**SESSION - 77**

**------------------**

--> Jenkins is a continuous integration

--> Create the servers --> launch the instances

--> connect the servers

--> Configure storage take 50GB

--> Give this one user-data then it will download automatically jenkins.

**cicd-tools/jenkins.sh**

#!/bin/bash

#resize disk from 20GB to 50GB

growpart /dev/nvme0n1 4

lvextend -L +10G /dev/RootVG/rootVol

lvextend -L +10G /dev/mapper/RootVG-varVol

lvextend -l +100%FREE /dev/mapper/RootVG-varTmpVol

xfs\_growfs /

xfs\_growfs /var/tmp

xfs\_growfs /var

curl -o /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

yum install fontconfig java-21-openjdk -y

yum install jenkins -y

systemctl daemon-reload

systemctl enable jenkins

systemctl start jenkins

--> create one more server -- jenkins-agent --> install java

**cicd-tools/jenkins-agent.sh**

#!/bin/bash

#resize disk from 20GB to 50GB

growpart /dev/nvme0n1 4

lvextend -L +10G /dev/mapper/RootVG-homeVol

lvextend -L +10G /dev/mapper/RootVG-varVol

lvextend -l +100%FREE /dev/mapper/RootVG-varTmpVol

xfs\_growfs /home

xfs\_growfs /var/tmp

xfs\_growfs /var

yum install java-21-openjdk -y

# Terraform Installation

yum install -y yum-utils

yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo

yum -y install terraform

# NodeJs installation

dnf module disable nodejs -y

dnf module enable nodejs:20 -y

dnf install nodejs -y

yum install zip -y

# docker

yum install -y yum-utils

yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo

yum install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

systemctl start docker

systemctl enable docker

usermod -aG docker ec2-user

curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3

chmod 700 get\_helm.sh

./get\_helm.sh

--> Take 50GB

--> Launch instances

--> create route53 also -- jenkins take public ip address

--> jenkins-agent - internal pipe address - i can give public ip

--> Connect jenkins -- installed plugins

--> Create a nodes

--> Number of execution -- 03

--> Remote root dirctory -- /home/ec2-user/jenkins-agent/

--> LABELS -- AGENT - 1

--> launch agent via SSH

--> Credentials -- ssh-auth

--> Host : jenkins-agent-daws-shankran.site

--> Let us start with catalogue -- catalogue only backend updation

--> Create a repo -- Catalogue

--> Plugins utility steps -- use this one

**Jenkinsfile.bkp**

pipeline {

agent {

label 'AGENT-1'

}

environment {

appVersion = ''

REGION = "us-east-1"

ACC\_ID = "315069654700"

PROJECT = "roboshop"

COMPONENT = "catalogue"

}

options {

timeout(time: 30, unit: 'MINUTES')

disableConcurrentBuilds()

}

parameters {

booleanParam(name: 'deploy', defaultValue: false, description: 'Toggle this value')

}

// Build

stages {

stage('Read package.json') {

steps {

script {

def packageJson = readJSON file: 'package.json'

appVersion = packageJson.version

echo "Package version: ${appVersion}"

}

}

}

stage('Install Dependencies') {

steps {

script {

sh """

npm install

"""

}

}

}

stage('Unit Testing') {

steps {

script {

sh """

echo "unit tests"

"""

}

}

}

/\* stage('Sonar Scan') {

environment {

scannerHome = tool 'sonar-7.2'

}

steps {

script {

// Sonar Server envrionment

withSonarQubeEnv(installationName: 'sonar-7.2') {

sh "${scannerHome}/bin/sonar-scanner"

}

}

}

} \*/

// Enable webhook in sonarqube server and wait for results

/\* stage("Quality Gate") {

steps {

timeout(time: 1, unit: 'HOURS') {

waitForQualityGate abortPipeline: true }

}

} \*/

stage('Check Dependabot Alerts') {

environment {

GITHUB\_TOKEN = credentials('github-token')

}

steps {

script {

// Fetch alerts from GitHub

def response = sh(

script: """

curl -s -H "Accept: application/vnd.github+json" \

-H "Authorization: token ${GITHUB\_TOKEN}" \

https://api.github.com/repos/daws-84s/catalogue/dependabot/alerts

""",

returnStdout: true

).trim()

// Parse JSON

def json = readJSON text: response

// Filter alerts by severity

def criticalOrHigh = json.findAll { alert ->

def severity = alert?.security\_advisory?.severity?.toLowerCase()

def state = alert?.state?.toLowerCase()

return (state == "open" && (severity == "critical" || severity == "high"))

}

if (criticalOrHigh.size() > 0) {

error "❌ Found ${criticalOrHigh.size()} HIGH/CRITICAL Dependabot alerts. Failing pipeline!"

