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/
- 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

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 Conflgure 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

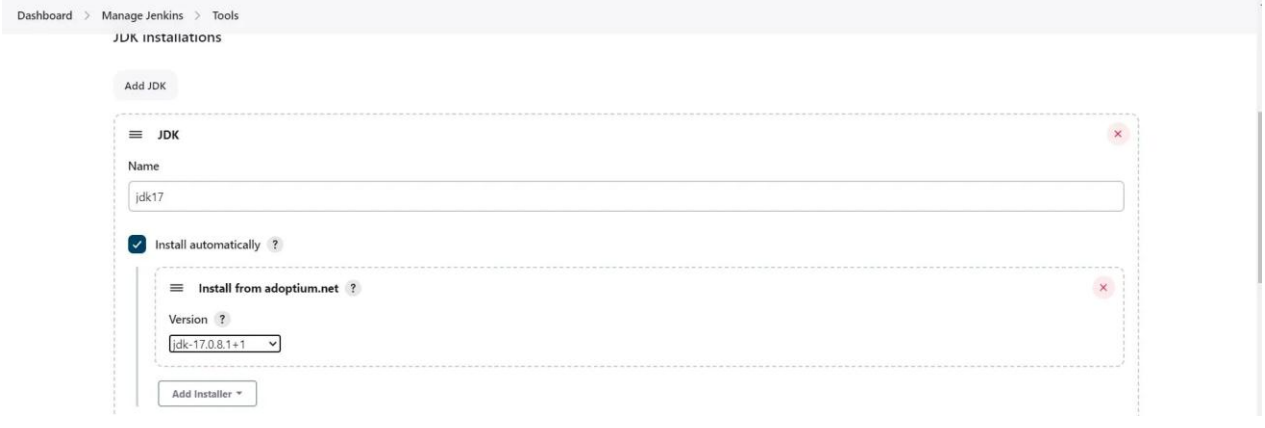
Administration → Configuration →Webhooks

