PROJECT: STREAMLINING SECURITY ACROSS ENVIRONMENTS WITH DEVSECOPS

PHASE 2 - SOLUTION ARCHITECTURE

College Name: Vemana Institute of Technology
Name: Monika P
CAN ID Number: CAN_33387138

1. Solution Architecture

The project focuses on streamlining security and deployment of containerized applications by incorporating DevSecOps principles. Below are the steps to set up the architecture:

Step 1: Set Up the Project Directory

- Use the following commands to create a directory structure for a containerized e-commerce application:

bash

CopyEdit

```
mkdir ecommerce-app

cd ecommerce-app

mkdir public public/css public/js

mkdir server server/controllers server/models server/routes

echo. > public/css/style.css

echo. > public/js/app.js

echo. > public/index.html

echo. > server/controllers/productController.js

echo. > server/models/productModel.js

echo. > server/routes/productRoutes.js

echo. > server/server.js

echo. > package.json

echo. > README.md
```

- The directory structure will look like:

pgsql

CopyEdit

```
ecommerce-app/
├── public/
│   ├── css/
│   │   └── style.css
│   ├── js/
│   │   └── app.js
│   └── index.html
├── server/
│   ├── controllers/
│   │   └── productController.js
│   ├── models/
│   │   └── productModel.js
│   ├── routes/
│   │   └── productRoutes.js
│   └── server.js
├── package.json
└── README.md
```

Step 2: Version Control Setup

- Initialize Git and create a repository:

bash

CopyEdit

```
git init
echo node_modules/ > .gitignore
echo .env >> .gitignore
```

```
git add .
git commit -m "Initial commit of ecommerce-app structure"
git remote add origin <repository_url>
git push -u origin master
```

---

2. CI/CD Pipeline Design and Implementation

To automate and secure the deployment process, a CI/CD pipeline is implemented using **Jenkins**:

Step 1: Jenkins Setup

1. Install Jenkins and required plugins:
   - Docker
   - Git
   - Kubernetes CLI
2. Configure Jenkins to trigger on code changes pushed to GitHub.

Step 2: Jenkinsfile Creation

- Create a Jenkinsfile with the following stages:
  1. **Checkout**: Pull the latest code.
  2. **Build**: Build Docker images.
  3. **Test**: Run unit and integration tests.
  4. **Push**: Push Docker images to IBM Cloud Container Registry.
  5. **Deploy**: Deploy the application to a Kubernetes cluster.

Example Jenkinsfile:

groovy

CopyEdit

```
pipeline {
    agent any
    environment {
        DOCKER_IMAGE = 'my-app'
        REGISTRY_URL = '<IBM_Container_Registry_URL>'
        CLUSTER_NAME = '<CLUSTER_NAME>'
    }
    stages {
        stage('Checkout') {
            steps {
                git 'https://github.com/<username>/ecommerce-app.git'
            }
        }
        stage('Build Docker Image') {
            steps {
                script {
                    sh 'docker build -t $REGISTRY_URL/$DOCKER_IMAGE .'
                }
            }
        }
        stage('Push Docker Image to IBM Cloud Container Registry') {
            steps {
                script {
                    sh 'docker push $REGISTRY_URL/$DOCKER_IMAGE'
                }
            }
        }
        stage('Deploy to Kubernetes') {
            steps {
                script {
                    sh '''
```

```
                    ibmcloud login --apikey <API_KEY> -r <REGION> -g
<RESOURCE_GROUP>
                    ibmcloud ks cluster config --cluster $CLUSTER_NAME
                    kubectl apply -f k8s/deployment.yaml
                    '''
                }
            }
        }
    }
    post {
        success {
            echo 'Pipeline executed successfully.'
        }
        failure {
            echo 'Pipeline failed. Please check the logs.'
        }
    }
}
```

---

3. Future Plan

1. **Container Image Management**:
   - Use **IBM Cloud Container Registry** to securely store Docker images.
2. **Kubernetes Deployment**:
   - Deploy applications to Kubernetes clusters using **Minikube** for testing and IBM Cloud for production.
3. **Security Enhancements**:
   - Implement **OpenSSL** for image signing and vulnerability scanning.

4. **CI/CD Integration**:
    - Automate using IBM Cloud Continuous Delivery or GitHub Actions.
5. **Focus on DevSecOps**:
    - Integrate security checks at every stage (e.g., static code analysis, dependency vulnerability scans).