OpenShift Network Policy Based SDN

OpenShift has a software defined network (SDN) inside the platform that is based on Open vSwitch. This SDN is used to provide connectivity between application components inside of the OpenShift environment. It comes with default network ranges pre-configured, although you can make changes to these should they conflict with your existing infrastructure, or for whatever other reason you may have.

The OpenShift Network Policy SDN plug-in allows projects to truly isolate their network infrastructure inside OpenShift's software defined network. While you have seen projects isolate resources through OpenShift's RBAC, the network policy SDN plugin is able to isolate pods in projects using pod and namespace label selectors.

The network policy SDN plugin was introduced in OpenShift 3.7, and more information about it and its configuration can be found in the networking documentation. Additionally, other vendors are working with the upstream Kubernetes community to implement their own SDN plugins, and several of these are supported by the vendors for use with OpenShift. These plugin implementations make use of appc/CNI, which is outside the scope of this lab.

Switch Your Project

Before continuing, make sure you are using a project that actually exists. If the last thing you did in the previous lab was delete a project, this will cause errors in the scripts in this lab.

```
oc project default
```

Execute the Creation Script

Only users with project or cluster administration privileges can manipulate **Project** networks.

Then, execute a script that we have prepared for you. It will create two **Projects** and then deploy a **DeploymentConfig** with a **Pod** for you:

```
bash {{ HOME_PATH }}/support/create-net-projects.sh
```

Examine the created infrastructure

Two **Projects** were created for you, `netproj-a` and `netproj-b`. Execute the following command to see the created resources:

```
oc get pods -n netproj-a
```

After a while you will see something like the following:

```
NAME           READY   STATUS              RESTARTS   AGE
ose-1-66dz2    0/1     ContainerCreating   0          7s
ose-1-deploy   1/1     Running             0          16s
```

Similarly:

```
oc get pods -n netproj-b
```

After a while you will see something like the following:

```
NAME           READY   STATUS      RESTARTS   AGE
ose-1-deploy   0/1     Completed   0          38s
ose-1-vj2gn    1/1     Running     0          30s
```

We will run commands inside the pod in the `netproj-a` **Project** that will connect to TCP port 5000 of the pod in the `netproj-b` **Project**.

Test Connectivity (should work)

Now that you have some projects and pods, let's test the connectivity between the pod in the `netproj-a` **Project** and the pod in the `netproj-b` **Project**.

To test connectivity between the two pods, run:

```
bash {{ HOME_PATH }}/support/test-connectivity.sh
```

You will see something like the following:

```
Getting Pod B's IP... 10.129.0.180
Getting Pod A's Name... ose-1-66dz2
Checking connectivity between Pod A and Pod B... worked
```

Note that the last line says `worked`. This means that the pod in the `netproj-a` **Project** was able to connect to the pod in the `netproj-b` **Project**.

This worked because, by default, with the network policy SDN, all pods in all projects can connect to each other.

Restricting Access

With the Network Policy based SDN, it's possible to restrict access in a project by creating a `NetworkPolicy` custom resource (CR).

For example, the following restricts all access to all pods in a **Project** where this `NetworkPolicy` CR is applied. This is the equivalent of a `DenyAll` default rule on a firewall:

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-by-default
spec:
  podSelector:
  ingress: []
```

Note that the `podSelector` is empty, which means that this will apply to all pods in this **Project**. Also note that the `ingress` list is empty, which means that there are no allowed `ingress` rules defined by this `NetworkPolicy` CR.

To restrict access to the pod in the `netproj-b` **Project** simply apply the above NetworkPolicy CR with:

```
oc create -n netproj-b -f {{ HOME_PATH }}/support/network-policy-block-all.yaml
```

Test Connectivity #2 (should fail)

Since the "block all by default" `NetworkPolicy` CR has been applied, connectivity between the pod in the `netproj-a` **Project** and the pod in the `netproj-b` **Project** should now be blocked.

Test by running:

```
bash {{ HOME_PATH }}/support/test-connectivity.sh
```

You will see something like the following:

```
Getting Pod B's IP... 10.129.0.180
Getting Pod A's Name... ose-1-66dz2
Checking connectivity between Pod A and Pod B..... FAILED!
```

Note the last line that says `FAILED!` . This means that the pod in the `netproj-a` **Project** was unable to connect to the pod in the `netproj-b` **Project** (as expected).

Allow Access

With the Network Policy based SDN, it's possible to allow access to individual or groups of pods in a project by creating multiple `NetworkPolicy` CRs.

The following allows access to port 5000 on TCP for all pods in the project with the label`run: ose`. The pod in the `netproj-b` project has this label.

The ingress section specifically allows this access from all projects that have the label`name: netproj-a`.

```
# allow access to TCP port 5000 for pods with the label "run: ose" specifically
# from projects with the label "name: netproj-a".
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: allow-tcp-5000-from-netproj-a-namespace
spec:
  podSelector:
    matchLabels:
      run: ose
  ingress:
  - ports:
    - protocol: TCP
      port: 5000
    from:
    - namespaceSelector:
        matchLabels:
          name: netproj-a
```

Note that the `podSelector` is where the local project's pods are matched using a specific label selector.

All `NetworkPolicy` CRs in a project are combined to create the allowed ingress access for the pods in the project. In this specific case the "deny all" policy is combined with the "allow TCP 5000" policy.

To allow access to the pod in the `netproj-b` **Project** from all pods in the `netproj-a` **Project**, simply apply the above NetworkPolicy CR with:

```
oc create -n netproj-b -f {{ HOME_PATH }}/support/network-policy-allow-all-from-netproj-a.yaml
```

Test Connectivity #3 (should work again)

Since the "allow access from `netproj-a` on port 5000" NetworkPolicy has been applied, connectivity between the pod in the `netproj-a` **Project** and the pod in the `netproj-b` **Project** should be allowed again.

Test by running:

```
bash {{ HOME_PATH }}/support/test-connectivity.sh
```

You will see something like the following:

```
Getting Pod B's IP... 10.129.0.180
Getting Pod A's Name... ose-1-66dz2
Checking connectivity between Pod A and Pod B... worked
```

Note the last line that says `worked`. This means that the pod in the `netproj-a` **Project** was able to connect to the pod in the `netproj-b` **Project** (as expected).

Last updated 2020-12-08 11:27:32 +0100