Multi-node Cluster Creation via Vagrant

What are we doing?

- Creating 3 node cluster with 1 master (namely **kubemaster**) and 2 worker (**kubenode01** & **kubenode02**)
- Setting up Docker, Weavenet (for pod network), Kubernetes

Pre-requisite Installation:

- Oracle Virtual Box
- Git Bash in Windows
- Vagrant 64 bit in Windows

Cloning repo to get vagrantfile to create cluster:

It is considered that you have Github account and access to clone repo

Go to your local Windows Path where you want to clone the repo (<<<<<click on repo hyperlink) and copy the clone link

Once you traversed to the local path where you want to clone the repo, *Right Click* > Click on *Git Bash Here*

Within Git Bash execute:

```
git clone <copied clone link>

Example:
git clone https://github.com/kodekloudhub/certified-kubernetes-administrator-course.git
```

You should be able to view the folder in your local, go inside the folder

```
cd <folder name>

Example:
cd certified-kubernetes-administrator-course
```

You can further continue in Git Bash or use Command Prompt / any other desired terminal to access vagrant:

```
pwd

/drives/c/Users/tanumgho/Oracle Content - Accounts/Oracle Content/Tanumoy/MSP/GITLAB_REPO/certified-kubernetes-administrator-course
```

ls

README.md docs ubuntu vagrant-ssh
Vagrantfile images ubuntu-bionic-18.04-cloudimg-console.log

Cluster Creation:

vagrant status

Current machine states:

kubemaster not created (virtualbox)
kubenode01 not created (virtualbox)
kubenode02 not created (virtualbox)

To bring up the cluster execute the below steps (It would take time based on your internet connectivity to download the images and set up the vagrant boxes):

vagrant up

Bringing machine 'kubemaster' up with 'virtualbox' provider...
Bringing machine 'kubenode01' up with 'virtualbox' provider...
Bringing machine 'kubenode02' up with 'virtualbox' provider...
…

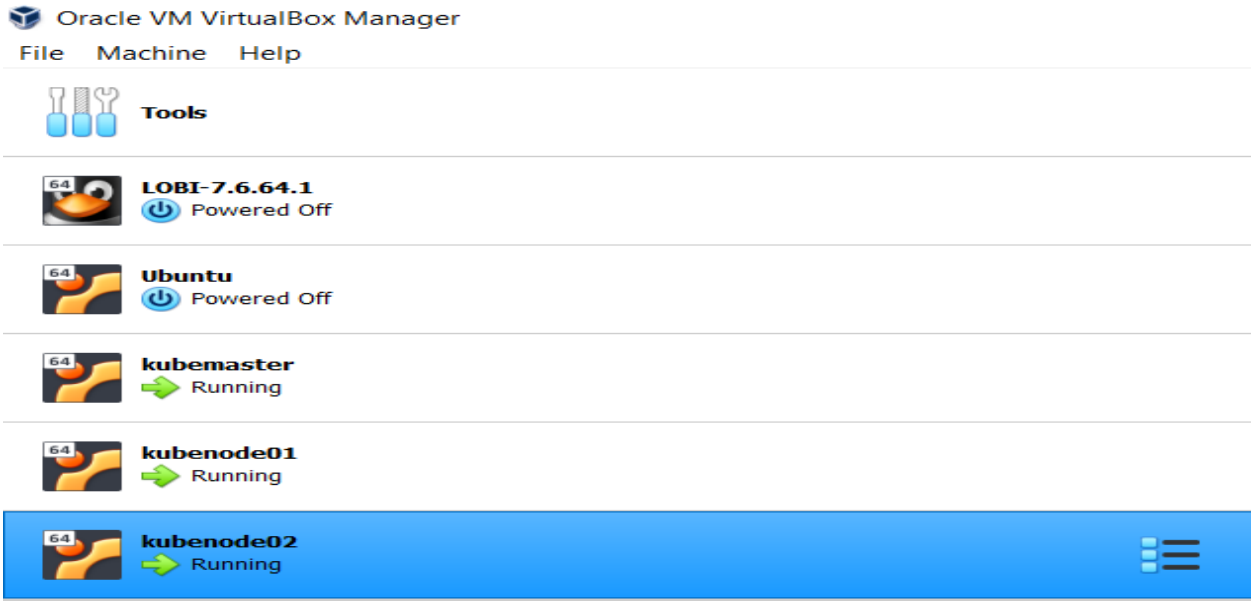
Check the status of the boxes again, it should be running now:

vagrant status

Current machine states:

kubemaster running (virtualbox)
kubenode01 running (virtualbox)
kubenode02 running (virtualbox)

