

Lab18 – Troubleshooting an issue with Network Policy

In this exercise, you will create a network policy allowing the web-app to query its database. The application stack consists of a web application implemented using node.js, and a MySQL database. The web application connects to the database upon requesting its endpoint. Web application and MySQL database run in a Pod. Both Pods have been exposed by a Service. The Service for the web application Pod is of type `NodePort`. The Service for the MySQL database is of type `ClusterIP`. By default, all ingress and egress traffic is not allowed in this namespace.



1. Create a new namespace named `network`. Within the namespace, setup the configuration `kubectl -n network apply -f setup.yaml`.

```

brahim@Training:~/lab18-troubleshooting-network-policy$ kubectl create ns network
namespace/network created
brahim@Training:~/lab18-troubleshooting-network-policy$ 
brahim@Training:~/lab18-troubleshooting-network-policy$ kubectl apply -f setup.yaml -n network
networkpolicy.networking.k8s.io/default-deny created
pod/mysql-db created
service/mysql-service created
pod/web-app created
service/web-app-service created
brahim@Training:~/lab18-troubleshooting-network-policy$ 
brahim@Training:~/lab18-troubleshooting-network-policy$ kubectl get all -n network
NAME                READY   STATUS             RESTARTS   AGE
pod/mysql-db        0/1     ContainerCreating   0           38s
pod/web-app         0/1     ContainerCreating   0           37s

NAME                TYPE          CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
service/mysql-service ClusterIP      10.104.234.172 <none>         3306/TCP          38s
service/web-app-service NodePort       10.100.53.97   <none>         3000:30506/TCP    37s
brahim@Training:~/lab18-troubleshooting-network-policy$ 
  
```

- The Pod running web application exposes the container port 3000. From your machine, use your browser or execute `curl` or `wget` to access the application through the Service endpoint from outside of the cluster. A successful response should render `Successfully connected to database!`, a failure response should render `Failed to connect to database: <error message>`.



- Add a new NetworkPolicy allowing web-app to connect to mysql-db only on port 3306.

```

---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector:
    matchLabels:
      app: mysql-service
  ingress:
    - ports:
        - port: 3306
      from:
        - podSelector:
            matchLabels:
              app: web-app
---
apiVersion: v1
kind: Pod
metadata:

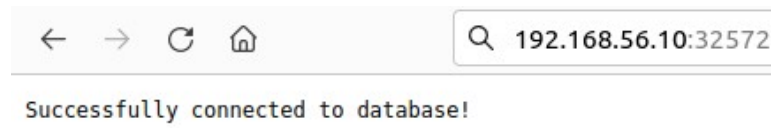
```

```

brahim@Training:~/lab17$ vim setup.yaml
brahim@Training:~/lab17$ kubectl apply -f setup.yaml -n network
networkpolicy.networking.k8s.io/default-deny configured
Warning: Detected changes to resource mysql-db which is currently being deleted.
pod/mysql-db configured
service/mysql-service configured
pod/web-app configured
service/web-app-service configured
brahim@Training:~/lab17$
brahim@Training:~/lab17$

```

4. Use your browser or ``curl`` or ``wget`` command should now render the message ``Successfully connected to database!``.



5. Delete the namespace ``network``.