**LAB#2 Network Policies with Calico**

**\*\*Tasks\*\*:**

**- Install Calico on EKS.**

**- Define and apply basic network policies.**

**- Test and validate policies.**

**- \*\*Documentation\*\*: Describe how**

**network policies enhance security within**

**Kubernetes.**

**Prerequisites:**

**Install kubectl**

**Install eks**

**Install awscli**

**Add access key**

**Add roles for cluster and node permissions**

### Install Calico on Amazon EKS:

* Install kubectl:
  + If you don't have kubectl installed, download and install it using the instructions in the official Kubernetes documentation: [Install and Set Up kubectl](https://kubernetes.io/docs/tasks/tools/install-kubectl/).
* Install eksctl:
  + If you don't have eksctl installed, follow the instructions on the official Amazon EKS documentation: [Installing eksctl](https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html).
* Create an Amazon EKS Cluster:
  + Use eksctl to create an EKS cluster. Make sure you have the necessary AWS CLI and IAM

**INSTALL CALICO:**

**kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml**

Create a Network Policy YAML file:

* Define a basic network policy in a YAML file. For example:

| **apiVersion: networking.k8s.io/v1 kind: NetworkPolicy metadata:  name: allow-nginx spec:  podSelector:  matchLabels:  run: nginx  policyTypes:  - Ingress  ingress:  - from:  - podSelector:  matchLabels:  app: frontend  ports:  - protocol: TCP  port: 80** |
| --- |

| kubectl apply -f your-network-policy.yaml |
| --- |

### Test and Validate Policies:

* Deploy Pods:
  + Deploy the necessary pods to test the network policies. For example, deploy an Nginx pod:

| kubectl run nginx --image=nginx --labels=run=nginx |
| --- |

Deploy Pods to Match Policy:

* Deploy pods that match the network policy. In the example policy, deploy a frontend pod:

# Deploy frontend pod

| **kubectl run frontend --image=nginx --labels=app=frontend** |
| --- |

### Verify Connectivity:

#### Test Communication from Frontend to Nginx:

* Access the Frontend Pod:

| kubectl exec -it <frontend-pod-name> -- /bin/sh |
| --- |

# Use the Nginx pod IP and port (in this example, port 80)

| **wget -qO- http://<nginx-pod-ip>:80** |
| --- |

# Use the Nginx pod IP and port (in this example, port 80)

| curl -s http://<nginx-pod-ip>:80 |
| --- |

| kubectl exec -it <nginx-pod-name> -- /bin/sh |
| --- |

# Use the Frontend pod IP and port (replace with the actual IP and port)

| wget -qO- http://<**frontend-pod-ip**>:<**frontend-port**> |
| --- |

If the network policy is correctly applied, you should get a successful response. If the policy is not applied correctly, the connection will fail.

============================================

**- \*\*Documentation\*\*:**

**Describe how network policies enhance security within Kubernetes.**

Calico is an open-source networking and network security solution commonly used in Kubernetes environments, including Amazon Elastic Kubernetes Service (EKS). It's designed to provide networking capabilities for containerized applications, enabling secure and scalable communication between pods and nodes within a Kubernetes cluster.

In Amazon EKS, Calico can be deployed as the networking plugin for managing network policies, pod-to-pod communication, and network traffic within the Kubernetes cluster.

### Key Features of Calico in EKS:

* Networking:
  + Calico sets up an overlay network that allows pods to communicate with each other regardless of the underlying infrastructure. It manages IP address assignment and routing between pods.
* Network Policies:
  + Calico enables the implementation of network policies that define how pods can communicate with each other. These policies control ingress and egress traffic between pods based on defined rules, enhancing security within the cluster.
* Scalability:
  + Calico is known for its scalability and performance, allowing for large-scale deployments in Kubernetes clusters without compromising performance.
* Integration with Kubernetes:
  + Calico integrates seamlessly with Kubernetes, leveraging Kubernetes’ native networking and security features while extending them with additional functionalities.

**Conclusion of the lab** 👍

Network policies in Kubernetes play a pivotal role in fortifying the security posture of containerized environments.

* Micro-Segmentation's Impact: Implementing network policies allows for micro-segmentation, establishing controlled communication pathways between pods, reducing the attack surface, and containing potential threats.
* Precise Traffic Control: The ability to define granular rules based on IP addresses, ports, and labels ensures precise control over traffic flow, preventing unauthorized access and minimizing the risks of lateral movement within the cluster.
* Enhanced Defense Mechanisms: Network policies act as a defense mechanism, enforcing isolation and enabling security teams to implement least privilege principles within the Kubernetes cluster.

The strategic implementation of network policies is integral in ensuring a robust security posture in Kubernetes deployments. By employing these policies effectively, organizations can reinforce their defense against potential threats, maintain compliance, and uphold the integrity and confidentiality of their containerized workloads.