Kubernetes Components

When you deploy Kubernetes, you get a cluster.

A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerised applications. Every cluster has at least one worker node.

The work node(s) host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.

https://kubernetes.io/docs/concepts/overview/components/

Nodes

Kubernetes runs your workload by placing containers into Pods to run on Nodes. A node may be a virtual or physical machine. Depending on the cluster. Each node is managed by the control plane and contains the service necessary to run Pods.

Building a Kubernetes 1.23 Cluster with Kubeadm

- To setup a CloudServer Control Plane Node and two Cloud Server Worker Nodes

1. Install all packages to server and two nodes
2. Log in to the control plane server firstly to install all packages and configuration
   1. The first thing is to enable some kernel modules
   2. Creating containerd.conf in modules-load.d

```
$ cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
> overlay
> br_netfilter
>EOF
```

Enable overlay and br_netfilter modules. The file of modules-load.d is going to make sure those modules are enabled whenever the server starts up

   3. Manually enable those modules right now without restarting the control plane server

```
$ sudo modprobe overlay
$ sudo modprobe br_netfilter
```

   4. The modules are enabled, now it is to set some system-level settings, so that my container networking will function as expected by creating a sysctl.d file called kubernetes-cri.conf

```
$ cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
> net.bridge.bridge-nf-call-iptables = 1
> net.ipv4.ip_forward = 1
> net.bridge.bridge-nf-call-ip6tables = 1
>EOF
```

```
$ sudo sysctl  --system

To load those settings immediately without restart the control plane server
```

5. Now, to install my containerd package

```
$ sudo apt-get update && sudo apt-get install -y containerd
```

6. Containerd is install, it is time to create a default configuration file for contianerd and save the configuration to the file of /etc/containerd/config.toml

```
$ sudo mkdir -p /etc/containerd
$ sudo containerd config default | sudo tee /etc/containerd/config.toml
```

7. To restart the containerd to make sure that this new configuration is used

```
$ sudo systemctl restart containerd
```

8. To check if containerd is up and running

```
$ sudo systemctl status containerd
```

9. Then it is time to install and configure my Kubernetes packages. In order to get Kubernetes work, I do need to disable swap

```
$ sudo swapoff -a
```

10. Use sed command to add /etc/fstab, and that is going to remove any entries that enable swap in that stab file if they exist

```
$ sudo sed -i ‘/ swap / s/^\(.*\)$/#\1/g’ /etc/fstab
```

11. Next, it is going to install some required packages

```
$ sudo apt-get update & sudo apt-get install -y apt-transport-https curl
```

12. Download the GPG key and add the GPG key for Kubernetes repository

```
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
```

13. And set up file to /etc/apt/sources.list.d/kubernetes.list

```
$ cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
> deb https://apt.kubernetes.io/ kubernetes-xenial main
> EOF
```

14. Update my package listings with the data from that new Kubernetes repository

```
$ sudo apt-get update
```

15. Install my Kubernetes packages, and it is IMPORTANT to make sure that all those are using the same version

```
$ sudo apt-get install -y kubelet=1.23.0-00 kubeadm=1.23.0-00 kubectl=1.23.0-00
```

16. And the last to hold all those three packages, that is to make sure that they are not automatically updated

```
$ sudo apt-mark hold kubelet kubeadm kubectl
```

3. Now I have installed and configured everything on the control plane node, the next is to repeat all of those steps on other two worker nodes.

- Initialise the Cluster (only on control plane node)

To Transition to a Kubernetes-based infrastructure for running their containerised applications, you will need a basic Kubernetes cluster to get started.

1. Initialise the Kubernetes control plane node

```
$ sudo kubeadm init --pod-network-cidr 192.168.0.0/16 --kubernetes-version 1.23.0

--pod-network-cidr 192.168.0.0/16 : a cidr IP address range, that is going to be used to
configure Kubernetes netowrking
--kubernetes-version 1.23.0: Specify Kubernetes version
```

2. To interact with the cluster, I need to set up my /kube/config

```
$ mkdir -p $HOME/.kube
$ sudo cp -I /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

3. Now I should be able to connect with the cluster using kubectl command

```
$ kubectl get nodes
```

4. If status of the node is NotReady, that is because I have not set up our networking yet. However, the cluster initialisation is well done.

- Install the Calico Network Add-On

1. To use a yaml file to install Calico

```
$ kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

Using kubectl apply to apply this yaml file to the cluster that I can use to set up the Calico
network add-on
```

2. Calico will need a while to get started, when it is ready, the status of the node should be Ready