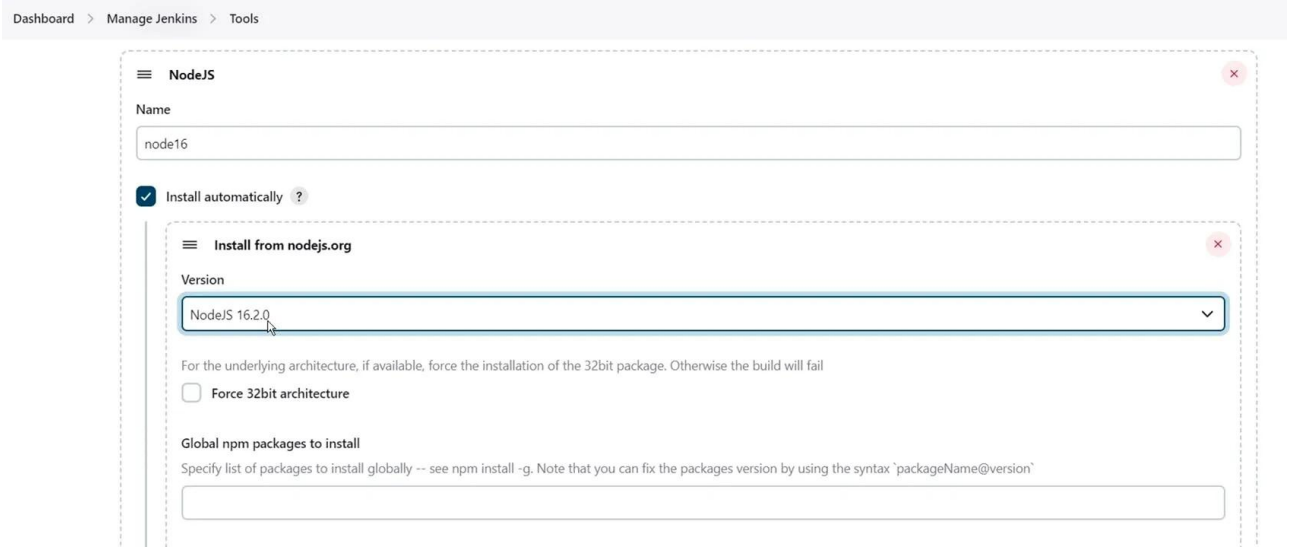
Click on Create

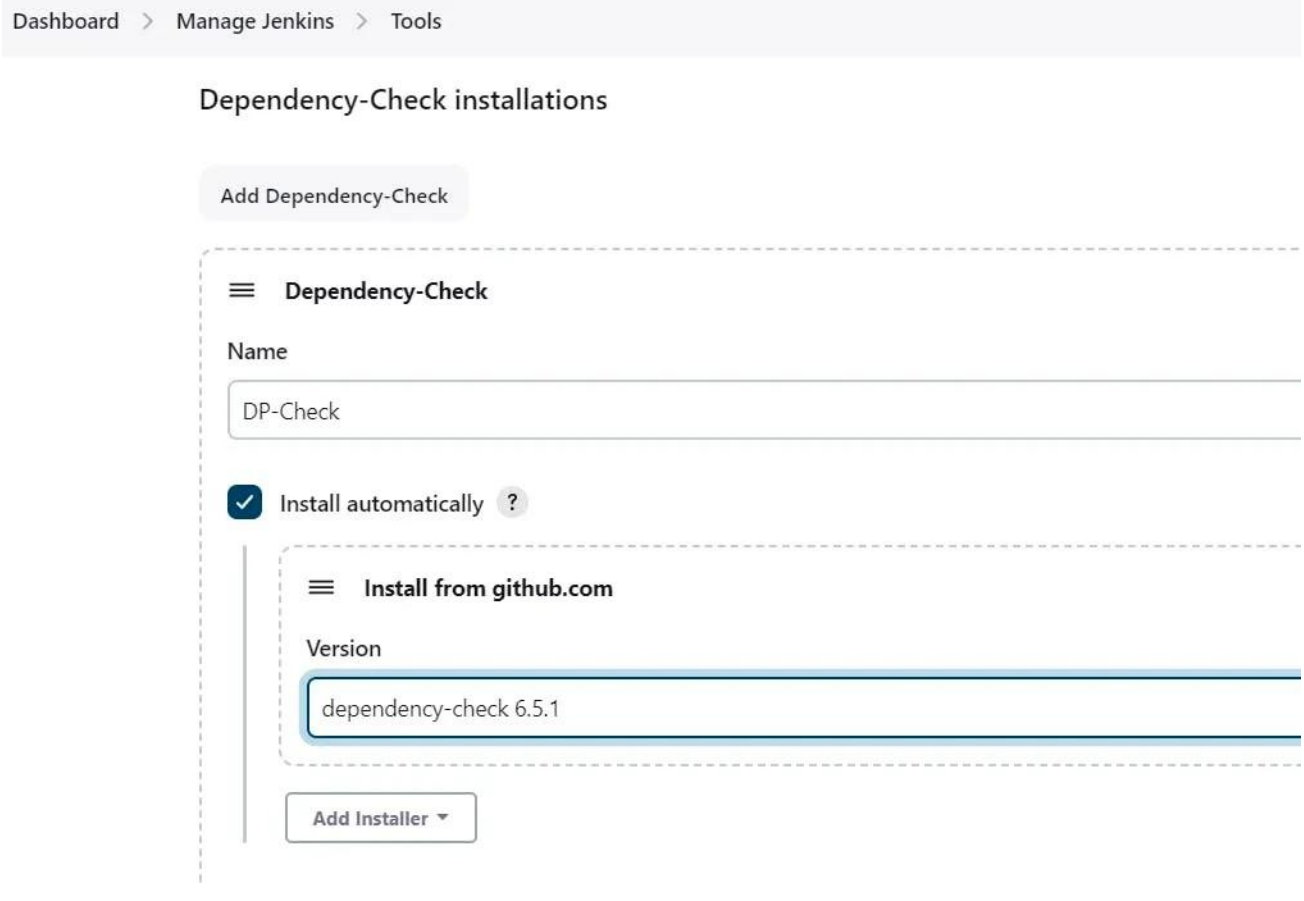
Name: Jenkins

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

Now conflgure NodeJs, Java & DP-Check







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 workspace"){
        steps{
            cleanWs()
        }
    }
    stage("code"){
        steps{
            git 'https://github.com/Varshita5233/Zomato-Project.git'
        }
    }
    stage("sonarqube Analysis"){
        steps{
            withSonarQubeEnv('mysonar') {
                sh """$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=CICDProject \
                    -Dsonar.projectKey=CICDProject"""
            }

        }
    }
        stage("Quality gates"){
            steps{
                script{
                    waitForQualityGate abortPipeline: false, credentialsId: 'sonar-password'
                }
            }
        }
        stage("node"){
            steps{
                sh 'npm install'
            }
        }
        stage("OWASP"){
            steps{
                dependencyCheck additionalArguments: '--scan ./ --disableYarnAudit --disableNodeAudit', odcInstallation: 'Dp-Check'
                dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
            }
        }
        stage("Build Image"){
            steps{
                sh 'docker build -t image1 .'
            }
        }
        stage("Trivy"){
            steps{
                sh 'trivy fs . >> trivyfs.txt'
            }
        }
        stage("Image scan"){
            steps{
                sh 'trivy image image1'
            }
        }
```

```
    stage("Dockerhub"){
        steps{
            script{
            withDockerRegistry(credentialsId: 'docker-password') {
                sh 'docker tag image1 jyotsna2181/cicdprojectimage:v1'
                sh 'docker image push jyotsna2181/cicdprojectimage:v1'
            }
        }
        }
    }
    stage("Deployment"){
        steps{
            sh 'docker run -itd --name zomatoproject -p 9999:3000 jyotsna2181/cicdprojectimage:v1'
        }
    }
  }
}
```