

DOCKER PROJECT

STEP-1: LAUNCH AN INSTANCE WITH T2.LARGE

STEP-2: INSTALL JENKINS, GIT, DOCKER & TRIVY

STEP-3: INSTALL THE FOLLOWING JENKINS PLUGINS

- SONAR SCANNER
- NODEJS
- OWASP DEPENDENCY CHECK
- DOCKER PIPELINE
- [Eclipse Temurin installerVersion](#)

STEP-4: CONFIGURE ALL THE PLUGINS INTO JENKINS

STEP-5: WRITE A PIPELINE

TRIVY INSTALLATION:

- `wget https://github.com/aquasecurity/trivy/releases/download/v0.18.3/trivy_0.18.3_Linux-64bit.tar.gz`
- `tar zxvf trivy_0.18.3_Linux-64bit.tar.gz`
- `sudo mv trivy /usr/local/bin/`
- `vim .bashrc`
- `export PATH=$PATH:/usr/local/bin/`
- `source .bashrc`

JENKINS INSTALLATION:

- `amazon-linux-extras install java-openjdk11 -y`
- `sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo`
- `sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key`
- `yum install jenkins -y`
- `systemctl start jenkins`

GIT & DOCKER INSTALLATION:

- `yum install git docker -y`

- `systemctl start docker`
- `chmod 777 ///var/run/docker.sock`

SETUP SONAR USING DOCKER:

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

After creating the sonar container, access the sonarqube with 9000 port number.

Login to the sonar dashboard with the following and credentials

- username: admin
- password: admin

The image shows the SonarQube login interface. At the top, the text "Log in to SonarQube" is centered. Below it are two input fields: the first is labeled "Login" and the second is labeled "Password". At the bottom right, there are two buttons: "Log in" and "Cancel". The background is a light gray.

After entering the credentials we have to set a new password.

CONFIGURE ALL THE PLUGINS INTO JENKINS:

Goto your Sonarqube Server. Click on Administration ----> Security ----> Users → Click on Tokens and Update Token ----> Give it a name ----> and click on Generate Token.

copy Token

Goto Jenkins Dashboard ----> Manage Jenkins ----> Credentials ----> Add Secret Text with id

sonar-token.

Goto Jenkins Dashboard → Manage Jenkins → Credentials → Add Secret Text.

Add sonarqube

Now, go to Dashboard --> Manage Jenkins -----> System and Add sonar servers with the name of **mysonar**

Click on Apply and Save

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

Click on add **SonarQube Scanner**

Name: mysonar

click on install automatically and proceed with default version.

In the Sonarqube Dashboard add a quality gate also

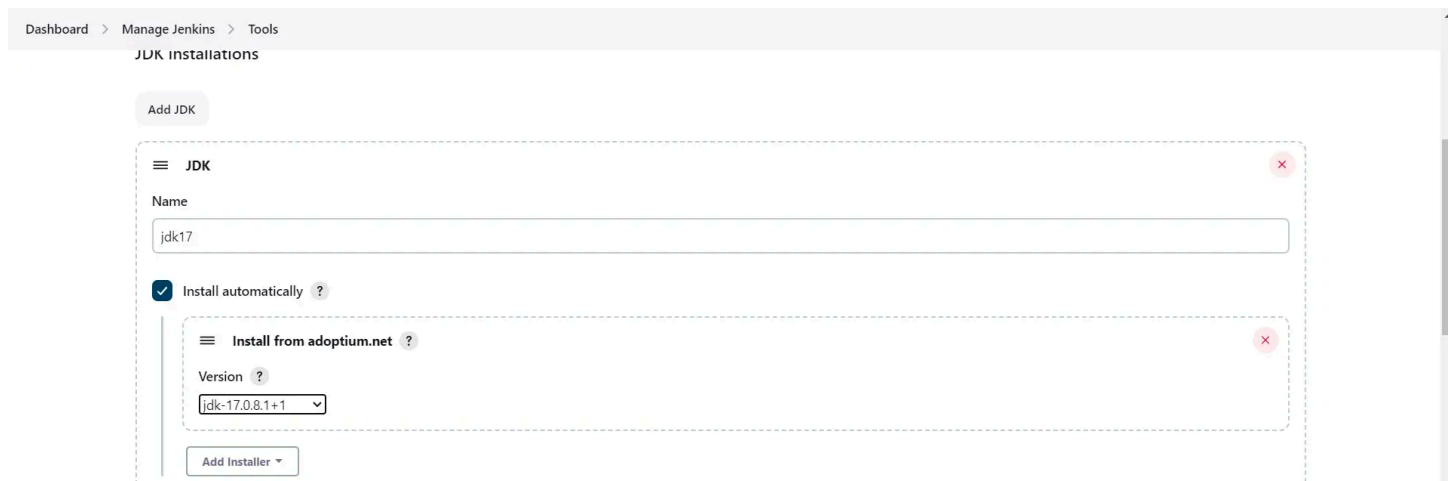
Administration → Configuration → Webhooks

Click on Create

Name: Jenkins

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

Now configure NodeJs, Java & DP-Check



node16

☒ Install automatically ?

≡ Install from nodejs.org

Version

NodeJS 16.2.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

☐ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax `packageName@ver`

Dashboard > Manage Jenkins > Tools

Dependency-Check installations

Add Dependency-Check

≡ Dependency-Check

Name

DP-Check

☒ Install automatically ?

≡ Install from github.com

Version

dependency-check 6.5.1

Add Installer ▾

Click on Apply and Save here.

START WRITING DECLARATIVE PIPELINE:

```
pipeline {
```

agent any

tools {

jdk 'jdk17'

nodejs 'node16'

}

environment {

SCANNER_HOME = tool 'mysonar'

}

stages {

stage("Clean WS") {

steps {

cleanWs()

}

}

stage("Code") {

steps {

git "<https://github.com/devops0014/Zomato-Project.git>"

}

}

stage("Sonarqube Analysis") {

steps {

withSonarQubeEnv('mysonar') {

sh """"\$SCANNER_HOME/bin/sonar-scanner \

-Dsonar.projectName=zomato \

-Dsonar.projectKey=zomato""""

```
    }  
  }  
}  
stage("Quality Gates") {  
  steps {  
    script {  
      waitForQualityGate abortPipeline: false, credentialsId: 'sonar-token'  
    }  
  }  
}  
stage("Install Dependencies") {  
  steps {  
    sh 'npm install'  
  }  
}  
stage("OWASP") {  
  steps {  
    dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --  
disableNodeAudit', odcInstallation: 'DP-Check'  
    dependencyCheckPublisher pattern: '**/dependency-check-report.xml'  
  }  
}  
stage("Trivy") {  
  steps {  
    sh 'trivy fs . > trivyfs.txt'  
  }  
}
```

```
}

stage("Build") {

    steps {

        sh 'docker build -t image1 .'

    }

}

stage("Tag & Push") {

    steps {

        script {

            withDockerRegistry(credentialsId: 'docker-password') {

                sh 'docker tag image1 shaikmustafa/mydockerproject:myzomatoimage'

                sh 'docker push shaikmustafa/mydockerproject:myzomatoimage'

            }

        }

    }

}

stage("Scan the Image") {

    steps {

        sh 'trivy image shaikmustafa/mydockerproject:myzomatoimage'

    }

}

stage("Container") {

    steps {

        sh 'docker run -d --name cont1 -p 3000:3000
shaikmustafa/mydockerproject:myzomatoimage'

    }

}
```

}

}

}