**RAGULANDIRAN M – 22CSR157 || KONGU ENGINNERING COLLEGE**

**DAY 5 Tasks – minikube deployment, Terraform**

**Minikube Deployment:**

**Step 1**: Config file updation

**Sudo visudo:** update this, jenkins ALL=(ALL) NOPASSWD: ALL

Data updation in config file:

* sudo cat /home/ragu\_ubuntu/.minikube/ca.crt
* sudo cat /home/ragu\_ubuntu/.minikube/ca.crt | base64 -w 0; echo
* sudo cat /home/ragu\_ubuntu/.minikube/profiles/minikube/client.crt | base64 -w 0; echo
* sudo cat /home/ragu\_ubuntu/.minikube/profiles/minikube/client.key | base64 -w 0; echo

A screenshot of a computer

AI-generated content may be incorrect.

**step 2:** Pipeline script for minikube deployment

pipeline {

agent any

tools {maven "maven"}

stages {

stage('SCM') {

steps {

git branch: 'main', url: 'https://github.com/Ragu162004/web-app.git'

}

}

stage('Build-clean') {

steps {

sh 'mvn clean'

}

}

stage('Build-validate') {

steps {

sh 'mvn validate'

}

}

stage('Build-complie') {

steps {

sh 'mvn compile'

}

}

stage('Build-package') {

steps {

sh 'mvn package'

}

}

stage('build to images') {

steps {

script {

sh 'docker build -t ragu162004/webapp1 .'

}

}

}

stage('push to hub') {

steps {

script {

withDockerRegistry(credentialsId: 'docker\_cred', toolName: 'docker', url: 'https://index.docker.io/v1/') {

sh 'docker push ragu162004/webapp1'

}

}

}

}

stage('deploy') {

steps {

script {

withKubeCredentials(kubectlCredentials: [[caCertificate: '', clusterName: 'minikube', contextName: 'minikube', credentialsId: 'kube\_cred', namespace: '', serverUrl: 'https://192.168.39.226:8443']]) {

sh 'kubectl apply -f deploy.yml --validate=false'

}

}

}

}

}

}

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Stage View of Pipeline:**

A screenshot of a computer

AI-generated content may be incorrect.

* minikube service my-service
* curl http://192.168.49.2:30002/my-web/

A screenshot of a computer

AI-generated content may be incorrect.

**Terraform:**

#terraform init

#terraform validate

#terraform plan

#terraform apply

#terraform destroy

terraform {

required\_providers {

aws = {

source = "hashicorp/aws"

version = "5.92.0"

}

}

}

provider "aws" {

region = "us-east-1"

}

resource "aws\_vpc" "myvpc" {

cidr\_block = "10.0.0.0/16"

tags = {

Name = "demovpc"

}

}

resource "aws\_subnet" "pubsub" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "10.0.1.0/24"

availability\_zone = "us-east-1a"

tags = {

Name = "sn1"

}

}

resource "aws\_subnet" "pub\_sub" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "10.0.2.0/24"

availability\_zone = "us-east-1a"

tags = {

Name = "sn1"

}

}

resource "aws\_subnet" "prisub" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "10.0.3.0/24"

availability\_zone = "us-east-1a"

tags = {

Name = "sn1"

}

}

resource "aws\_subnet" "pri\_sub" {

vpc\_id = aws\_vpc.myvpc.id

cidr\_block = "10.0.4.0/24"

availability\_zone = "us-east-1a"

tags = {

Name = "sn1"

}

}

resource "aws\_internet\_gateway" "tfigw" {

vpc\_id = aws\_vpc.myvpc.id

tags = {

Name = "tfigw"

}

}

resource "aws\_route\_table" "tfpubrt" {

vpc\_id = aws\_vpc.myvpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_internet\_gateway.tfigw.id

}

tags = {

Name = "tfpublicroute"

}

}

resource "aws\_route\_table\_association" "pubsn1" {

subnet\_id = aws\_subnet.pubsub.id

route\_table\_id = aws\_route\_table.tfpubrt.id

}

resource "aws\_route\_table\_association" "pubsn2" {

subnet\_id = aws\_subnet.pub\_sub.id

route\_table\_id = aws\_route\_table.tfpubrt.id

}

resource "aws\_eip" "tfeip" {

domain = "vpc"

}

resource "aws\_nat\_gateway" "tfnat" {

allocation\_id = aws\_eip.tfeip.id

subnet\_id = aws\_subnet.pub\_sub.id

tags = {

Name = "gw NAT"

}

}

resource "aws\_route\_table" "tfprirt" {

vpc\_id = aws\_vpc.myvpc.id

route {

cidr\_block = "0.0.0.0/0"

gateway\_id = aws\_nat\_gateway.tfnat.id

}

tags = {

Name = "tfprivateroute"

}

}

resource "aws\_route\_table\_association" "prisn3" {

subnet\_id = aws\_subnet.prisub.id

route\_table\_id = aws\_route\_table.tfprirt.id

}

resource "aws\_route\_table\_association" "prisn4" {

subnet\_id = aws\_subnet.pri\_sub.id

route\_table\_id = aws\_route\_table.tfprirt.id

}

resource "aws\_security\_group" "allow\_tfsg" {

name = "allow\_tfsg"

description = "Allow TLS inbound traffic"

vpc\_id = aws\_vpc.myvpc.id

ingress {

description = "HTTPS "

from\_port = 443

to\_port = 443

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

description = "HTTP "

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

ingress {

description = "SSH"

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"]

}

egress {

from\_port = 0

to\_port = 0

protocol = "-1"

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

Name = "TfsecurityGroup"

}

}

resource "aws\_instance" "pub\_ins" {

ami = "ami-0fc5d935ebf8bc3bc"

instance\_type = "t2.micro"

subnet\_id = aws\_subnet.pub\_sub.id

vpc\_security\_group\_ids = [aws\_security\_group.allow\_tfsg.id]

key\_name = ""

associate\_public\_ip\_address = "true"

}