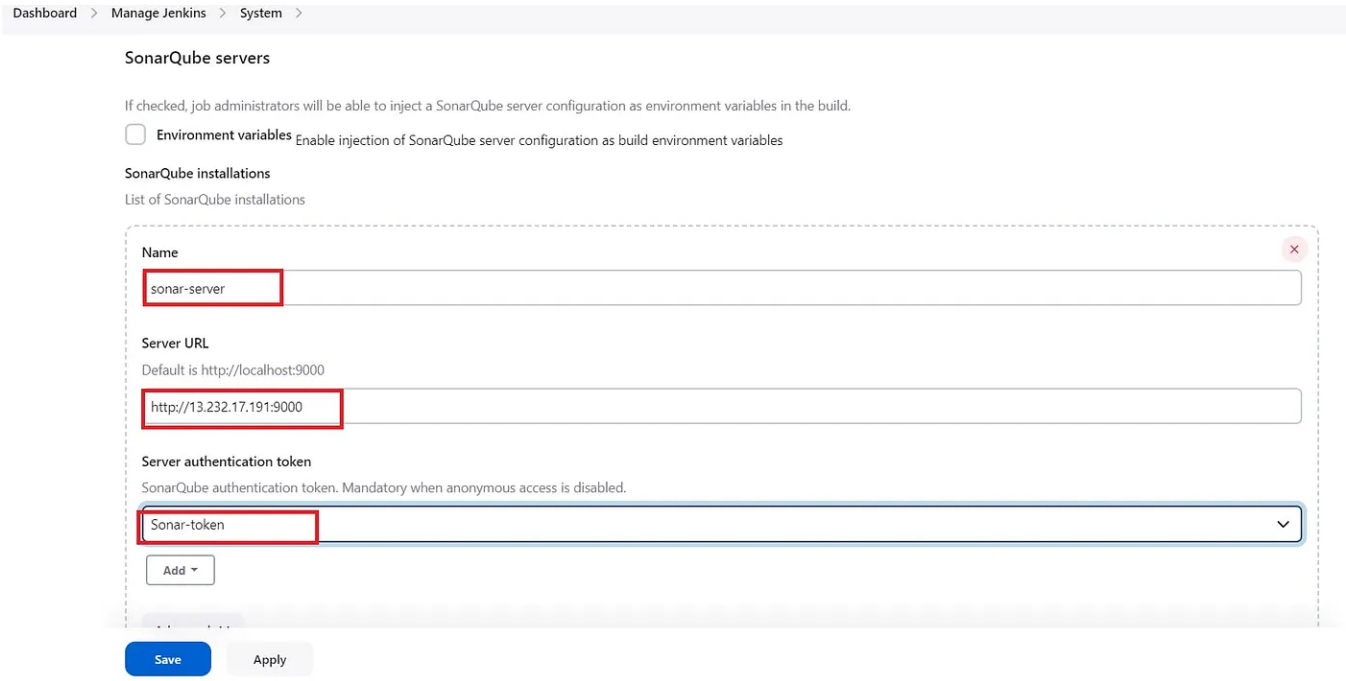
copy Token

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



You will this page once you click on create

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

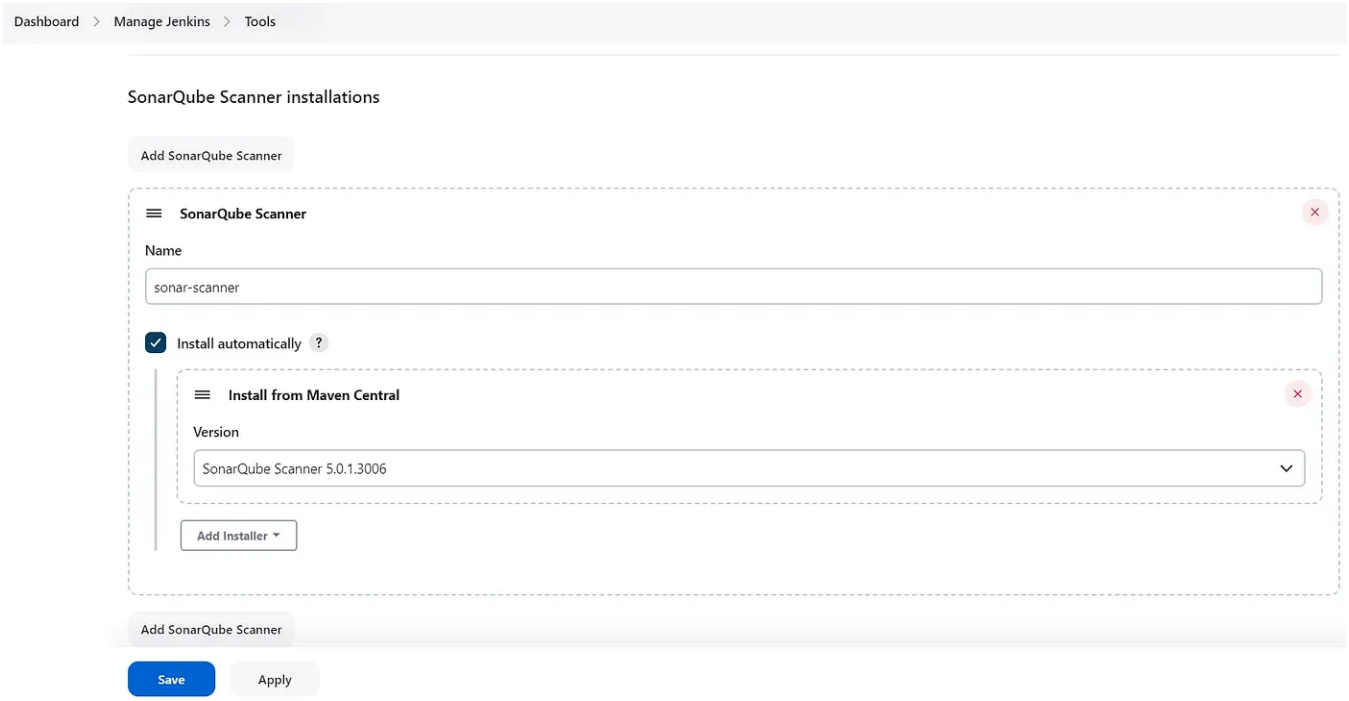


Click on Apply and Save

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

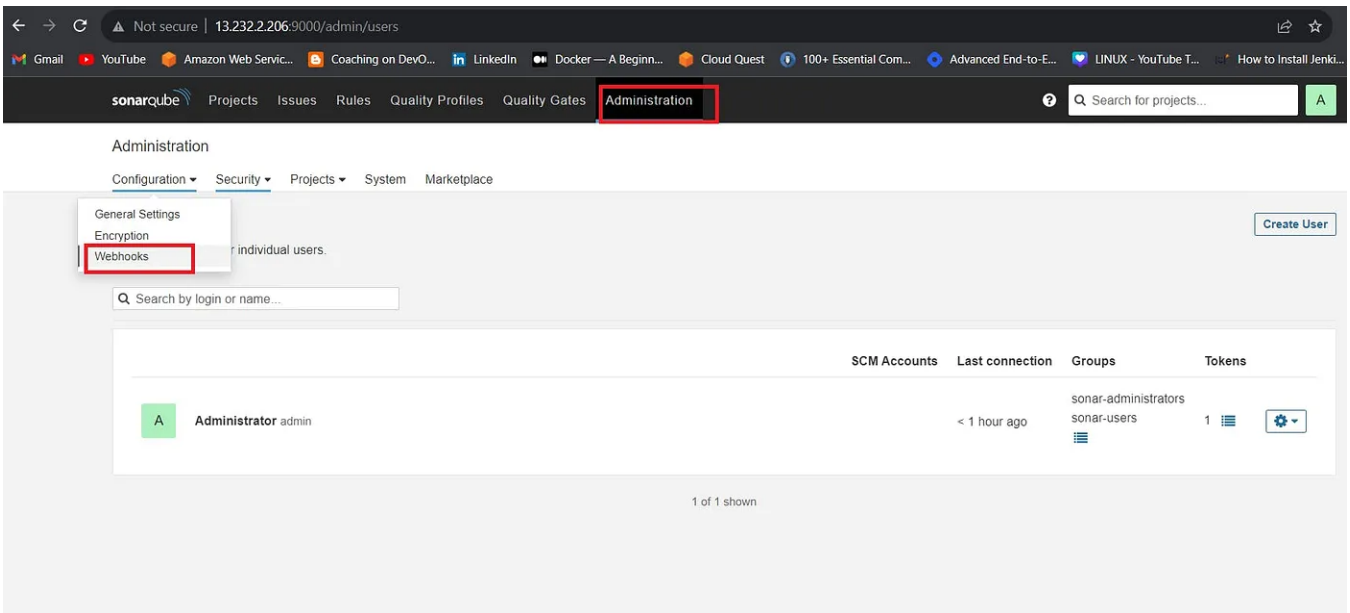
Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

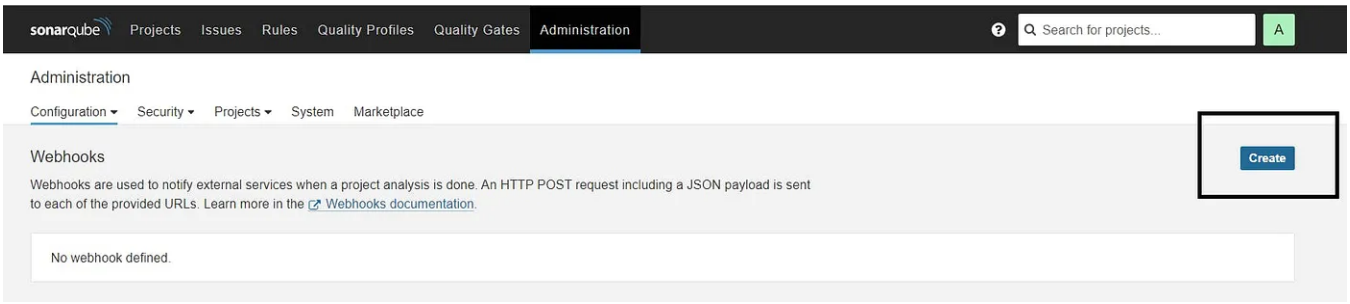


In the Sonarqube Dashboard add a quality gate also

Administration → Configuration →Webhooks

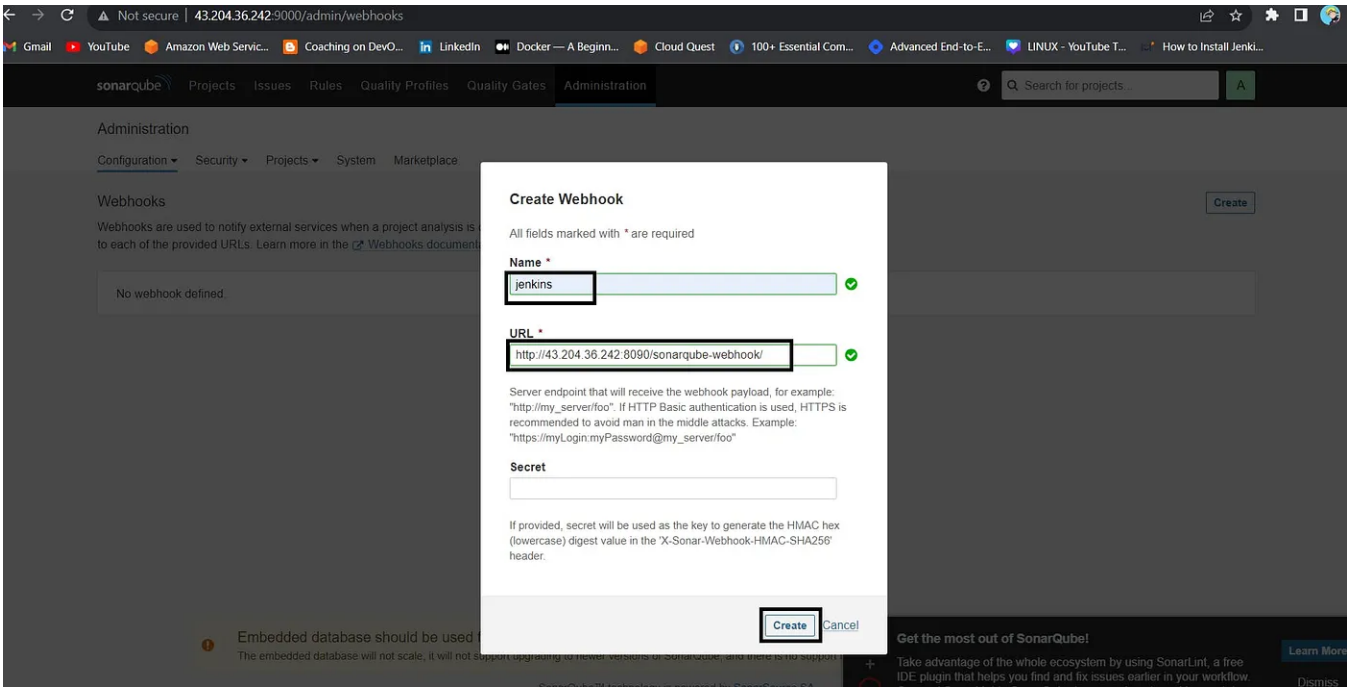


Click on Create



Add details

```
#in url section of quality gate
<http://jenkins-public-ip:8080>/sonarqube-webhook/
```

