**SESSION - 78**

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

--> 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

upstream downstream

--> Security group depended on SG

1. **sg/Jenkinsfile**

pipeline {

agent {

label 'AGENT-1'

}

environment {

appVersion = ''

REGION = "us-east-1"

ACC\_ID = "315069654700"

PROJECT = "roboshop"

}

options {

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

disableConcurrentBuilds()

ansiColor('xterm')

}

/\* parameters {

string(name: 'PERSON', defaultValue: 'Mr Jenkins', description: 'Who should I say hello to?')

text(name: 'BIOGRAPHY', defaultValue: '', description: 'Enter some information about the person')

booleanParam(name: 'TOGGLE', defaultValue: true, description: 'Toggle this value')

choice(name: 'CHOICE', choices: ['One', 'Two', 'Three'], description: 'Pick something')

password(name: 'PASSWORD', defaultValue: 'SECRET', description: 'Enter a password')

} \*/

// Build

stages {

stage('init') {

steps {

script {

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

sh """

cd 10-sg

terraform init -reconfigure

"""

}

}

}

}

stage('plan') {

steps {

script {

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

sh """

cd 10-sg

terraform plan

"""

}

}

}

}

stage('apply') {

steps {

script {

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

sh """

cd 10-sg

terraform apply -auto-approve

"""

}

}

}

}

stage('Bastion EKS ACM ALB') {

parallel {

stage('Trigger Bastion') {

steps {

script {

build job: '20-bastion',

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

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

}

}

}

stage('Trigger EKS') {

steps {

script {

build job: '80-eks',

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

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

}

}

}

stage('ACM') {

steps {

script {

build job: '60-acm',

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

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

}

}

}

}

}

stage('Trigger ALB') {

steps {

script {

build job: '70-frontend-alb',

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'

}

}

}

--> Repo Created

--> backend repo -> CI

backend deploy -> CD

--> Let’s create helm chart

**Catalogue-cd/Chart.yaml**

apiVersion: v2

name: catalogue

version: 1.0.0 # chart version

description: catalogue helm chart description

appVersion: v1 # application version

**Catalogue-cd/Jenkinsfile**

pipeline {

agent {

label 'AGENT-1'

}

environment {

appVersion = ''

REGION = "us-east-1"

ACC\_ID = "160885265516"

PROJECT = "roboshop"

COMPONENT = "catalogue"

}

options {

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

disableConcurrentBuilds()

}

parameters {

string(name: 'appVersion', description: 'Image version of the application')

choice(name: 'deploy\_to', choices: ['dev', 'qa', 'prod'], description: 'Pick the Environment')

}

// Build

stages {

stage('Check Status'){

steps{

script{

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

def deploymentStatus = sh(returnStdout: true, script: "kubectl rollout status deployment/catalogue --timeout=30s -n $PROJECT || echo FAILED").trim()

if (deploymentStatus.contains("successfully rolled out")) {

echo "Deployment is success"

} else {

sh """

helm rollback $COMPONENT -n $PROJECT

sleep 20

"""

def rollbackStatus = sh(returnStdout: true, script: "kubectl rollout status deployment/catalogue --timeout=30s -n $PROJECT || echo FAILED").trim()

if (rollbackStatus.contains("successfully rolled out")) {

error "Deployment is Failure, Rollback Success"

}

else{

error "Deployment is Failure, Rollback Failure. Application is not running"

}

}

}

}

}

}

stage('Deploy') {

steps {

script {

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

sh """

aws eks update-kubeconfig --region $REGION --name "$PROJECT-${params.deploy\_to}"

kubectl get nodes

kubectl apply -f 01-namespace.yaml

sed -i "s/IMAGE\_VERSION/${params.appVersion}/g" values-${params.deploy\_to}.yaml

#helm upgrade --install $COMPONENT -f values-${params.deploy\_to}.yaml -n $PROJECT .

kubectl apply -f application.yaml

"""

}

}

}

}

// API Testing

stage('Functional Testing'){

when{

expression { params.deploy\_to = "dev" }

}

steps{

script{

echo "Run functional test cases"

}

}

}

// All components testing

stage('Integration Testing'){

when{

expression { params.deploy\_to = "qa" }

}

steps{

script{

echo "Run Integration test cases"

}

}

}

stage('PROD Deploy') {

when{

expression { params.deploy\_to = "prod" }

}

steps {

script {

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

sh """

echo "get cr number"

echo "check with in the deployment window"

echo "is CR approved"

echo "trigger PROD deploy"

"""

}

}

}

}

}

post {

always {

echo 'I will always say Hello again!'

deleteDir()

}

success {

echo 'Hello Success'

}

failure {

echo 'Hello Failure'

}

}

}

1. create infra

2. EKS setup

allow jenkins agent sg in eks control plane

create namespace

create DB pods

mongodb redis mysql rabbitmq

3. we can deploy applications

DEV --> QA --> CR --> PROD