Create a Cluster Using `kubectl`

Introduction

In this lab, you will be provided two servers. They have been partially configured for use with Kubernetes and containerd. You will need to initialize a Kubernetes cluster, add a network plugin, and add a Node to the cluster.

Solution

Log in to the servers, `kcontrol` and `knode1`, using the credentials provided:

```
ssh cloud_user@<PUBLIC_IP_ADDRESS>
```

> **Note:** *When copying and pasting code into Vim from the lab guide, first enter `:set paste` (and then `i` to enter insert mode) to avoid adding unnecessary spaces and hashes. To save and quit the file, press **Escape** followed by `:wq`. To exit the file **without** saving, press **Escape** followed by `:q!`.*

Log In to the Provided Servers

1. List the contents of the `kcontrol` server to see if the `SERVER_READY` file is available:

   ```
   ls
   ```

2. List the contents of the `knode1` server to see if the `SERVER_READY` file is available:

   ```
   ls
   ```

3. Once the `SERVER_READY` file is present on both servers, continue on to the next objective.

   > **Note:** It can take some time for the `SERVER_READY` file to appear. You made need to wait a few moments and list the files a few times before the file displays and you can move on.

Initialize the Cluster and Set Up Your User

1. In the `kcontrol` server, confirm `kubeadm` is configured properly by pulling images:

   ```
   sudo kubeadm config images pull
   ```

2. When prompted, enter the `cloud_user` password.

3. Initialize the cluster (this may take a few minutes):

```
sudo kubeadm init --pod-network-cidr=172.16.0.0/16   --control-plane-endpoint=kcontrol
```

4. Copy all the cluster information displayed, starting with the line `Your Kubernetes control-plane has initialized successfully!`.
5. Create a file named `cluster-info.txt`:

```
vim cluster-info.txt
```

6. Press **i** to enter insert mode, then paste the information you just copied.
7. Press **Escape** followed by `:wq` to save and quit the file.
8. Create a home folder:

```
mkdir -p $HOME/.kube
```

9. Copy a file from Kubernetes to the `$HOME` folder:

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

10. Change ownership of the folder to the `cloud_user`:

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

11. Confirm the control planes are running:

```
kubectl cluster-info
```

Add a Network Plugin for Use

1. Apply the Calico network plugin:

```
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml
```

2. Verify that plugin has been added successfully:

```
kubectl get pods --all-namespaces
```

Add the Worker Node to the Cluster

1. List the names of the files:

```
ll
```

2. Move into the `cluster-info` file:

```
cat cluster-info.txt
```

3. Copy `kubeadm join kcontrol:6443 --token <STRING>`. Make sure you don't copy the trailing `\`.

4. Move to the `knode1` server.

5. Type `sudo`, paste in the `kubeadm join kcontrol:6443 --token <STRING>` command, and add a space after it.

6. Then, navigate back to the `kcontrol` server and copy rest of the command (i.e., `--discovery-token-ca-cert-hash <STRING>`). Paste this in as well; the complete command should look like this:

```
sudo kubeadm join kcontrol:6443 --token <STRING> --discovery-token-ca-cert-hash <STRING>
```

7. When prompted, enter the `cloud_user` password.

Test the Worker Node Has Been Added

1. Once the join is complete, back on the `kcontrol` server, verify that the node has joined the cluster:

```
kubectl get nodes
```

You should see that the `knode1` server has been added to the cluster.

> Note: If its status is `NotReady`, wait a few moments, run the command again, and confirm that it's status has updated to `Ready`.

Conclusion

Congratulations — you've completed this hands-on lab!