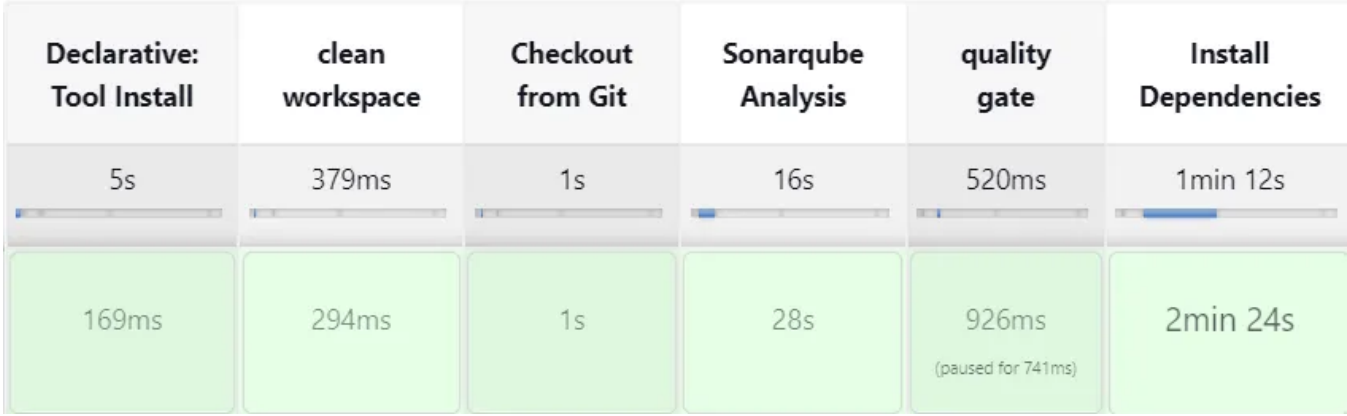


Let's go to our Pipeline and add the script in our Pipeline Script.

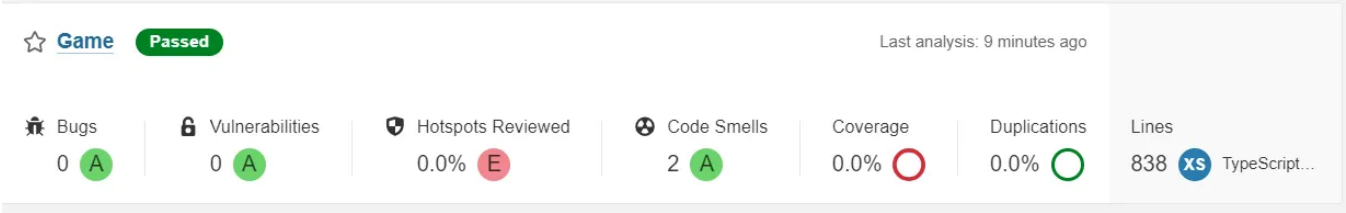
```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
```

```
        stage('clean workspace'){
            steps{
                cleanWs()
            }
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'master', url: 'https://github.com/Aj7Ay/2048-React
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=
                    -Dsonar.projectKey=Game '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId: 'Sc
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh "npm install"
            }
        }
    }
}
```

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



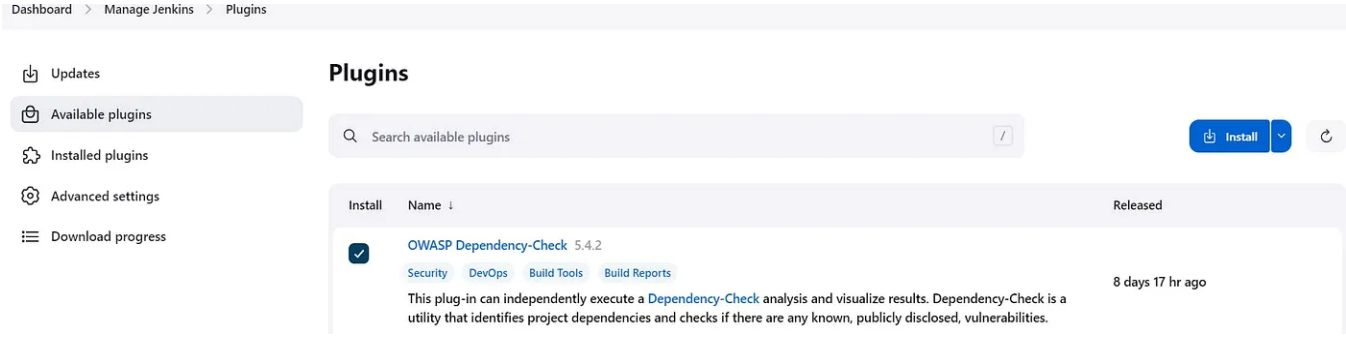
To see the report, you can go to Sonarqube Server and go to Projects.



You can see the report has been generated and the status shows as passed. You can see that there are 838 lines. To see a detailed report, you can go to issues.

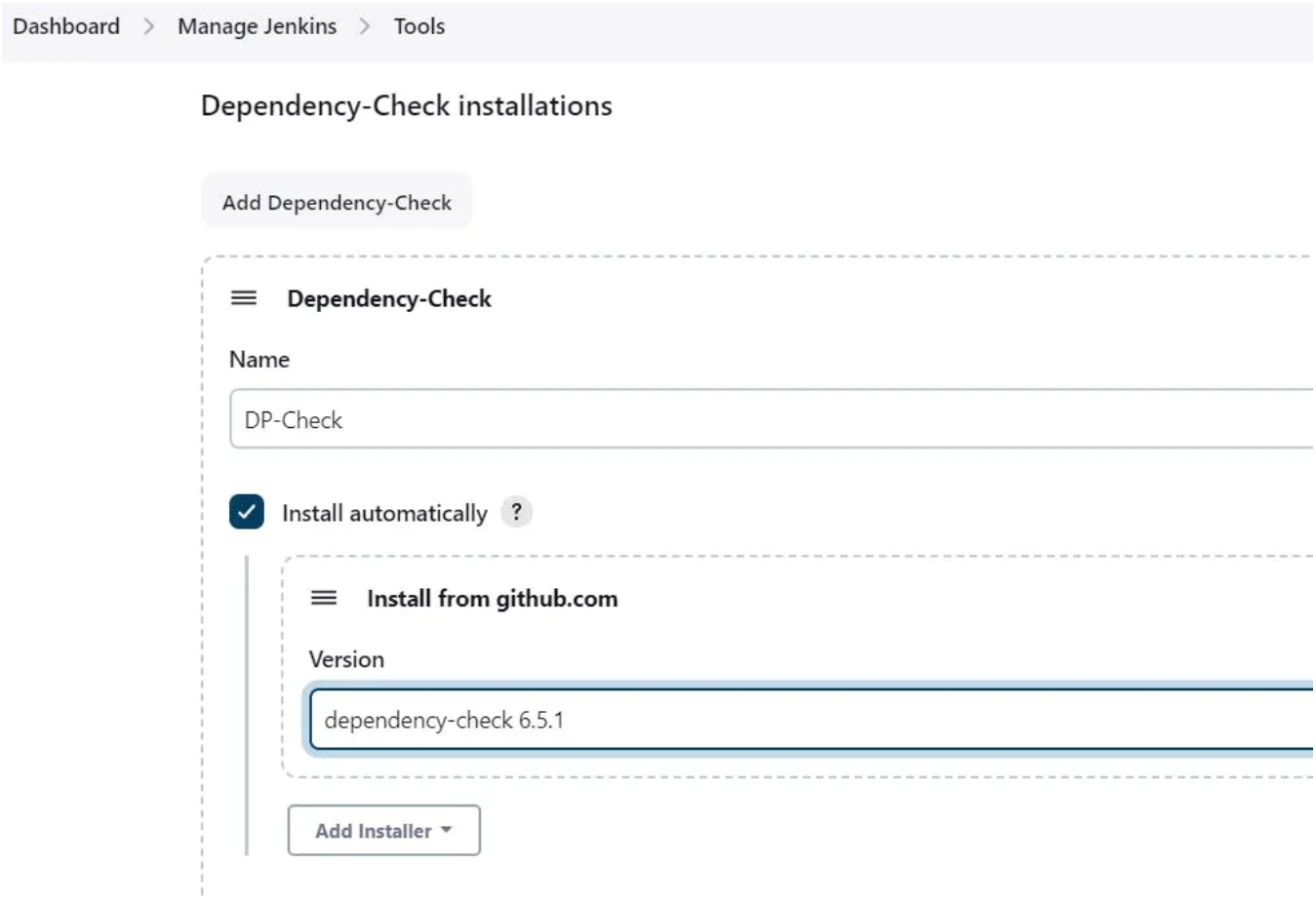
Step 5 — Install OWASP Dependency Check Plugins