} else {

echo "✅ No HIGH/CRITICAL Dependabot alerts found."

}

}

}

}

stage('Docker Build') {

steps {

script {

withAWS(credentials: 'aws-creds', region: 'us-east-1') {

sh """

aws ecr get-login-password --region ${REGION} | docker login --username AWS --password-stdin ${ACC\_ID}.dkr.ecr.us-east-1.amazonaws.com

docker build -t ${ACC\_ID}.dkr.ecr.us-east-1.amazonaws.com/${PROJECT}/${COMPONENT}:${appVersion} .

docker push ${ACC\_ID}.dkr.ecr.us-east-1.amazonaws.com/${PROJECT}/${COMPONENT}:${appVersion}

"""

}

}

}

}

stage('Check Scan Results') {

steps {

script {

withAWS(credentials: 'aws-creds', region: 'us-east-1') {

// Fetch scan findings

def findings = sh(

script: """

aws ecr describe-image-scan-findings \

--repository-name ${PROJECT}/${COMPONENT} \

--image-id imageTag=${appVersion} \

--region ${REGION} \

--output json

""",

returnStdout: true

).trim()

// Parse JSON

def json = readJSON text: findings

def highCritical = json.imageScanFindings.findings.findAll {

it.severity == "HIGH" || it.severity == "CRITICAL"

}

if (highCritical.size() > 0) {

echo "❌ Found ${highCritical.size()} HIGH/CRITICAL vulnerabilities!"

currentBuild.result = 'FAILURE'

error("Build failed due to vulnerabilities")

} else {

echo "✅ No HIGH/CRITICAL vulnerabilities found."

}

}

}

}

}

stage('Trigger Deploy') {

when{

expression { params.deploy }

}

steps {

script {

build job: 'catalogue-cd',

parameters: [

string(name: 'appVersion', value: "${appVersion}"),

string(name: 'deploy\_to', value: 'dev')

],

propagate: false, // even SG fails VPC will not be effected

wait: false // VPC will not wait for SG pipeline completion

}

}

}

}

post {

always {

echo 'I will always say Hello again!'

deleteDir()

}

success {

echo 'Hello Success'

}

failure {

echo 'Hello Failure'

}

}

}

--> settings --> webhook --> <http://jenkins.daws-84s.site:8080/github-webhook>/

--> update webhook

--> Github hook trigger for GITscm polling

--> Amazon ECR --> Private Registry --> Scanning

Clone

Reading version

Install dependencies

docker build -t URL/catalogue:appVersion

--> Amazon CR --> Private registry --> Repositories --> create private repositories --> create

jenkins pipeline --> AWS credentials

--> Jenkins --> Manage Jenkins --> Credentials --> System --> Global Credentials

--> AWS credentials

--> ID : aws-auth

--> Description : aws-creas

--> AWS steps jenkins

--> api hitting

--> Install AWS steps

--> I need aws environment for running multiple commands.

--> Jenkins pipeline with aws

--> Scanning -> scanning configuration--> basic scanning

--> Scan and push

--> sudo systemctl start docker

--> sudo systemtl enable docker

--> sudo usermod -aG docker ec2-user

**Testing**

**=======**

unit testing --> this should be done by developer

functions are basic blocks of the programming

login(username, password){

}

**functional testing --> DEV**

**=================**

either developers/testers

login(username, password){

connectSQL()

checkUserName()

checkPassword()

}

**integration testing --> SIT/UAT/QA**

**================**

testing team writes integration test cases

catalogue cart

cart is calling catalogue

--> sudo yum install -y yum-utils

--> sudo yum-config-manager --add-repo <https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo>

--> sudo yum -y install terraform

--> Jenkins --> new item --> output

--> sudo systemctl restart jenkins