Installing Kubernets

How to Install Kubernetes Cluster on Ubuntu 22.04

Platform: Azure Portal

VNET: Create an Azure VNET in any Region with Address Space = 192.168.1.0/16

Subnet: Create a subnet with Address Space = 192.168.1.0/24

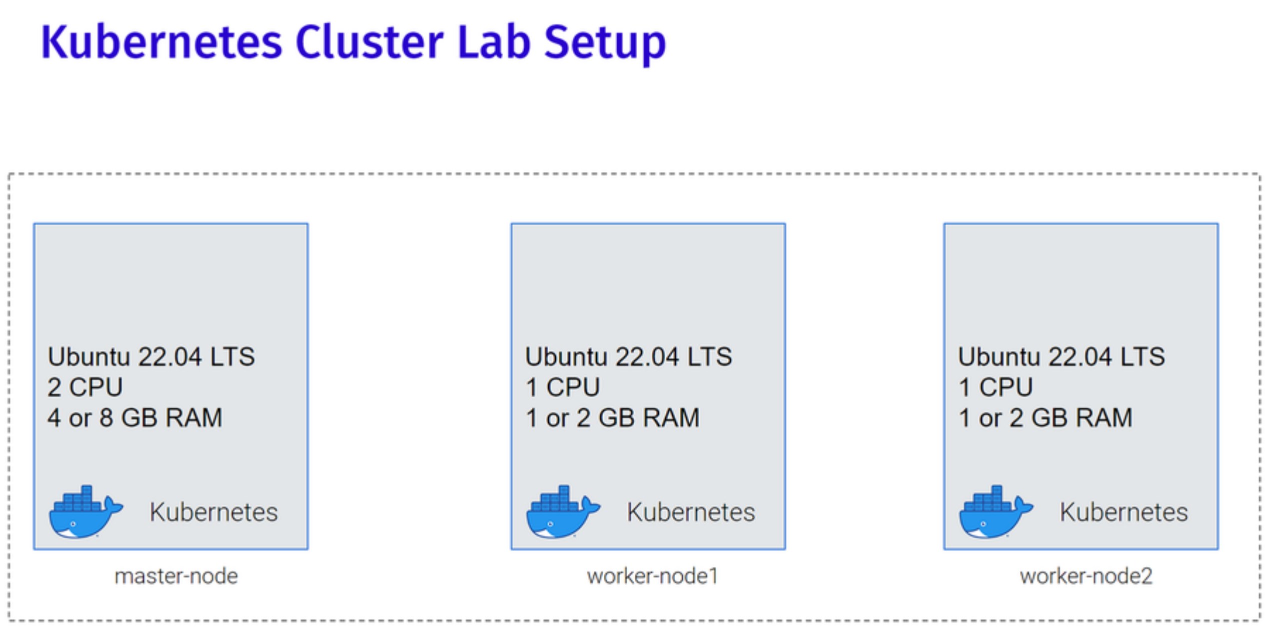
Following are system requirements on Master node

- Ubuntu 22.04
- Minimum 2GB RAM
- Minimum 2 CPU cores
- Host name = master-node

Following are system requirements on Worker nodes

- Ubuntu 22.04
- Minimum 1 GB RAM
- Minimum 1 CPU cores
- Host names = worker-node1, worker-node2

Lab Setup Diagram



Steps should be performed on both Master and Worker Nodes

Except as indicated where an additional command is required to be run on only master-node

Step 1) Set hostname and add entries in the hosts file

Login to **both master node and worker nodes**, add Name,IP to /etc/hosts files

Add the following entries in /etc/hosts file on each node

```
192.168.1.4 master-node
192.168.1.5 worker-node1
192.168.1.6 worker-node2
```

Step 2) Disable swap & add kernel settings

Disable swap **on all the nodes**

(In Azure by default swap memory is disabled, so need to worry, but if you are practicing on your Laptop, this is a must)

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

Load the following kernel modules **on all the nodes**

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

Set the following Kernel parameters for Kubernetes **on all the nodes**

```
$ 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

```

Reload the above changes:

```
$ sudo sysctl --system
```

Step 3) Install containerd run time

Perform this **on all nodes**

```
$ sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates
```

Enable docker repository **on all nodes**:

```
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/dc
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Now, run the following apt command on all nodes:

```
$ sudo apt update
$ sudo apt install -y containerd.io
```

Configure containerd **on all the nodes** so that it starts using systemd as cgroup

```
$ containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
$ sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml
```

Restart and enable containerd service **on all nodes**

```
$ sudo systemctl restart containerd
$ sudo systemctl enable containerd
```

Step 4) Add apt repository for Kubernetes

Execute following commands to add apt repository for Kubernetes

```
$ curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmour -o /etc/apt/trusted.gpg.d/k
$ sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

Step 5) Install Kubernetes components Kubectl, kubeadm & kubelet

Install Kubernetes components like kubectl, kubelet and Kubeadm utility on all the nodes. Run following set of commands,

```
$ sudo apt update
$ sudo apt install -y kubelet kubeadm kubectl
$ sudo apt-mark hold kubelet kubeadm kubectl
```

Step 6) Initialize Kubernetes cluster with Kubeadm command

Now, we are all set to initialize Kubernetes cluster. Run the following Kubeadm command from the master node only.

```
$ sudo kubeadm init --control-plane-endpoint=master
```

Join both the worker nodes to the cluster, command is already there is output, just copy paste on the worker nodes,

```
$ sudo kubeadm join master:6443 --token vt4ua6.wcma2y8pl4menxh2 \
   --discovery-token-ca-cert-hash sha256:0494aa7fc6ced8f8e7b20137ec0c5d2699dc5f8e616656932ff9173c94962a36
```

Check the nodes status from master node using kubectl command,

```
$ kubectl get nodes
```

As we can see nodes status is ‘NotReady’, so to make it active. We must install CNI (Container Network Interface) or network add-on plugins like Calico, Flannel and Weave-net.

Step 6) Install Calico Pod Network Add-on

Run following kubectl command to install Calico network plugin from the master node,

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

Verify the status of pods in kube-system namespace,

```
$ kubectl get pods -n kube-system
```

Perfect, check the nodes status as well.

```
$ kubectl get nodes
```

Great, above confirms that nodes are active node. Now, we can say that our Kubernetes cluster is functional.