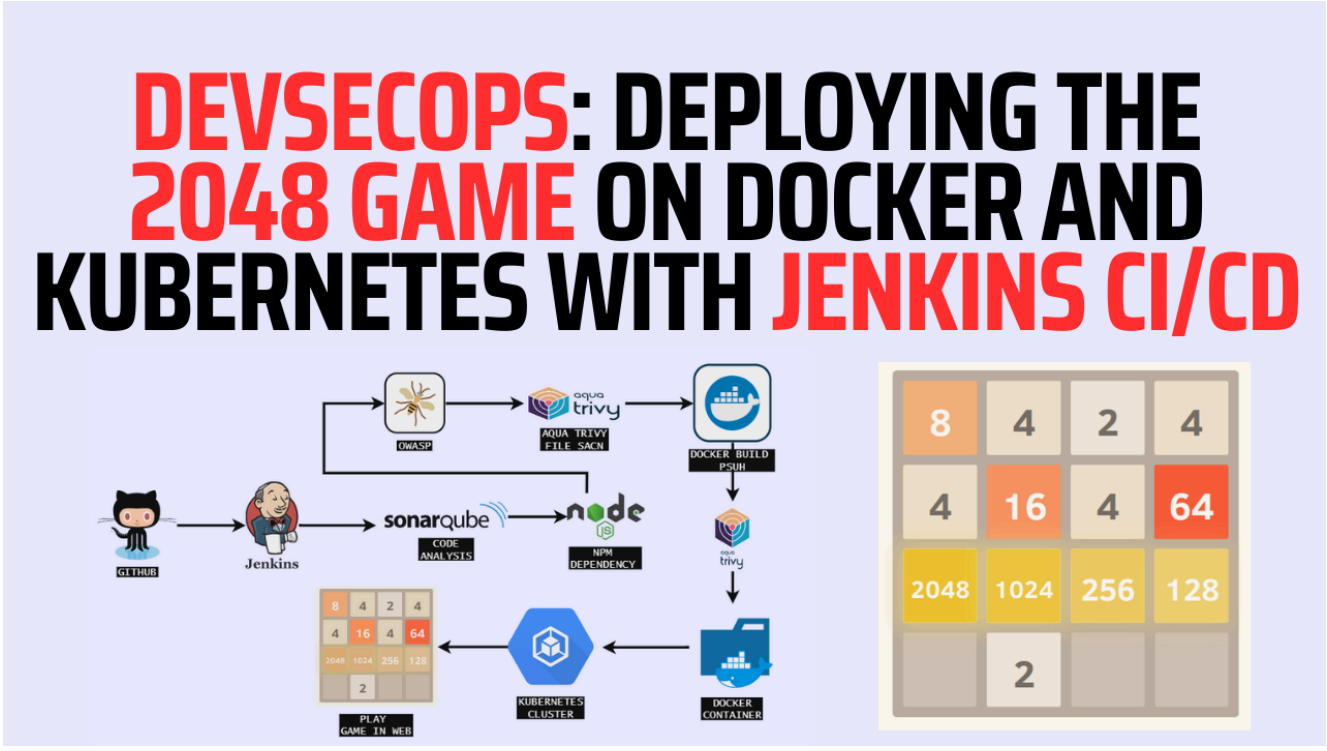
DevOps

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

mrcloudbook.com · 8 January 2024



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.

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 – 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.

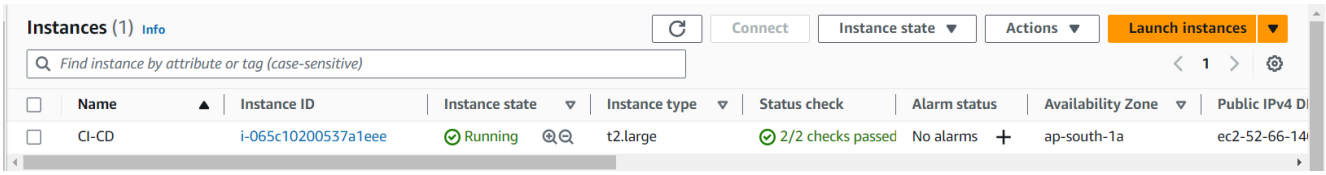
Now, let's get started and dig deeper into each of these steps:-

Contents [hide]

STEP1:Launch an Ubuntu(22.04) T2 Large Instance
Step 2 – Install Jenkins, Docker and Trivy
2A – To Install Jenkins
2B – Install Docker
2C – Install Trivy
Step 3 – Install Plugins like JDK, Sonarqube Scanner, NodeJs, OWASP Dependency Check
3A – Install Plugin
3B – Configure Java and Nodejs in Global Tool Configuration
3C – Create a Job
Step 4 – Configure Sonar Server in Manage Jenkins
Step 5 – Install OWASP Dependency Check Plugins
Step 6 – Docker Image Build and Push
Step 8 – Kuberenetes Setup
Kubectl is to be installed on Jenkins also
Part 1 ––––Master Node––––
––––Worker Node––––
Part 2 ––––Both Master & Node ––––
Part 3 ––––– Master –––––
––––Worker Node––––
STEP9:Access from a Web browser with
Step 10: Terminate instances.
Complete Pipeline

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
```

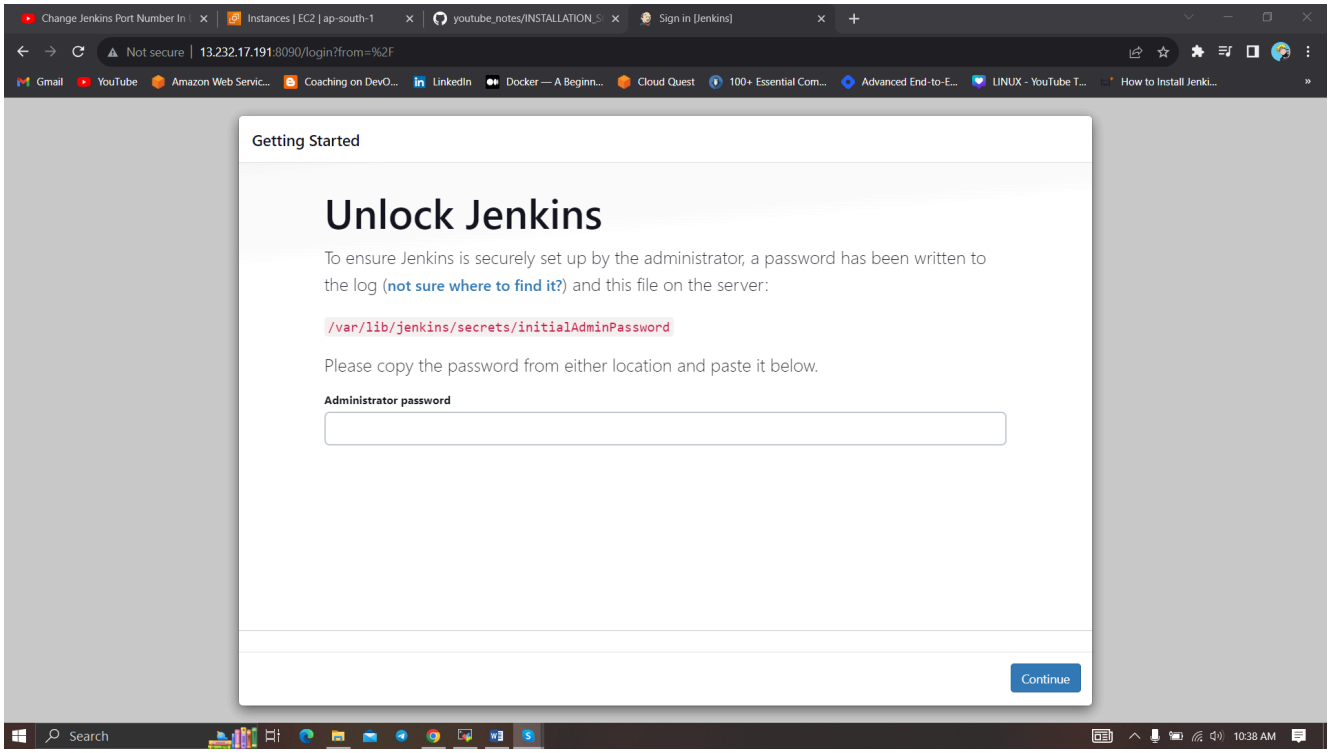
```
#!/bin/bash
sudo apt update -y
#sudo apt upgrade -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.ad
sudo apt update -y
sudo apt install temurin-17-jdk -y
/usr/bin/java --version
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key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

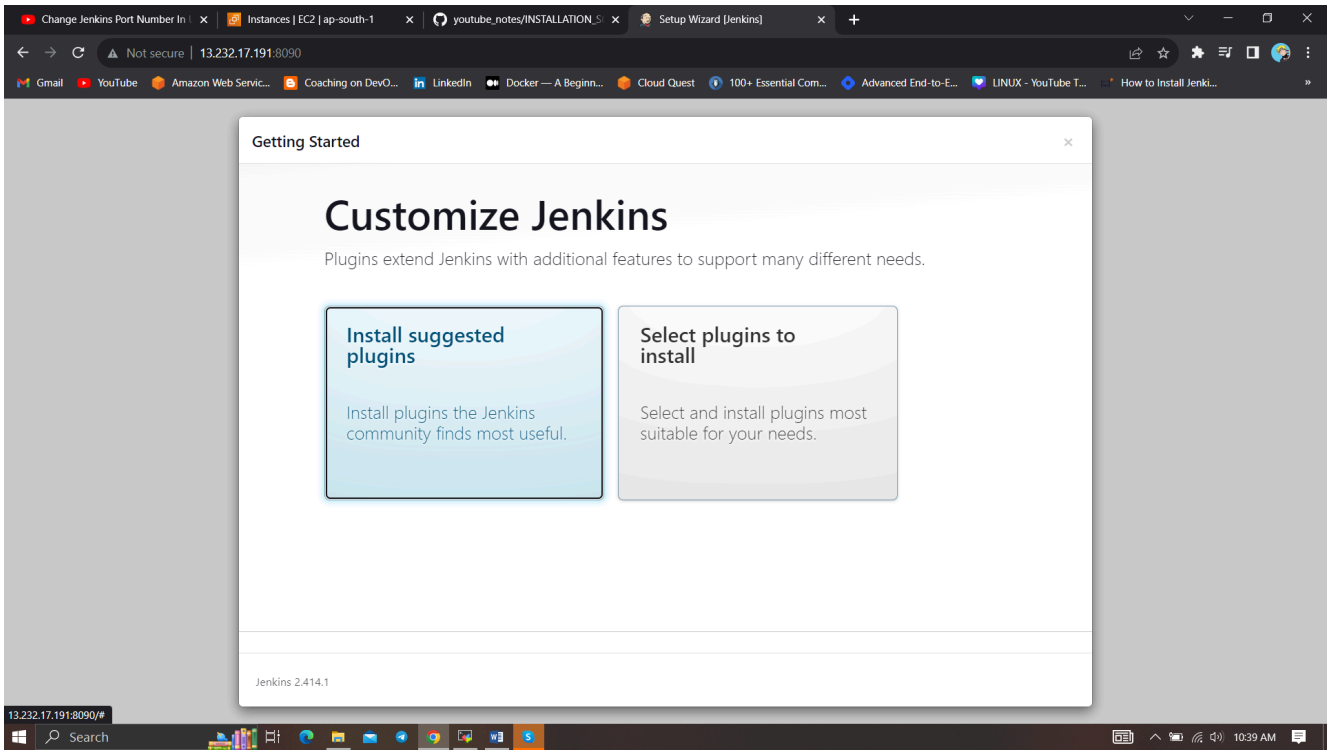
Once Jenkins is installed, you will need to go to your AWS EC2 Security Group and open Inbound Port 8080, since Jenkins works on Port 8080.

