Deploy Kubernetes Cluster on Ubuntu 20.04 with Containerd:

1- Install containerd

- To install containerd, follow these steps on both Vms:
- Load the br_netfilter module required for networking.

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

To allow iptables to see bridged traffic, as required by Kubernetes, we need to set the values of certain fields to 1.

```
sudo tee /etc/sysctl.d/kubernetes.conf<<EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

- Apply the new settings without restarting:

```
sudo sysctl -system
```

- Get the apt-key and then add the repository from which we will install containerd.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release - cs) stable"
```

Or:

```
apt install containerd -y
```

- Set up the default configuration file.

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

- Next up, we need to modify the containerd configuration file and ensure that the cgroupDriver is set to systemd. To do so, edit the following file:

```
sudo vim /etc/containerd/config.toml
```

and search *plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options* , then change *SystemdCgroup = false* to true.

```
SystemdCgroup = true
```

- Finally, to apply these changes, we need to restart containerd.

```
sudo systemctl restart containerd
```

2- With our container runtime installed and configured, we are ready to install Kubernetes. Add the repository key and the repository.

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

echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

2-1. Update your system and install the 3 Kubernetes modules.

```
sudo apt update -y
sudo apt install -y kubelet kubeadm kubectl
```

2-2. Set the appropriate hostname for each machine.

```
sudo hostnamectl set-hostname "master-node"
exec bash
```

2-3. To allow kubelet to work properly, we need to disable swap on both machines.

```
sudo swapoff –a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

or Open the **/etc/fstab** file with a text editor. You can use nano, vim, or any other text editor you are comfortable with. For example:

```
sudo vim /etc/fstab
```

Look for the line that references the swap file. It will usually look something like this:

```
/swapfile none swap sw 0 0
```

Then Reboot the system.

Finally, enable the kubelet service on both systems so we can start it.

```
 sudo systemctl enable kubelet
```

3. Run the following command on the master node to allow Kubernetes to fetch the required images before cluster initialization:

```
sudo kubeadm config images pull
```

3-1. Initialize the cluster

```
kubeadm init --control-plane-endpoint "192.168.1.40:6443" --pod-network-cidr=10.244.0.0/16  --upl
oad-certs
```

To start using your cluster, you need to run the following as a regular user:

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

- Alternatively, if you are the root user, you can run:

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

- **Install Flannel**

```
# kubectl apply -f https://github.com/coreos/flannel/raw/master/Documentation/kube-flannel.yml
```