DevOps-Swiggy-Clone-Project-Using-Terraform-Jenkins-SonarQube-K8S

Step 1:Terraform to create EC2 instance.

Install terraform and aws cli on your local system. Add your aws secret key and access key.

Now start using terraform .

While creating ec2 instance using terraform we also add shell script in user data to install jdk,jenkins,docker,soanarcube as container,trivy.

provider.tf

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}


# Configure the AWS Provider
provider "aws" {
  region = "ap-south-1"          #change region as per you requirement
}
```

main.tf

```
resource "aws_instance" "web" {
```

```
  ami                         = "ami-0287a05f0ef0e9d9a"          #change ami id for different
region
  instance_type               = "t2.large"
  key_name                    = "Linux-VM-Key7"                  #change key name as per
your setup
  vpc_security_group_ids = [aws_security_group.Jenkins-VM-SG.id]
  user_data                   = templatefile("./install.sh", {})

  tags = {
    Name = "Jenkins-SonarQube"
  }

  root_block_device {
    volume_size = 40
  }
}

resource "aws_security_group" "Jenkins-VM-SG" {
  name          = "Jenkins-VM-SG"
  description = "Allow TLS inbound traffic"

  ingress = [
    for port in [22, 80, 443, 8080, 9000, 3000] : {
      description      = "inbound rules"
      from_port        = port
      to_port          = port
```

```
        protocol                = "tcp"
        cidr_blocks             = ["0.0.0.0/0"]
        ipv6_cidr_blocks = []
        prefix_list_ids    = []
        security_groups    = []
        self                    = false
      }
  ]


  egress {
      from_port      = 0
      to_port        = 0
      protocol       = "-1"
      cidr_blocks = ["0.0.0.0/0"]
  }


  tags = {
      Name = "Jenkins-VM-SG"
  }
}
```

install.sh

```
#!/bin/bash

sudo apt update -y

wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee
/etc/apt/keyrings/adoptium.asc
```

```
echo "deb [signed-by=/etc/apt/keyrings/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list

sudo apt update -y

sudo apt install temurin-17-jdk -y

/usr/bin/java --version

curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keyrings/jenkins-keyring.asc > /dev/null

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update -y

sudo apt-get install jenkins -y

sudo systemctl start jenkins

sudo systemctl status jenkins


##Install Docker and Run SonarQube as Container

sudo apt-get update

sudo apt-get install docker.io -y

sudo usermod -aG docker ubuntu

sudo usermod -aG docker jenkins

newgrp docker

sudo chmod 777 /var/run/docker.sock

docker run -d --name sonar -p 9000:9000 sonarqube:lts-community


#install trivy

sudo apt-get install wget apt-transport-https gnupg lsb-release -y
```

wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null

echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list

sudo apt-get update

sudo apt-get install trivy -y

Step 2: Configure jenkins with install reqired plugins.

Open jenkins in browser -> <public ip of ec2>:8080

Install thease plugins(Eclipse temurin installer,sonarcube scanner,sonar quality gates, quality gates, nodejs,docker,docker commons,docker pipeline,docker api,docker build step)

Add installations under tools(1.nodejs installations,2.jdk installations(add installer as adaptium.net),3.Docker installations(download from docker.com),4.Sonarcube scanner installations)

Step 3: Configure sonarcube and integrate with jenkins

Open soanr cube in browser-> <punlic ip of ec2>:9000

default user and password : admin

generate tioke in soanrcube

add token under credentials in jenkisn

add sonarcube installations in jenkisn(url:http:<private ip of ec2>:9000, select soanrcube token)

go to sonarcube dashobaoard->click create quality gate->save

)

go to sonarcube dashobaoard->click adminstaration->create webhook(Name:jenkins,url: http//<private ip of ec2>:8080/sonarcube-webhook/)

Step 4: Create jenkins pipeline to build and push docker image to dockerhub

Create jeniisn pipeline project->discard old build, max builds 2-> paste below pipleline script

```
pipeline{
    agent any

    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME=tool 'sonarqube-scanner'
    }

    stages {
        stage('Clean Workspace'){
            steps{
                cleanWs()
            }
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/Ashfaque-9x/a-swiggy-clone.git'
            }
        }
        stage("Sonarqube Analysis "){
            steps{
```

```
                withSonarQubeEnv('SonarQube-Server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Swiggy-CI \
                    -Dsonar.projectKey=Swiggy-CI '''
                }
            }
        }
        stage("Quality Gate"){
            steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId: 'SonarQube-Token'
                }
            }
        }
        stage('Install Dependencies') {
            steps {
                sh "npm install"
            }
        }
        stage('TRIVY FS SCAN') {
            steps {
                sh "trivy fs . > trivyfs.txt"
            }
        }
}
```

Build now an check sonarcube projects dahsboard you can see swiggy project there.

Add dockerhub token in jenkins and add below pipeline script under your job

```
stage("Docker Build & Push"){
                steps{
                    script{
                        withDockerRegistry(credentialsId: 'dockerhub', toolName: 'docker'){
                            sh "docker build -t swiggy-clone ."
                            sh "docker tag swiggy-clone ashfaque9x/swiggy-clone:latest "
                            sh "docker push ashfaque9x/swiggy-clone:latest "
                        }
                    }
                }
            }
            stage("TRIVY"){
                steps{
                    sh "trivy image ashfaque9x/swiggy-clone:latest > trivyimage.txt"
                }
            }
```

Build now, you can see new image is there in your dockerhub.

Step 5: Configure aws eks cluster

Login your ec2 instance which is created by terraform

Install kubectl on Jenkins

```
sudo apt update
sudo apt install curl
```

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version --client

Install AWS Cli

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

sudo apt install unzip

unzip awscliv2.zip

sudo ./aws/install

aws --version

Installing eksctl

curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp

cd /tmp

sudo mv /tmp/eksctl /bin

eksctl version

Create Iam role by attached policy of adminstator access to access eks cluster in aws console and attach it to your ec2 instance.

Setup Kubernetes using eksctl

Refer--https://github.com/aws-samples/eks-workshop/issues/734

eksctl create cluster --name virtualtechbox-cluster \

--region ap-south-1 \

--node-type t2.small \

--nodes 3 \

Verify Cluster with below command

$ kubectl get nodes

$ kubectl get svc

Step 6 :Configure kubernetes with jenkins and deploy application on EKS using jenkins

Install thease plugins(Kubernetes,Kubernetes credentlas,Kubernetes client api,kubernetes cli

download kubeconfige file from ec2 which have installed with eksctl,kubectl to your local machine

Add credentials(kind:secret file, selcect your kubeconfig file from local)

Modify pipeline script in your job.

```
 stage('Deploy to Kubernets'){
                steps{
                    script{
                        dir('Kubernetes') {
                            kubeconfig(credentialsId: 'kubernetes', serverUrl: '') {
                            sh 'kubectl delete --all pods'
                            sh 'kubectl apply -f deployment.yml'
                            sh 'kubectl apply -f service.yml'
                            }
                        }
                    }
                }
            }
```

Go to your jenkins job-> configure->slect github project(give your swiggy repo url)->select github hook trigger for git scm polling(go to your githup repo->settings->webhook->add webhook by give this url <jekins-url>/github-webhook/)->apply and save.

Build the job by making change in github repo.

kubectl get svc(copy the dns and enter in browser, you can see your application is running)