Go to Dashboard → 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

Go to Dashboard → Manage Jenkins → Tools →



Click on Apply and Save here.

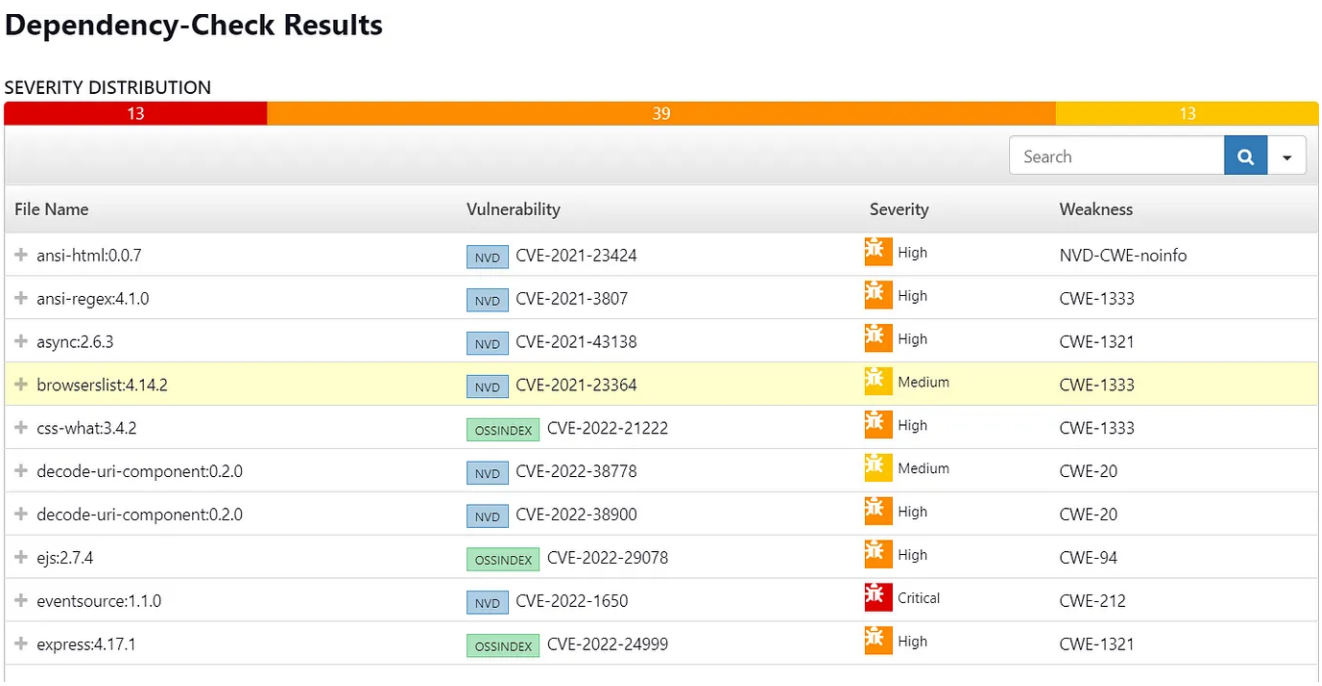
Now go configure → Pipeline and add this stage to your pipeline and build.

```
stage('OWASP FS SCAN') {
            steps {
                dependencyCheck additionalArguments: '--scan ./ --disableYarnAu
                dependencyCheckPublisher pattern: '**/dependency-check-report.x
            }
        }
        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.

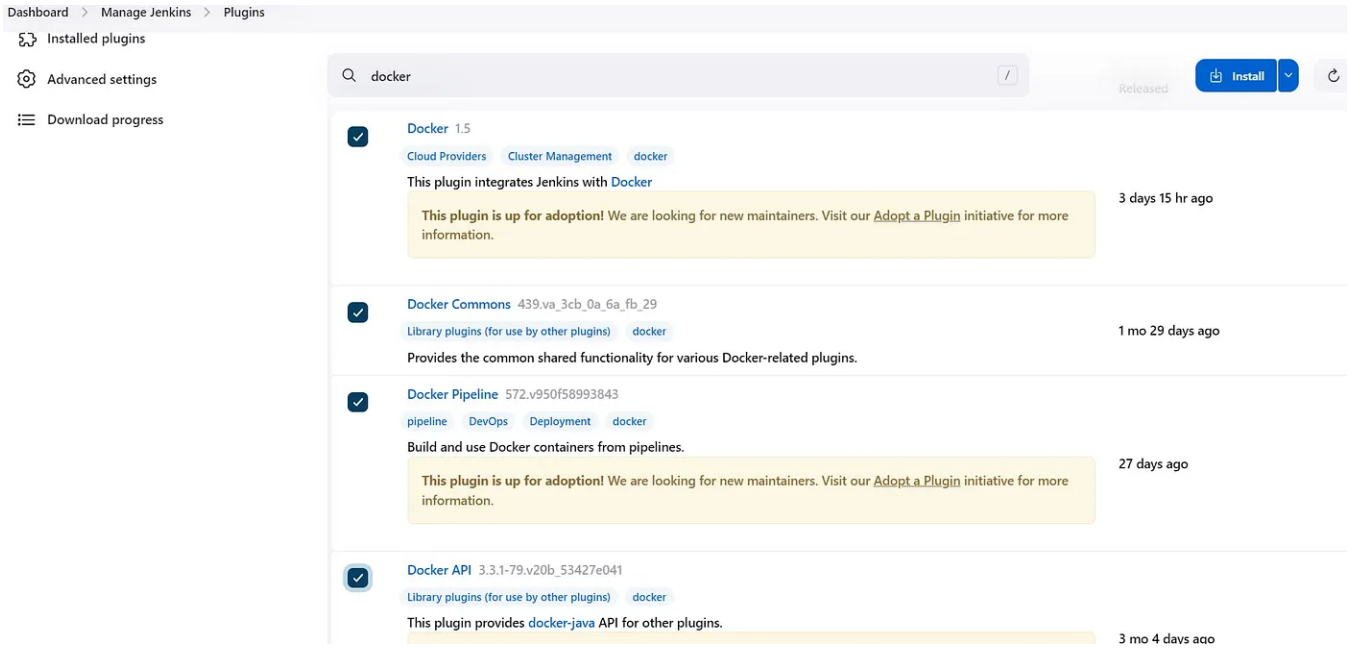


Step 6 — Docker Image Build and Push

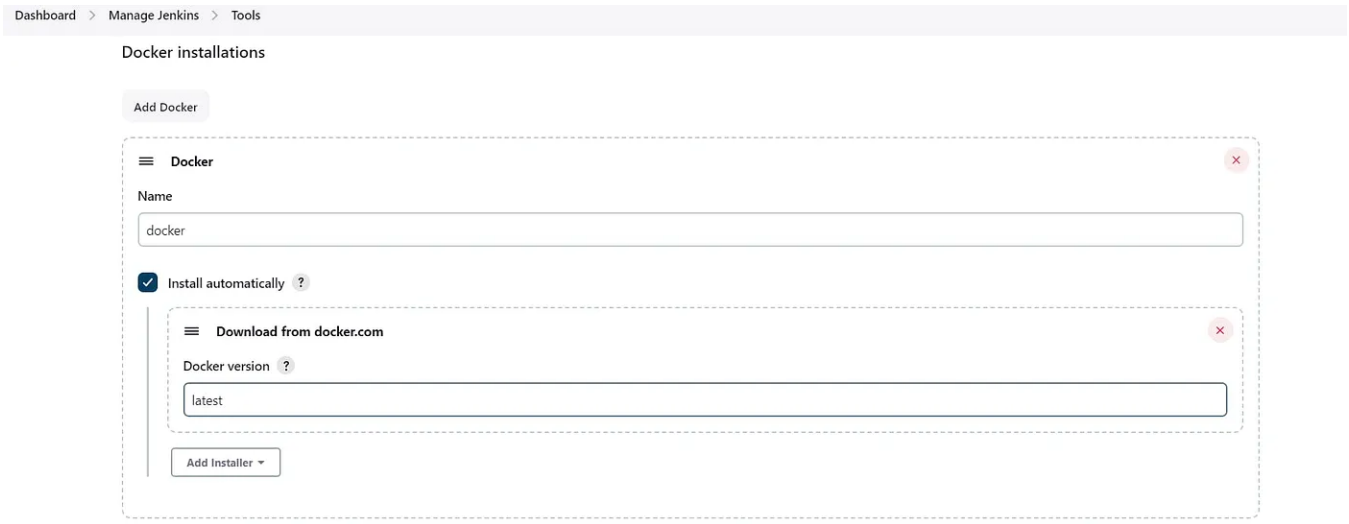
We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

