# PROJECT: STREAMLINING SECURITY ACROSS ENVIRONMENTS WITH DEVSECOPS

## PHASE 3 - SOLUTION DEVELOPMENT AND TESTING

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

### SOLUTION DEVELOPMENT

**Setting up Cloud Environment and Security Tools**

**Step 1: Create a Cloud Environment**

1. **Select a Cloud Provider** (AWS, Azure, GCP, or IBM Cloud).
2. **Sign up and configure security permissions** using IAM roles.
3. **Enable billing** to access security-related services.

**Step 2: Install and Configure Required Tools**

1. **Install CLI Tools** (AWS CLI, Azure CLI, IBM Cloud CLI, etc.).
2. **Install kubectl** for Kubernetes management.
3. **Set up Terraform** for Infrastructure as Code (IaC).
4. **Install security tools** (Trivy, Aqua Security, Snyk, etc.).

---

**Implementing DevSecOps Security in CI/CD Pipeline**

**Step 1: Dockerizing the Application with Security Best Practices**

**Create Secure Dockerfiles**

- **Frontend Dockerfile (/public/Dockerfile):**

dockerfile

CopyEdit

```
FROM node:16-alpine
WORKDIR /app
COPY . .
RUN npm install
EXPOSE 3000
CMD ["npm", "start"]
```

- **Backend Dockerfile (/server/Dockerfile):**

dockerfile

CopyEdit

FROM node:16-alpine

WORKDIR /app

COPY . .

RUN npm install

EXPOSE 5000

CMD ["node", "server.js"]

- **Security Enhancements:**
    - Use **multi-stage builds** to minimize attack surface.
    - **Limit privileges** by using non-root users.

**Step 2: Push Secure Docker Images to Container Registry**

1. **Enable vulnerability scanning** before pushing:

sh

CopyEdit

trivy image frontend-app:1.0

trivy image backend-app:1.0

2. **Tag the images securely:**

sh

CopyEdit

docker tag frontend-app:1.0 <cloud_registry>/<namespace>/frontend-app:1.0

docker tag backend-app:1.0 <cloud_registry>/<namespace>/backend-app:1.0

3. **Push to Registry:**

sh

CopyEdit

docker push <cloud_registry>/<namespace>/frontend-app:1.0

docker push <cloud_registry>/<namespace>/backend-app:1.0

---

**SECTION 2: TESTING THE SECURITY SOLUTION**

**Step 1: Deploy Secure Applications to Kubernetes**

**Create Kubernetes Deployment Files**

- **Frontend Deployment (frontend-deployment.yaml):**

yaml

CopyEdit

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: <cloud_registry>/<namespace>/frontend-app:1.0
        securityContext:
          runAsNonRoot: true
        ports:
        - containerPort: 3000
```

- **Backend Deployment (backend-deployment.yaml):**

yaml

CopyEdit

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment
spec:
```

```
  replicas: 2
  selector:
    matchLabels:
      app: backend
  template:
    metadata:
      labels:
        app: backend
    spec:
      containers:
      - name: backend
        image: <cloud_registry>/<namespace>/backend-app:1.0
        securityContext:
          runAsNonRoot: true
        ports:
        - containerPort: 5000
```

**Step 2: Apply Security Policies and Scanning**

1. **Apply Kubernetes Deployments:**

```
sh
CopyEdit
kubectl apply -f frontend-deployment.yaml
kubectl apply -f backend-deployment.yaml
```

2. **Check Running Pods and Services:**

```
sh
CopyEdit
kubectl get pods
kubectl get svc
```

3. **Implement Role-Based Access Control (RBAC)** for security enforcement.

**Step 3: Integrate CI/CD Security Testing**

1. **Set up GitHub Actions / Jenkins Pipeline with security scanning:**

```
yaml
CopyEdit
```

```
jobs:
  security_scan:
    runs-on: ubuntu-latest
    steps:
    - name: Run Trivy Security Scan
      run: trivy image <cloud_registry>/<namespace>/frontend-app:1.0
```

2. **Automate deployment with security checks** before merging code.

---

**SECTION 3: SECURITY TESTING & FUTURE IMPROVEMENTS**

**Step 1: Security Testing & Compliance**

1. **Run Static Application Security Testing (SAST)** using Snyk or SonarQube.
2. **Perform Dynamic Application Security Testing (DAST)** with OWASP ZAP.
3. **Use penetration testing tools** like Metasploit for security assessments.

**Step 2: Future Enhancements**

1. **Implement Kubernetes Network Policies** to restrict traffic between services.
2. **Enable Autoscaling & Self-Healing Mechanisms** for better security resilience.
3. **Integrate AI-based Threat Detection** using cloud-native security tools.

---

**Conclusion**

This project integrates **DevSecOps** principles to **streamline security across environments** by automating security scanning, enforcing access controls, and deploying applications securely using Kubernetes and cloud security tools.