Now, grab your Public IP Address

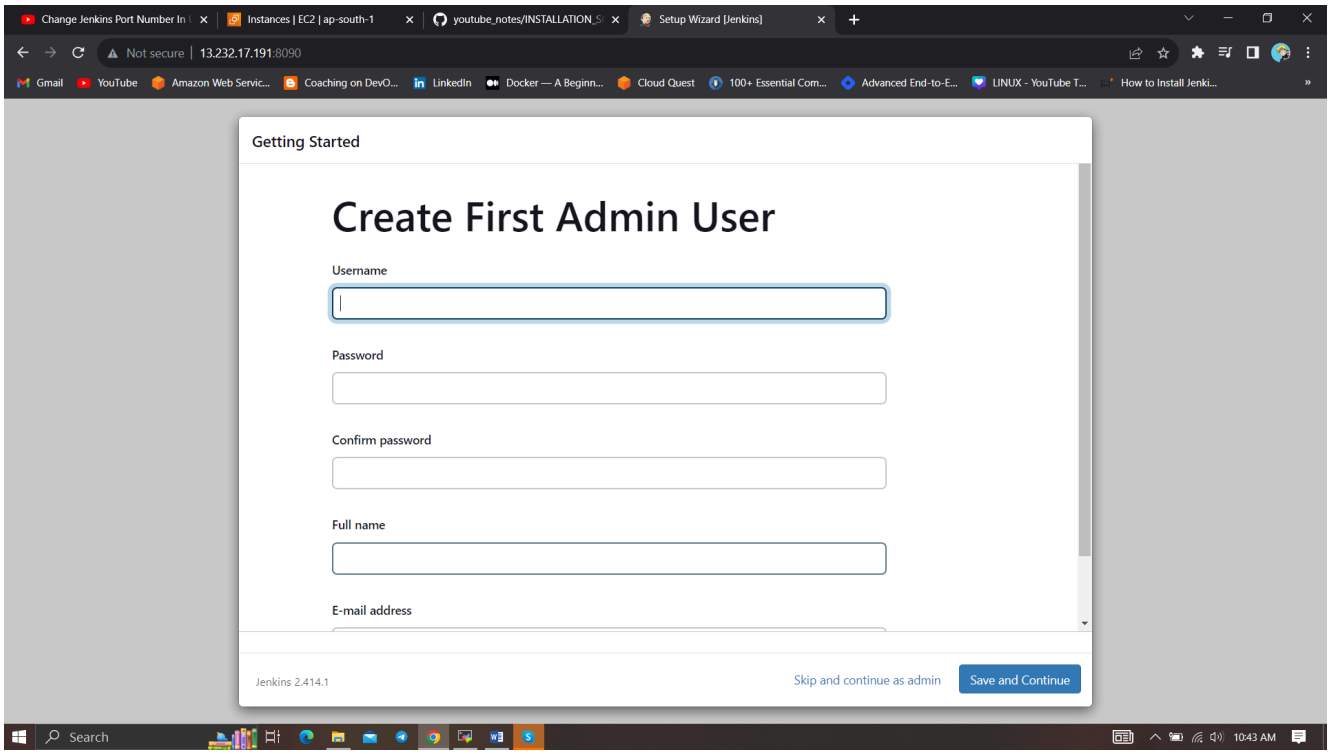
```
EC2 Public IP Address:8080
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



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

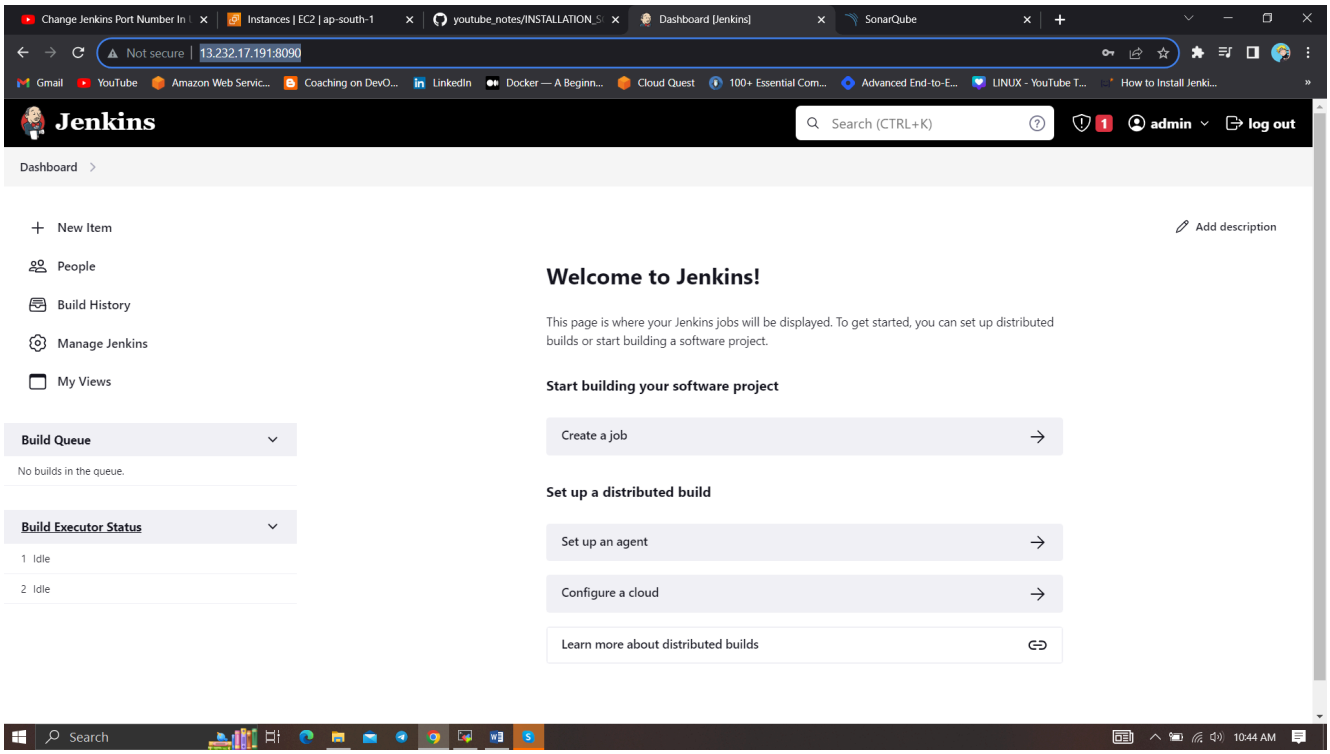


Jenkins will now get installed and install all the libraries.



Create a user click on save and continue.

Jenkins Getting Started Screen.



2B – Install Docker

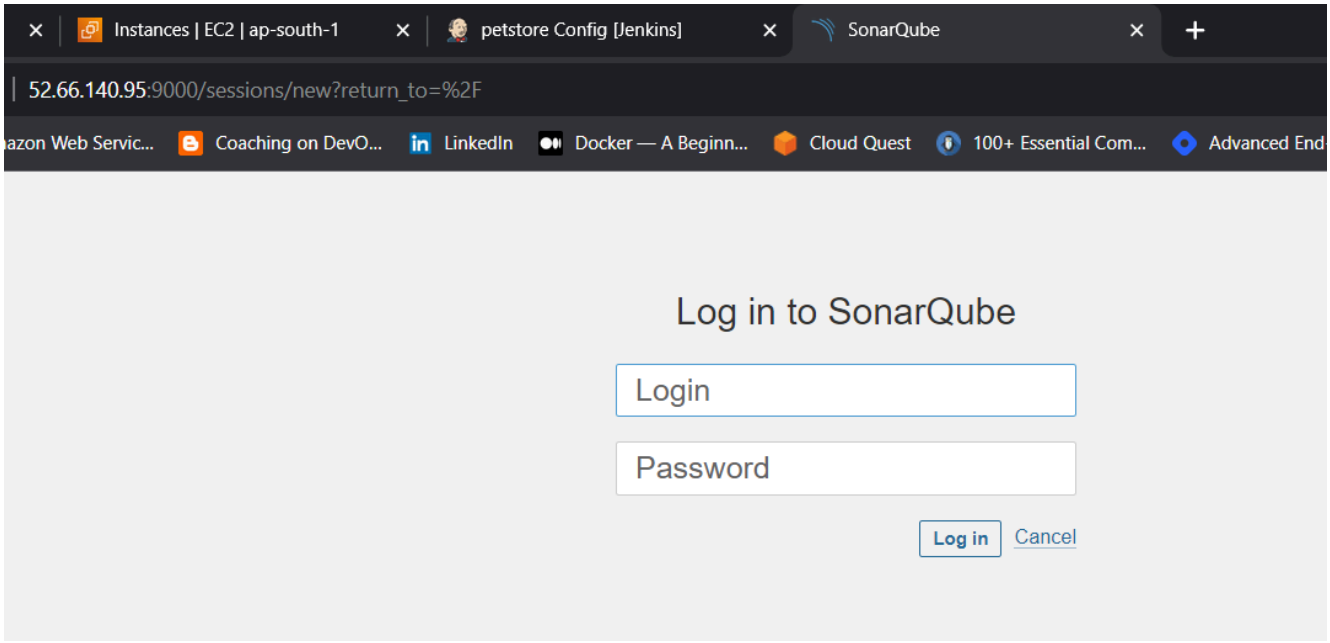
```
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER
newgrp docker
sudo chmod 777 /var/run/docker.sock
```

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
```

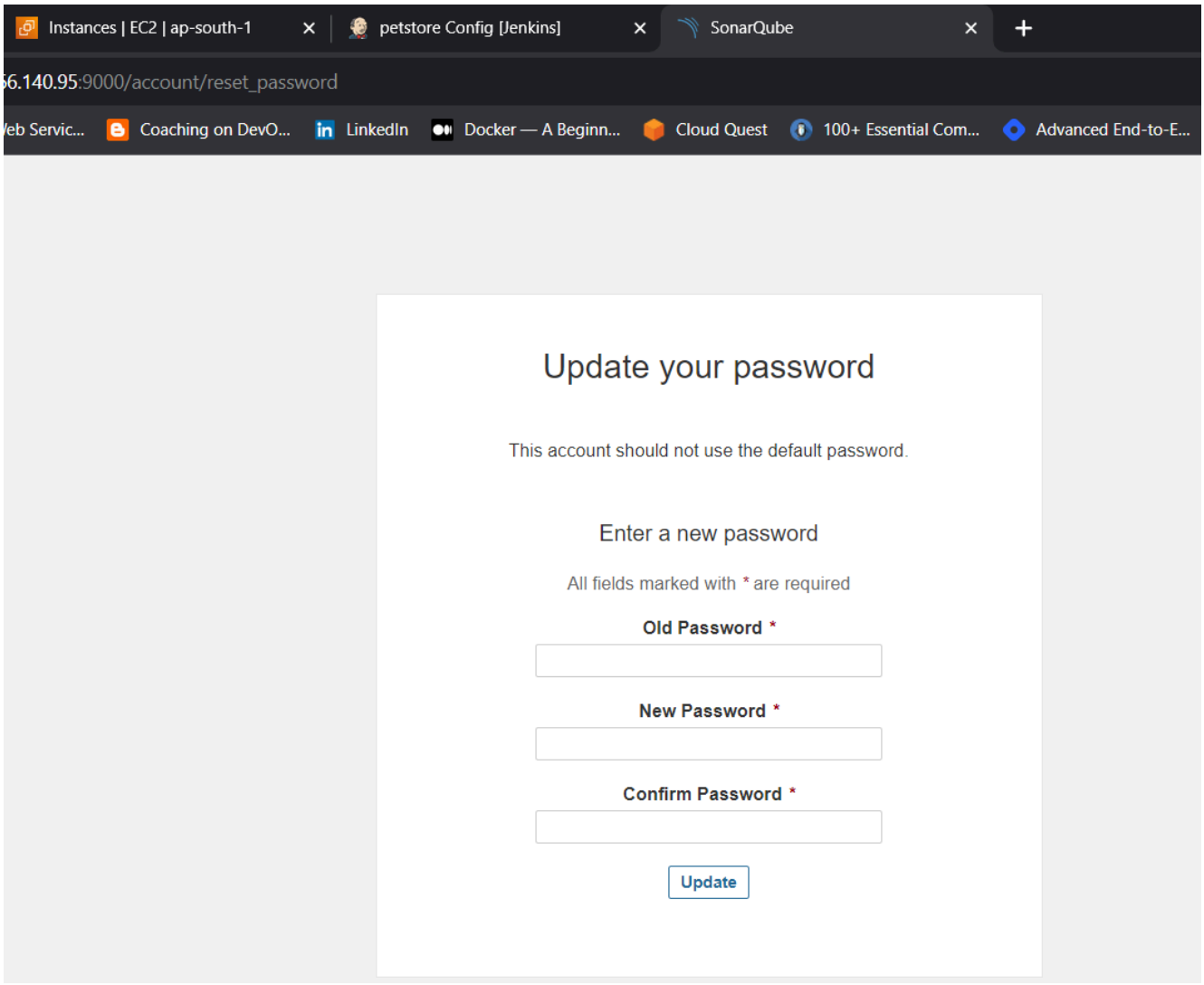
```
ubuntu@ip-172-31-42-253:~$ sudo chmod 777 /var/run/docker.sock
ubuntu@ip-172-31-42-253:~$ docker run -d --name sonar -p 9000:9000 sonarqube:lts-community
Unable to find image 'sonarqube:lts-community' locally
lts-community: Pulling from library/sonarqube
44ba2882f8eb: Pull complete
2cabec57fa36: 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

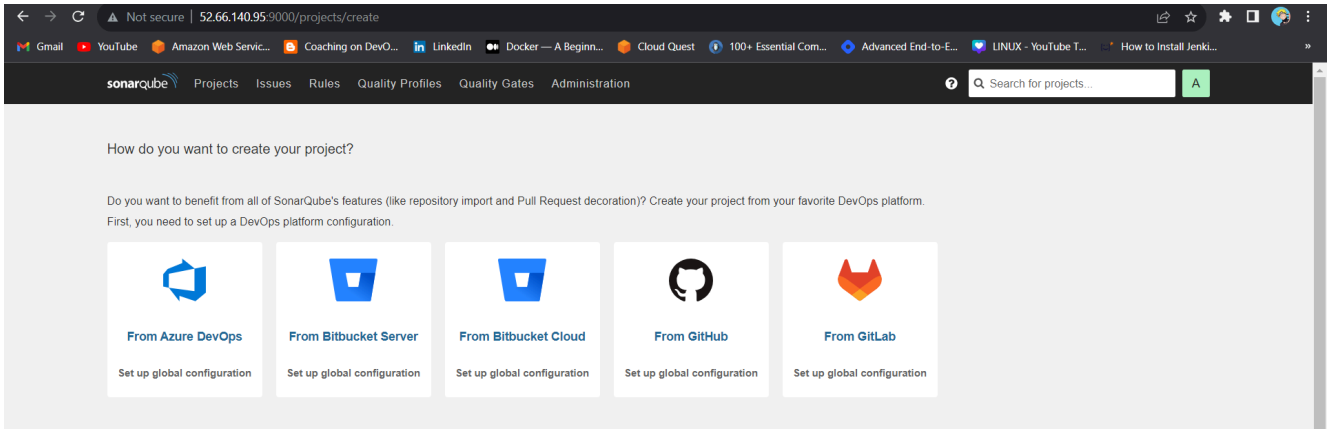


Enter username and password, click on login and change password

