Exercise 2.2: Deploy a New Cluster

Deploy a Control Plane Node using Kubeadm

1. Log into your nodes using **PuTTY** or using **SSH** from a terminal window. Unless the instructor tells you otherwise the user name to use will be **student**. You may need to change the permissions on the `pem` (or `ppk` on windows) file as shown in the following commands. Your file name and node IP address will probably be different.

   If using PuTTY, search for instructions on using a `ppk` key to access an instance. Use the **student** username when asked by PuTTY.

   ```
   localTerm:~$ chmod 400 LF-Class.pem
   localTerm:~$ ssh -i LF-Class.pem student@WW.XX.YY.ZZ
   ```

   ```
   student@cp:~$
   ```

2. Use the **wget** and **tar** commands shown above to download and extract the course tarball to your cp node. Install **wget** if the command is not found on your instance.

3. Be sure to type commands found in the exercises, instead of trying to copy and paste. Typing will cause you to learn the commands, whereas copy and paste tends not to be absorbed. Also, use the YAML files included in the course tarball, as white space issues can slow down the labs and be discouraging.

4. The virtual machine's hostname might vary depending on how it was created. If you want your VM to have the same hostname as in our lab examples, run the following command. This is an OPTIONAL step.

   ```
   student@cp:~$ sudo hostnamectl set-hostname cp
   student@cp:~$ bash
   ```

5. Review the script to install and begin the configuration of the cp kubernetes server. You may need to change the **find** command search directory for your home directory depending on how and where you downloaded the tarball.

   A **find** command is shown if you want to locate and copy to the current directory instead of creating the file. Mark the command for reference as it may not be shown for future commands.

   ```
   student@cp:~$ find $HOME -name <YAML File>
   student@cp:~$ cp LFD259/<Some Path>/<YAML File> .
   ```

   ```
   student@cp:~$ find $HOME -name k8scp.sh
   ```

   ```
   student@cp:~$ more LFD259/SOLUTIONS/s_02/k8scp.sh
   ```

   ```
   ....
   # Install the Kubernetes software, and lock the version
   sudo apt update
   sudo apt-get -y install kubelet=1.32.1-1.1 kubeadm=1.32.1-1.1 kubectl=1.32.1-1.1
   sudo apt-mark hold kubelet kubeadm kubectl

   # Ensure Kubelet is running
   sudo systemctl enable --now kubelet
   ```

```
# Disable swap just in case
sudo swapoff -a
....
```

6. 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 worker/minion node in a future step. Use the **tee** command to save the output of the script, in case you cannot scroll back to find the `kubeadm join` in the script output. Please note the following is one command and then its output.

   Using **Ubuntu 24.04** you may be asked questions during the installation. Allow restarts (yes) and use the local, installed software if asked during the update, usually (option 2).

   Copy files to your home directory first.

```
student@cp:~$ cp LFD259/SOLUTIONS/s_02/k8scp.sh .

student@cp:~$ bash k8scp.sh | tee $HOME/cp.out
```

```
<output_omitted>

Your Kubernetes control-plane 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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.128.0.93:6443 --token zu7icq.2o0ouw7s044yi9w9 \
        --discovery-token-ca-cert-hash
        ↪  sha256:a26f2166ad39b6aba0fc0dd663ac2b3455af19526bd9fe80af5439255dd
<output_omitted>

NAME        STATUS     ROLES           AGE       VERSION
cp          NotReady   control-plane   20s       v1.32.1


Continue to the next step
```

Deploy a Worker Node

7. Open a separate terminal into your **second node**, which will be your worker. Having both terminal sessions allows you to monitor the status of the cluster while adding the second node. Change the color or other characteristic of the second terminal to make it visually distinct from the first. This will keep you from running commands on the incorrect instance, which probably won't work.

8. The virtual machine's hostname might vary depending on how it was created. If you want your VM to have the same hostname as in our lab examples, run the following command. This is an OPTIONAL step.

```
student@worker:~$ sudo hostnamectl set-hostname worker
student@worker:~$ bash
```

9. Use the previous **wget** command download the tarball to the worker node. Extract the files with **tar** as before. Find and copy the k8sWorker.sh file to student's home directory then view it. You should see the same early steps as found in the cp setup script.

```
student@worker:~$ more k8sWorker.sh


....
# Install the Kubernetes software, and lock the version
sudo apt update
sudo apt-get -y install kubelet=1.32.1-1.1 kubeadm=1.32.1-1.1 kubectl=1.32.1-1.1
sudo apt-mark hold kubelet kubeadm kubectl

# Ensure Kubelet is running
sudo systemctl enable --now kubelet

# Disable swap just in case
sudo swapoff -a
....
```

10. Run the script on the **second node**. Again please note you may have questions during the update. Allow daemons to restart, type yes, and use the local installed version, usually option 2. For troubleshooting you may want to write output to a a file using the **tee** command.

```
student@worker:~$ bash k8sWorker.sh | tee worker.out
```

```
<output_omitted>
```

11. When the script is done the worker node is ready to join the cluster. The kubeadm join statement can be found near the end of the kubeadm init output on the cp node. It should also be in the file cp.out as well. Your nodes will use a different IP address and hashes than the example below. You'll need to pre-pend **sudo** to run the script copied from the cp node. Also note that some non-Linux operating systems and tools insert extra characters when multi-line samples are copied and pasted. Copying one line at a time solves this issue. A helpful command to find a line in a file may be **grep -A2 join cp.out** on the control plane node.

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

```
<output_omitted>
```

Configure the Control Plane Node

12. Return to the cp node. Install a text editor. While the lab uses **vim**, any text editor such as **emacs** or **nano** will work. Be aware that Windows editors may have issues with special characters. Also install the **bash-completion** package, if not already installed. Use the locally installed version of a package if asked.

```
student@cp:~$ sudo apt-get install bash-completion vim -y
```

```
<output_omitted>
```

13. We will configure command line completion and verify 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. You may need to exit the shell and log back in for command completion to work without error.

```
student@cp:~$ source <(kubectl completion bash)

student@cp:~$ echo "source <(kubectl completion bash)" >> $HOME/.bashrc
```

14. Verify that both nodes are part of the cluster. And show a `Ready` state.

```
student@cp:~$ kubectl get node
```

```
NAME     STATUS   ROLES           AGE     VERSION
cp       Ready    control-plane   4m11s   v1.32.1
worker   Ready    <none>          61s     v1.32.1
```

15. 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@cp:~$ 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

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

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

```
student@cp:~$ 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>
```

17. By default the cp 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. This is not typically done in a production environment for security and resource contention reasons. The following command will remove the taint from all nodes, so you should see one success and one `not found` error. The worker/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@cp:~$  kubectl describe nodes | grep -i taint
```

```
Taints:             node-role.kubernetes.io/control-plane:NoSchedule
Taints:             <none>
```

```
student@cp:~$ kubectl taint nodes --all node-role.kubernetes.io/control-plane-
```

```
node/cp untainted
error: taint "node-role.kubernetes.io/control-plane" not found
```

18. Check that both nodes are without a `Taint`. If they both are without taint the `nodes` should now show as `Ready`. It may take a minute or two for all infrastructure pods to enter `Ready` state, such that the `nodes` will show a `Ready` state.

```
student@cp:~$ kubectl describe nodes | grep -i taint
```

```
Taints:           <none>
Taints:           <none>
```

```
student@cp:~$ kubectl get nodes
```

```
NAME     STATUS   ROLES           AGE     VERSION
cp       Ready    control-plane   6m1s    v1.32.1
worker   Ready    <none>          5m31s   v1.32.1
```