**Project: Automated Deployment and Scaling for Analytics Pvt Ltd**

**Project Overview**

Analytics Pvt Ltd needs a comprehensive DevOps lifecycle for automating deployment, scaling, and operations of application containers. The setup involves using Jenkins for CI/CD, Docker for containerization, Kubernetes for orchestration, Terraform for infrastructure provisioning, and Ansible for configuration management. The project ensures that the release happens on the 25th of every month, with a focus on using Docker containers, Kubernetes clusters, and Jenkins pipelines.

**1. Infrastructure Creation with Terraform**

**Objective:** Provision AWS EC2 instances for Jenkins, Kubernetes master, and Kubernetes workers.

**1.1. Create Terraform Configuration**

**File: main.tf**

provider "aws" {

  region = "us-east-1"

}

resource "aws\_vpc" "main" {

  cidr\_block           = "10.0.0.0/16"

  enable\_dns\_support   = true

  enable\_dns\_hostnames = true

  tags = {

    Name = "main-vpc"

  }

}

resource "aws\_subnet" "public" {

  vpc\_id                  = aws\_vpc.main.id

  cidr\_block              = "10.0.1.0/24"

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

  map\_public\_ip\_on\_launch = true

  tags = {

    Name = "public-subnet"

  }

}

resource "aws\_internet\_gateway" "main" {

  vpc\_id = aws\_vpc.main.id

  tags = {

    Name = "main-igw"

  }

}

resource "aws\_route\_table" "public" {

  vpc\_id = aws\_vpc.main.id

  route {

    cidr\_block = "0.0.0.0/0"

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

  }

  tags = {

    Name = "public-route-table"

  }

}

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

  subnet\_id      = aws\_subnet.public.id

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

}

resource "aws\_instance" "jenkins\_master" {

  ami                         = "ami-02dbb4e06f1bec81b"

  instance\_type               = "t2.medium"

  key\_name                    = "ansible" # Name of the existing key pair

  subnet\_id                   = aws\_subnet.public.id

  associate\_public\_ip\_address = true # Automatically assign a public IP

  tags = {

    Name = "Jenkins Master"

  }

}

resource "aws\_instance" "kubernetes\_master" {

  ami                         = "ami-02dbb4e06f1bec81b"

  instance\_type               = "t2.medium"

  key\_name                    = "ansible" # Name of the existing key pair

  subnet\_id                   = aws\_subnet.public.id

  associate\_public\_ip\_address = true # Automatically assign a public IP

  tags = {

    Name = "Kubernetes Master"

  }

}

resource "aws\_instance" "kubernetes\_worker" {

  count                       = 2

  ami                         = "ami-02dbb4e06f1bec81b"

  instance\_type               = "t2.medium"

  key\_name                    = "ansible" # Name of the existing key pair

  subnet\_id                   = aws\_subnet.public.id

  associate\_public\_ip\_address = true # Automatically assign a public IP

  tags = {

    Name = "Kubernetes Worker ${count.index + 1}"

  }

}

output "jenkins\_master\_ip" {

  value = aws\_instance.jenkins\_master.public\_ip

}

output "kubernetes\_master\_ip" {

  value = aws\_instance.kubernetes\_master.public\_ip

}

output "kubernetes\_worker\_ips" {

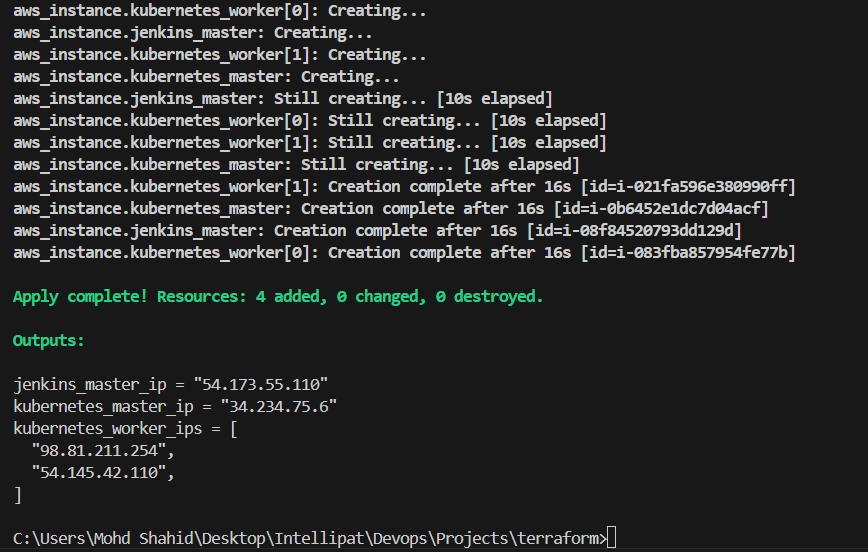
  value = aws\_instance.kubernetes\_worker[\*].public\_ip