```
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 | gp
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity
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

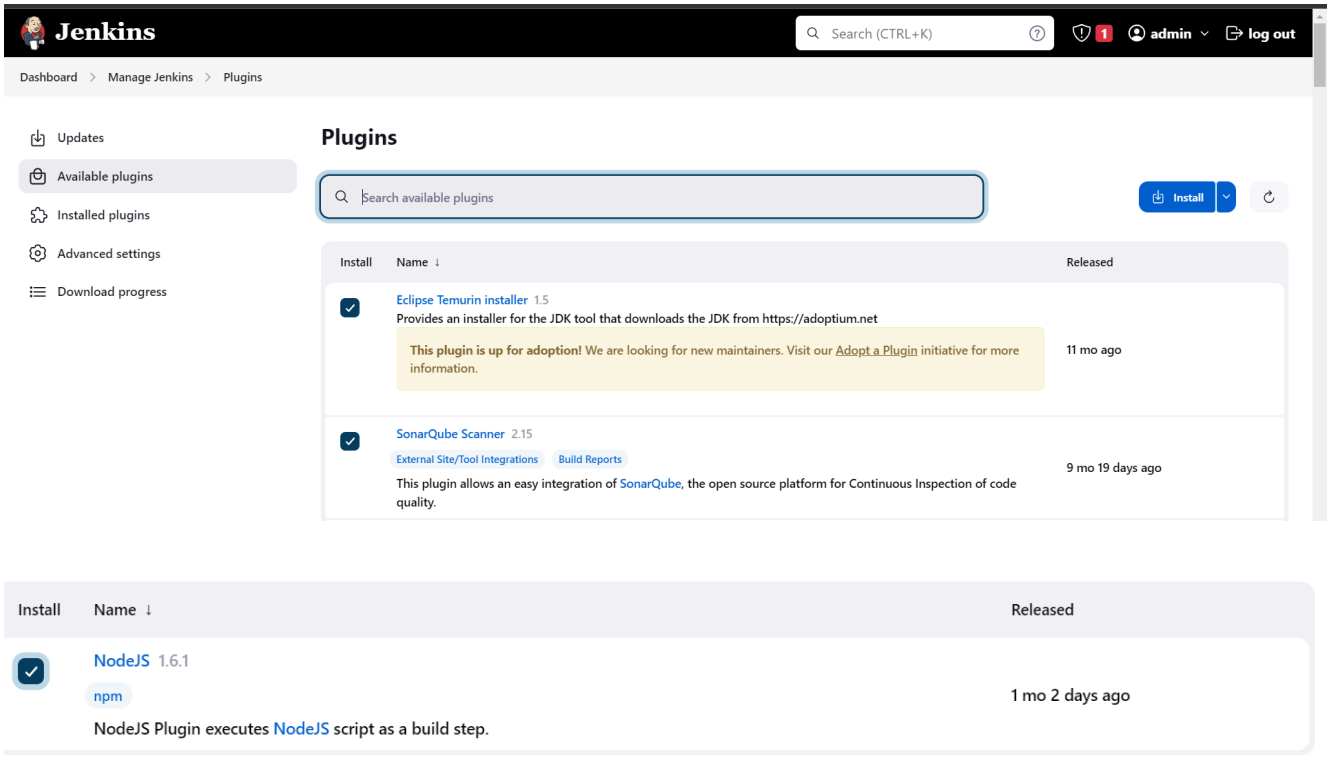
Goto Manage Jenkins →Plugins → Available Plugins →

Install below plugins

1 → Eclipse Temurin Installer (Install without restart)

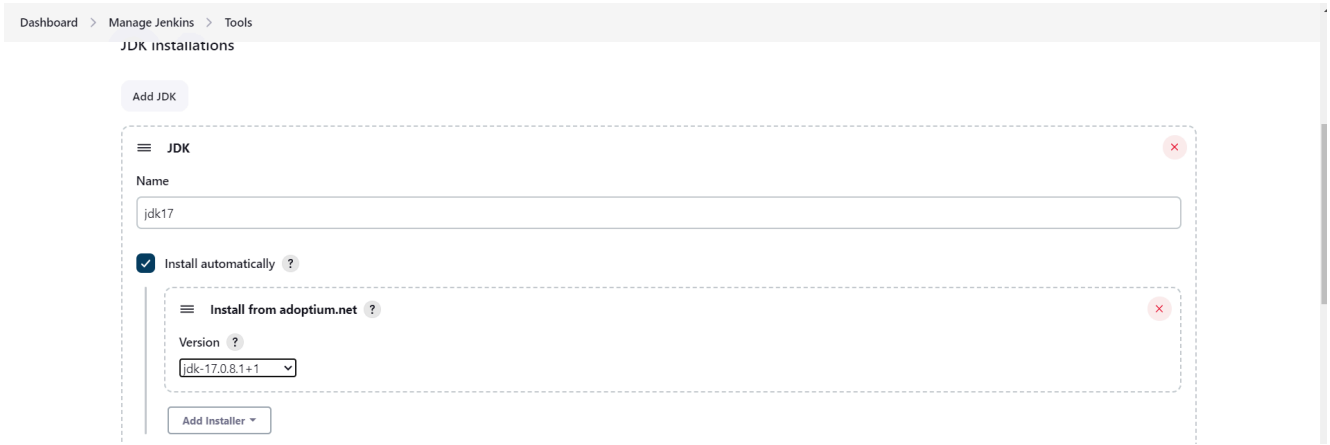
2 → SonarQube Scanner (Install without restart)

3 → NodeJs Plugin (Install Without restart)



3B – Configure Java and Nodejs in Global Tool Configuration

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

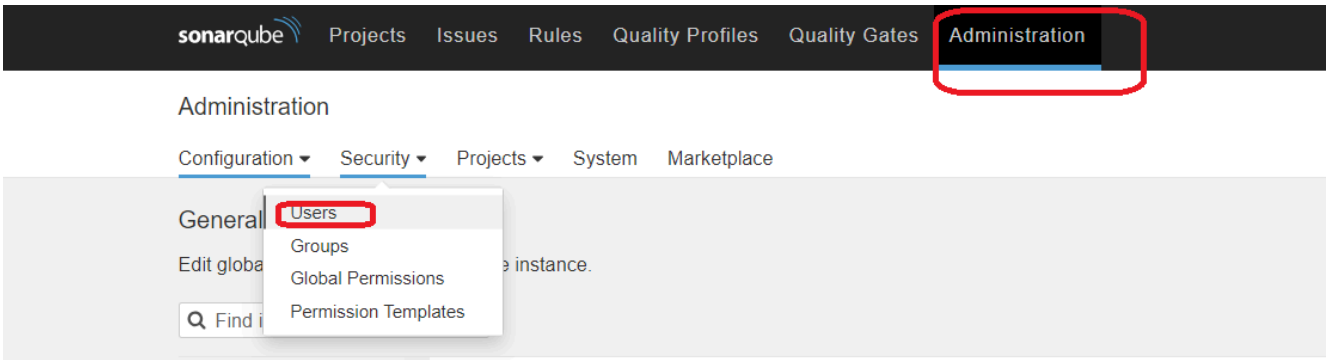


3C – Create a Job

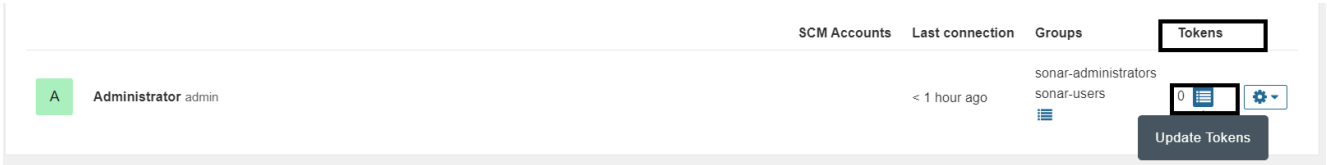
create a job as 2048 Name, select pipeline and click on ok.

Step 4 – Configure Sonar Server in Manage Jenkins

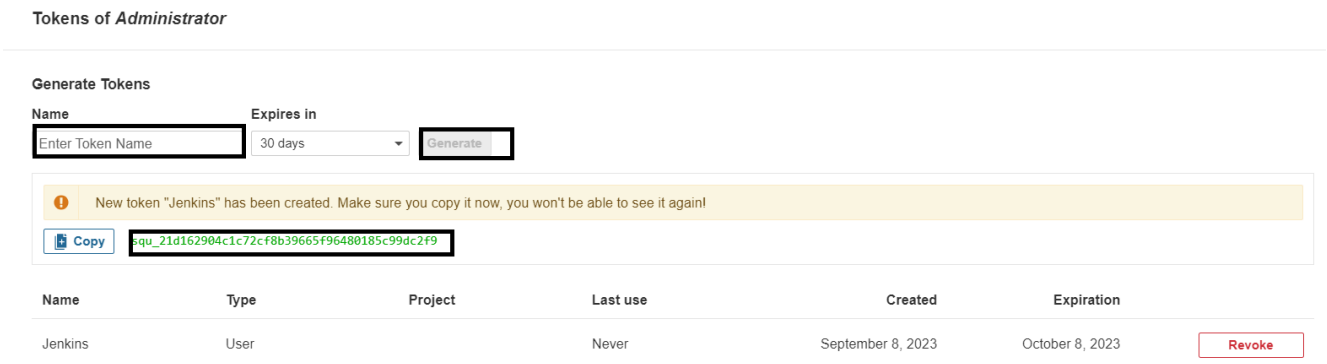
Grab the Public IP Address of your EC2 Instance, Sonarqube works on Port 9000, so <Public IP>:9000. Goto your Sonarqube Server. Click on Administration → Security → Users → Click on Tokens and Update Token → Give it a name → and click on Generate Token



