**Complete Jenkins Pipeline Documentation**

This document describes the steps to build a Jenkins pipeline that deploys a Java application to an EKS (Elastic Kubernetes Service) cluster using Maven, SonarQube, Nexus, Docker, DockerHub, and Trivy.

**Overview**

This Jenkins pipeline includes the following stages:

* **Build**: Uses Maven to compile the Java application.
* **Code Quality**: Analyzes code with SonarQube.
* **Artifact Upload**: Uploads the .war file to Nexus.
* **Dockerization**: Builds and pushes a Docker image to DockerHub.
* **Security Scan**: Scans Docker images with Trivy.
* **Deployment**: Deploys to AWS EKS.

**Prerequisites**

* Jenkins installed
* Docker installed
* SonarQube (running in Docker)
* Nexus installed
* Trivy installed
* DockerHub account
* EKS cluster created

**Jenkins Setup**

**Installation**

1. Launch an EC2 instance in AWS.
2. Create a shell script jenkins.sh:

#!/bin/bash

sudo apt update

sudo apt install fontconfig openjdk-17-jre -y

sleep 2

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 -y

sudo apt install jenkins -y

1. Run the script: sh jenkins.sh
2. Allow port 8080 in your instance's security group.
3. Access Jenkins via: http://<public-ip>:8080

**Jenkins Plugin Installation**

Install the following plugins:

* Pipeline
* Git
* Docker Pipeline
* SonarQube Scanner
* Nexus Artifact Uploader
* Slack Notification
* AWS CLI (optional)

**Global Tool Configuration**

* Add JDK 17 (jdk-17)
* Add Maven (maven)
* Configure SonarQube server (sonar)
* Add credentials: DockerHub, GitHub, Nexus, AWS IAM

**Docker Setup**

Create a script docker.sh:

#!/bin/bash

sudo apt-get update

sudo apt-get install -y ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin docker-compose

sudo usermod -aG docker ubuntu

newgrp docker

* + Run it using sh docker.sh.

**SonarQube Setup**

Run in Docker:

docker run --name sonarqube -d -p 9000:9000 sonarqube:latest

* Allow port 9000 in your security group.
* Access: http://<public-ip>:9000
* Create a webhook and token.
* Configure Jenkins > Global Tool Config:

Name: sonar

URL: http://<your-sonar-host>:9000

Token: Stored in Jenkins credentials

**Nexus Setup**

Run in Docker:

For installing Nexus

-run the below commands

sudo apt update -y

sudo apt install openjdk-11-jdk -y

sudo wget https://download.sonatype.com/nexus/3/nexus-3.80.0-06-linux-x86\_64.tar.gz

tar -xvf nexus-3.80.0-06-linux-x86\_64.tar.gz

mv nexus-3.80.0-06 nexus3

sudo mv nexus3 /opt

sudo mv sonatype-work /opt

sudo useradd nexus

cd /opt

sudo chown -R nexus:nexus nexus3

sudo chown -R nexus:nexus /opt/sonatype-work/nexus3

then change to root user

sudo -i

Then we have to create a service that will run in the background

vi /etc/systemd/system/nexus.service

[Unit]

Description=Nexus Repository Manager

After=network.target

[Service]

Type=forking

LimitNOFILE=65536

User=nexus

Group=nexus

ExecStart=/bin/bash /opt/nexus3/bin/nexus start

ExecStop=/bin/bash /opt/nexus3/bin/nexus stop

Restart=on-abort

[Install]

WantedBy=multi-user.target

After that run below commands to start the Nexus

sudo systemctl daemon-reload

sudo systemctl start nexus

sudo systemctl status nexus

* Allow port 8081.
* Access via: http://<public-ip>:8081
* Create a Maven hosted repository (e.g., my-artifact)
* Store Nexus credentials in Jenkins (Nexus-Credentials)

**Trivy Setup**

Create trivy.sh:

#!/bin/bash

sudo apt-get install -y wget apt-transport-https gnupg lsb-release

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

echo "deb https://aquasecurity.github.io/trivy-repo/deb $(lsb\_release -sc) main" | sudo tee /etc/apt/sources.list.d/trivy.list

sudo apt-get update -y

sudo apt-get install -y trivy

Run it using sh trivy.sh.

**DockerHub Setup**

* Create a DockerHub account.
* Store your DockerHub credentials in Jenkins as DockerHub-Credentials.

**EKS Setup**

**AWS CLI Setup**

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

aws --version

aws configure

Provide:

* AWS Access Key ID
* AWS Secret Access Key
* Region (e.g., ap-south-1)
* Output format

Store these as Jenkins credentials (aws-credentials).

**Install eksctl**

curl --silent --location "https://github.com/eksctl-io/eksctl/releases/latest/download/eksctl\_Linux\_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version

**Install kubectl**

curl -LO "https://dl.k8s.io/release/v1.30.1/bin/linux/amd64/kubectl"

chmod +x kubectl

sudo mv kubectl /usr/local/bin/

kubectl version –client

**Install NGINX Ingress Controller**

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.10.1/deploy/static/provider/aws/deploy.yamlchmod +x kubectl

kubectl get pods -n ingress-nginx

**Create EKS Cluster**

eksctl create cluster --name my-cluster --region us-east-1 --zones us-east-1a,us-east-1b --nodegroup-name my-nodes --node-type t3.medium --nodes 2 --nodes-min 1 --nodes-max 3 --managed

**Verify Cluster**

aws eks update-kubeconfig --region ap-south-1 --name my-cluster

kubectl get nodes

kubectl get svc

kubectl get pods --all-namespaces

To delete:

eksctl delete cluster --name my-cluster --region ap-south-1

**Jenkins Declarative Pipeline**

pipeline {

agent any

tools {

jdk 'jdk-17'

maven 'maven'