You could check in Oracle VirtualBox where it would also show as Running:



To login to the boxes:

vagrant ssh <vagrant box name>

Example:
vagrant ssh kubemaster
vagrant ssh kubenode01
vagrant ssh kubenode02

To stop the boxes temporarily once your work is done so that you can resume with command “vagrant up”:

vagrant halt

To remove / shutdown the boxes (Make sure it will remove the downloaded images from the local as well, so when you try to create the cluster next time, it will actually download the images from the remote repo again which will take time):

vagrant destroy

Installation of Docker Container Runtime & Kubernetes into the cluster:

Below steps to be executed in all boxes:

- Check if br_netfilter module is loaded

```
lsmod | grep br_netfilter
```

- If it is not there, you can explicitly load it

```
sudo modprobe br_netfilter
```

- Now check again for the module

```
lsmod | grep br_netfilter

br_netfilter        24576  0
bridge             155648  1 br_netfilter
```

- IPTable settings

```
cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sudo sysctl --system
```

- Installing Docker Engine in Ubuntu

```
sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl gnupg lsb-release

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

- Check if docker is installed

```
docker -v

Example:
Docker version 20.10.6, build 370c289
```

- Installing Docker Daemon (Below /etc/docker folder might be already existing)

```
sudo mkdir /etc/docker

cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
```

- Restart Docker

```
sudo systemctl enable docker

sudo systemctl daemon-reload

sudo systemctl restart docker

service docker status
```

- Installing kubeadm, kubelet and kubectl

```
sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl

sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl
```

To execute only in kubemaster:

- Initializing master (Check master ip via ifconfig command)

```
sudo -i

kubeadm init --pod-network-cidr <cidr range> --apiserver-advertise-address=<masterip>


Example:
vagrant@kubemaster:~$ sudo -i

root@kubemaster:~# kubeadm init --pod-network-cidr 10.244.0.0/16 --apiserver-advertise-address=192.168.56.2
```

Extract of the output after executing the above command which would be required:

```
…
Your Kubernetes control-plane has initialized successfully!

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

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.56.2:6443 --token <redacted> --discovery-token-ca-cert-hash <redacted>
```

Above green highlighted commands to run in master and yellow highlighted to run in workers.

To execute only in kubemaster:

- Enabling kubernetes from homepath of normal user in kubemaster

```
vagrant@kubemaster:~$ whoami; cd
vagrant

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

- Using pod network solution (Installing weave)

```
vagrant@kubemaster:~$ sudo kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
```

To execute in worker nodes (kubenode01 & kubenode02):

- Making the worker node join the cluster

```
vagrant@kubenode01:~$ sudo kubeadm join 192.168.56.2:6443 –token <redacted> --discovery-token-ca-cert-hash <redacted>
…
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.
…


vagrant@kubenode02:~$ sudo kubeadm join 192.168.56.2:6443 --token <redacted> --discovery-token-ca-cert-hash <redacted>
…
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.
…
```

To execute in kubemaster:

- Checking for cluster status from kubemaster

```
vagrant@kubemaster:~$ kubectl get no
NAME         STATUS ROLES                  AGE     VERSION
kubemaster   Ready  control-plane,master   62m     v1.21.1
kubenode01   Ready  <none>                 18m     v1.21.1
kubenode02   Ready  <none>                 2m31s   v1.21.1
```

- Installing metrics-server in cluster

```
vagrant@kubemaster:~$ git clone https://github.com/kodekloudhub/kubernetes-metrics-server.git

vagrant@kubemaster:~$ kubectl apply -f kubernetes-metrics-server/

vagrant@kubemaster:~$ kubectl -n kube-system get po -l k8s-app=metrics-server
NAME                      READY   STATUS    RESTARTS   AGE
metrics-server-774b56d589-kfs7c   1/1     Running   0          5m44s
```

To execute from your local CMD:

- Once work is done, stop the vagrant boxes after exiting from the vagrant box and executing from your local CMD:

```
vagrant halt
```

- When resume, execute below command, your work would be restored:

```
vagrant up
```

- If you don't need the cluster anymore, you can terminate it, make sure all the progress would be lost

```
vagrant destroy
```