click on update Token

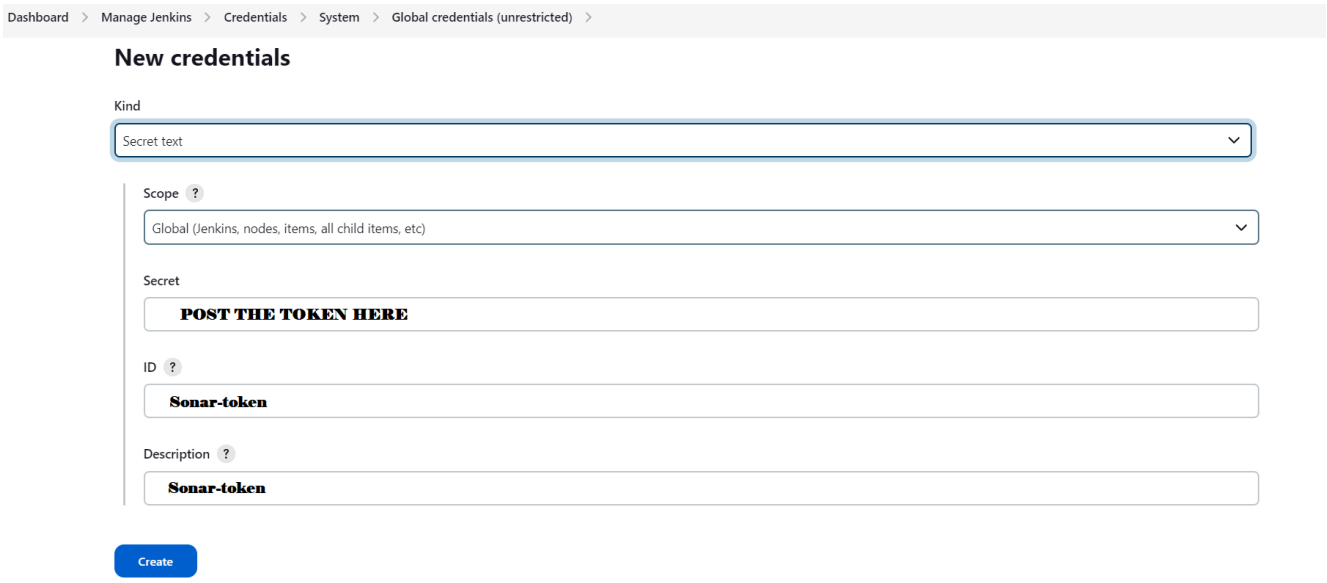


Create a token with a name and generate



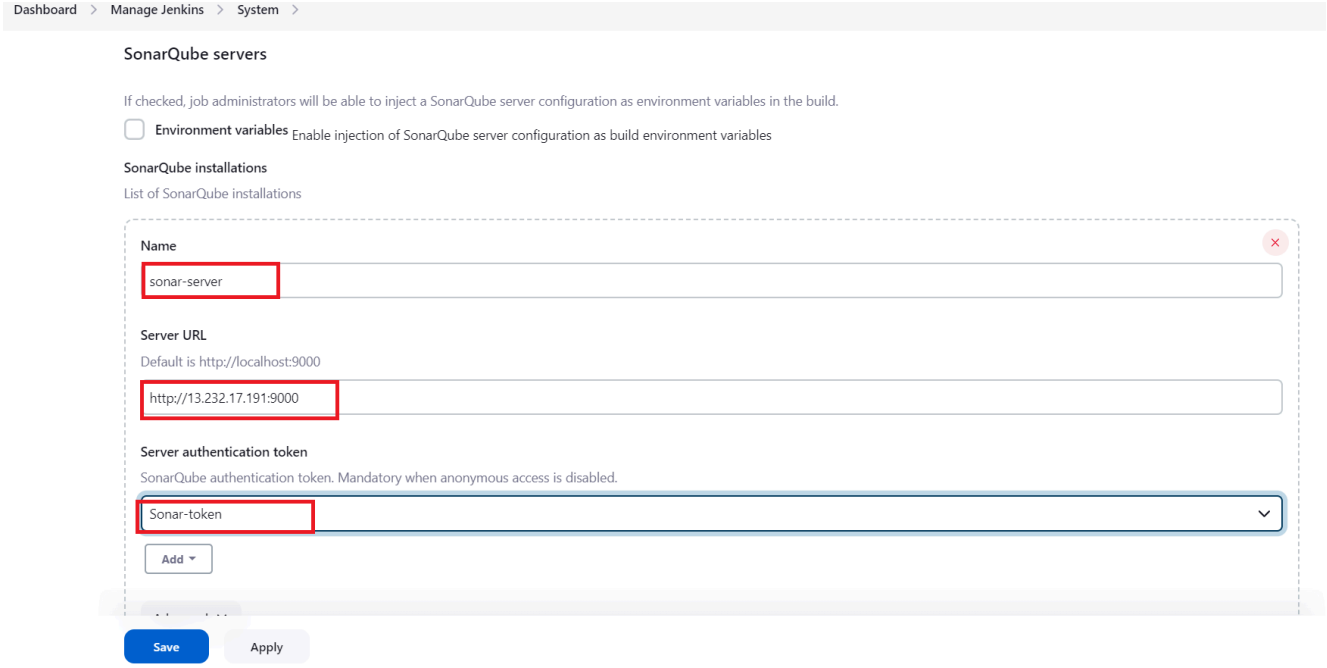
copy Token

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.

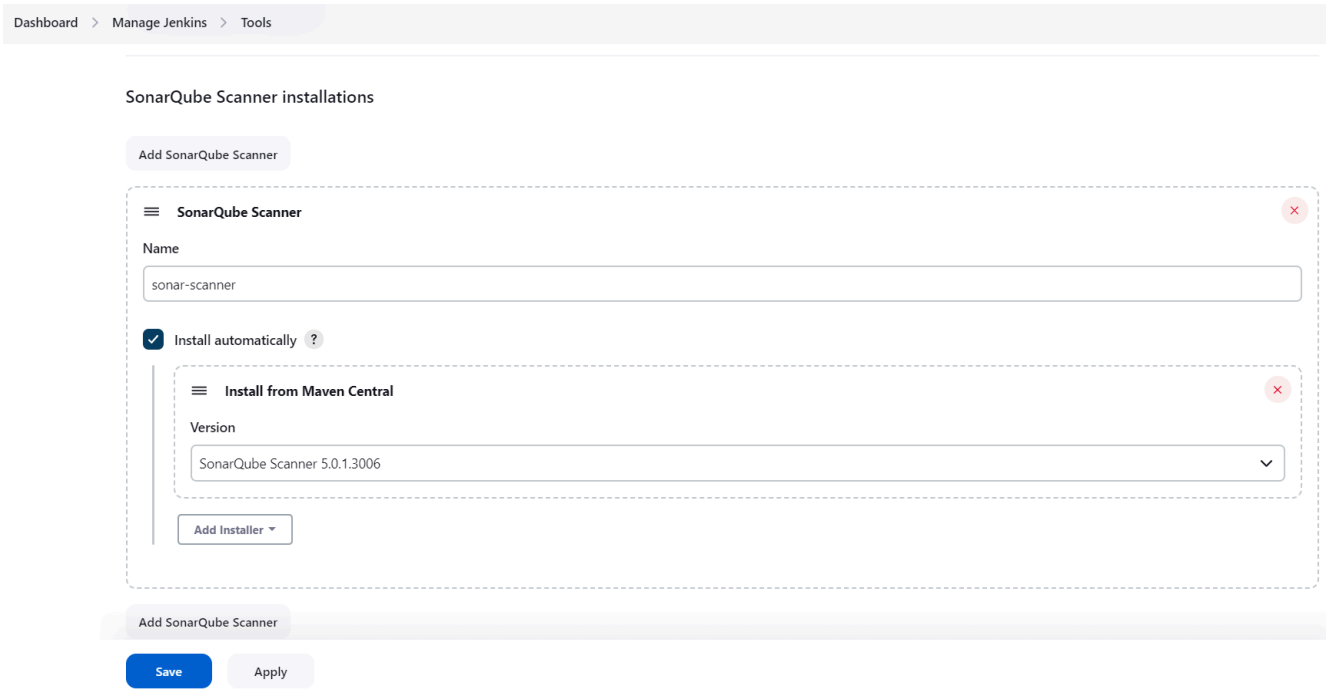


Click on Apply and Save

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

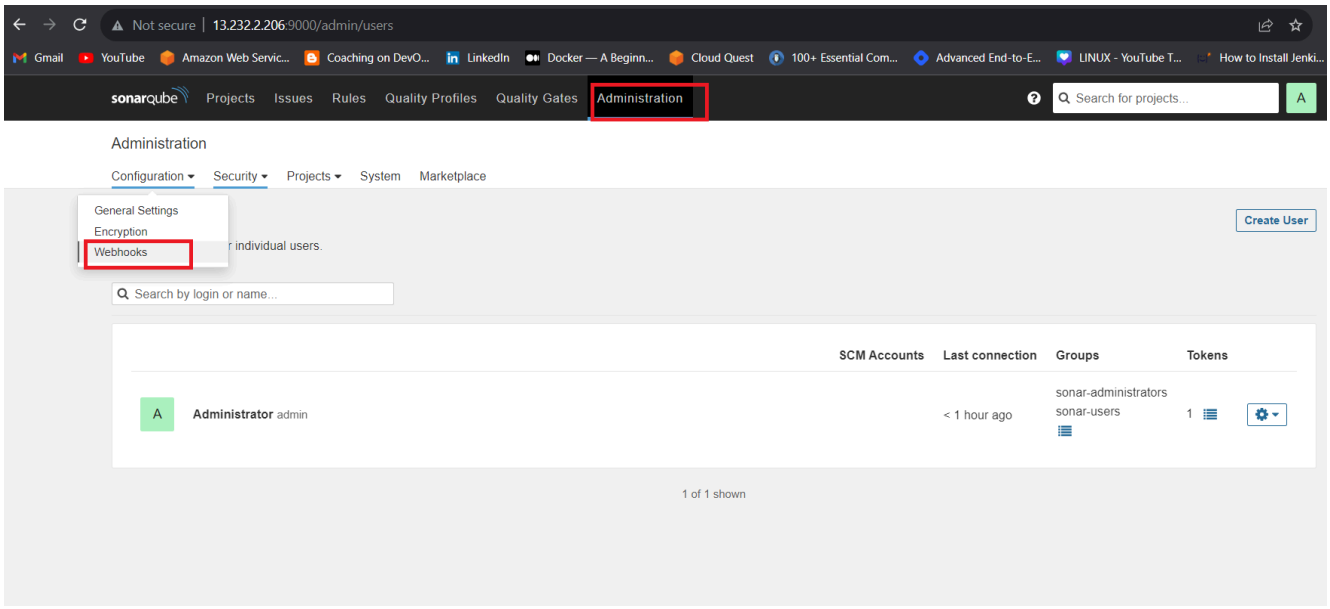
Global Tool Configuration is used to configure different tools that we install using Plugins

We will install a sonar scanner in the tools.