```
Docker

Docker Commons

Docker Pipeline

Docker API

docker-build-step
```

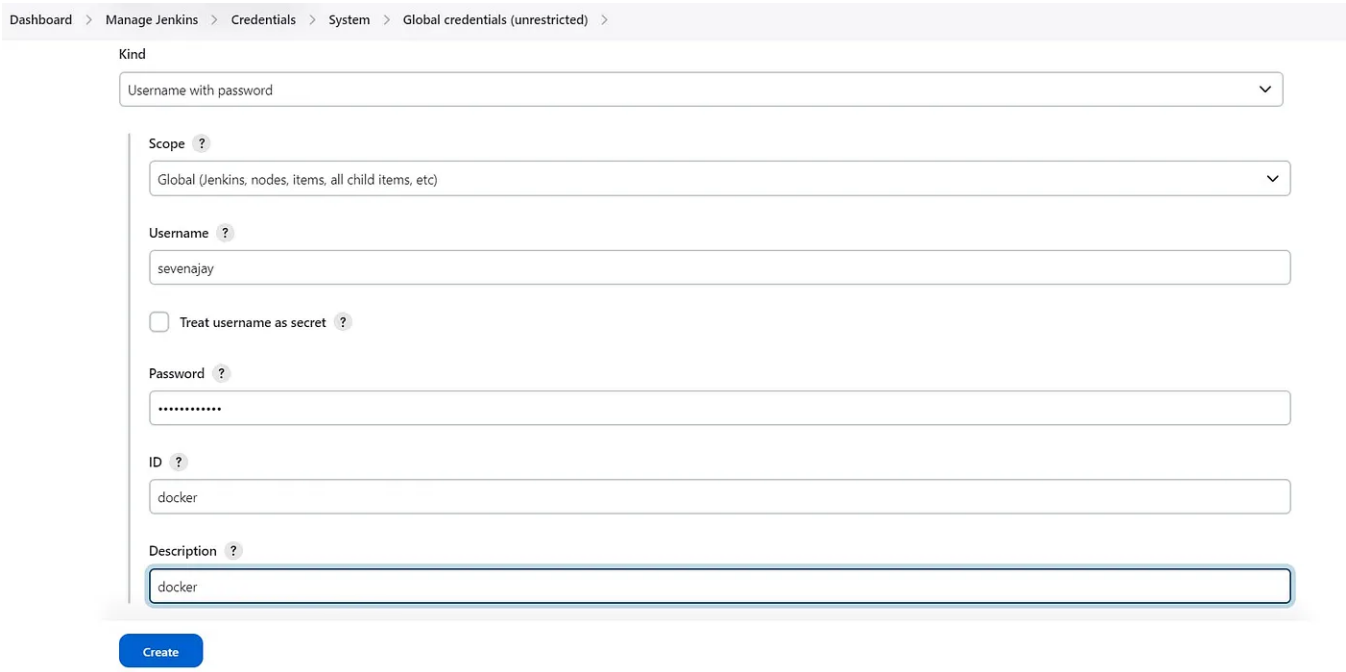
and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



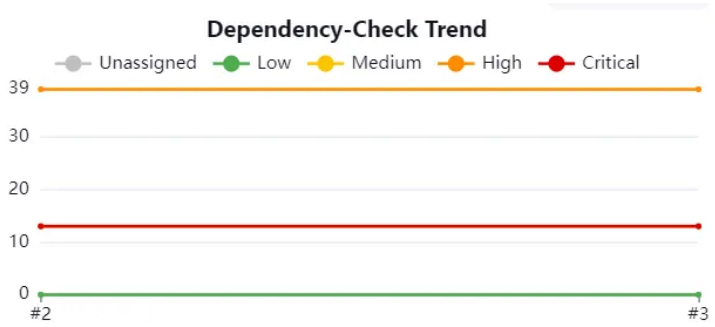
Add Docker Hub Username and Password under Global Credentials



Add this stage to Pipeline Script

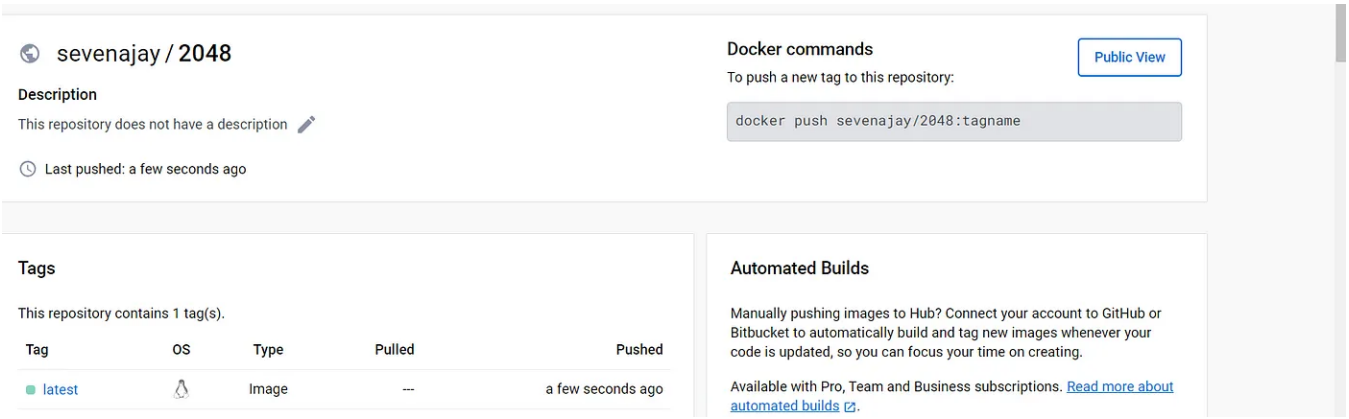
```
stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName: 'docke
                       sh "docker build -t 2048 ."
                       sh "docker tag 2048 sevenajay/2048:latest "
                       sh "docker push sevenajay/2048:latest "
                    }
                }
            }
        }
        stage("TRIVY"){
            steps{
                sh "trivy image sevenajay/2048:latest > trivy.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 Docker hub, you will see a new image is created



