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 master 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.

1. 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.

   ```
   student@worker:~$ sudo -i

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

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

   (a) **IF** you chose Docker on the master:

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

   (b) **IF** you chose cri-o on the master:

   ```
   See several previous ten steps for cri-o installation details.
   ```

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

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

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

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

   root@worker:~# apt-get update

   root@worker:~# apt-get install -y \
         kubeadm=1.19.1-00 kubelet=1.19.1-00 kubectl=1.19.1-00

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

2. Find the IP address of your **master** 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 master node IP is `10.128.0.3`.

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

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

3. At this point we could copy and paste the **join** command from the master 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 master 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@master:~$ sudo kubeadm token list
```

```
TOKEN                     TTL   EXPIRES                USAGES...
27eee4.6e66ff60318da929   23h   2017-11-03T13:27:33Z   authe....
```

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.

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

```
27eee4.6e66ff60318da929
```

5. Create and use a Discovery Token CA Cert Hash created from the master to ensure the node joins the cluster in a secure manner. Run this on the master node or wherever you have a copy of the CA file. You will get a long string as output.

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

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

6. On the **worker node** add a local DNS alias for the master server. Edit the `/etc/hosts` file and add the master IP address and assign the name `k8smaster`.

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

10.128.0.3 k8smaster    #<-- 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 master server and port 6443. The output of the **kubeadm init** on the master also has an example to use, should it still be available.

```
root@worker:~# kubeadm join \
     --token 27eee4.6e66ff60318da929 \
     k8smaster:6443 \
     --discovery-token-ca-cert-hash \
     sha256:6d541678b05652e1fa5d43908e75e67376e994c3483d6683f2a18673e5d2a1b0
```

```
[preflight] Running pre-flight checks
       [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended \
          driver is "systemd". Please follow the guide at https://kubernetes.io/docs/setup/cri/
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.15" ConfigMap in the \
                kube-system namespace
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Activating the kubelet service
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

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
```

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

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

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