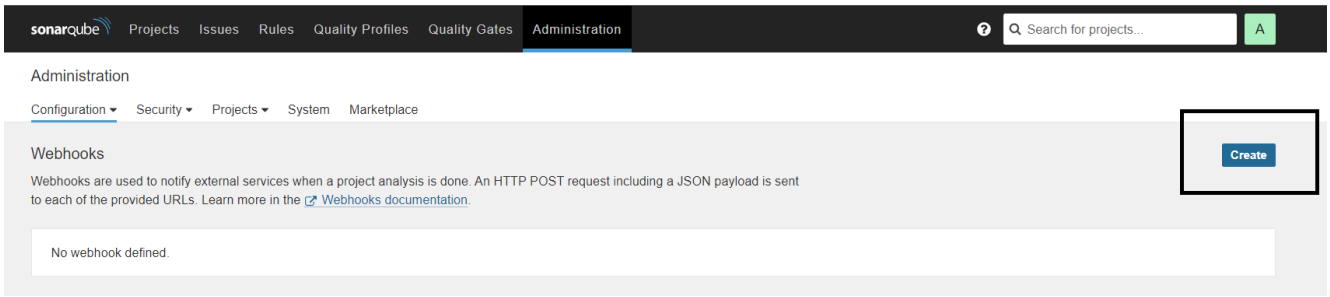


In the Sonarqube Dashboard add a quality gate also

Administration–> Configuration–>Webhooks

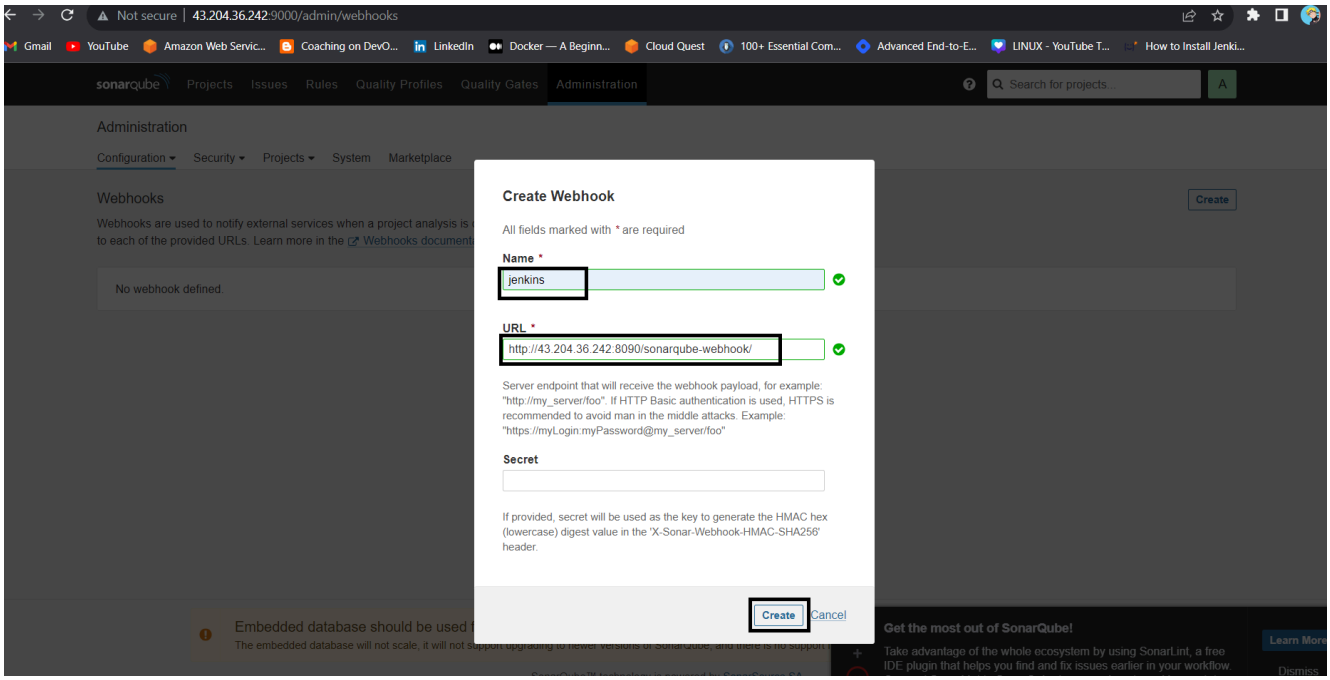


Click on Create



Add details

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

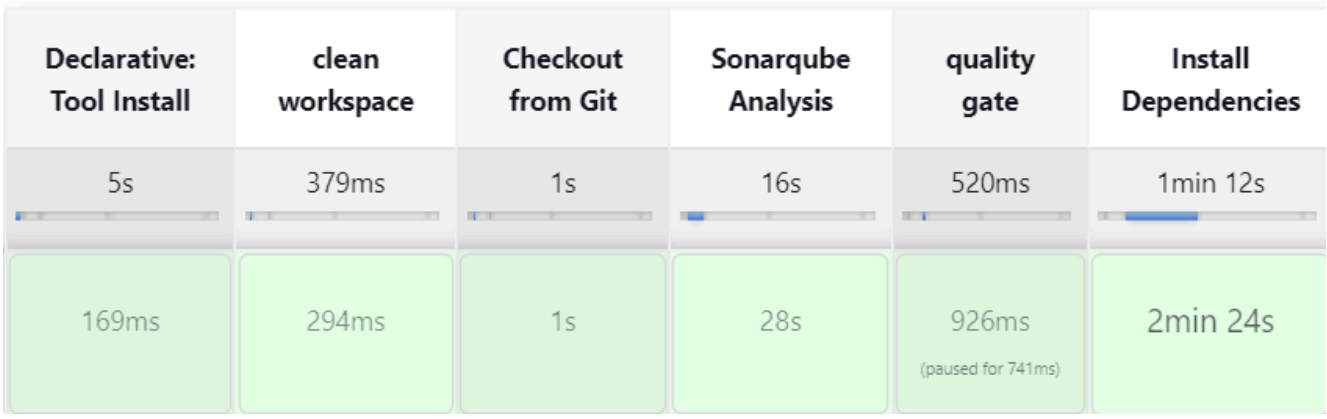


Let's go to our Pipeline and add the script in our Pipeline Script.

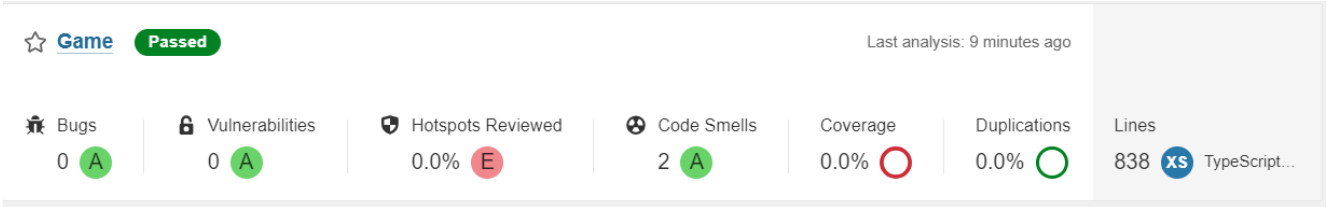
```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanWs()
            }
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'master', url: 'https://github.com/Aj7Ay/20
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.proje
```

```
                    -Dsonar.projectKey=Game '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentials
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh "npm install"
            }
        }
    }
}
```

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



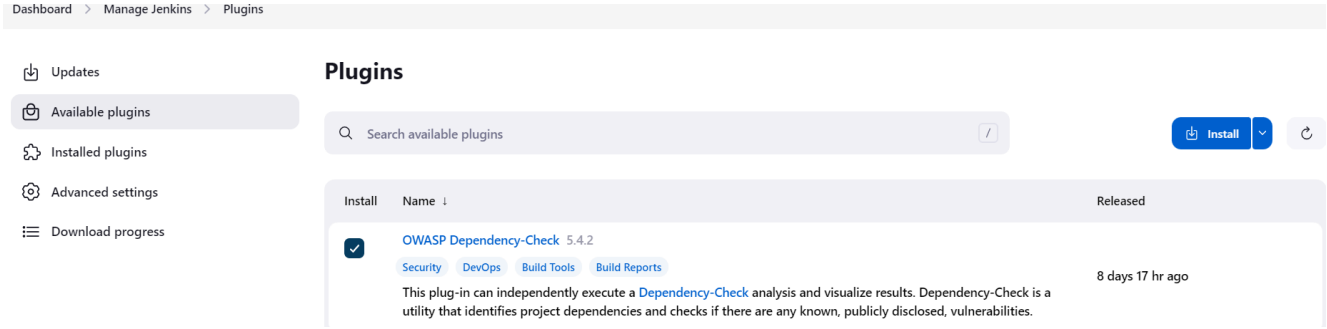
To see the report, you can go to Sonarqube Server and go to Projects.



You can see that there are 838 lines. To see a detailed report, you can go to issues.

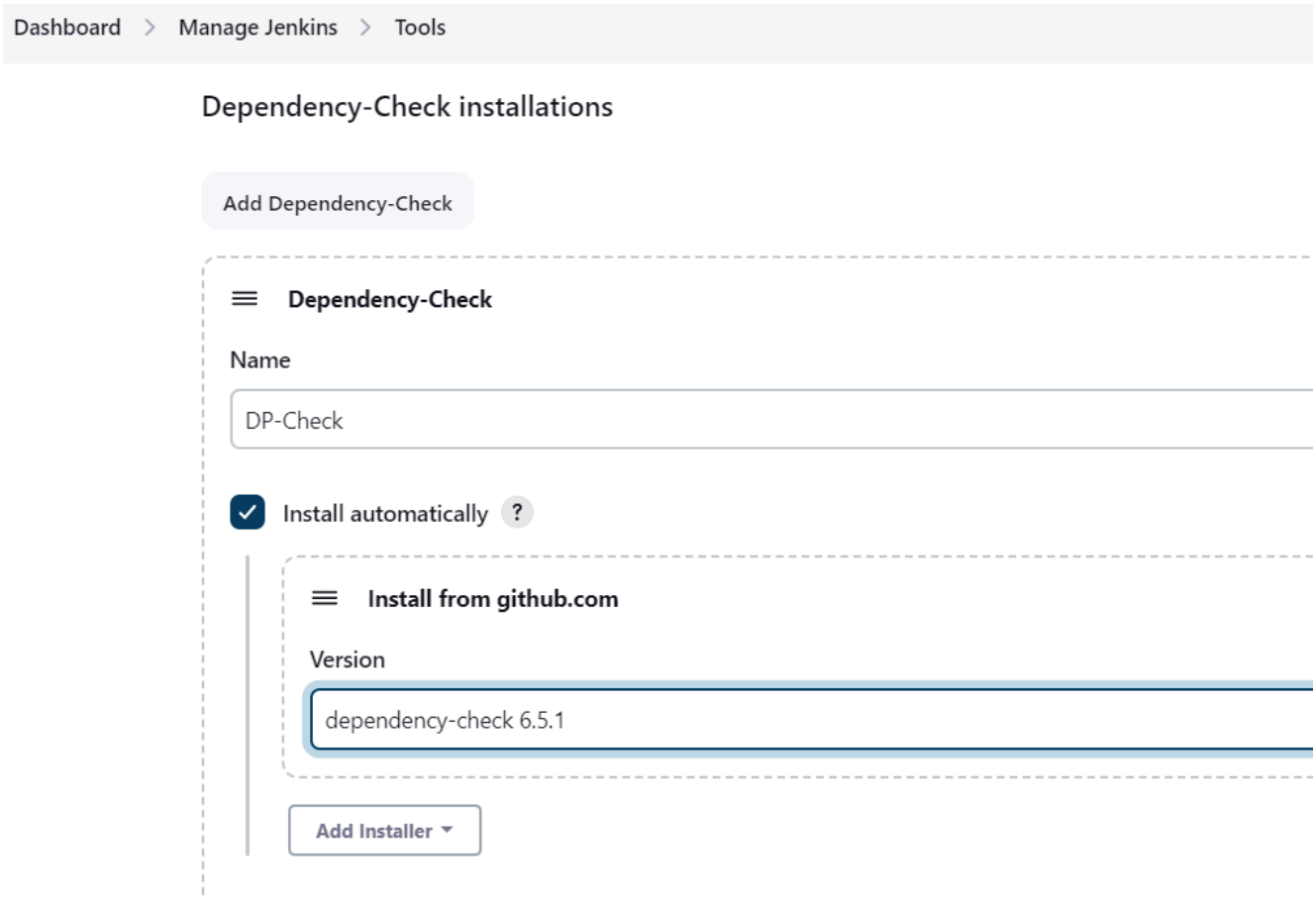
Step 5 – Install OWASP Dependency Check Plugins

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 →

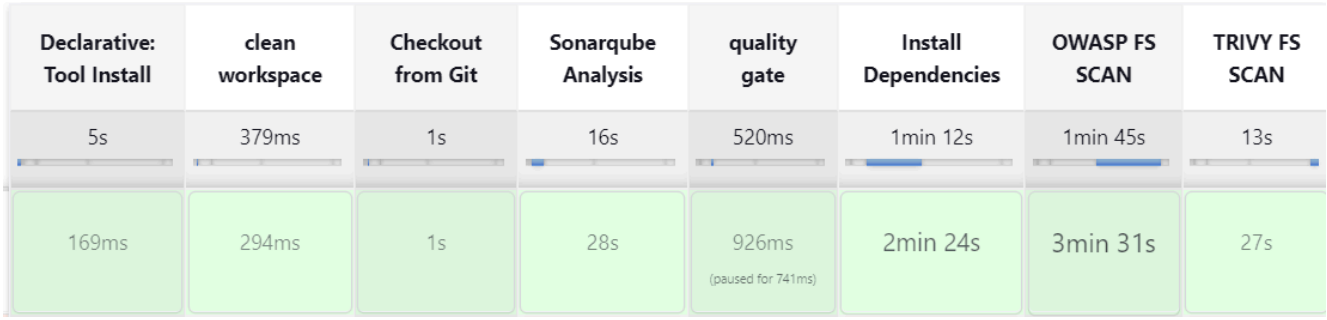


Click on Apply and Save here.

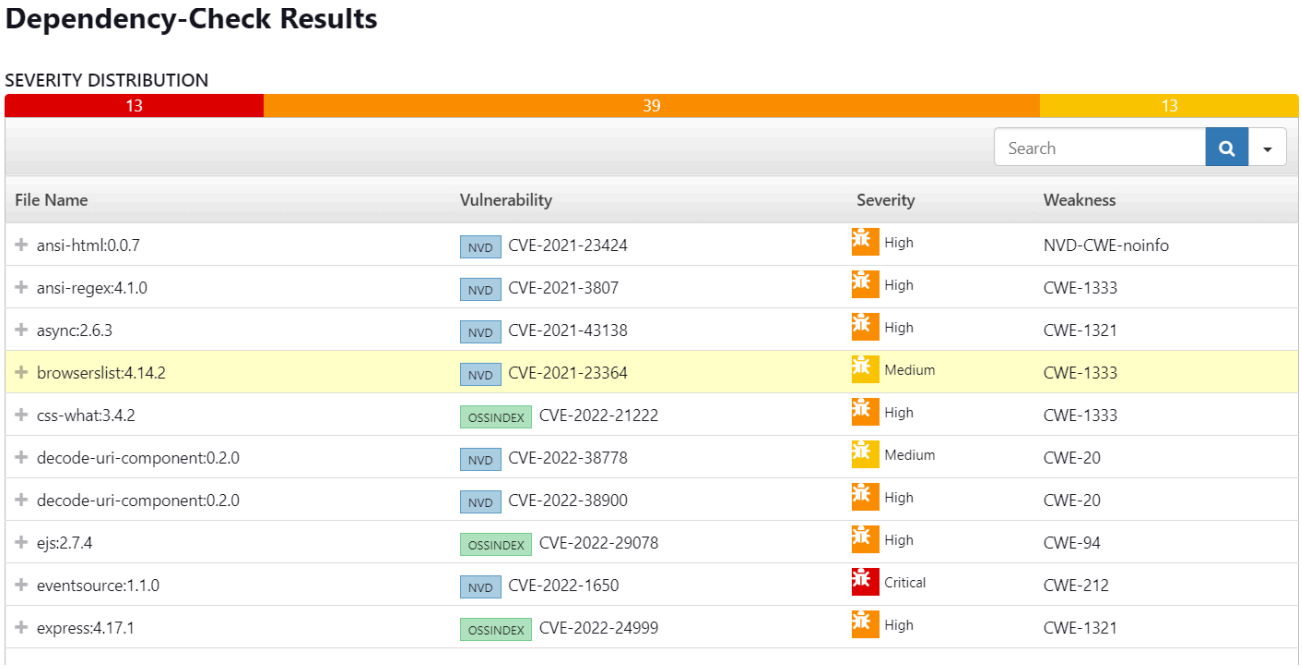
Now go configure → Pipeline and add this stage to your pipeline and build.

```
stage('OWASP FS SCAN') {
            steps {
                dependencyCheck additionalArguments: '--scan ./ --disab
                dependencyCheckPublisher pattern: '**/dependency-check-r
            }
        }
        stage('TRIVY FS SCAN') {
            steps {
                sh "trivy fs . > trivyfs.txt"
            }
        }
```

