

Exploring Kubernetes Networking

You are working for a company called BeeBox, a subscription service that ships weekly shipments of bees to customers. The company is using Kubernetes to run their infrastructure of containerized applications.

The company has set up a new development cluster to be used by an external contractor's development team. The cluster seems to be working and the team is able to access it, but they are reporting there is an issue.

You have received the below email describing the problem. Fix the issue, and verify pods can communicate with one another using the Kubernetes network.

Hello,

We've been trying to set up a network connection between two pods, called "cyberdyne-frontend" and "testclient". However, whenever we try to create these pods, they never get up and running. Can you look into this and figure out what is going on? Once the pods are up and running, we need you to verify they can communicate via network as well.

Thanks for your help!

Brando Smith

CyberDyne System

So there are 2 task we need to do. First we are going to need to identify the problem that is preventing the pods from starting up. Eventually fix that problem so that those pods can go ahead and get up and running. Once that is completed we need to verify the 2 pods can communicate with each other via the cluster network.

Fix the Issue Causing Pods Not to Start Up

Lets take a look at the pods

1. Take a look at the pods

```
kubectl get pods
```

Figure 1-1

```
cloud_user@k8s-control:~$ kubectl get pods
NAME                READY   STATUS    RESTARTS   AGE
cyberdyne-frontend  0/1     Pending   0           75m
testclient           0/1     Pending   0           75m
```

Above you can see that both pods in this lab is in the pending status

2. Take a look at the node status

```
kubectl get nodes
```

Figure 1-2

```
cloud_user@k8s-control:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
k8s-control         NotReady  control-plane  77m   v1.24.0
k8s-worker1         NotReady  <none>      77m   v1.24.0
```

Above you can see the nodes are set to not ready as well.

3. Lets get more info on the k8s-worker1

```
kubectl describe node k8s-worker1
```

You can do this to check events, CPU, versions, and so on. Nothing looked out the ordinary. There was the coredns in the pending state but that shouldn't be the reason why it is not running.

4. Check the networking plugin pods

```
kubectl get pods -n kube-system
```

Figure 1-3

```
cloud_user@k8s-control:~$ kubectl get pods -n kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-6d4b75cb6d-bcwsj            0/1     Pending   0           80m
coredns-6d4b75cb6d-cqd24            0/1     Pending   0           80m
etcd-k8s-control                    1/1     Running   0           81m
kube-apiserver-k8s-control           1/1     Running   0           81m
kube-controller-manager-k8s-control 1/1     Running   0           81m
kube-proxy-7v2q8                    1/1     Running   0           80m
kube-proxy-hmbbh                    1/1     Running   0           80m
kube-scheduler-k8s-control           1/1     Running   0           81m
```

Looks like a networking pod(s) is missing. So the pods may not be running due to a networking plugin never being installed.

*In previous labs we used Calico plugin

5. Install the Calico plugin

```
kubectl apply -f https://docs.projectcalico.org/v3.15/manifests/calico.yaml
```

6. Once that is installed, check the status of the nodes again

```
kubectl get nodes
```

Figure 1-4

```
cloud_user@k8s-control:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
k8s-control         Ready     control-plane   90m   v1.24.0
k8s-worker1         NotReady  <none>         90m   v1.24.0
cloud_user@k8s-control:~$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
k8s-control         Ready     control-plane   91m   v1.24.0
k8s-worker1         Ready     <none>         90m   v1.24.0
```

As you can see above I initiated the command and waited for another min and saw that both status turned to ready. We still need to check the nodes next. Which checking the nodes as well in the ready status.

Verify You Can Communicate between Pods Using the Cluster Network

So we need to test each pod by using the kubectl exec command

7. Check the pods -o wide

Figure 1-5

```
cloud_user@k8s-control:~$ kubectl get pods -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP             NODE
cyberdyne-frontend  1/1     Running   0           94m   192.168.194.65 k8s-worker1
testclient          1/1     Running   0           94m   192.168.194.66 k8s-worker1
```

8. Run curl on the IP address of the cyberdyne-frontend Pod (which will be listed in the output from the previous command)

```
kubectl exec testclient -- curl 192.168.194.65
```

Figure 1-6

```
cloud_user@k8s-control:~$ kubectl exec testclient -- curl 192.168.194.65
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             Dload  Upload  Total   Spent    Left   Speed
100  612  100  612    0     0  551k      0  --:--:-- --:--:-- --:--:--  597k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

We will test the cyberdyne-frontend IP address. We got the HTML of an Nginx page. Which means the pods are able to communicate.