}

environment {

NEXUS\_VERSION = "nexus3" # provide nexus version you are using

NEXUS\_PROTOCOL = "http" # provide the protocol on which you are running the Nexus

NEXUS\_URL = "public\_IP:8081" # provide the url on which Nexus is running (public\_IP with port number)

NEXUS\_REPOSITORY = "artifact-folder-name" # provide the repository name created in Nexus for storing artifacts

NEXUS\_CREDENTIAL\_ID = "Nexus-Credentials" # provide the ID in which Nexus credentials are stored in Jenkins

ARTVERSION = "${BUILD\_ID}" # It will get from the build

DOCKER\_IMAGE\_NAME = "repo\_name/image\_name"

DOCKER\_IMAGE\_TAG = "${BUILD\_NUMBER}"

}

stages {

stage('Clean Workspace') {

steps {

sh '''

echo "Cleaning up Docker and workspace..."

docker system prune -af || true

rm -rf /tmp/trivy\* || true

df -h

'''

}

}

stage('Git Checkout') {

steps {

checkout scmGit(branches: [[name: '\*/main']], extensions: [], userRemoteConfigs: [[credentialsId: 'GitHub-Credentials', [url:<”github\_url](url:%3c)”>]]) # [<”github\_url](file:///C:\Users\rdine\OneDrive\Desktop\%3c)”> replace with github url

}

}

stage('Maven Build') {

steps {

sh 'mvn clean package'

}

}

stage('Code Analysis with SonarQube') {

steps {

withSonarQubeEnv('sonar') {

sh 'mvn sonar:sonar'

}

timeout(time: 10, unit: 'MINUTES') {

waitForQualityGate abortPipeline: true

}

}

}

stage('Upload Artifact to Nexus') {

steps {

nexusArtifactUploader(

nexusVersion: "${NEXUS\_VERSION}",

protocol: "${NEXUS\_PROTOCOL}",

nexusUrl: "${NEXUS\_URL}",

version: "${ARTVERSION}",

groupId: "com.vprofile",

repository: "${NEXUS\_REPOSITORY}",

credentialsId: "${NEXUS\_CREDENTIAL\_ID}",

artifacts: [[

artifactId: "vprofile",

classifier: "",

file: "target/vprofile-v2.war",

type: "war"

]]

)

}

}

stage('Docker Image Build') {

steps {

echo "Building Docker image: ${DOCKER\_IMAGE\_NAME}:${DOCKER\_IMAGE\_TAG}"

sh "docker build -t ${DOCKER\_IMAGE\_NAME}:${DOCKER\_IMAGE\_TAG} ."

}

}

stage('Docker Image Scan with Trivy') {

steps {

echo "Scanning Docker image with Trivy: ${DOCKER\_IMAGE\_NAME}:${DOCKER\_IMAGE\_TAG}"

sh '''

IMAGE\_NAME="${DOCKER\_IMAGE\_NAME}:${DOCKER\_IMAGE\_TAG}"

echo "Scanning image: $IMAGE\_NAME"

docker run --rm \

-v /var/run/docker.sock:/var/run/docker.sock \

-v $HOME/.cache/trivy:/root/.cache/ \

-v $WORKSPACE:/app \

aquasec/trivy:latest \

image --exit-code 0 --severity CRITICAL,HIGH \

-f json -o /app/trivy-report.json \

"$IMAGE\_NAME"

'''

sh 'ls -lh trivy-report.json'

}

}

stage('Docker Push to DockerHub') {

steps {

script {

echo "Logging into Docker Registry and pushing image: ${DOCKER\_IMAGE\_NAME}:${DOCKER\_IMAGE\_TAG}"

docker.withRegistry('https://index.docker.io/v1/', 'DockerHub-Credentials') {

def image = docker.build("${DOCKER\_IMAGE\_NAME}:${DOCKER\_IMAGE\_TAG}")

image.push()

}

}

}

}

stage('Deploy to EKS') {

steps {

withCredentials([usernamePassword(credentialsId: 'aws-credentials', usernameVariable: 'AWS\_ACCESS\_KEY\_ID', passwordVariable: 'AWS\_SECRET\_ACCESS\_KEY')]) {

sh '''

export AWS\_ACCESS\_KEY\_ID=$AWS\_ACCESS\_KEY\_ID

export AWS\_SECRET\_ACCESS\_KEY=$AWS\_SECRET\_ACCESS\_KEY

export AWS\_DEFAULT\_REGION=ap-south-1

echo "Setting up KUBECONFIG for EKS cluster..."

aws eks update-kubeconfig --region ap-south-1 --name my-cluster

echo "Deploying to Amazon EKS..."

sed -i "s|image: .\*|image: ${DOCKER\_IMAGE\_NAME}:${DOCKER\_IMAGE\_TAG}|g" k8s/deployment.yaml

kubectl apply -f k8s/

'''

}

}

}

}

post {

always {

echo 'Archiving Trivy report and sending Slack notification...'

archiveArtifacts artifacts: 'trivy-report.json'

slackSend (

channel: '<salck\_channel>', # ‘<salck\_channel>’ replace with your channel name

color: currentBuild.currentResult == 'SUCCESS' ? 'good' : 'danger',

message: """\

\*${currentBuild.currentResult}\*: Job <${env.BUILD\_URL}|${env.JOB\_NAME} #${env.BUILD\_NUMBER}\*

\*Branch\*: ${env.GIT\_BRANCH ?: 'N/A'}

\*Commit\*: ${env.GIT\_COMMIT ?: 'N/A'}

\*Duration\*: ${currentBuild.durationString}

\*Details\*: <${env.BUILD\_URL}|Click to view console log>

""".stripIndent()

)

}

}

}