Lab 2.1 - Deploying a New Cluster

Overview

We will create a two-node Ubuntu 16.04 cluster. Using two nodes allows an understanding of issues and configurations found in a production environment. While 2 vCPU and 8G of memory allows for quick labs, you could also use much smaller VMs. Other Linux distributions should work in a very similar manner, but have not been tested.

Regardless of the platform used, VirtualBox, VMWare, AWS, GCE, or even bare metal, please remember that security software like SELinux and Firewalls can prevent the labs from working. While this is not something you should do in production, consider disabling the firewall and security software. GCE requires a new VPC to be created and a rule allowing all traffic to be included. The use of Wireshark can be a helpful place to start with troubleshooting, if you're unable to open all ports. Currently, **kubeadm** requires that swap to be turned off on every node. The **`swapoff -a`** command will do this until your next reboot, with various methods to disable swap persistently. Cloud providers typically provide instances with swap disabled.

To assist with setting up your cluster, please download the tarball of shell scripts and YAML files. The **`k8sMaster.sh`** and **`k8sSecond.sh`** scripts deploy a Kubernetes cluster using **kubeadm** and use **Project Calico** for networking.

```
student@ckad-1:~$ wget https://training.linuxfoundation.org/cm/LFD259/
     --user=LFtraining --password=Penguin2014
 student@ckad-1:~$ tar -xvf lfd259-example-files.tar
```

Deploy a Master Node using kubeadm

Review the script to install and begin the configuration of the master Kubernetes server.

```
student@ckad-1:~$ cat lfd259/k8sMaster.sh
#!/bin/bash -x
echo "This script is written to work with Ubuntu 16.04"
sleep 3
echo
echo "Disable swap until next reboot"
echo
sudo swapoff -a

echo "Update the local node"
sudo apt-get update && sudo apt-get upgrade -y
echo
echo "Install Docker"
sleep 3

sudo apt-get install -y docker.io
echo
echo "Install kubeadm and kubectl"
sleep 3

sudo sh -c "echo 'deb http://apt.kubernetes.io/ kubernetes-xenial main' >>
/etc/apt/sources.list.d/kubernetes.list”

<output_omitted>
```

Run the script as an argument to the bash shell. You will need the **kubeadm join** command shown near the end of the output when you add the minion node in a future step.

```
     student@ckad-1:~$ bash lfd259/k8sMaster.sh
<output_omitted>
```

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

```
mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

You should now deploy a pod network to the cluster.

Run **kubectl apply -f [podnetwork].yaml** with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node as root:

```
kubeadm join --token 118c3e.83b49999dc5dc034 10.128.0.3:6443
--discovery-token-ca-cert-hash
sha256:40aa946e3f53e38271bae24723866f56c86d77efb49aedeb8a70cc189bfe2e1d

--2018-03-30 19:32:41--  https://goo.gl/eWLkzb

<output_omitted>

NAME            STATUS     ROLES      AGE       VERSION
ckad-1          Ready      master     1m        v1.9.1
```

Deploy a Minion Node

Open a separate terminal into your second node. Having both terminal sessions allows you to monitor the status of the cluster while adding the second node.

```
student@ckad-2:~$ cat lfd259/k8sSecond.sh
#!/bin/bash -x
sudo apt-get update && sudo apt-get upgrade -y

<output_omitted>
```

Run the script on the second node:

```
student@ckad-2:~$ bash lfd259/k8sSecond.sh

<output_omitted>
```

When the script is done, the minion node is ready to join the cluster. The **kubeadm join** statement can be found near the end of the **kubeadm init** output. Your nodes will use a different IP address and hashes than the example below. You will need to prepend **sudo** to run the script copied from the master node.

```
student@ckad-2:~$ sudo kubeadm join --token 118c3e.83b49999dc5dc034 \
 10.128.0.3:6443 --discovery-token-ca-cert-hash \
sha256:40aa946e3f53e38271bae24723866f56c86d77efb49aedeb8a70cc189bfe2e1d
```

Configure the Master Node

Return to the master node. We will configure command line completion and verify that both nodes have been added to the cluster. The first command will configure completion in the current shell. The second command will ensure future shells have completion:

```
student@ckad-1:~$ source <(kubectl completion bash)
student@ckad-1:~$ echo "source <(kubectl completion bash)" >> ~/.bashrc
```

Verify that both nodes are part of the cluster. It may take a minute for the second node to reach a *Ready* state.