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

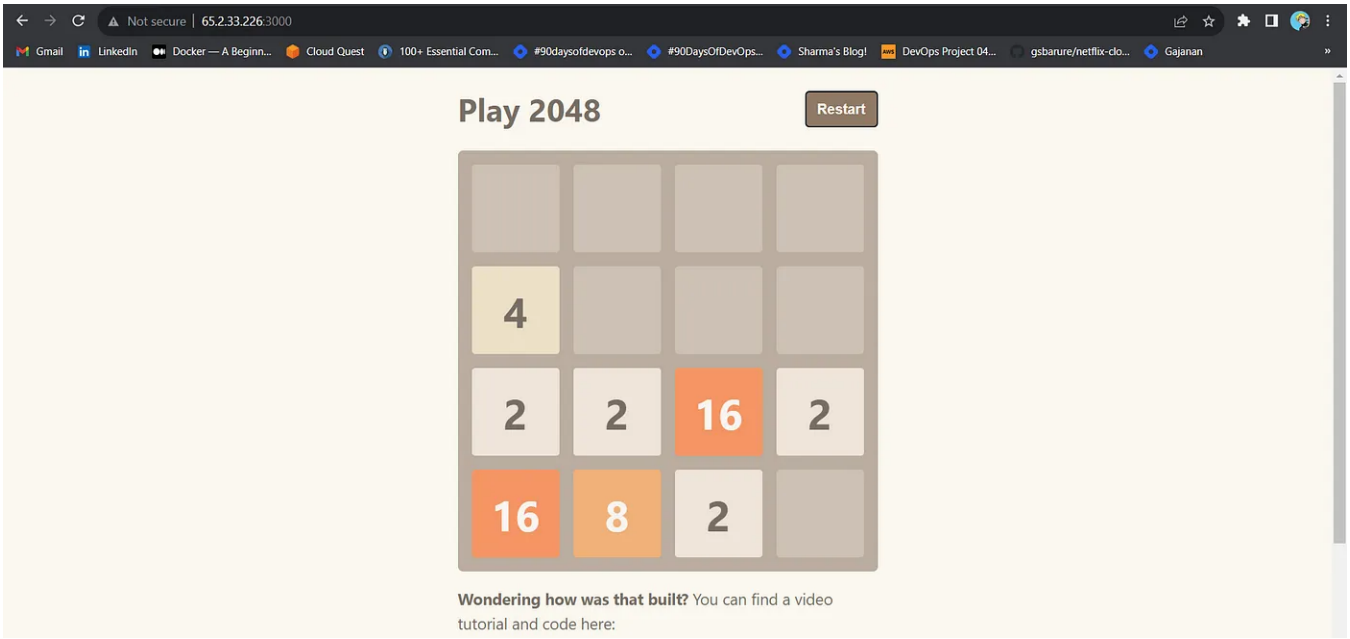
```
stage('Deploy to container'){
            steps{
                sh 'docker run -d --name 2048 -p 3000:3000 sevenajay/2048:lates
            }
        }
```

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:3000>
```

You will get this output



Play the game and make it 2048

Step 8 — 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 -y
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stabl
```

```
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
sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-k

sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main     # 3lines same commanc
EOF

sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo snap install kube-apiserver
```