```
$ kubectl get nodes
```

- Join the worker nodes to the cluster

1. Right now, the cluster only contains the control plane node, so we will join our worker nodes. Run the following in the control plane node to get the other command that I need in order to join my worker nodes

```
$ kubeadm token create --print-join-command
```

2. The above command will give me a join command

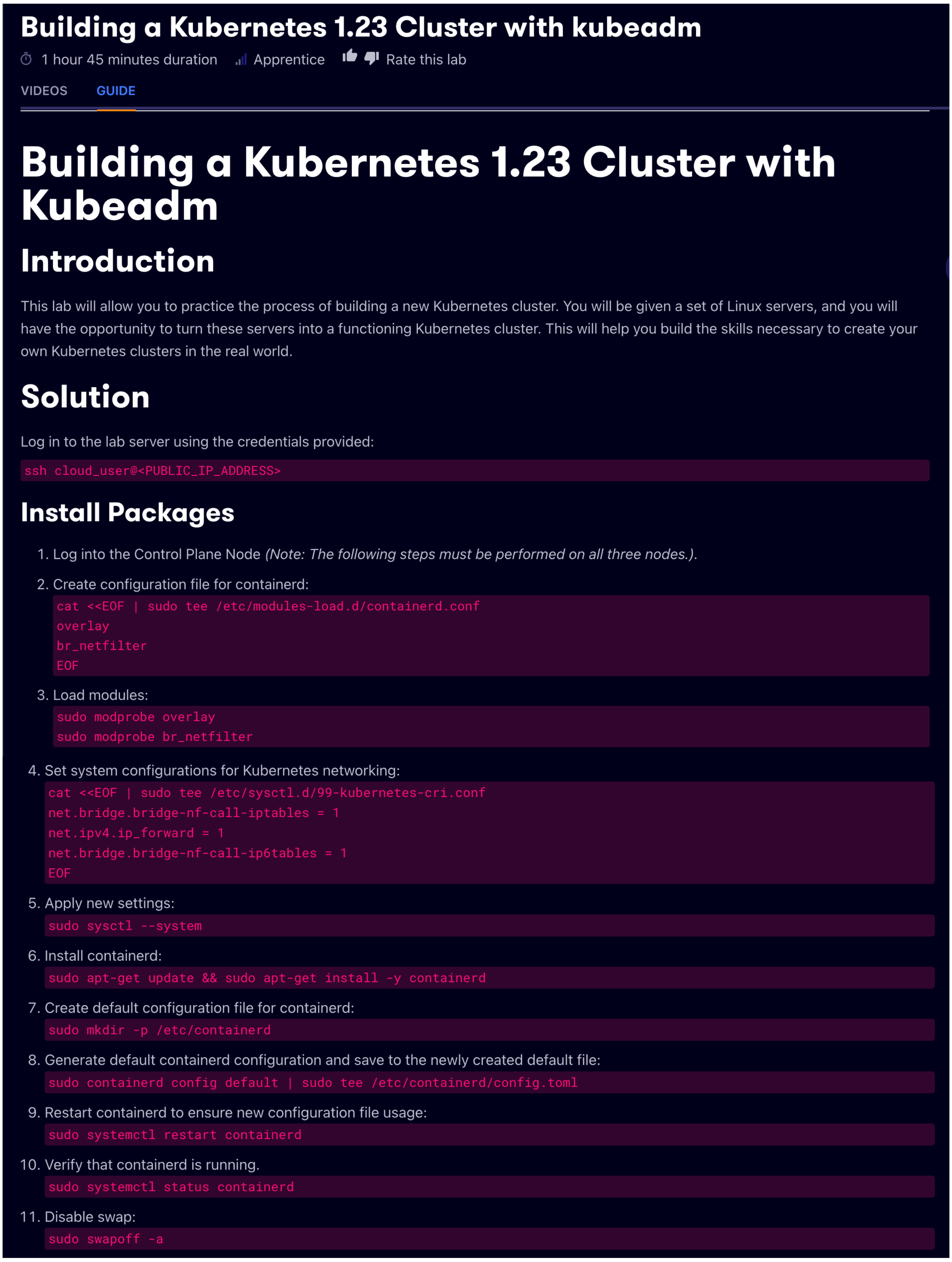
```
$ kubeadm join 10.0.1.101:6443 --token 311fcb.ozZa5ve6hy1zhmz6 --discovery-token-ac-cert-
hash sha256:50a59d87f120d7801b17ed69b0bebb5a0710ecb8ea8a2a15738cb4c331d063e5
```

3. Switch to the worker nodes, copy that entire command and paste and run in worker nodes

