Exercise 4.1: Planning the Deployment

Overview

In this exercise we will investigate common network plugins. Each **kubelet** agent uses one plugin at a time. Due to complexity, the entire cluster uses one plugin which is configured prior to application deployment. Some plugins don't honor security configurations such as network policies. Should you design a deployment which and use a network policy there wouldn't be an error; the policy would have no effect. While developers typically wouldn't care much about the mechanics of it can affect the availability of features and troubleshooting of newly decoupled microservices.

While still new, the community is moving towards the **Container Network Interface** (**CNI**) specification (https://github.com/containernetworking/cni). This provides the most flexibility and features in the fast changing space of container networking.

A common alternative is **kubenet**, a basic plugin which relies on the cloud provider to handle routing and cross-node networking. In a previous lab exercise we configured **Project Cilium**. Classic and external modes are also possible. Several software defined network projects intended for Kubernetes have been created recently, with new features added regularly. Be aware that **Cilium** is a dynamic project with ongoing and frequent changes.

Quick Understanding of Network Plugins

While developers don't need to configure cluster networking, they may need to understand the plugin in use, and it's particular features and quirks. This section is to ensure you have made a quick review of the most commonly used plugins, and where to find more information as necessary.

1. Verify your nodes are using a CNI plugin. Read through the startup process of CNI. Each message begins with a type of message and a time stamp, followed by the details of the message. Use **TAB** to complete for your node name. Examine both the controller and one of your proxy pods.

```
student@cp:~$ kubectl -n kube-system logs kube-controller-manager-<TAB>
```

```
I0216 17:06:11.729548       1 serving.go:348] Generated self-signed cert in-memory
I0216 17:06:12.318263       1 controllermanager.go:196] Version: v1.31.1
I0216 17:06:12.321279       1 dynamic_cafile_content.go:156] "Starting controller" name=....
I0216 17:06:12.321311       1 secure_serving.go:200] Serving securely on 127.0.0.1:10257
....
```

```
student@cp:~$ kubectl -n kube-system logs kube-proxy-<TAB>
```

```
I0216 17:06:19.299029       1 node.go:163] Successfully retrieved node IP: 10.128.0.67
I0216 17:06:19.299337       1 server_others.go:138] "Detected node IP" address="10.128.0.67"
I0216 17:06:19.299515       1 server_others.go:561] "Unknown proxy mode, assuming iptables proxy"
↪  proxyMode=""
```

2. There are many CNI providers possible. The following list represents some of the more common choices, but it is not exhaustive. With many new plugins being developed there may be another which better serves your needs. Use these websites to answer questions which follow. While we strive to keep the answers accurate, please be aware that this area has a lot of attention and development and changes often.

   - **Project Calico**

https://docs.projectcalico.org/v3.0/introduction/

- **Calico with Canal**
  https://docs.projectcalico.org/v3.0/getting-started/kubernetes/installation/hosted/canal
- **Flannel**
  https://github.com/coreos/flannel
- **Cilium**
  http://cilium.io/
- **Kube Router**
  https://www.kube-router.io

3. Which of the plugins allow vxlans?
4. Which are layer 2 plugins?
5. Which are layer 3?
6. Which allow network policies?
7. Which can encrypt all TCP and UDP traffic?

Multi-container Pod Considerations

Using the information learned from this chapter, consider the following questions:

1. Which deployment method would allow the most flexibility, multiple applications per pod or one per pod?
2. Which deployment method allows for the most granular scalability?
3. Which have the best performance?
4. How many IP addresses are assigned per pod?
5. What are some ways containers can communicate within the same pod?
6. What are some reasons you should have multiple containers per pod?

Do you really know?

When and why would you use a multi-container pod?

Have you found a YAML example online?

Go back and review multi-container pod types and content on decoupling if you can't easily answer these questions. We touched on adding a second logging and a readiness container in a previous chapter and will work more with logging a future exercise.

Solution 4.1

Plugin Answers

1. Which of the plugins allow vxlans?

   Canal, Project Calico, Flannel, Weave Net, Cilium

2. Which are layer 2 plugins?

   Canal, Weave Net

3. Which are layer 3?

   Project Calico, Romana, Kube Router

4. Which allow network policies?

   Project Calico, Canal, Kube Router, Weave Net, Cilium

5. Which can encrypt all TCP and UDP traffic?

   Project Calico, Weave Net, Cilium

Multi Pod Answers

1. Which deployment method would allow the most flexibility, multiple applications per pod or one per Pod?

   One per pod

2. Which deployment method allows for the most granular scalability?

   One per pod

3. Which have the best inter-container performance?

   Multiple per pod.

4. How many IP addresses are assigned per pod?

   One

5. What are some ways containers can communicate within the same pod?

   IPC, loopback or shared filesystem access.

6. What are some reasons you should have multiple containers per pod?

   Lean containers may not have functionality like logging. Able to maintain lean execution but add functionality as necessary, like Ambassadors and Sidecar containers.