}

**Initialize and Apply Terraform**

**terraform init**

**terraform apply**

****

**2. Configuration Management with Ansible**

**Objective:** Install and configure necessary software on the provisioned instances using Ansible.

**2.1. Prepare Ansible Inventory**

**File: /etc/ansible/host**

[jenkins]

jenkins ansible\_ssh\_host=<slave ip address>

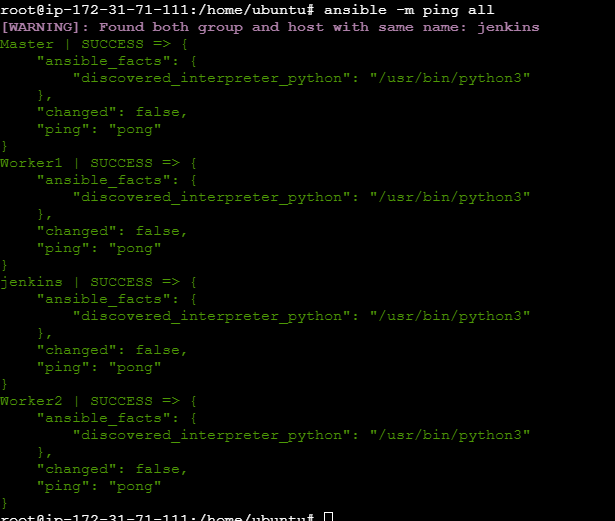
[master]

Master ansible\_ssh\_host=<slave ip address>

[worker]

Worker1 ansible\_ssh\_host=<slave ip address>

Worker2 ansible\_ssh\_host=<slave ip address>

****

**2.2. Create Ansible Playbooks**

**2.2.1. Install Jenkins and Java**

**File: jenkins.yml**

- hosts: Jenkins

  become: yes

  tasks:

    - name: Download Jenkins key

      get\_url:

        url: https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

        dest: /usr/share/keyrings/jenkins-keyring.asc

    - name: Add Jenkins repository to sources list

      lineinfile:

        path: /etc/apt/sources.list.d/jenkins.list

        line: "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/"

        create: yes

    - name: Update package index

      apt:

        update\_cache: yes

    - name: Install Jenkins

      apt:

        name: jenkins

        state: present

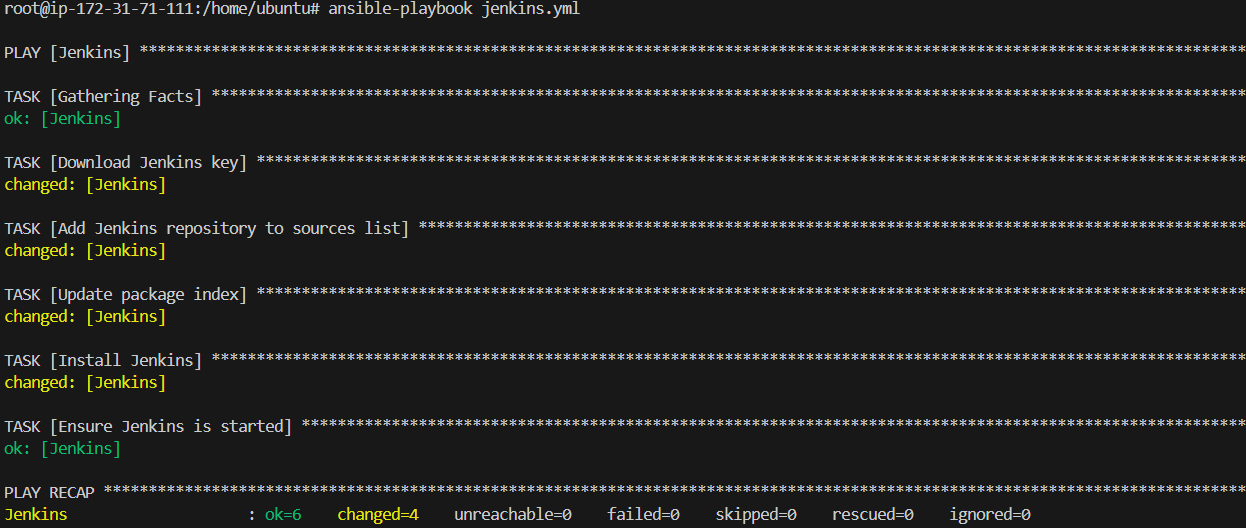
    - name: Ensure Jenkins is started

      service:

        name: jenkins

        state: started

        enabled: yes

****

**2.2.2. Install Docker, Kubernetes, and Java**

**File: master.yml**

- hosts: Kubernetes\_master

  become: yes

  tasks:

