Networking

» Network Policies

Overview

This exercise provides an overview of the NetworkPolicy resource along with the Calico Network Plugin.

Namespace-to-Namespace Communication

Let's start by preventing communication between namespaces via a Calico Network Policy.

First, create two Namespaces.

```
kubectl create namespace accounting
kubectl create namespace logistics
```

Confirm that the namespaces were successfully created.

```
kubectl get namespaces
```

Create a demo Pod in each Namespace

```
kubectl run accounting-pod -n accounting --image busybox --restart Never -- sleep 8000
kubectl run logistics-pod -n logistics --image busybox --restart Never -- sleep 8000
```

Confirm that the pods were successfully created.

```
kubectl get pods -n accounting -o wide
kubectl get pods -n logistics -o wide
```

Save the IP address for the `Accounting` pod in an environment variable.

```
ACCOUNTING_POD_IP=$(kubectl get pods accounting-pod -n accounting -o jsonpath={$.status.podIP})
echo $ACCOUNTING_POD_IP
```

Save the IP address for the `Logistics` pod in an environment variable.

```
LOGISTICS_POD_IP=$(kubectl get pods logistics-pod -n logistics -o jsonpath={$.status.podIP})
echo $LOGISTICS_POD_IP
```

Test communication from the Accounting pod to the Logistics pod. We should receive a reply.

```
kubectl exec -it accounting-pod -n accounting -- ping -c 3 $LOGISTICS_POD_IP
```

Test communication from the Logistics pod to the Accounting pod. We should receive a reply.

```
kubectl exec -it logistics-pod -n logistics -- ping -c 3 $ACCOUNTING_POD_IP
```

Confirm no current Network Policies are in the cluster.

```
kubectl get NetworkPolicy --all-namespaces
```

Create a manifest that defines a NetworkPolicy that will deny all ingress traffic to all pods in the `Accounting` namespace.

```
cat > network-policy.yaml <<EOF
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  name: default-deny
  namespace: accounting
spec:
  podSelector:
EOF
```

Create the NetworkPolicy resource based on the manifest.

```
kubectl create -f network-policy.yaml
```

Confirm that the NetworkPolicy was successfully created.

```
kubectl get NetworkPolicy -n accounting
```

Now test communication to a pod in the `Accounting` namespace from a pod in the `Logistics` namespace. You should **not** receive a reply this time.

```
kubectl exec -it logistics-pod -n logistics -- ping -c 3 $ACCOUNTING_POD_IP
```

However, all **egress** traffic from the `Logistics` namespace will be allowed.

```
kubectl exec -it logistics-pod -n logistics -- ping -c 3 coreos.com
```

The `Accounting` pod can still receive established traffic as well.

```
kubectl exec -it accounting-pod -n accounting -- ping -c 3 $LOGISTICS_POD_IP
```

Pod-to-Pod Communication

Deploy an Nginx Deployment to the Accounting Namespace and expose it as a service

```
kubectl run -n accounting nginx --image=nginx
```

```
kubectl expose -n accounting deployment nginx --port=80
```

Create a new Pod in the Accounting namespace

```
kubectl run --restart=Never --namespace=accounting accounting-client -l run=accounting-client --image busybox sleep 8000
```

Confirm that the new pod CANNOT connect to the Nginx Service due to existing Deny NetworkPolicy

```
kubectl exec -it accounting-client -n accounting -- wget -q -O - nginx
```

If we want to allow all ingress traffic within the `Accounting` namespace only, we could create the following.

```
network-policy-allow.yaml <<EOF
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  name: allow-all-accounting
  namespace: accounting
spec:
  podSelector:
    matchLabels: {}
  ingress:
    - from:
      - podSelector:
          matchLabels: {}
```

Allow Access

Identify the labels assigned to the pods within the `Accounting` namespace.

```
kubectl get pods -n accounting --show-labels
```

Create a manifest that defines a NetworkPolicy that will allow the `accounting-client` to access `nginx .`

```
cat > network-policy-nginx.yaml <<EOF
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  name: access-nginx
  namespace: accounting
spec:
  podSelector:
    matchLabels:
      run: nginx
  ingress:
    - from:
      - podSelector:
          matchLabels:
            run: accounting-client
      ports:
       - protocol: TCP
         port: 80
EOF
```

Create the NetworkPolicy resource based on the manifest.

```
kubectl create -f network-policy-nginx.yaml
```

Confirm that the new pod **can** connect to the nginx Service due to the NetworkPolicy in place.

```
kubectl exec -it accounting-client -n accounting -- wget -q -O - nginx
```

Clean Up

Clean up the namespaces that we created to remove all additional resources.

```
kubectl delete namespace accounting logistics
```

View any remaining resources.

```
kubectl get all
```