Guided Exercise

Configure Network Policies

Create network policies and review pod isolation that these network policies created.

Outcomes

• Create network policies to control communication between pods.

```
[student@workstation ~]$ lab start network-policy
```

Instructions

1. Log in to the OpenShift cluster and create the network-policy project.

1.1. Log in to the cluster as the developer user with the developer password.

```
[student@workstation ~]$ oc login -u developer -p developer \
 https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

1.2. Create the network-policy project.

```
[student@workstation ~]$ oc new-project network-policy
Now using project "network-policy" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

2. Create two identical deployments named hello and test. Create a route to the hello deployment.

2.1. Create the hello deployment that uses the registry.ocp4.example.com:8443/redhattraining/hello-worldnginx:v1.0 container image.

```
[student@workstation ~]$ oc new-app --name hello \
 --image registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:v1.0
...output omitted...
--> Creating resources ...
 imagestream.image.openshift.io "hello" created
 deployment.apps "hello" created
 service "hello" created
--> Success
...output omitted...
```

2.2. Create the test deployment that uses the registry.ocp4.example.com:8443/
redhattraining/hello-world-nginx:v1.0 container image.

```
[student@workstation ~]$ oc new-app --name test \
 --image registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:v1.0
...output omitted...
--> Creating resources ...
 imagestream.image.openshift.io "test" created
 deployment.apps "test" created
 service "test" created
--> Success
...output omitted...
```

2.3. Use the oc expose command to create a route to the hello service.

```
[student@workstation ~]$ oc expose service hello
route.route.openshift.io/hello exposed
```

3. Verify that the test pod can access the hello pod by using the oc rsh and curl commands.

3.1. Open a second terminal and run the script at ~/DO280/labs/network-policy/
display-project-info.sh. This script provides information about the pods,
service, and route that are used in the rest of this exercise.

```
[student@workstation ~]$ ~/DO280/labs/network-policy/display-project-info.sh
===================================================================
PROJECT: network-policy
POD NAME IP ADDRESS
hello-6c4984d949-g28c4 10.8.0.13
test-c4d74c9d5-5pq9s 10.8.0.14
SERVICE NAME CLUSTER-IP
hello 172.30.137.226
test 172.30.159.119
ROUTE NAME HOSTNAME PORT
hello hello-network-policy.apps.ocp4.example.com 8080-tcp
===================================================================
```

3.2. Access the hello pod IP address from the test pod by using the oc rsh and curl commands.

```
[student@workstation ~]$ oc rsh test-c4d74c9d5-5pq9s \
 curl 10.8.0.13:8080 | grep Hello
 <h1>Hello, world from nginx!</h1>
```

3.3. Access the hello service IP address from the test pod by using the oc rsh and curl commands.

```
[student@workstation ~]$ oc rsh test-c4d74c9d5-5pq9s \
 curl 172.30.137.226:8080 | grep Hello
 <h1>Hello, world from nginx!</h1>
```

3.4. Access the hello route hostname by using the curl command.

```
[student@workstation ~]$ curl -s hello-network-policy.apps.ocp4.example.com | \
 grep Hello
 <h1>Hello, world from nginx!</h1>
```

4. Create a project named different-namespace that contains a deployment named sample-app.

4.1. Create the different-namespace project.

```
[student@workstation ~]$ oc new-project different-namespace
Now using project "different-namespace" on server "https://
api.ocp4.example.com:6443".
...output omitted...
```

4.2. Create the sample-app deployment from the registry.ocp4.example.com:8443/redhattraining/hello-world�nginx:v1.0 image. The web app listens on port 8080.

```
[student@workstation ~]$ oc new-app --name sample-app \
 --image registry.ocp4.example.com:8443/redhattraining/hello-world-nginx:v1.0
...output omitted...
--> Creating resources ...
 imagestream.image.openshift.io "sample-app" created
 deployment.apps "sample-app" created
 service "sample-app" created
--> Success
...output omitted...
```

5. Access the hello and test pods in the network-policy project from the sample-app pod in the different-namespace project.