    - name: Install Java

      apt:

        name: openjdk-11-jdk

        state: present

        update\_cache: yes

    - name: Install Docker

      apt:

        name: docker.io

        state: present

        update\_cache: yes

    - name: Install Kubernetes Components

      apt:

        name: "{{ item }}"

        state: present

        update\_cache: yes

      loop:

        - kubelet

        - kubeadm

        - kubectl

**2.2.3. Install Docker, Kubernetes, and Java**

**File: worker.yml**

- hosts: Kubernetes\_workers

  become: yes

  tasks:

    - name: Install Java

      apt:

        name: openjdk-11-jdk

        state: present

        update\_cache: yes

    - name: Install Docker

      apt:

        name: docker.io

        state: present

        update\_cache: yes

    - name: Install Kubernetes Components

      apt:

        name: "{{ item }}"

        state: present

        update\_cache: yes

      loop:

        - kubelet

        - kubeadm

        - kubectl

**2.3. Run Ansible Playbooks**

ansible-playbook jenkins.yml

ansible-playbook master.yml

ansible-playbook worker.yml

**3. Jenkins and Kubernetes Configuration**

**Objective:** Configure Jenkins for CI/CD and Kubernetes for cluster management.

**3.1. Set Up Jenkins**

1. **Access Jenkins:** Open Jenkins in your web browser using the Jenkins master public IP address.
2. **Install Plugins:** Go to Manage Jenkins > Manage Plugins and install necessary plugins (Docker, Kubernetes, Git, etc.).
3. **Configure Jenkins:**
   * Add Docker Hub credentials for pushing images.
   * Configure Jenkins to interact with Kubernetes (set up Kubernetes plugin and provide cluster credentials).

**3.2. Configure Kubernetes**

1. **Initialize Kubernetes Cluster:**

**kubeadm init**

# On worker nodes (with token and hash from the master output)

**kubeadm join <master-ip>:6443 --token <token> --discovery-token-ca-cert-hash sha256:<hash>**

1. **Configure kubectl:**

**mkdir -p $HOME/.kube**

**cp -i /etc/kubernetes/admin.conf $HOME/.kube/config**

1. **Deploy CoreDNS and Network Add-ons:** Apply necessary YAML files for network setup.

**4. CI/CD Pipeline with Jenkins**

**Objective:** Create a Jenkins pipeline to automate Docker build, push, and deployment to Kubernetes.

**File: Jenkinsfile**

**\**

pipeline {

    agent any

    environment {

        DOCKER\_IMAGE = 'hshar/website'

        REPO\_URL = 'https://github.com/hshar/website.git'

        IMAGE\_TAG = "${env.BUILD\_ID}"

    }

    triggers {

        cron('0 0 25 \* \*') // Trigger build on the 25th of every month

    }

    stages {

        stage('Checkout') {

            steps {

                git branch: 'master', url: "${REPO\_URL}"

            }

        }

        stage('Build Docker Image') {

            steps {

                script {

                    docker.build("${DOCKER\_IMAGE}:${IMAGE\_TAG}")

                }

            }

        }

        stage('Push Docker Image') {

            steps {

                script {

                    docker.withRegistry('https://index.docker.io/v1/', 'dockerhub-credentials-id') {

                        docker.image("${DOCKER\_IMAGE}:${IMAGE\_TAG}").push("${IMAGE\_TAG}")

                    }

                }

            }

        }

        stage('Deploy to Kubernetes') {

            steps {

                script {

                    kubectl.apply('-f k8s/deployment.yaml')

                    kubectl.apply('-f k8s/service.yaml')

                }

            }

        }

    }

    post {

        always {

            cleanWs()

        }

    }

}

**5. Kubernetes Deployment**

**Objective:** Deploy the Dockerized application to Kubernetes.

**File: k8s/deployment.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: website-deployment

spec:

  replicas: 2

  selector:

    matchLabels:

      app: website

  template:

    metadata:

      labels:

        app: website

    spec:

      containers:

      - name: website

        image: hshar/website:${IMAGE\_TAG}

        ports:

        - containerPort: 8080

File: k8s/service.yaml

yaml

Copy code

apiVersion: v1

kind: Service

metadata:

  name: website-service

spec:

  type: NodePort

  selector:

    app: website

  ports:

    - protocol: TCP

      port: 80

      targetPort: 8080

      nodePort: 30008

**Summary**

* **Terraform:** Provision AWS EC2 instances for Jenkins and Kubernetes infrastructure.
* **Ansible:** Install and configure Jenkins, Java, Docker, and Kubernetes on the respective instances.
* **Jenkins:** Set up a pipeline with a cron trigger to automate Docker builds, pushes, and Kubernetes deployments on the 25th of every month.
* **Kubernetes:** Configure and deploy applications using Kubernetes manifests.

This setup ensures a streamlined and automated approach to deployment and scaling, meeting the requirements of Analytics Pvt Ltd.