DevOps Engineer Assignment Documentation

Objective:
This assignment demonstrates automating infrastructure setup, deploying a Node.js application, managing a cloud environment (AWS), and setting up a CI/CD pipeline using tools like Jenkins, Docker, Terraform, Ansible, Kubernetes, Prometheus, Grafana, and ELK.

Step-by-Step Process

---

1. Infrastructure Setup:
- **Cloud Provider**: We are using **AWS** as the hosting platform.
- **Infrastructure as Code (IaC)**: Terraform will be used to provision AWS resources like EC2, VPC, and Load Balancers. Security groups and IAM roles are set up for security.

Terraform Code:
Here is the code to provision AWS resources using Terraform:

```hcl
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "main_vpc" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "public_subnet" {
  vpc_id     = aws_vpc.main_vpc.id
  cidr_block = "10.0.1.0/24"
}

resource "aws_internet_gateway" "igw" {
  vpc_id = aws_vpc.main_vpc.id
}

resource "aws_route_table" "public_route" {
  vpc_id = aws_vpc.main_vpc.id
  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.igw.id
  }
}

resource "aws_instance" "app_server" {
  ami           = "ami-0c55b159cbfafe1f0"  # Use a compatible AMI
  instance_type = "t2.micro"
  subnet_id     = aws_subnet.public_subnet.id
  tags = {
    Name = "NodeAppServer"
  }
}
```

Auto Scaling & Load Balancer:
Auto Scaling can be set up using Terraform to adjust the number of EC2 instances dynamically based on traffic.

---

2. CI/CD Pipeline Setup:
We use **Jenkins** for CI/CD pipeline implementation to automate testing and deployment of the Node.js application. After running tests, the app is containerized and deployed using Terraform and Kubernetes.

Jenkins Pipeline Script:

```groovy
pipeline {
    agent any
    environment {
        AWS_ACCESS_KEY_ID = credentials('aws-access-key')
        AWS_SECRET_ACCESS_KEY = credentials('aws-secret-key')
    }

    stages {
        stage('Checkout Code') {
            steps {
                git branch: 'main', url:
'https://github.com/sumannayak6207/CWC.git'
            }
        }

        stage('Build') {
            steps {
                script {
                    sh 'npm install'
                    sh 'npm test'
                }
            }
        }

        stage('Docker Build') {
            steps {
                script {
                    sh 'docker build -t sumannayak6207/cwc:latest .'
                }
            }
        }

        stage('Push Docker Image') {
            steps {
                script {
                    sh 'docker push YOUR_DOCKER_REGISTRY/cwc:latest'
                }
            }
        }

        stage('Deploy with Terraform and Ansible') {
            steps {
                script {
                    dir('infra/terraform') {
                        sh 'terraform init'
                        sh 'terraform apply -auto-approve'
                    }
                    dir('infra/ansible') {
                        sh 'ansible-playbook -i hosts playbook.yml'
                    }
                }
            }
        }

        stage('Deploy to Kubernetes') {
```

```
            steps {
                script {
                    sh 'kubectl apply -f k8s/deployment.yaml'
                }
            }
        }
    }

    post {
        always {
            cleanWs()
        }
    }
}
```

---

3. Containerization and Orchestration:
- **Docker**: We create a Dockerfile to containerize the Node.js app.
- **Kubernetes**: The app is deployed to a Kubernetes cluster to ensure scalability.

Dockerfile:
```Dockerfile
FROM node:14

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

Kubernetes Deployment YAML:
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nodejs-app-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nodejs-app
  template:
    metadata:
      labels:
        app: nodejs-app
    spec:
      containers:
      - name: nodejs-app
        image: sumannayak6207/cwc:latest
        ports:
        - containerPort: 3000
```

---

4. Monitoring and Logging:
- **Prometheus & Grafana**: Monitoring tools to track system health, CPU/memory usage, and set up alerts.
- **ELK Stack**: ElasticSearch, Logstash, and Kibana for logging and log visualization.

Prometheus Setup:
```yaml
apiVersion: v1
kind: Service
metadata:
  name: prometheus
spec:
  ports:
    - port: 9090
      protocol: TCP
      targetPort: 9090
  selector:
    app: prometheus
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: prometheus
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus
  template:
    metadata:
      labels:
        app: prometheus
    spec:
      containers:
      - name: prometheus
        image: prom/prometheus
        ports:
        - containerPort: 9090
```

Grafana Setup:
```yaml
apiVersion: v1
kind: Service
metadata:
  name: grafana
spec:
  ports:
    - port: 3000
      protocol: TCP
      targetPort: 3000
  selector:
    app: grafana
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grafana
spec:
  replicas: 1
  selector:
    matchLabels:
```

```
      app: grafana
  template:
    metadata:
      labels:
        app: grafana
    spec:
      containers:
      - name: grafana
        image: grafana/grafana
        ports:
        - containerPort: 3000
```

---

5. Security Best Practices:
- **IAM Roles**: We implement proper IAM roles and policies to ensure least privilege.
- **SSL/TLS**: Secure communication with SSL certificates.
- **Vulnerability Scanning**: Using tools like **Trivy** for scanning Docker images in the Jenkins pipeline.

---

6. Documentation:

- **Setup**:
  - Clone the repository `https://github.com/sumannayak6207/CWC.git`.
  - Navigate to the `infra/terraform` directory and run `terraform apply` to provision AWS infrastructure.
  - Use `kubectl` to manage Kubernetes deployments and services.

- **CI/CD Workflow**:
  - Jenkins pipeline automates the entire flow: building, testing, deploying, and scaling.

- **Scaling**:
  - Use Terraform’s auto-scaling configurations and Kubernetes Horizontal Pod Autoscaler.

Deliverables:
1 **GitHub Repository**:
   - Repository: [CWC GitHub Repository](https://github.com/sumannayak6207/CWC.git)
   - Include Terraform, Docker, Jenkinsfile, and application code in the repository.