Part 3 — — — — — — — — Master — — — — — — — -

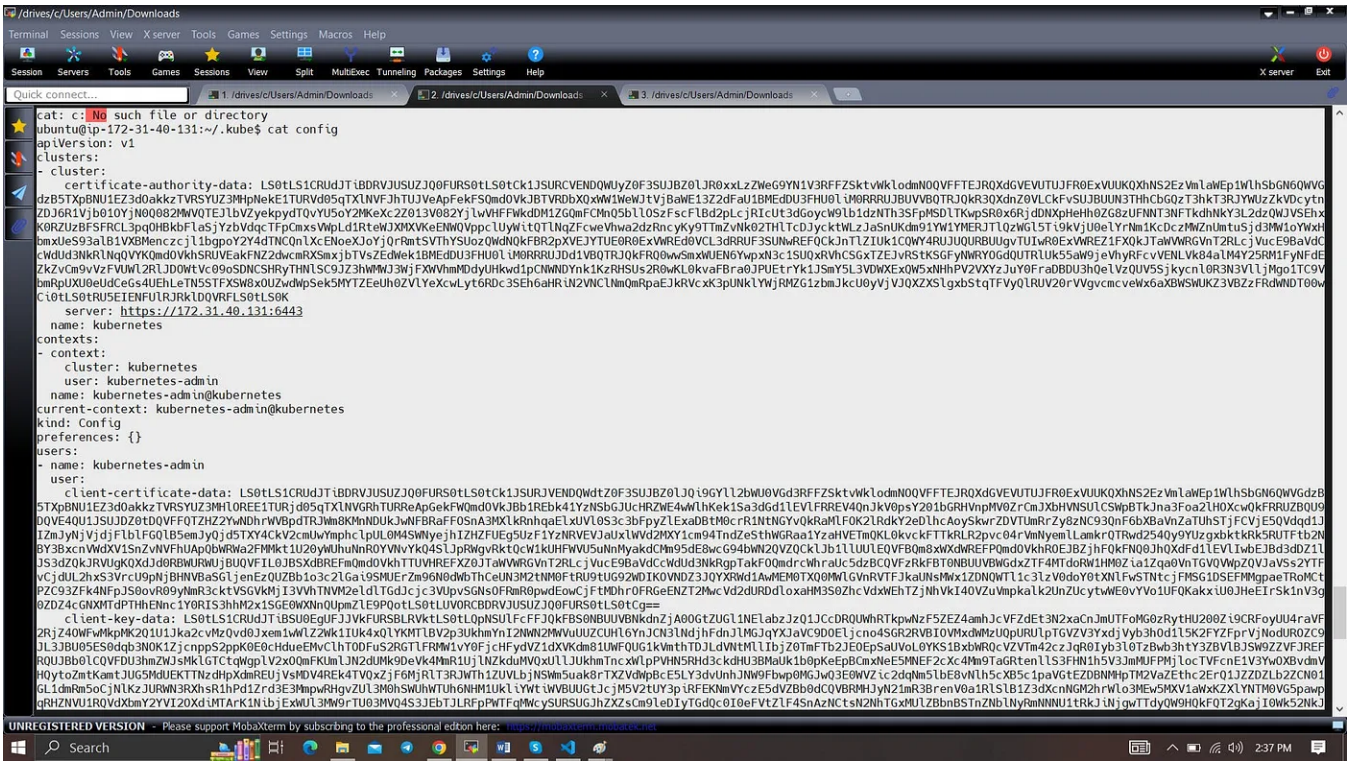
```
sudo kubeadm init --pod-network-cidr=10.244.0.0/16
# in case your in root exit from it and run below commands
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
sudo chown $(id -u):$(id -g) $HOME/.kube/config
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Docume
```

— — — — — Worker Node — — — — — —

```
sudo kubeadm join <master-node-ip>:<master-node-port> --token <token> --discove
```

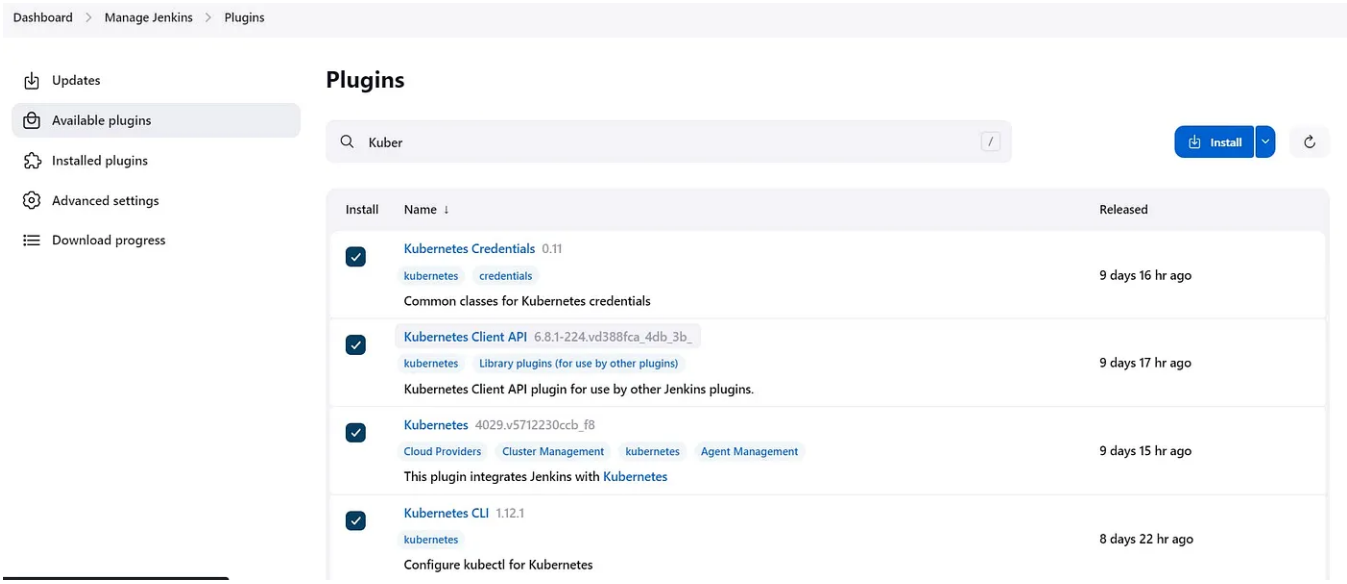
Copy the config file to Jenkins master or the local file manager and save it



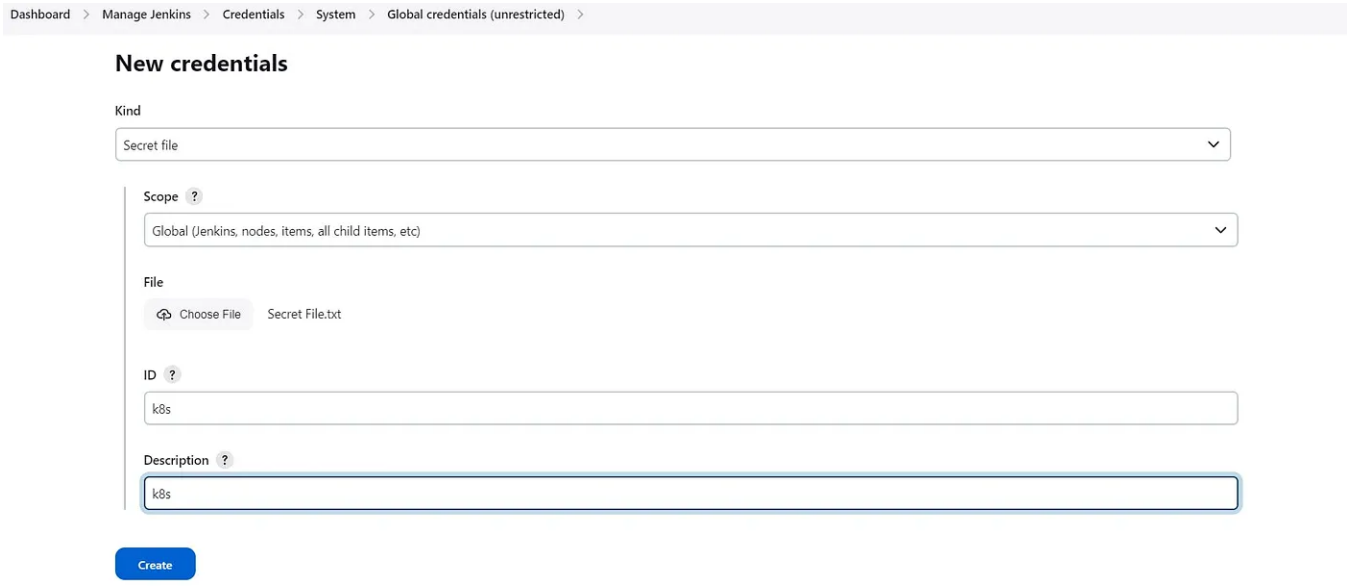
copy it and save it in documents or another folder save it as secret-file.txt

