**Objective: Develop into Networking Policy within EKS.**

**Tasks:**

**1.** Deploy multi-tier applications on EKS.

**2.** Implement network policies using Amazon VPC CNI.

**3.** Test network segmentations.

**Documentation:**

- Basics of Kubernetes networking.

- Amazon VPC CNI overview.

- Crafting network policies.

**Prerequisites:**

1. **AWS Account:** Ensure you have access to an AWS account.
2. **AWS CLI:** Installed and configured with access keys.
3. **kubectl tool:** Installed to manage EKS clusters.
4. **EKS Cluster:** Running with nodes launched.
5. **Amazon VPC CNI:** Ensure it's installed in your EKS cluster. By default, EKS uses the Amazon VPC CNI.

**Tasks:**

**1. Deploy multi-tier applications on EKS:**

For demonstration purposes, let's deploy a basic frontend and backend.

| Chmod +x deployments.sh bash deployments.sh |
| --- |

| # Create a namespace kubectl create namespace multi-tier-app  # Deploy backend kubectl apply -f - <<EOF apiVersion: apps/v1 kind: Deployment metadata:  name: backend-deployment  namespace: multi-tier-app spec:  replicas: 2  selector:  matchLabels:  app: backend  template:  metadata:  labels:  app: backend  spec:  containers:  - name: backend  image: nginx EOF  # Deploy frontend kubectl apply -f - <<EOF apiVersion: apps/v1 kind: Deployment metadata:  name: frontend-deployment  namespace: multi-tier-app spec:  replicas: 2  selector:  matchLabels:  app: frontend  template:  metadata:  labels:  app: frontend  spec:  containers:  - name: frontend  image: nginx EOF |
| --- |

| **kubectl apply -f - <<EOF apiVersion: v1 kind: Service metadata:  name: backend-deployment  namespace: multi-tier-app spec:  selector:  app: backend  ports:  - protocol: TCP  port: 80  targetPort: 80 EOF** |
| --- |

**2. Implement network policies using Amazon VPC CNI:**

The following policy allows only the frontend to communicate with the backend:

| kubectl apply -f - <<EOF apiVersion: networking.k8s.io/v1 kind: NetworkPolicy metadata:  name: backend-network-policy  namespace: multi-tier-app spec:  podSelector:  matchLabels:  app: backend  policyTypes:  - Ingress  ingress:  - from:  - podSelector:  matchLabels:  app: frontend EOF |
| --- |

**3. Test network segmentations:**

Now, try communicating from the frontend pod to the backend pod.

| # Get a frontend pod name FRONTEND\_POD=$(kubectl get pods -n multi-tier-app -l app=frontend -o jsonpath='{.items[0].metadata.name}')  # Try reaching backend from frontend kubectl exec -n multi-tier-app $FRONTEND\_POD -- curl backend-deployment |
| --- |

**4. Verify the Network Policy:**

Use the following command to check if the network policy has been applied:

| kubectl get networkpolicy -n multi-tier-app |
| --- |

You should see backend-network-policy in the list, which means the policy is applied successfully.

If your network policy is set up correctly, the above should work. If you try to communicate from a different pod (not labeled as frontend), it shouldn't work.

**Troubleshooting Tips and Tricks:**

1. **Networking:** If pods can't communicate, ensure that they have proper IP addresses and that no Network ACLs or Security Groups are blocking the traffic in AWS.
2. **Logs:** Check logs of the pods (kubectl logs POD\_NAME).
3. **Network Policies:** Ensure that the network policies are correctly applied and that they don't conflict.
4. **Amazon VPC CNI:** Ensure the VPC CNI is functioning correctly. Check its logs for any issues.
5. **Pod Status:** Ensure that all pods are in the RUNNING state.

**Documentation:**

1. **Basics of Kubernetes networking:**

Kubernetes offers a flat networking model where every Pod gets its IP. Pods can communicate with other Pods, irrespective of the node they're on.

1. **Amazon VPC CNI overview:**

Amazon VPC CNI provides native AWS networking capabilities for Kubernetes. It assigns Elastic Network Interfaces (ENIs) directly to Pods, providing high-performance networking.

1. **Crafting network policies:**

Network policies define how pods communicate. They specify ingress and egress rules based on IP protocols, ports, and selectors.