5.1. In the second terminal, view the full name of the sample-app pod with the display-project-info.sh script.

```
 [student@workstation ~]$ ~/DO280/labs/network-policy/display-project-info.sh
===================================================================
PROJECT: network-policy
POD NAME IP ADDRESS
hello-6c4984d949-g28c4 10.8.0.13
test-c4d74c9d5-5pq9s 10.8.0.14
SERVICE NAME CLUSTER-IP
hello 172.30.137.226
test 172.30.159.119
ROUTE NAME HOSTNAME PORT
hello hello-network-policy.apps.ocp4.example.com 8080-tcp
===================================================================
PROJECT: different-namespace
POD NAME
sample-app-d5f945-spx9q
===================================================================
```

5.2. In the first terminal, access the hello pod IP address from the sample-app pod by using the oc rsh and curl commands.

```
[student@workstation ~]$ oc rsh sample-app-d5f945-spx9q \
 curl 10.8.0.13:8080 | grep Hello
 <h1>Hello, world from nginx!</h1>
```

5.3. Access the test pod IP address from the sample-app pod by using the oc rsh and curl commands. Target the IP address that was previously retrieved for the test

pod.

```
[student@workstation ~]$ oc rsh sample-app-d5f945-spx9q \
 curl 10.8.0.14:8080 | grep Hello
 <h1>Hello, world from nginx!</h1>
```

6. In the network-policy project, create a deny-all network policy by using the resource file at ~/DO280/labs/network-policy/deny-all.yaml.

6.1. Switch to the network-policy project.

```
[student@workstation ~]$ oc project network-policy
Now using project "network-policy" on server "https://api.ocp4.example.com:6443".
```

6.2. Change to the ~/DO280/labs/network-policy directory.

```
[student@workstation ~]$ cd ~/DO280/labs/network-policy
```

6.3. Use a text editor to update the deny-all.yaml file with an empty podSelector

field to target all pods in the network-policy project.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
 name: deny-all
spec:
 podSelector: {}
```

Note

A solution is provided at ~/DO280/solutions/network-policy/deny�all.yaml.

6.4. Create the network policy with the oc create command.

```
[student@workstation network-policy]$ oc create -f deny-all.yaml
networkpolicy.networking.k8s.io/deny-all created
```

7. Verify that the deny-all network policy forbids network access to pods in the network�policy project.

7.1. Verify that the test pod can no longer access the IP address of the hello pod. Wait

a few seconds, and then press Ctrl+C to exit the curl command that does not reply.

```
[student@workstation network-policy]$ oc rsh test-c4d74c9d5-5pq9s \
 curl 10.8.0.13:8080 | grep Hello
^C
command terminated with exit code 130
```

7.2. Switch to the different-namespace project.

```
[student@workstation network-policy]$ oc project different-namespace
Now using project "different-namespace" on server "https://
api.ocp4.example.com:6443".
```

7.3. Verify that the sample-app pod can no longer access the IP address of the test pod. Wait a few seconds, and then press Ctrl+C to exit the curl command that does not reply.

```
[student@workstation network-policy]$ oc rsh sample-app-d5f945-spx9q \
 curl 10.8.0.14:8080 | grep Hello
^C
command terminated with exit code 130
```

8. Create a network policy to allow traffic to the hello pod in the network-policy project from the sample-app pod in the different-namespace project via TCP on port 8080. Use the resource file at ~/DO280/labs/network-policy/allow-specific.yaml.

8.1. Use a text editor to replace the CHANGE_ME sections in the allow-specific.yaml file as follows:

\

```
...output omitted...
spec:
 podSelector:
 matchLabels:
 deployment: hello
 ingress:
 - from:
 - namespaceSelector:
 matchLabels:
```

```
 network: different-namespace
 podSelector:
 matchLabels:
 deployment: sample-app
 ports:
 - port: 8080
 protocol: TCP
```

Note

A solution is provided at ~/DO280/solutions/network-policy/allowspecific.yaml.

8.2. Apply the network policy from the allow-specific.yaml file with the oc create command.

```
[student@workstation network-policy]$ oc create -n network-policy -f \
 allow-specific.yaml
networkpolicy.networking.k8s.io/allow-specific created
```

8.3. View the network policies in the network-policy project.

```
[student@workstation network-policy]$ oc get networkpolicies -n network-policy
NAME POD-SELECTOR AGE
allow-specific deployment=hello 11s
deny-all <none> 5m6s
```

9. As the admin user, label the different-namespace namespace with the network=different-namespace label.

9.1. Log in as the admin user.

```
[student@workstation network-policy]$ oc login -u admin -p redhatocp
Login successful.
...output omitted...
```

9.2. Apply the network=different-namespace label with the oc label command.