The stage view would look like this,



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

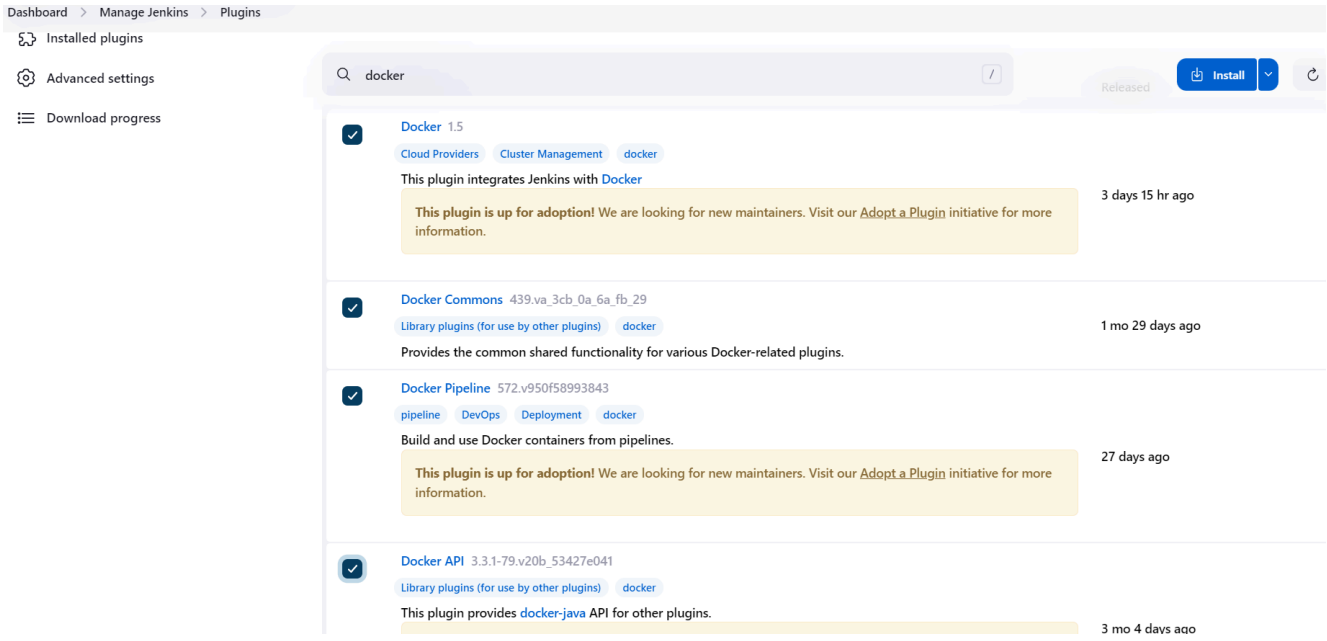


Step 6 – Docker Image Build and Push

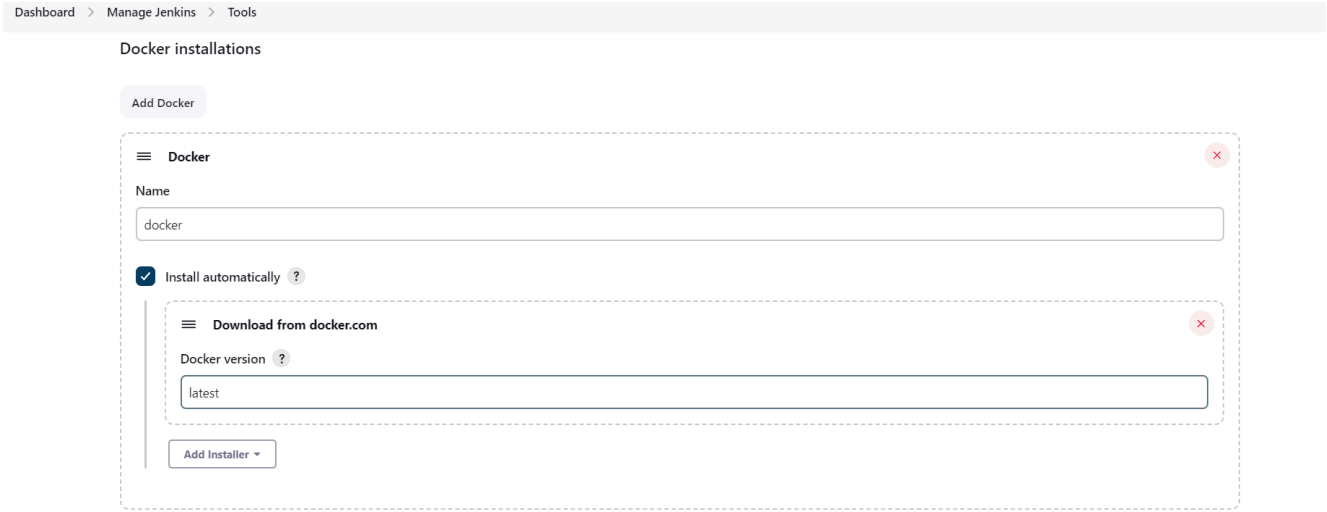
We need to install the Docker tool in our system, Goto Dashboard → Manage Plugins → Available plugins → Search for Docker and install these plugins

```
Docker

Docker Commons

Docker Pipeline

Docker API

docker-build-step
```

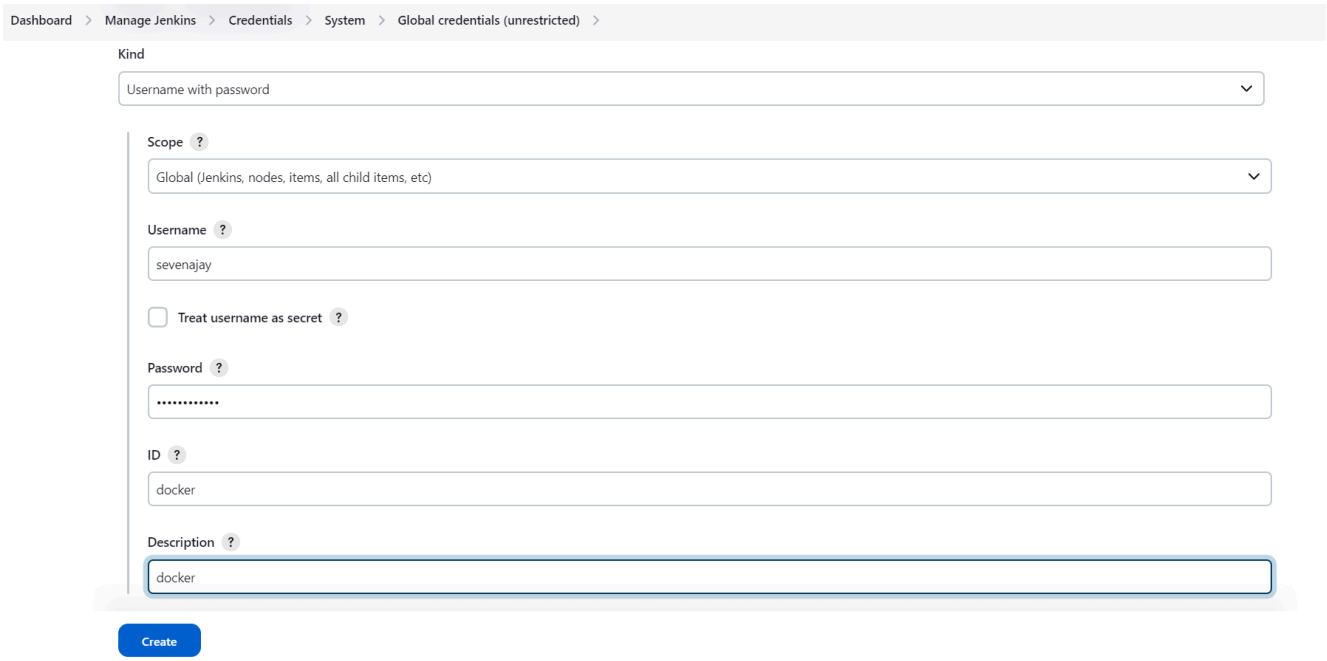
and click on install without restart



Now, goto Dashboard → Manage Jenkins → Tools →



Add DockerHub Username and Password under Global Credentials

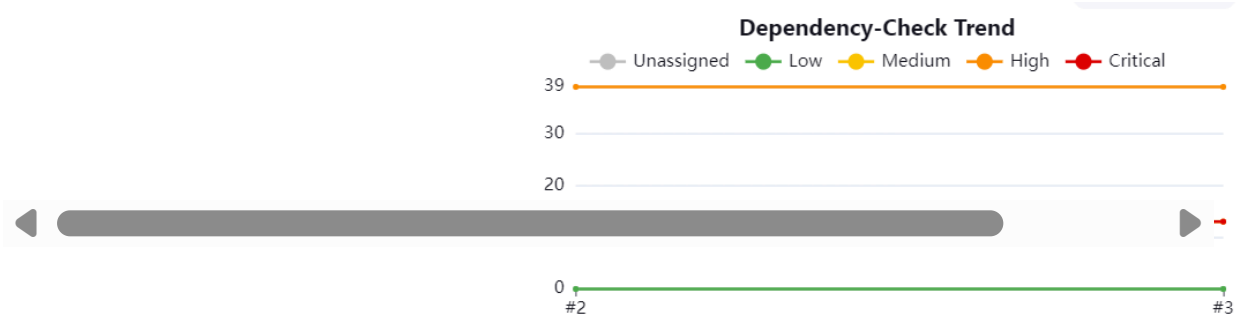


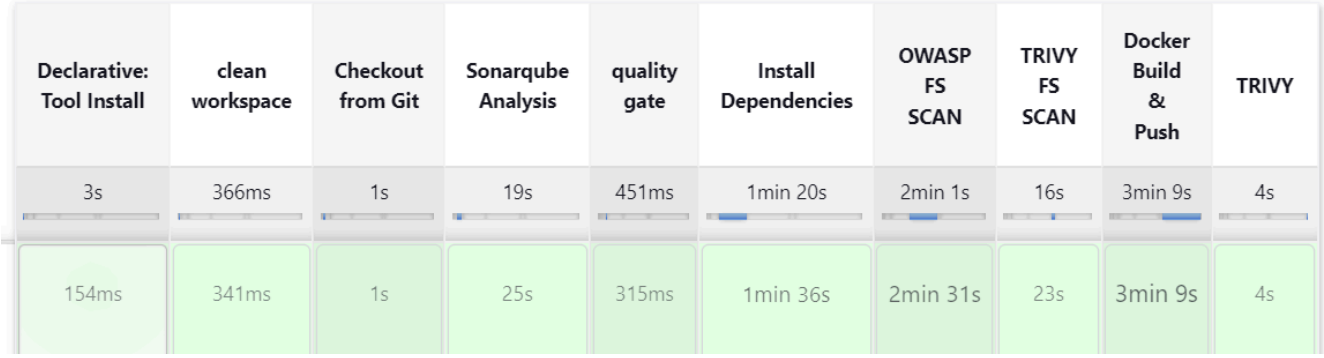
Add this stage to Pipeline Script

```
stage("Docker Build & Push"){
           steps{
               script{
                  withDockerRegistry(credentialsId: 'docker', toolName
                      sh "docker build -t 2048 ."
                      sh "docker tag 2048 sevenajay/2048:latest "
                      sh "docker push sevenajay/2048:latest "
                   }
               }
           }
```

