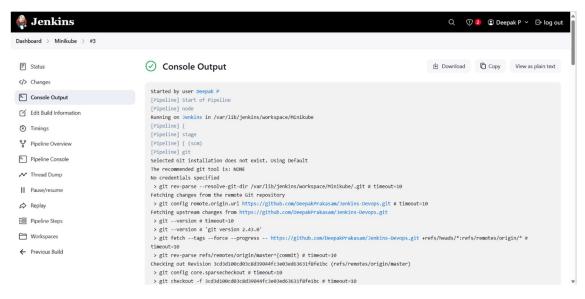
Day 5 Task: Minikube, Kubernetes & Terraform

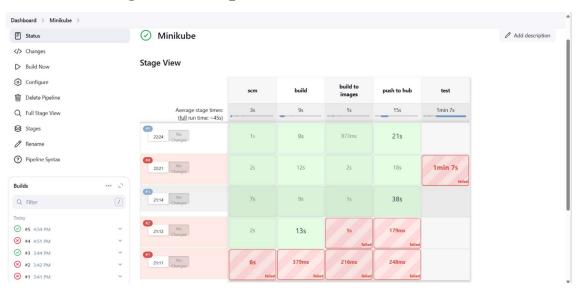
- Practiced Minikube and Kubernetes deployment
- Worked with Terraform for infrastructure automation

SSH-service output

Minikube pipeline console output



Minikube stage view output



Command prompt output



Terraform Cheatsheet



Jenkins and System Management Commands:

1. Grant Jenkins Sudo Permissions (without password):

• *jenkins ALL=(ALL) NOPASSWD: ALL*

2. Restarting Services:

- sudo systemctl restart ssh.service
- sudo systemctl restart sshd.service

3. Install and Manage OpenSSH:

- sudo apt update
- sudo apt install openssh-server
- sudo systemctl status ssh

4. File Permissions and Access:

• sudo chmod 666 /var/run/docker.sock

5. Minikube Certificate Base64 Encoding:

• cat/home/david/.minikube/ca.crt | base64 -w 0; echo

6. Service URL Access via Minikube:

minikube service my-service --url | xargs curl

Jenkins Pipeline (CI/CD):

1. SCM (Source Code Management):

- Get the branch from Git:
 - git branch: ''

2. Build Phases (using Maven):

- Clean:
- mvn clean
- Validate:
- mvn validate
- Compile:
- mvn compile
- Run Tests:
 - mvn test
- Package:
- mvn package

3. Docker Operations:

- Build Docker image:
 - docker build -t
- Push image to Docker Hub:
 - docker push <image_name>

4. Deploy to Kubernetes (Minikube):

- Apply Kubernetes deployment YAML:
 - kubectl apply -f deployment.yml --validate=false
- Test deployed service:
 - minikube service my-service --url | xargs curl

Terraform Code:

1. Terraform AWS Provider Configuration:

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "5.92.0"
    }
  }
  provider "aws" {
    region = "us-east-1"
  }
}
```

2. AWS Resources Configuration:

• VPC:

```
resource "aws_vpc" "myvpc" {
    cidr_block = "10.0.0.0/16"
    tags = {
        Name = "demovpc"
    }
}
```

• Subnet:

```
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" }
}
```

• Internet Gateway:

```
resource "aws_internet_gateway" "tfigw" {
   vpc_id = aws_vpc.myvpc.id
   tags = { Name = "tfigw" }
}
```

• Route Table and Association:

```
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
}
```

• Security Group:

```
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" }
}
```

• EC2 Instance:

```
resource "aws_instance" "pub_ins" {
    ami = "ami-0fc5d935ebf8bc3bc"
    instance_type = "t2.micro"
    subnet_id = aws_subnet.pubsub.id
    vpc_security_group_ids =
    [aws_security_group.allow_tfsg.id]
    key_name = "David"
    associate_public_ip_address = true
}
```

- 3. Commands:
- Terraform initialization:
 - terraform init
- Validate configuration:
 - terraform validate
- Plan the changes:
 - terraform plan
- Apply the changes:
 - terraform validate
- Destroy the infrastructure:
 - terraform destroy

Jenkins Pipeline Script and Breakdown:

```
pipeline {
agent any
stages {
  // SCM Stage: Checkout the code from the Git repository
  stage('scm') {
    steps {
       git url: 'https://github.com/DeepakPrakasam/Jenkins-Devops.git'
  // Build Stage: Maven build steps
  stage('build') {
    steps {
       sh "mvn clean"
       sh "mvn install"
  // Docker Image Build Stage
  stage('build to images') {
    steps {
       script {
         sh 'docker build -t deepakp2003/simpleapplication .'
```

```
}
    // Push Docker Image to Docker Hub
    stage('push to hub') {
       steps {
         script {
            withDockerRegistry(credentialsId: 'Deepak Docker Cred', url:
'https://index.docker.io/v1/') {
              sh 'docker push deepakp2003/simpleapplication'
    // Test/Deploy Stage (Uncommented)
    stage('test') {
       steps {
         withKubeConfig(caCertificate: ", clusterName: ", contextName: ",
credentialsId: 'minikube', namespace: ", restrictKubeConfigAccess: false,
serverUrl: 'https://192.168.39.226:8443') {
            sh 'kubectl apply -f deployment.yml --validate=false'
```

Breakdown of the pipeline:

1. SCM Stage:

• It pulls the source code from the GitHub repository.

2. Build Stage:

- Runs mvn clean to clean up any existing builds.
- Then it runs mvn install to build and install the dependencies.

3. Build Docker Image Stage:

 Uses Docker to build an image for the application and tags it as deepakp2003/simpleapplication.

4. Push Docker Image to Docker Hub:

- Logs in to Docker Hub using the credentials stored in Jenkins (Deepak Docker Cred).
- Pushes the image deepakp2003/simpleapplication to Docker Hub.

5. Test Stage:

- Deploys the application to a Kubernetes cluster using kubectl apply and the deployment.yml file.
- This stage assumes you're using Minikube or a Kubernetes cluster and have the minikube credentials set up in Jenkins.