```
[student@workstation network-policy]$ oc label namespace different-namespace \
 network=different-namespace
namespace/different-namespace labeled
```

Important

The allow-specific network policy uses labels to match the different�namespace namespace. By default, namespaces and projects do not get any labels

automatically.

9.3. Confirm that the different-namespace label was applied.

```
[student@workstation network-policy]$ oc describe namespace different-namespace
Name: different-namespace
Labels: network=different-namespace
...output omitted...
```

9.4. Log in as the developer user.

```
[student@workstation network-policy]$ oc login -u developer -p developer
Login successful.
...output omitted...
```

10. Verify that the sample-app pod can access the IP address of the hello pod, but cannot

access the IP address of the test pod.

10.1. Switch to the different-namespace project.

```
[student@workstation network-policy]$ oc project different-namespace
Already on project "different-namespace" on server "https://
api.ocp4.example.com:6443".
```

10.2. Access the hello pod in the network-policy namespace with the oc rsh and

curl commands via the 8080 port.

```
[student@workstation network-policy]$ oc rsh sample-app-d5f945-spx9q \
 curl 10.8.0.13:8080 | grep Hello
 <h1>Hello, world from nginx!</h1>
```

10.3. Verify that the hello pod cannot be accessed on another port. Because the network

policy allows access only to port 8080 on the hello pod, requests to any other port

are ignored and eventually time out. Wait a few seconds, and then press Ctrl+C to

exit the curl command when no response occurs.

```
[student@workstation network-policy]$ oc rsh sample-app-d5f945-spx9q \
 curl 10.8.0.13:8181 | grep Hello
^C
command terminated with exit code 130
```

10.4. Verify that the test pod cannot be accessed from the sample-app pod. Wait a few seconds, and then press Ctrl+C to exit the curl command when no response occurs.

```
[student@workstation network-policy]$ oc rsh sample-app-d5f945-spx9q \
 curl 10.8.0.14:8080 | grep Hello
^C
command terminated with exit code 130
```

11. Verify if the hello route cannot access the hello pod.

11.1. Verify if the hello pod cannot be accessed via its exposed route.

```
[student@workstation network-policy]$ curl -s \
 hello-network-policy.apps.ocp4.example.com
 <h1>Hello, world from nginx!</h1>
```

The lab environment is a single-node cluster. Because the ingress pods use host networking and the application pods are in the same node, the network policy does not block the traffic.

12. Create a network policy that allows traffic to the hello pod via the exposed route. Use the resource file at ~/DO280/labs/network-policy/allow-from-openshift￾ingress.yaml.

This step does not have an effect on the lab environment, because the lab environment is a single-node cluster. On a cluster with multiple nodes, this step is required for correct ingress operation.

12.1. Use a text editor to replace the CHANGE_ME values in the allow-from￾openshift-ingress.yaml file as follows:

```
...output omitted...
spec:
 podSelector: {}
 ingress:
 - from:
 - namespaceSelector:
 matchLabels:
 policy-group.network.openshift.io/ingress: ""
```

Note

A solution is provided at ~/DO280/solutions/network-policy/allow-from�openshift-ingress.yaml.

12.2. Apply the network policy from the allow-from-openshift-ingress.yaml file

with the oc create command.

```
[student@workstation network-policy]$ oc create -n network-policy -f \
 allow-from-openshift-ingress.yaml
networkpolicy.networking.k8s.io/allow-from-openshift-ingress created
```

12.3. View the network policies in the network-policy namespace.

```
[student@workstation network-policy]$ oc get networkpolicies -n network-policy
NAME POD-SELECTOR AGE
allow-from-openshift-ingress <none> 10s
allow-specific deployment=hello 8m16s
deny-all <none> 13m
```

12.4. Log in as the admin user.

```
[student@workstation network-policy]$ oc login -u admin -p redhatocp
Login successful.
...output omitted...
```

12.5. Access the hello pod via the exposed route with the curl command.

```
[student@workstation network-policy]$ curl -s \
 hello-network-policy.apps.ocp4.example.com | grep Hello
 <h1>Hello, world from nginx!</h1>
```

13. Close the terminal window that contains the output of the display-project-info.sh

script, and navigate to the home directory.

```
[student@workstation network-policy]$ cd
```

Finish

On the workstation machine, use the lab command to complete this exercise. This step is

important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish network-policy
```