



## Exercise 2.2: Deploy a New Cluster

### Deploy a Master 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 file as shown in the following commands. Your file and node IP address will probably be different.

```
localTerm:~$ chmod 400 LFD459.pem
localTerm:~$ ssh -i LFD459.pem student@WW.XX.YY.ZZ
```

```
1 student@ckad-1:~$
```

2. Review the script to install and begin the configuration of the master kubernetes server. You may need to change the **find** command search directory which uses tilde 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@ckad-1:~$ find $HOME -name <YAML File>
student@ckad-1:~$ cp LFD259/<Some Path>/<YAML File> .
```

```
student@ckad-1:~$ find $HOME -name k8sMaster.sh
```

```
student@ckad-1:~$ more LFD259/SOLUTIONS/s_02/k8sMaster.sh
```

SH

k8sMaster.sh

```
#!/bin/bash -x
## TxS 08-2020
## v1.19.0 CKAD
echo "This script is written to work with Ubuntu 18.04"
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, kubelet, and kubect1"
sleep 3

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

SH

```

sudo sh -c "curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -"

sudo apt-get update

sudo apt-get install -y kubeadm=1.19.0-00 kubelet=1.19.0-00 kubect1=1.19.0-00

sudo apt-mark hold kubelet kubeadm kubect1

echo
echo "Installed - now to get Calico Project network plugin"

## If you are going to use a different plugin you'll want
## to use a different IP address, found in that plugins
## readme file.

sleep 3

sudo kubeadm init --kubernetes-version 1.19.0 --pod-network-cidr 192.168.0.0/16

sleep 5

echo "Running the steps explained at the end of the init output for you"

mkdir -p $HOME/.kube

sleep 2

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sleep 2

sudo chown $(id -u):$(id -g) $HOME/.kube/config

echo "Apply Calico network plugin from ProjectCalico.org"
echo "If you see an error they may have updated the yaml file"
echo "Use a browser, navigate to the site and find the updated file"

kubect1 apply -f https://docs.projectcalico.org/manifests/calico.yaml

echo
echo
sleep 3
echo "You should see this node in the output below"
echo "It can take up to a mintue for node to show Ready status"
echo
kubect1 get node
echo
echo
echo "Script finished. Move to the next step"

```

3. 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 18** you will be asked questions during the installation. Allow restarts and use the local, installed software if asked during the update, usually option 2.

Copy files to your home directory first.

```

student@ckad-1:~$ cp LFD259/SOLUTIONS/s_02/k8sMaster.sh .

student@ckad-1:~$ bash k8sMaster.sh | tee $HOME/master.out
1 <output_omitted>
2
3 Your Kubernetes master has initialized successfully!
4
5     To start using your cluster, you need to run the
6     following as a regular user:
7
8     mkdir -p $HOME/.kube
9     sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
10    sudo chown $(id -u):$(id -g) $HOME/.kube/config
11
12    You should now deploy a pod network to the cluster.
13    Run \verb?kubectl apply -f [podnetwork].yaml? with one
14    of the options listed at:
15    https://kubernetes.io/docs/concepts/cluster-administration/addons/
16
17    You can now join any number of machines by running the
18    following on each node as root:
19
20    kubeadm join 10.128.0.3:6443 --token 69rdjq.2x2012j9ncexy37b
21    --discovery-token-ca-cert-hash
22
23    sha256:72143e996ef78301191b9a42184124416aebcf0c7f363adf9208f9fa599079bd
24
25 <output_omitted>
26
27 + kubectl get node
28 NAME                STATUS    ROLES    AGE    VERSION
29 ckad-1              NotReady master   18s    v1.19.0
30 + echo
31 + echo 'Script finished. Move to the next step'
32 Script finished. Move to the next step

```

## 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. 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.

Use the previous **wget** command to download the tarball to the second node. Extract the files with **tar** as before. Find and copy the **k8sSecond.sh** file to student's home directory then view it. You should see the same early steps as on the master system.

```
student@ckad-2:~$ more k8sSecond.sh
```

```

SH k8sSecond.sh
#!/bin/bash -x
## TxS 08-2020
## CKAD for 1.19.0
##
echo " This script is written to work with Ubuntu 18.04"
echo
sleep 3
echo " Disable swap until next reboot"