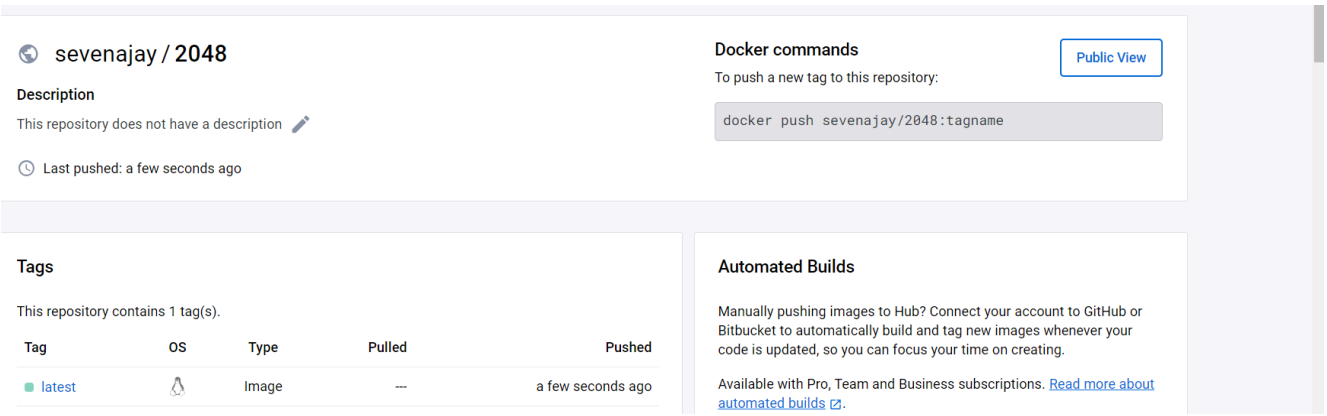
```
        }
        stage("TRIVY"){
            steps{
                sh "trivy image sevenajay/2048:latest > trivy.txt"
            }
        }
```

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





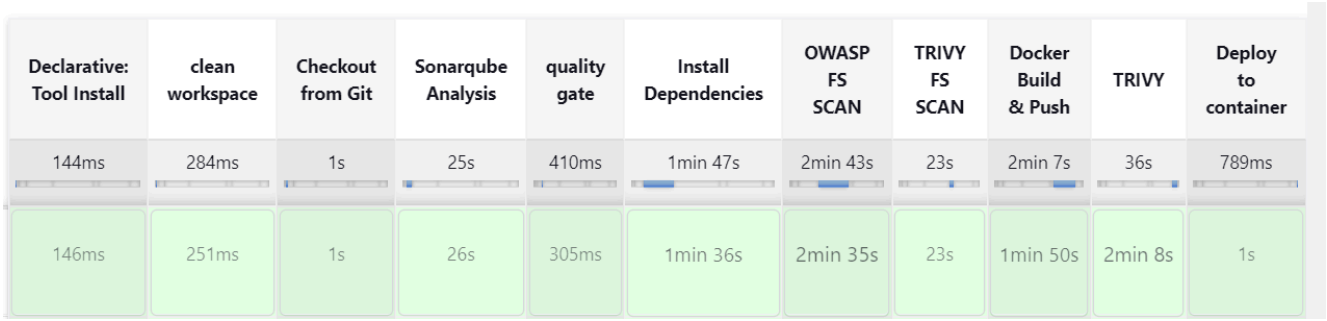
When you log in to Dockerhub, you will see a new image is created



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

```
    stage('Deploy to container'){
            steps{
                sh 'docker run -d --name 2048 -p 3000:3000 sevenajay/20
            }
        }
```

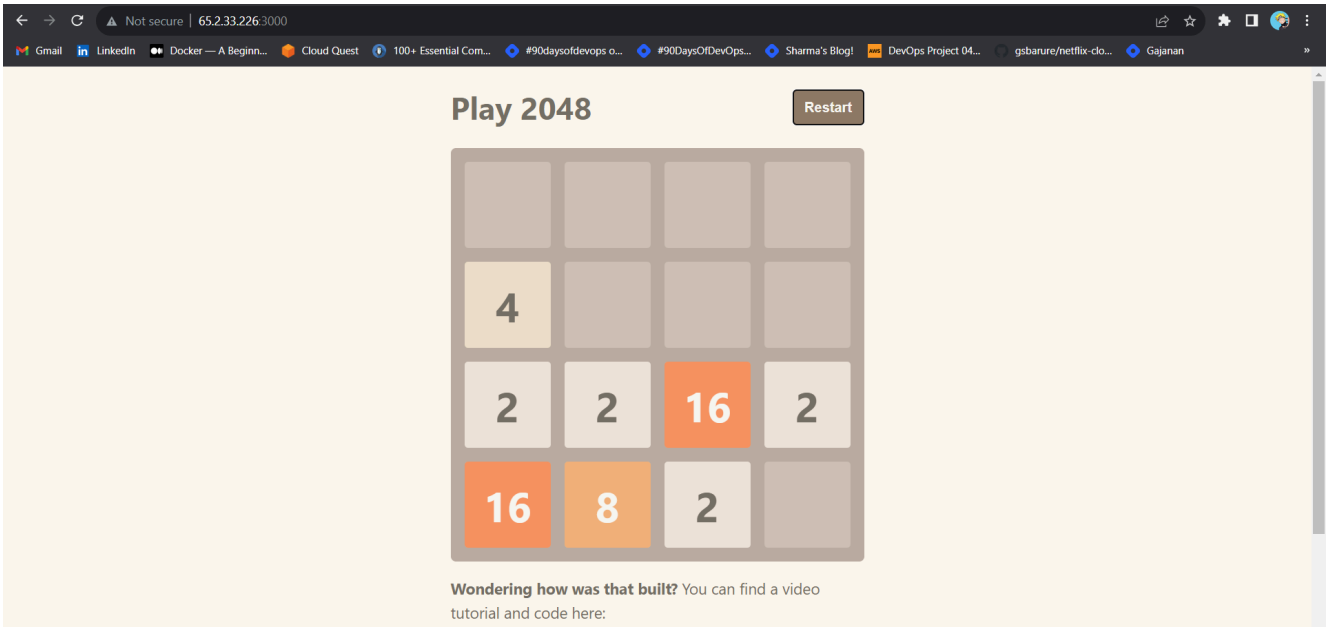
stage view



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
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/releas
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 | suc
sudo tee /etc/apt/sources.list.d/kubernetes.list <<EOF
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo snap install kube-apiserver
```

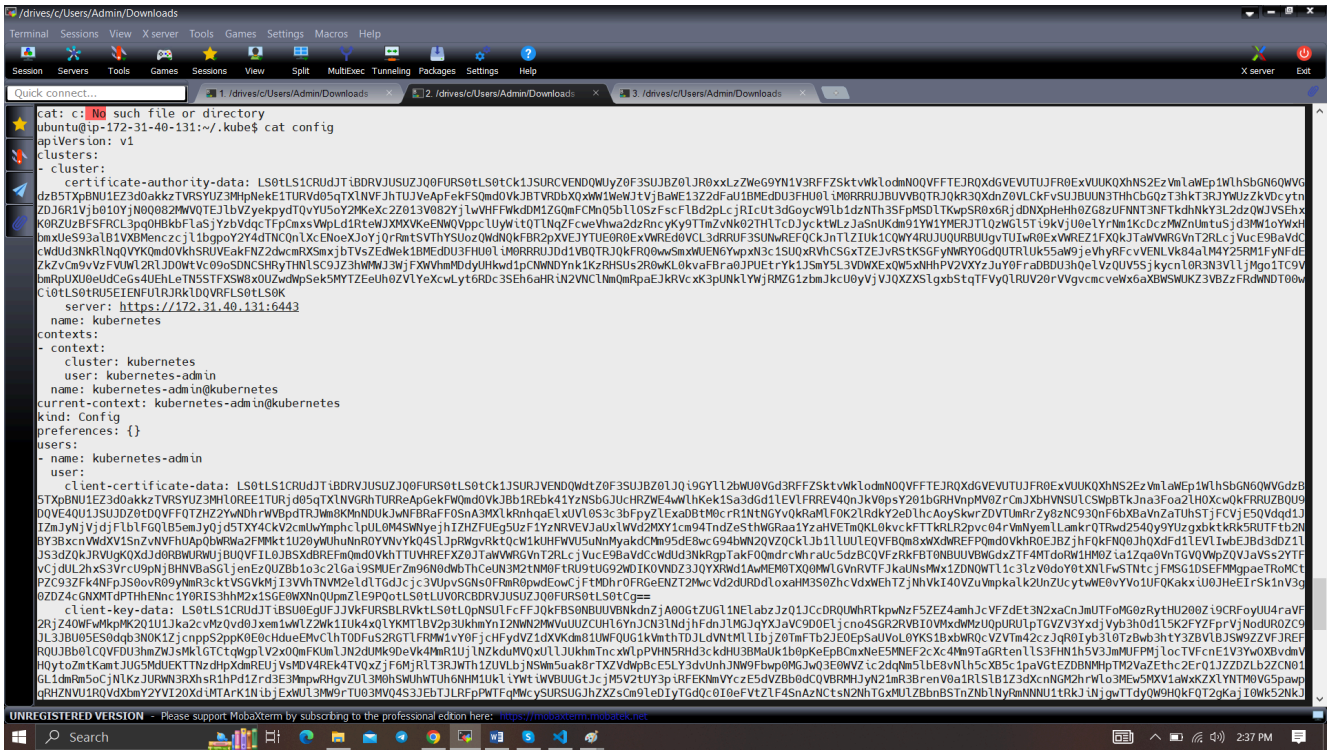
Part 3 ––––– Master –––––