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

Install Kubernetes Plugin, Once it’s installed successfully



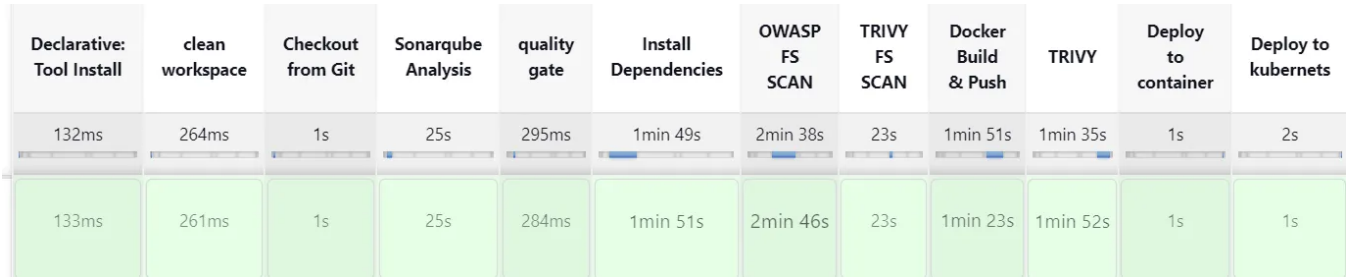
go to manage Jenkins → manage credentials → Click on Jenkins global → add credentials



final step to deploy on the Kubernetes cluster

```
stage('Deploy to kubernets'){
            steps{
                script{
                    withKubeConfig(caCertificate: '', clusterName: '', contextN
                       sh 'kubectl apply -f deployment.yaml'
                  }
                }
            }
        }
```

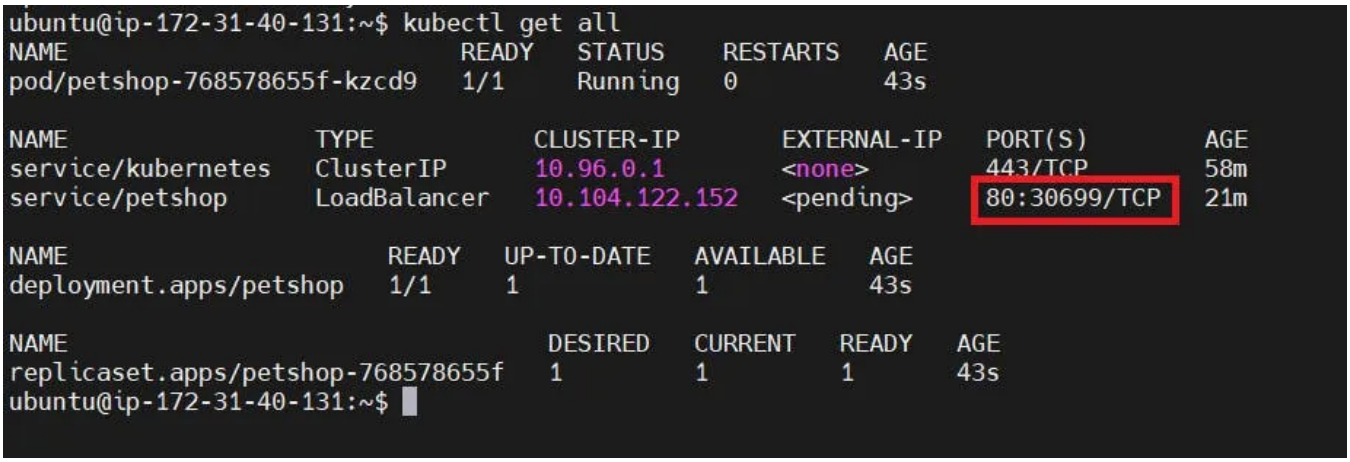
stage view



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

In the Kubernetes cluster give this command

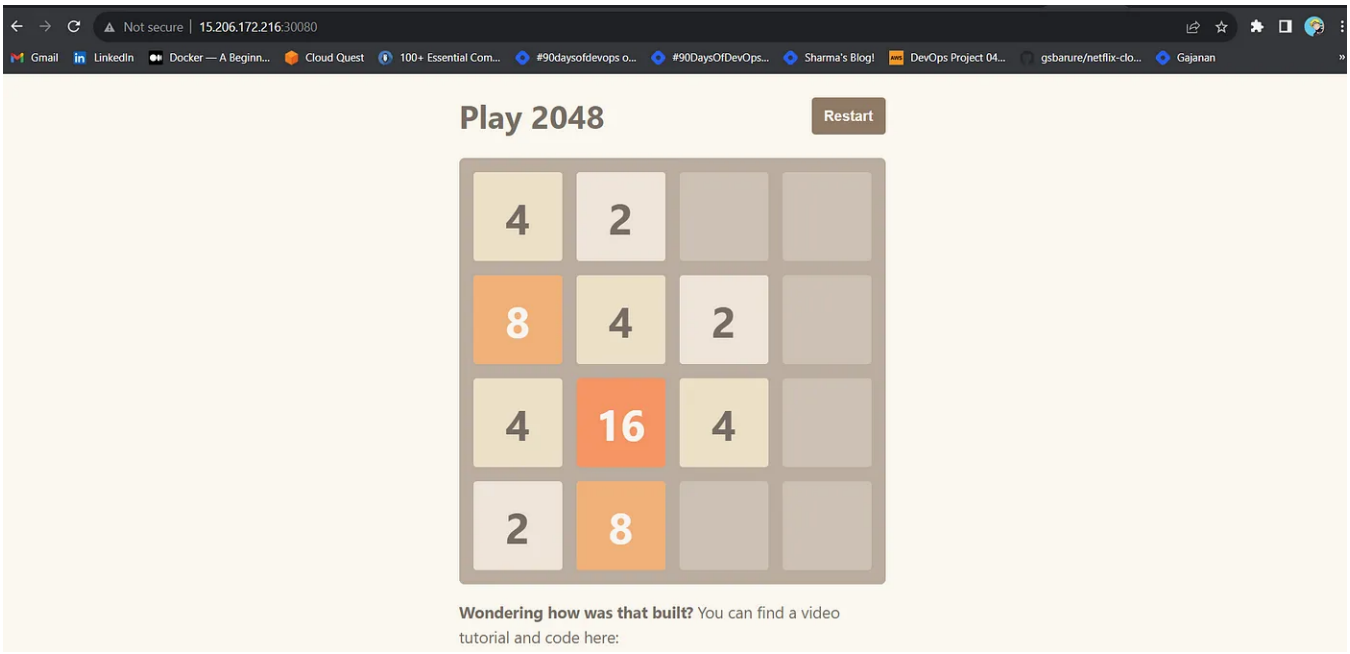
```
kubectl get all
kubectl get svc #use anyone
```



STEP9:Access from a Web browser with

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

output:



Step 10: Terminate instances.