



## Exercise 3.2: Grow the Cluster

Open another terminal and connect into a your second node. Install **Docker** and Kubernetes software. These are the many, but not all, of the steps we did on the cp node.

This book will use the **worker** prompt for the node being added to help keep track of the proper node for each command. Note that the prompt indicates both the user and system upon which run the command.

- Using the same process as before connect to a second node. If attending an instructor-led class session, use the same .pem key and a new IP provided by the instructor to access the new node. Giving a different title or color to the new terminal window is probably a good idea to keep track of the two systems. The prompts can look very similar.

**PLEASE NOTE:** If you chose to use **crio** instead of **Docker** as the container engine you should reference the previous portion of the lab for detailed installation steps.

(a) `student@worker:~$ sudo -i`

(b) `root@worker:~# apt-get update && apt-get upgrade -y`

```
1 <Again allow services to restart and keep the local version of software>
```

(c) Install a container engine

i. **IF** you chose Docker on the cp:

```
root@worker:~# apt-get install -y docker.io
```

ii. **IF** you chose cri-o on the cp:

See several previous steps for cri-o installation details.  
Bash history on cp may make copy paste easy.

(d) `root@worker:~# apt-get install -y vim`

(e) `root@worker:~# vim /etc/apt/sources.list.d/kubernetes.list`

```
1 deb http://apt.kubernetes.io/ kubernetes-xenial main
```

(f) `root@worker:~# curl -s \`  
`https://packages.cloud.google.com/apt/doc/apt-key.gpg \`  
`| apt-key add -`

(g) `root@worker:~# apt-get update`

(h) `root@worker:~# apt-get install -y \`  
`kubeadm=1.20.1-00 kubelet=1.20.1-00 kubectl=1.20.1-00`

(i) `root@worker:~# apt-mark hold kubeadm kubelet kubectl`

- Find the IP address of your **cp** server. The interface name will be different depending on where the node is running. Currently inside of **GCE** the primary interface for this node type is `ens4`. Your interfaces names may be different. From the output we know our cp node IP is 10.128.0.3.

`student@cp:~$ hostname -i`

```
1 10.128.0.3
```

`student@cp:~$ ip addr show ens4 | grep inet`

```
1 inet 10.128.0.3/32 brd 10.128.0.3 scope global ens4
2 inet6 fe80::4001:aff:fe8e:2/64 scope link
```

3. At this point we could copy and paste the **join** command from the cp node. That command only works for 2 hours, so we will build our own **join** should we want to add nodes in the future. Find the token on the cp node. The token lasts 2 hours by default. If it has been longer, and no token is present you can generate a new one with the **sudo kubeadm token create** command, seen in the following command.

```
student@cp:~$ sudo kubeadm token list
```

TOKEN	TTL	EXPIRES	USAGES
DESCRIPTION		EXTRA GROUPS	
bml44w.3owxl50rrtymamt7	2h	2021-05-27T18:49:41Z	authentication,signing
<none>			system:bootstrappers:kubeadm:default-node-token

4. We'll assume you are adding a node more than two hours later and create a new token, to use as part of the **join** command. You may get a Docker not found warning in output if using cri-o.

```
student@cp:~$ sudo kubeadm token create
```

```
1 27eee4.6e66ff60318da929
```

5. Create and use a Discovery Token CA Cert Hash created from the cp to ensure the node joins the cluster in a secure manner. Run this on the cp node or wherever you have a copy of the CA file. You will get a long string as output. Also note that a copy and paste from a PDF sometimes has issues with the caret (^) and the single quote (') found at the end of the command.

```
student@cp:~$ openssl x509 -pubkey \
    -in /etc/kubernetes/pki/ca.crt | openssl rsa \
    -pubin -outform der 2>/dev/null | openssl dgst \
    -sha256 -hex | sed 's/^.* //'
```

```
1 (stdin)= 6d541678b05652e1fa5d43908e75e67376e994c3483d6683f2a18673e5d2a1b0
```

6. On the **worker node** add a local DNS alias for the cp server. Edit the `/etc/hosts` file and add the cp IP address and assign the name `k8scp`.

```
root@worker:~# vim /etc/hosts
```

```
10.128.0.3 k8scp      #<-- Add this line
127.0.0.1 localhost
....
```

7. Use the token and hash, in this case as `sha256:long-hash` to join the cluster from the **second/worker** node. Use the **private** IP address of the cp server and port 6443. The output of the **kubeadm init** on the cp also has an example to use, should it still be available.

```
root@worker:~# kubeadm join \
    --token 27eee4.6e66ff60318da929 \
    k8scp:6443 \
    --discovery-token-ca-cert-hash \
    sha256:6d541678b05652e1fa5d43908e75e67376e994c3483d6683f2a18673e5d2a1b0
```

```
1 [preflight] Running pre-flight checks
2   [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended \
3   driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
4 [preflight] Reading configuration from the cluster...
5 [preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
6 [kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.15" ConfigMap in the \
7   kube-system namespace
8 [kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
9 [kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
10 [kubelet-start] Activating the kubelet service
11 [kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...
12
13 This node has joined the cluster:
14 * Certificate signing request was sent to apiservert and a response was received.
```

```
15 * The Kubelet was informed of the new secure connection details.  
16  
17 Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

8. Try to run the **kubectl** command on the secondary system. It should fail. You do not have the cluster or authentication keys in your local `.kube/config` file.

```
root@worker:~# exit
```

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

```
1 The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

```
student@worker:~$ ls -l .kube
```

```
1 ls: cannot access '.kube': No such file or directory
```