```
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
```

––––Worker Node––––

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

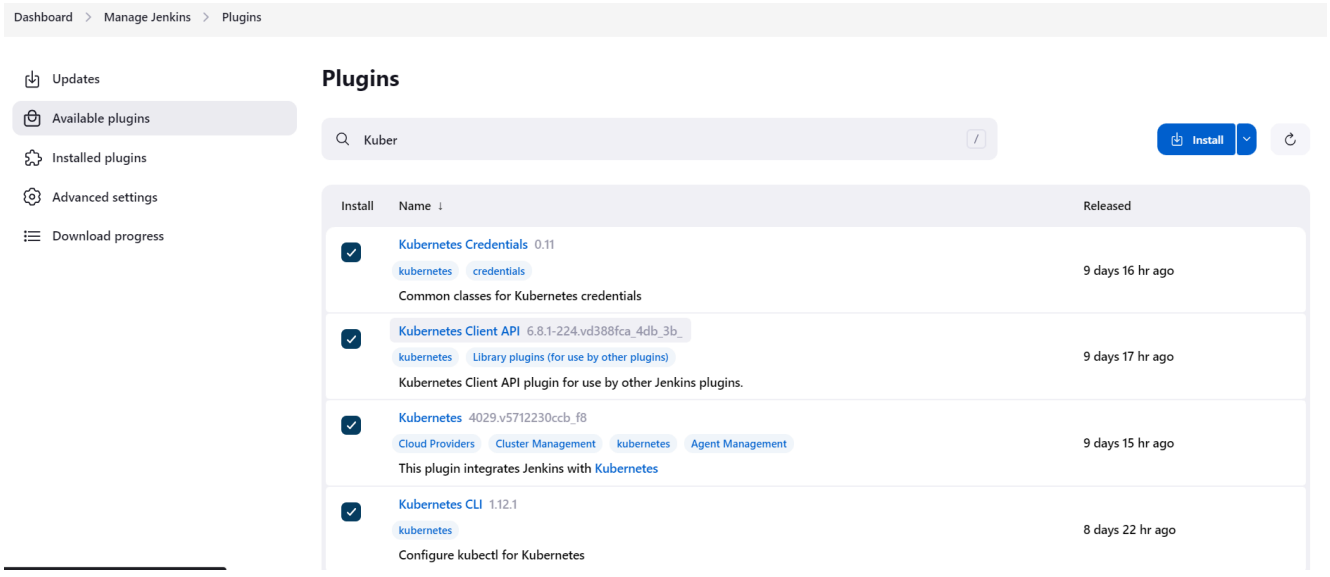
Copy the config file to Jenkins master or the local file manager and save it



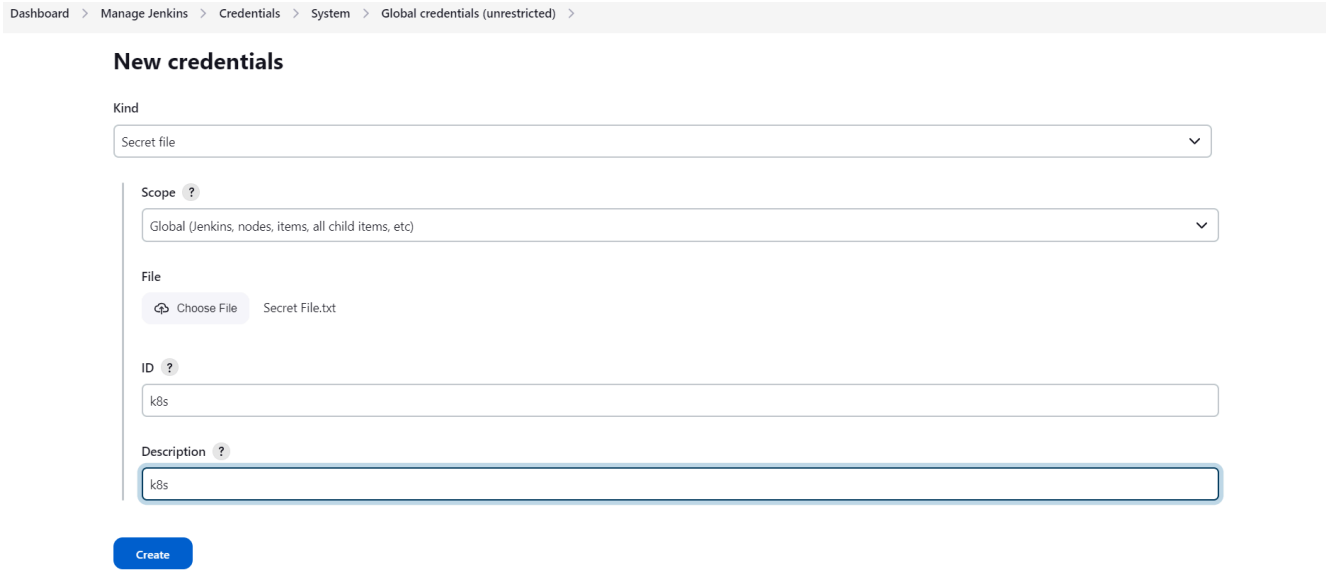
copy it and save it in documents or another folder save it as secret-file.txt

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

Install Kubernetes Plugin, Once it's installed successfully



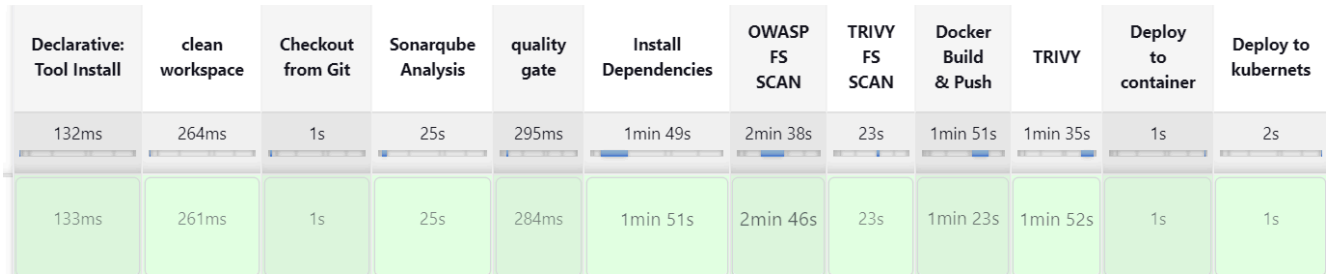
goto manage Jenkins –> manage credentials –> Click on Jenkins global –> add credentials



final step to deploy on the Kubernetes cluster

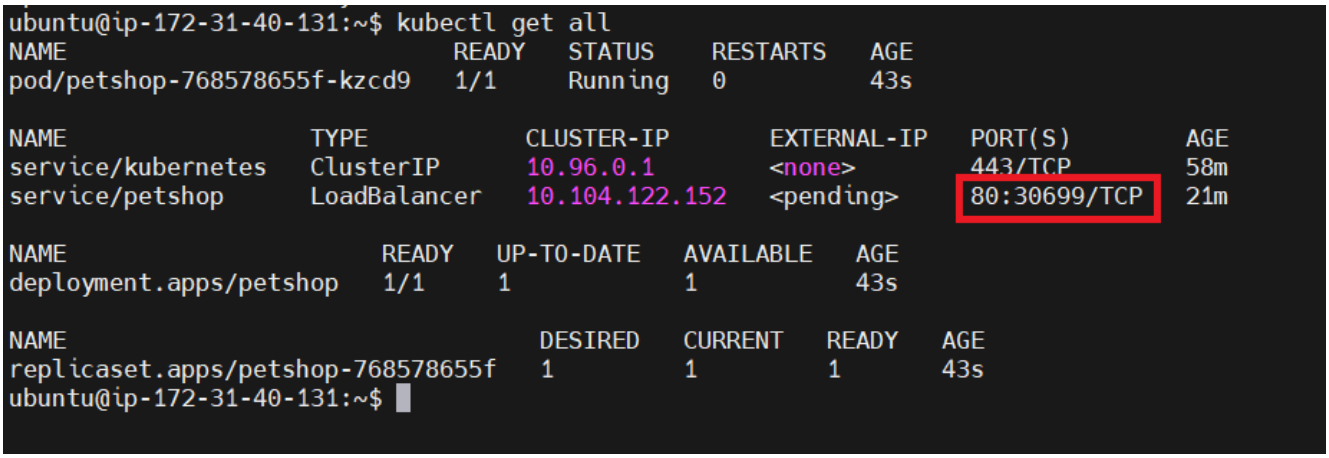
```
stage('Deploy to kubernets'){
            steps{
                script{
                    withKubeConfig(caCertificate: '', clusterName: '', c
                       sh 'kubectl apply -f deployment.yaml'
                  }
                }
            }
        }
```

stage view



In the Kubernetes cluster give this command

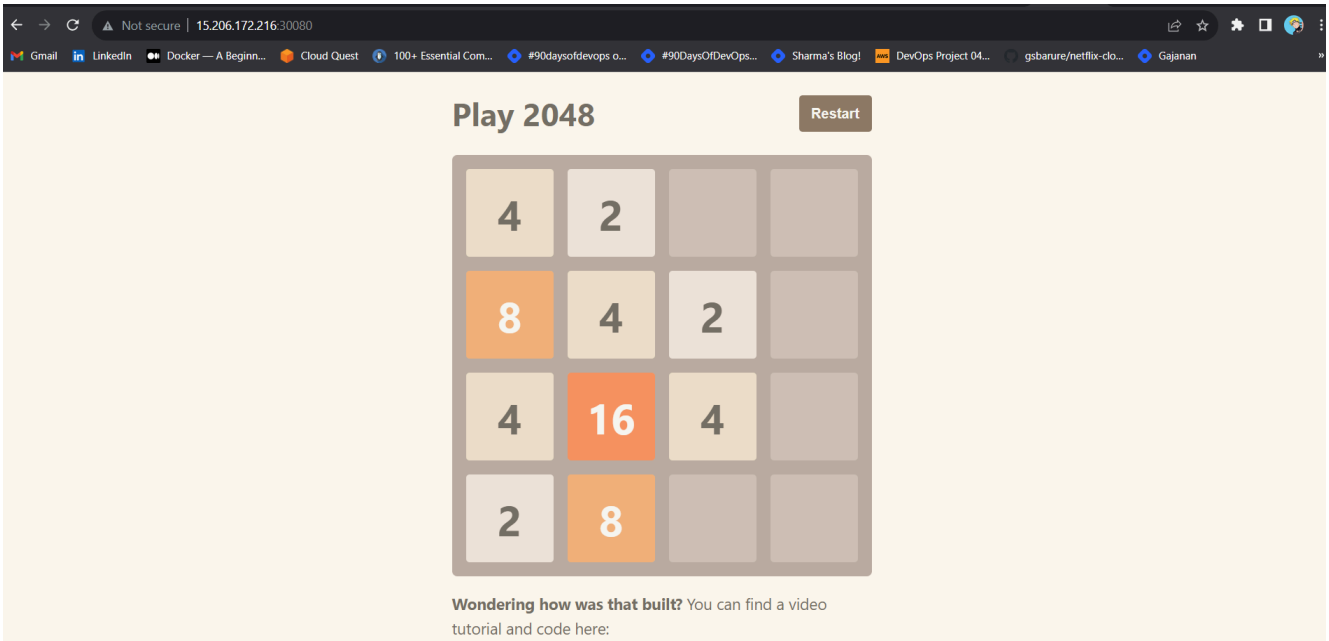
```
kubectl get all
kubectl get svc #use anyone
```



STEP9:Access from a Web browser with

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

output:



Step 10: Terminate instances.

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: 'master', url: 'https://github.com/Aj7Ay/204
            }
        }
        stage("Sonarqube Analysis "){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.proje
                    -Dsonar.projectKey=Game '''
                }
            }
        }
        stage("quality gate"){
           steps {
                script {
                    waitForQualityGate abortPipeline: false, credentials
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh "npm install"
            }
        }
        stage('OWASP FS SCAN') {
            steps {
```

```
                dependencyCheck additionalArguments: '--scan ./ --disab
                dependencyCheckPublisher pattern: '**/dependency-check-r
            }
        }
        stage('TRIVY FS SCAN') {
            steps {
                sh "trivy fs . > trivyfs.txt"
            }
        }
        stage("Docker Build & Push"){
            steps{
                script{
                   withDockerRegistry(credentialsId: 'docker', toolName
                       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"
            }
        }
        stage('Deploy to container'){
            steps{
                sh 'docker run -d --name 2048 -p 3000:3000 sevenajay/204
            }
        }
        stage('Deploy to kubernets'){
            steps{
                script{
                    withKubeConfig(caCertificate: '', clusterName: '', c
                       sh 'kubectl apply -f deployment.yaml'
                  }
                }
            }
        }
    }
}
```

Ajay Kumar Yegireddi is a DevSecOps Engineer and System Administrator, with a passion for sharing real-world DevSecOps projects and tasks. **Mr. Cloud Book**, provides hands-on tutorials and practical insights to help others master DevSecOps tools and workflows. Content is designed to bridge the gap between development, security, and operations, making complex concepts easy to understand for both beginners and professionals.

Comments

2 responses to “Deploying 2048 Game on Docker and Kubernetes with Jenkins CI/CD”

vikranth
15 February 2024

Hi Ajay,

We i am trying to build and push to dockerhub, it is failing due to this error.
Please find the below error
+ docker build -t 2048 .

DEPRECATED: The legacy builder is deprecated and will be removed in a future release.

Install the buildx component to build images with BuildKit:

https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 251.8MB

Step 1/8 : FROM node:16

–> 1ddc7e4055fd

Step 2/8 : WORKDIR /app

–> Using cache

–> 75bf19fac687

Step 3/8 : COPY package*.json ./

—> 15f541247f56

Step 4/8 : RUN npm install

—> Running in 28829db9839b

npm WARN deprecated urix@0.1.0: Please see https://github.com/lydell/urix#deprecated

npm WARN deprecated w3c-hr-time@1.0.2: Use your platform's native performance.now() and performance.timeOrigin.

npm WARN deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library is deprecated. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#browser_compatibility

npm WARN deprecated source-map-url@0.4.1: See https://github.com/lydell/source-map-url#deprecated

npm WARN deprecated sourcemap-codec@1.4.8: Please use @jridgewell/sourcemap-codec instead

npm WARN deprecated source-map-resolve@0.6.0: See https://github.com/lydell/source-map-resolve#deprecated

npm WARN deprecated sane@4.1.0: some dependency vulnerabilities fixed, support for node < 10 dropped, and newer ECMAScript syntax/features added

npm WARN deprecated rollup-plugin-terser@5.3.1: This package has been deprecated and is no longer maintained. Please use @rollup/plugin-terser

npm WARN deprecated resolve-url@0.2.1: https://github.com/lydell/resolve-url#deprecated

npm WARN deprecated workbox-google-analytics@5.1.4: It is not compatible with newer versions of GA starting with v4, as long as you are using GAv3 it should be ok, but the package is not longer being maintained

npm WARN deprecated querystring@0.2.1: The querystring API is considered Legacy. new code should use the URLSearchParams API instead.

npm WARN deprecated rollup-plugin-babel@4.4.0: This package has been deprecated and is no longer maintained. Please use @rollup/plugin-babel.

npm ERR! code ENOSPC

npm ERR! syscall write

npm ERR! errno -28

npm ERR! nospc ENOSPC: no space left on device, write

npm ERR! nospc There appears to be insufficient space on your system to finish.

npm ERR! nospc Clear up some disk space and try again.

npm ERR! A complete log of this run can be found in:

npm ERR! /root/.npm/_logs/2024-02-15T12_39_36_120Z-debug-0.log

No space left on device

Reply

tejas
11 September 2024

make sure you have added jenkins user to docker group (same like adding ubuntu user)
checker docker credentials are right
your tag and your docker hub user name should be same

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

I'm not a robot
reCAPTCHA
Privacy - Terms

Post Comment

Uncategorized

How to Automate Incident Response : How Q Developer Helped Me Automate a Daily Pain Point

22 July 2025

AI

How to Run Docker Model Runner on Ubuntu 24.04

11 July 2025

AI, DevOps

How to Install docker-ai on Ubuntu 24.04

15 June 2025

Upskill with Ajay: DevSecOps Mastery

Join Mr Cloud book to master DevSecOps through real-world projects. Learn CI/CD, security integration, automation, and more, gaining hands-on skills for industry-level challenges.

Important Links

Privacy Policy

Terms & Conditions

Contact

Resources

Blog

YouTube Channel

Follow Us on YouTube