```

SH

```

echo
sudo swapoff -a

echo " Update the local node"
sleep 2
sudo apt-get update && sudo apt-get upgrade -y
echo
sleep 2

echo " Install Docker"
sleep 3
sudo apt-get install -y docker.io

echo
echo " Install kubeadm, kubelet, and kubectl"
sleep 2
sudo sh -c
↪ "echo 'deb http://apt.kubernetes.io/ kubernetes-xenial main' >> /etc/apt/sources.list.d/kubernetes.list"

sudo sh -c "curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -"

sudo apt-get update

sudo apt-get install -y kubeadm=1.19.0-00 kubelet=1.19.0-00 kubectl=1.19.0-00

sudo apt-mark hold kubelet kubeadm kubectl
echo
echo " Script finished. You now need the kubeadm join command"
echo " from the output on the master node"
echo

```

5. Run the script on the **second node**. Again please note you may have questions during the update. Allow daemons to restart and use the local installed version, usually option 2.

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

```
1 <output_omitted>
```

6. 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 on the master node. It should also be in the file `master.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 master 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.

```
student@ckad-2:~$ sudo kubeadm join --token 118c3e.83b49999dc5dc034 \
10.128.0.3:6443 --discovery-token-ca-cert-hash \
sha256:40aa946e3f53e38271bae24723866f56c86d77efb49aede8a70cc189bfe2e1d
```

```
1 <output_omitted>
```

## Configure the Master Node

7. Return to the master 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@ckad-1:~$ sudo apt-get install bash-completion vim -y
```

```
1 <output_omitted>
```

8. 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@ckad-1:~$ source <(kubectl completion bash)
```

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

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

```
student@ckad-1:~$ kubectl get node
```

```
1 NAME          STATUS    ROLES    AGE      VERSION
2 ckad-1        NotReady master   4m11s    v1.19.1
3 ckad-2        NotReady <none>    3m6s     v1.19.1
```

10. 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
```

```
1 kubectl controls the Kubernetes cluster manager.
2
3 Find more information at:
4   https://kubernetes.io/docs/reference/kubectl/overview/
5
6 Basic Commands (Beginner):
7   create      Create a resource from a file or from stdin.
8   expose      Take a replication controller, service,
9   deployment or pod and expose it as a new Kubernetes Service
10  run          Run a particular image on the cluster
11  set          Set specific features on objects
12
13 Basic Commands (Intermediate):
14 <output_omitted>
```

11. 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@ckad-1:~$ kubectl taint --help
```

```
1 Update the taints on one or more nodes.
2
3 * A taint consists of a key, value, and effect. As an argument
4   here, it is expressed as key=value:effect.
5 * The key must begin with a letter or number, and may contain
6   letters, numbers, hyphens, dots, and underscores, up to
7   253 characters.
8 * Optionally, the key can begin with a DNS subdomain prefix
9   and a single '/',
10 like example.com/my-app
11 <output_omitted>
```

12. 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. 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@ckad-1:~$ kubectl describe nodes | grep -i Taint
```

```
1 Taints:                node-role.kubernetes.io/master:NoSchedule
2 Taints:                <node>
```

```
student@ckad-1:~$ kubectl taint nodes --all node-role.kubernetes.io/master-
```

```
1 node/ckad-1 untainted
2 taint "node-role.kubernetes.io/master:" not found
```

13. 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@ckad-1:~$ kubectl describe nodes | grep -i taint
```

```
1 Taints:                <none>
2 Taints:                <none>
```

```
student@ckad-1:~$ kubectl get nodes
```

	NAME	STATUS	ROLES	AGE	VERSION
1	ckad-1	Ready	master	6m1s	v1.19.0
2	ckad-2	Ready	<none>	5m31s	v1.19.0