```
student@ckad-1:~$ kubectl get node
NAME          STATUS    ROLES     AGE       VERSION
ckad-1        Ready     master    5m        v1.9.1
ckad-2        Ready     <none>    2m        v1.9.1
```

Create a Simple Deployment

We will use the **kubectl** command for the majority of work with Kubernetes. Review the help output to become familiar with commands options and arguments:

```
student@ckad-1:~$ kubectl --help
kubectl controls the Kubernetes cluster manager.
```

Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/.

```
Basic Commands (Beginner):
  create         Create a resource from a file or from stdin.
  expose         Take a replication controller, service, deployment or pod
and
```

```
expose it as a new Kubernetes Service
  run             Run a particular image on the cluster
  set             Set specific features on objects
  run-container   Run a particular image on the cluster. This command is
deprecated, use "run" instead

Basic Commands (Intermediate):
<output_omitted>
```

With more than 40 arguments, you can explore each by also using the **`--help`** option. Take a closer look at a few, starting with *taint*, for example:

```
student@ckad-1:~$ kubectl taint --help
Update the taints on one or more nodes.

  * A taint consists of a key, value, and effect. As an argument here, it
is
expressed as key=value:effect.
  * The key must begin with a letter or number, and may contain letters,
numbers, hyphens, dots, and underscores, up to  253 characters.
  * Optionally, the key can begin with a DNS subdomain prefix and a single
'/',
like example.com/my-app
<output_omitted>
```

By default, the master node will not allow general containers to be deployed for security reasons. This is via a *taint*. Only containers which tolerate this taint will be scheduled on this node. As we only have two nodes in our cluster, we will remove the taint, allowing containers to be deployed on both nodes. The following command will remove the taint from all nodes, so you should see one success and one “not found” error. The minion node does not have the taint to begin with. Note the minus sign at the end of the command, which removes the preceding value.

```
student@ckad-1:~$ kubectl taint nodes --all node-role.kubernetes.io/master-
node “ckad-1” untainted
taint "node-role.kubernetes.io/master:" not found
```

Now, run a containerized webserver **nginx**. Use **`kubectl run`** to create a simple, single replica deployment running the **nginx** web server.

```
student@ckad-1:~$ kubectl run firstpod --image=nginx
deployment.apps "firstpod" created
```

Verify the new deployment exists and that the desired number of Pods matches the current number. Using a comma, you can request two resource types at once. The **`<Tab>`** key can be helpful. Type enough of the word to be unique and press the Tab key - it should complete the word. The deployment should show a number 1 for each value, such that the desired number of pods matches the up-to-date and running number. The pod should show zero restarts.

```
student@ckad-1:~$ kubectl get deployment,pod
NAME                                DESIRED   CURRENT   UP-TO-DATE   AVAILABLE
AGE
deployment.extension/firstpod      1         1         1            1
13s
NAME                                READY     STATUS    RESTARTS   AGE
pod/firstpod-7d99ffc75-247kl       1/1       Running   0          13s
```

View the details of the deployment, then the pod. Work through the output slowly. Knowing what a healthy deployment and pod look like can be helpful when troubleshooting issues. Again, the **`<Tab>`** key can be helpful when using long auto-generated object names. You should be able to type **`firstpod<Tab>`** and the name will complete when viewing the pod.

```
student@ckad-1:~$ kubectl describe deployment firstpod
Name:                   firstpod
Namespace:              default
CreationTimestamp:      Fri, 30 Mar 2018 16:46:57 +0000
Labels:                 run=firstpod
Annotations:            deployment.kubernetes.io/revision=1
Selector:               run=firstpod
Replicas:               1 desired | 1 updated | 1 total | 1 available | 0
unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
<output_omitted>

student@ckad-1:~$ kubectl describe pod firstpod-7d99ffc75-247kl
     Name:            firstpod-7d99ffc75-247kl
     Namespace:       default
     Node:            ckad-2/10.128.0.2
     Start Time:      Fri, 30 Mar 2018 16:46:57 +0000
     Labels:          pod-template-hash=385599731
                      run=firstpod
     Annotations:     <none>
     Status:          Running
     IP:              192.168.55.100
     Controlled By:   ReplicaSet/firstpod-7d99ffc75
     Containers:
       firstpod:
     <output_omitted>
```

Note that the resources are in the **default** namespace. Get a list of available namespaces:

```
student@ckad-1:~$ kubectl get namespaces
NAME          STATUS    AGE
default       Active    20m
kube-public   Active    20m
kube-system   Active    20m
```

There are two other namespaces. Look at the pods in the **kube-system** namespace:

```
student@ckad-1:~$ kubectl get pod -n kube-system
NAME                                      READY     STATUS    RESTARTS
AGE

calico-etcd-rvrpk                         1/1       Running   1
20m
calico-kube-controllers-d554689d5-lm687   1/1       Running   1
20m
calico-node-2ck9g                         2/2       Running   4
19m
calico-node-kkxvl                         2/2       Running   3
20m
etcd-ckad-1                               1/1       Running   1
20m
<output_omitted>
```

Now, look at the pods in a namespace that does not exist. Note that you do not receive an error:

```
student@ckad-1:~$ kubectl get pod -n fakenamespace
No resources found.
```

You can also view resources in all namespaces at once:

```
student@ckad-1:~$ kubectl get pod --all-namespaces
NAMESPACE     NAME                                      READY     STATUS
RESTARTS   AGE
default       firstpod-7d99ffc75-247kl                   1/1       Running
0          5m
```

```
kube-system   calico-etcd-z49kx                          1/1       Running
0          31m
kube-system   calico-kube-controllers-d554689d5-pfszw    1/1       Running
0          31m
<output_omitted>
```

View several resources at once. Note that most resources have a short name, such as **rs** for ReplicaSet, **po** for Pod, **svc** for Service, and **ep** for endpoint.

```
student@ckad-1:~$ kubectl get deploy,rs,po,svc,ep
NAME                             DESIRED    CURRENT    UP-TO-DATE    AVAILABLE
AGE
deployment.extensions/firstpod   1          1          1             1
4m

NAME                                        DESIRED    CURRENT    READY     AGE
replicaset.extensions/firstpod-7d99ffc75    1          1          1         4m

NAME                        READY     STATUS     RESTARTS    AGE
pod/firstpod-7d99ffc75      1/1       Running    0           4m

NAME                     TYPE         CLUSTER-IP    EXTERNAL-IP    PORT(S)    AGE
service/kubernetes       ClusterIP    10.96.0.1     <none>         443/TCP    21m

NAME                      ENDPOINTS          AGE
endpoints/kubernetes      10.128.0.3:6443    21m
```

Delete the *ReplicaSet* and view the resources again. Note that the time on the *ReplicaSet* and the Pod it controls is now less than a minute. The deployment controller restarted the *ReplicaSet*, which restarted the Pod when the desired configuration did not match the current status.

```
student@ckad-1:~$ kubectl delete rs firstpod-7d99ffc75
replicaset.extensions "firstpod-7d99ffc75" deleted


student@ckad-1:~$ kubectl get deployment,rs,po,svc,ep
NAME                             DESIRED    CURRENT    UP-TO-DATE    AVAILABLE
AGE

deployment.extensions/firstpod   1          1          1             1
7m
```

```
NAME                                       DESIRED   CURRENT   READY
AGE
replicaset.extensions/firstpod-7d99ffc75   1         1         1
12s

NAME                           READY     STATUS    RESTARTS   AGE
pod/firstpod-7d99ffc75-p9hbw   1/1       Running   0          12s

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   24m

NAME                   ENDPOINTS         AGE
endpoints/kubernetes   10.128.0.2:6443   24m
```

This time, delete the top-level controller. After about 30 seconds for everything to shut down, you should only see the cluster service and endpoint remain:

```
student@ckad-1:~$ kubectl delete deployment firstpod
deployment.extensions "firstpod" deleted

student@ckad-1:~$ kubectl get deployment,rs,po,svc,ep
NAME         TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   24m

NAME         ENDPOINTS         AGE
kubernetes   10.128.0.3:6443   24m
```