# **Install Jenkins on Amazon Linux 2 instance:**

#### 

#### 

**Step-1: Launch the basic EC2 instance with 8080 port enabled in the security group.**

**Step-2: Update the software package of the instance using the below command:**

sudo yum update -y

**Step-3: Install java packages and remove the oldest version of java if any:**

sudo yum install java-1.8.0

**Step-4: Add the Jenkins repo:**

sudo wget -O /etc/yum.repos.d/jenkins.repo <http://pkg.jenkins-ci.org/redhat/jenkins.repo>

**Step-5: Import the key file:**

**Note:** Below is the2 hyphens attached to the import keyword.

sudo rpm -- import <https://pkg.jenkins.io/redhat/jenkins.io.key>

**Step-6: Install Jenkins using the below command:**

sudo yum install jenkins -y

**Step-7: Start the Jenkins service:**

sudo service jenkins start

**Step-8: Open the browser and hit the public IP along with the 8080 port.**

**Image for post**

Install Git inside the system and Go to Jenkins -> global tool configuration -> Git -> Git Installations -> Path to Git executable.

Path to Git executable = Type which git in your linux system and you will get the executable

In the build section of the job pick up invoke top-level maven targets and give the maven goal which you wish to execute

After execution you will be able to see the target folder under the workspace of the jenkins

Also try to give poll scm option to check whether the scm is working on peculiar changes by developers

Reference docs for jenkins installation :--

https://medium.com/@kalyanshettynetra/install-jenkins-on-amazon-linux-2-instance-52105b5bc7e1

Reference docs for maven installation :--

<https://docs.aws.amazon.com/neptune/latest/userguide/iam-auth-connect-prerq.html>

Vim jenkins.sh

#########################################################################

sudo yum update -y

sudo yum install java-1.8.0 -y

sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins-ci.org/redhat/jenkins.repo

sudo rpm --import <https://pkg.jenkins.io/redhat/jenkins.io.key>

sudo amazon-linux-extras install epel -y

sudo yum install jenkins -y

sudo service jenkins start

#########################################################################

Vim maven.sh

##########################################################################

sudo wget https://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo

sudo sed -i s/\$releasever/6/g /etc/yum.repos.d/epel-apache-maven.repo

sudo yum install -y apache-maven

sudo yum install java-1.8.0-devel -y

#########################################################################

Vim externalpackages.sh

################################################################

yum install git -y

which git

yum install docker -y

service docker start

#################################################################

Plugins to be installed

[Pipeline Maven Integration](https://plugins.jenkins.io/pipeline-maven)

Sonarqube scanner

Jacoco

Sonar setup :---

Install plugin sonarqube scanner

Go to global tool config and install scanner automatically

Go to configure system and pass installation details



Since sonar uses java 11 only we have install java 11 on the jenkins server and setup a path in the jenkins global tools config

Steps to install java 11

Extract the binary in any path using wget eg in :-- /var/lib/jenkins/

Binary file:-- <https://download.java.net/java/GA/jdk11/13/GPL/openjdk-11.0.1_linux-x64_bin.tar.gz>

Use tar -xvzf <binary file name>

Set the path in the Global tool config by the name java11

Pipeline script for jenkins

pipeline {

agent any

stages {

stage ('cloning'){

steps{

git branch: 'master', url: 'https://github.com/eswanthdev/jpetstore-6.git'

}

}

stage ('Building'){

steps {

sh 'mvn clean package'

}

}

stage ('sonar testing'){

tools {

jdk "java11"

}

environment {

Scanner\_Home = tool 'sonar'

}

steps {

withSonarQubeEnv('sonarqube'){

sh '''$Scanner\_Home/bin/sonar-scanner \

-Dsonar.projectKey=sonar \

-Dsonar.java.binaries=. \

-Dsonar.host.url=http://50.17.160.162:9000/ \

-Dsonar.login=e55e18dbc38f7a2ebbe5c327a3bdbde86089204e'''

}

}

}

stage('jacoco'){

steps {

jacoco()

}

}

}

}

1c3400d6c196dbee51884e114f63bac94de60a53

Installation of sonarqube using docker

yum install docker -y

Service docker start

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

Go with Nexus Artifact uploader:--

# Install nexus artifact uploader plugin

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

pipeline {

agent any

stages {

stage ('cloning'){

steps{

git branch: 'main', url: 'https://github.com/eswanthdev/Maven-ci.git'

}

}

stage ('Building'){

steps {

sh 'mvn clean package'

}

}

stage ('sonar testing'){

steps {

sh 'mvn sonar:sonar \

-Dsonar.projectKey=test \

-Dsonar.host.url=http://52.56.247.95:9000/ \

-Dsonar.login=d6a473680a12a15e9de3e3ad3a59a4e142cd6dda'

}

}

stage('jacoco'){

steps {

jacoco()

}

}

stage('nexus') {

steps {

nexusArtifactUploader artifacts: [

[

artifactId: 'project1',

classifier: '',

file: 'target/project1-1.0.0.jar',

type: 'jar'

]