```
$ sudo kubeadm join 10.0.1.101:6443 --token 311fcb.ozZa5ve6hy1zhmz6 --discovery-token-ac-
cert-hash
sha256:50a59d87f120d7801b17ed69b0bebb5a0710ecb8ea8a2a15738cb4c331d063e5
```

4. Switch back to the control plane node and to check. It will take a while to get all nodes up and running

```
$ kubectl get nodes
```



Building a Kubernetes 1.23 Cluster with kubeadm

1 hour 45 minutes duration   Apprentice   Rate this lab

VIDEOS   GUIDE

Building a Kubernetes 1.23 Cluster with Kubeadm

Introduction

This lab will allow you to practice the process of building a new Kubernetes cluster. You will be given a set of Linux servers, and you will have the opportunity to turn these servers into a functioning Kubernetes cluster. This will help you build the skills necessary to create your own Kubernetes clusters in the real world.

Solution

Log in to the lab server using the credentials provided:

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

Install Packages

1. Log into the Control Plane Node *(Note: The following steps must be performed on all three nodes.)*.
2. Create configuration file for containerd:
   ```
   cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
   overlay
   br_netfilter
   EOF
   ```
3. Load modules:
   ```
   sudo modprobe overlay
   sudo modprobe br_netfilter
   ```
4. Set system configurations for Kubernetes networking:
   ```
   cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
   net.bridge.bridge-nf-call-iptables = 1
   net.ipv4.ip_forward = 1
   net.bridge.bridge-nf-call-ip6tables = 1
   EOF
   ```
5. Apply new settings:
   ```
   sudo sysctl --system
   ```
6. Install containerd:
   ```
   sudo apt-get update && sudo apt-get install -y containerd
   ```
7. Create default configuration file for containerd:
   ```
   sudo mkdir -p /etc/containerd
   ```
8. Generate default containerd configuration and save to the newly created default file:
   ```
   sudo containerd config default | sudo tee /etc/containerd/config.toml
   ```
9. Restart containerd to ensure new configuration file usage:
   ```
   sudo systemctl restart containerd
   ```
10. Verify that containerd is running.
    ```
    sudo systemctl status containerd
    ```
11. Disable swap:
    ```
    sudo swapoff -a
    ```

12. Disable swap on startup in `/etc/fstab`:
    ```
    sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
    ```
13. Install dependency packages:
    ```
    sudo apt-get update && sudo apt-get install -y apt-transport-https curl
    ```
14. Download and add GPG key:
    ```
    curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
    ```
15. Add Kubernetes to repository list:
    ```
    cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
    deb https://apt.kubernetes.io/ kubernetes-xenial main
    EOF
    ```
16. Update package listings:
    ```
    sudo apt-get update
    ```
17. Install Kubernetes packages (Note: If you get a dpkg lock message, just wait a minute or two before trying the command again):
    ```
    sudo apt-get install -y kubelet=1.23.0-00 kubeadm=1.23.0-00 kubectl=1.23.0-00
    ```
18. Turn off automatic updates:
    ```
    sudo apt-mark hold kubelet kubeadm kubectl
    ```
19. Log into both Worker Nodes to perform previous steps.

Initialize the Cluster

1. Initialize the Kubernetes cluster on the control plane node using kubeadm *(Note: This is only performed on the Control Plane Node)*:
   ```
   sudo kubeadm init --pod-network-cidr 192.168.0.0/16 --kubernetes-version 1.23.0
   ```
2. Set kubectl access:
   ```
   mkdir -p $HOME/.kube
   sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
   sudo chown $(id -u):$(id -g) $HOME/.kube/config
   ```
3. Test access to cluster:
   ```
   kubectl get nodes
   ```

Install the Calico Network Add-On

1. On the Control Plane Node, install Calico Networking:
   ```
   kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
   ```
2. Check status of the control plane node:
   ```
   kubectl get nodes
   ```

Join the Worker Nodes to the Cluster

1. In the Control Plane Node, create the token and copy the kubeadm join command *(NOTE:The join command can also be found in the output from `kubeadm init` command)*:
   ```
   kubeadm token create --print-join-command
   ```
2. In both Worker Nodes, paste the kubeadm join command to join the cluster. Use sudo to run it as root:
   ```
   sudo kubeadm join ...
   ```
3. In the Control Plane Node, view cluster status (Note: You may have to wait a few moments to allow all nodes to become ready):
   ```
   kubectl get nodes
   ```