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

3. Run the script: sh jenkins.sh
4. Allow port 8080 in your instance's security group.
5. 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:

docker run --name nexus -d -p 8081:8081 sonatype/nexus3

- 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/$(curl -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

chmod +x kubectl

sudo mv kubectl /usr/local/bin/

kubectl version –client

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">]]) # <"github_url"> 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()
            )
        }
    }
}
```