],

credentialsId: 'nexus3',

groupId: 'wipro',

nexusUrl: '172.31.35.114:8081',

nexusVersion: 'nexus3',

protocol: 'http',

repository: 'testrepo',

version: '1.0.0'

}

}

}

}

**Docker**

Docker Installation

Yum install docker -y

Service docker start

Work with ec2-user

Vim Dockerfile

# Centos based container with Java and Tomcat

**FROM centos:centos7**

**MAINTAINER eswanth**

**# Install prepare infrastructure**

**RUN yum -y update && \**

**yum -y install wget && \**

**yum -y install tar**

**# Prepare environment**

**ENV JAVA\_HOME /usr/java/latest**

**ENV CATALINA\_HOME /opt/tomcat**

**ENV PATH $PATH:$JAVA\_HOME/bin:$CATALINA\_HOME/bin:$CATALINA\_HOME/scripts**

**# Install Oracle Java15**

**ENV JAVA\_VERSION 15.0.1**

**ENV JAVA\_BUILD 15.0.1+9**

**ENV JAVA\_DL\_HASH 51f4f36ad4ef43e39d0dfdbaf6549e32**

**RUN wget --no-check-certificate --no-cookies --header "Cookie: oraclelicense=accept-securebackup-cookie" \**

**https://edelivery.oracle.com/otn-pub/java/jdk/${JAVA\_BUILD}/${JAVA\_DL\_HASH}/jdk-${JAVA\_VERSION}\_linux-x64\_bin.rpm && \**

**yum -y localinstall jdk\***

**# Install Tomcat**

**ENV TOMCAT\_MAJOR 9**

**ENV TOMCAT\_VERSION 9.0.39**

**RUN wget http://mirror.linux-ia64.org/apache/tomcat/tomcat-${TOMCAT\_MAJOR}/v${TOMCAT\_VERSION}/bin/apache-tomcat-${TOMCAT\_VERSION}.tar.gz && \**

**tar -xvf apache-tomcat-${TOMCAT\_VERSION}.tar.gz && \**

**rm apache-tomcat\*.tar.gz && \**

**mv apache-tomcat\* ${CATALINA\_HOME}**

**RUN chmod +x ${CATALINA\_HOME}/bin/\*sh**

**# Create Tomcat admin user**

**ADD create\_admin\_user.sh $CATALINA\_HOME/scripts/create\_admin\_user.sh**

**ADD tomcat.sh $CATALINA\_HOME/scripts/tomcat.sh**

**RUN chmod +x $CATALINA\_HOME/scripts/\*.sh**

**# Create tomcat user**

**RUN groupadd -r tomcat && \**

**useradd -g tomcat -d ${CATALINA\_HOME} -s /sbin/nologin -c "Tomcat user" tomcat && \**

**chown -R tomcat:tomcat ${CATALINA\_HOME}**

**WORKDIR /opt/tomcat**

**EXPOSE 8080**

**EXPOSE 8009**

**USER tomcat**

**CMD ["tomcat.sh"]**

docker build -t <name of the image> .

If you wish to run the container then

docker run -d --name <name of the <container> <imagename>

**Checking the container commits**

Create the container with the base image of centos and then do some changes in it

And commit it using

sudo docker commit <containerid> testim

**Checking the Copy from Host to container**

docker cp <source path> containerid:<Destpath>

ECR and EKS

Install kubectl on the system

<https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html>

Pull the nginx image from the hub.docker.com

Authenticate with ecr

By following below steps

Aws configure

---- pass the access key and secret key and region as you wish to work with

Create an ECR repo in the region given above

Click on the view push commands

-- execute those to push the nginx images

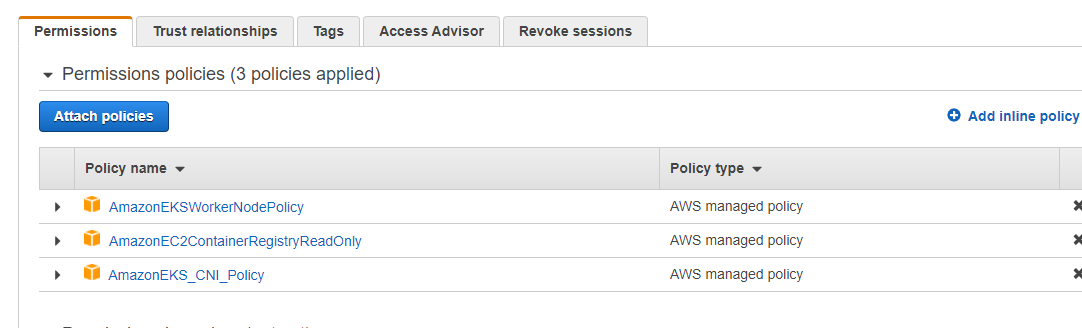
Create an EKS cluster

Check the status by using

aws eks --region us-east-1 describe-cluster --name <reponame>--query cluster.status

After the cluster state is active then create a nodepool

And it will ask for the permissions give the permissions mentioned below



Update the kubeconfig in to the server in which you have setup (brings the master configs to the local server)

aws eks --region eu-west-2 update-kubeconfig --name testcluster

aws eks --region us-east-1 update-kubeconfig --name test-eks