Kubernets installaton Steps :

1. Install **Docker** in all VMs:

a) Install docker on all 3 VMs

$ curl -fsSL https://get.docker.com -o get-docker.sh

$ sh get-docker.sh

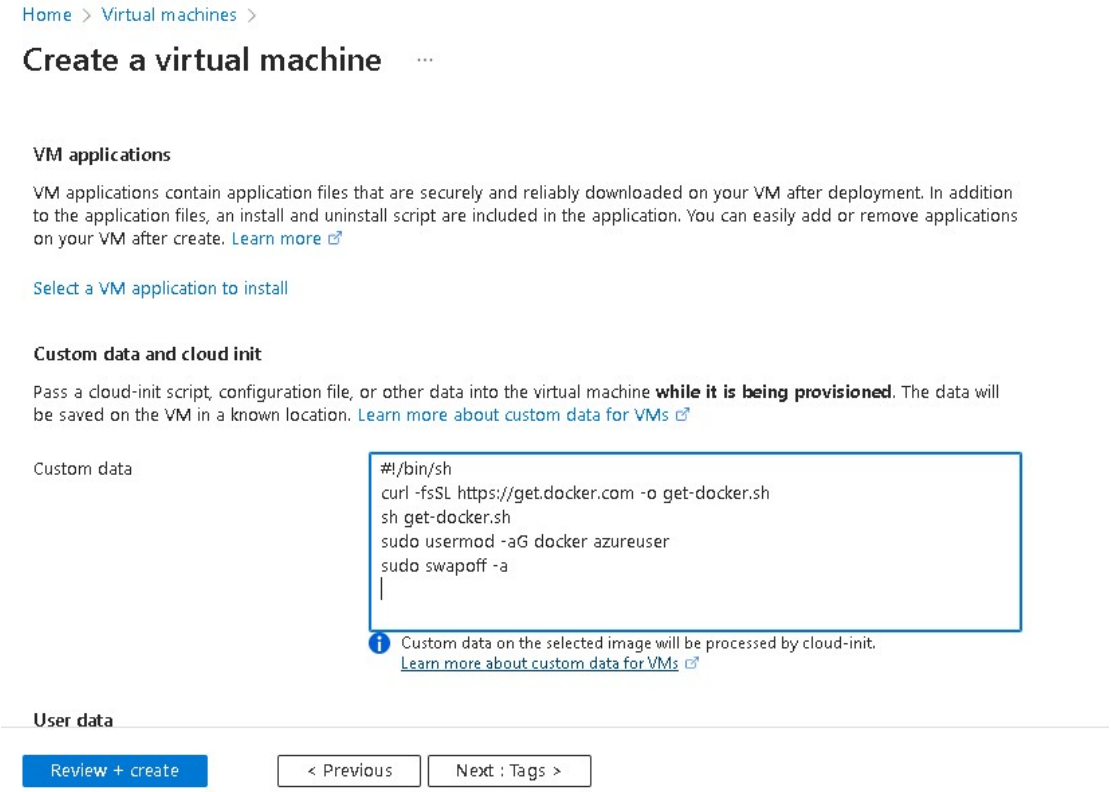
b) Add users to Docker group (default users for AWS - ubuntu/Azure - azureuser)

$ sudo usermod -aG docker ubuntu

c) Turn off swap

$ sudo swapoff -a

c) Exit and re-login



2. Install Go lang in all 3 VMs as root user

a) Run these commands to switch to root user (in all 3 VMs)

root $ sudo -i

b) once we are root user, run below commands:

root $ wget https://storage.googleapis.com/golang/getgo/installer_linux

root $ chmod +x ./installer_linux

root $ ./installer_linux

root $ source /root/.bash_profile

```
azureuser@K8sMaster1: $ sudo -i
root@K8sMaster1:~# wget https://storage.googleapis.com/golang/getgo/installer_linux
--2023-04-26 09:01:52--  https://storage.googleapis.com/golang/getgo/installer_linux
Resolving storage.googleapis.com (storage.googleapis.com)... 172.253.63.128, 142.250.31.128, 142.251.111.128, ...
Connecting to storage.googleapis.com (storage.googleapis.com)|172.253.63.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5179246 (4.9M) [application/octet-stream]
Saving to: 'installer_linux'

installer_linux                     100%[=================================================================================>]   4.94M  --.-KB/s    in 0.0

2023-04-26 09:01:52 (201 MB/s) - 'installer_linux' saved [5179246/5179246]

root@K8sMaster1:~# chmod +x ./installer_linux
root@K8sMaster1:~# ./installer_linux
Welcome to the Go installer!
Downloading Go version go1.20.3 to /root/.go
This may take a bit of time...
Downloaded!
Setting up GOPATH
GOPATH has been set up!

One more thing! Run `source /root/.bash_profile` to persist the
new environment variables to your current session, or open a
new shell prompt.
root@K8sMaster1:~# source /root/.bash_profile
root@K8sMaster1:~#
```

3. Installing cri-dockerd (all nodes as root)

root $ git clone https://github.com/Mirantis/cri-dockerd.git

root $ cd cri-dockerd

root $ mkdir bin

root $ go build -o bin/cri-dockerd

```
root@K8sMaster1:~# git clone https://github.com/Mirantis/cri-dockerd.git
Cloning into 'cri-dockerd'...
remote: Enumerating objects: 16619, done.
remote: Counting objects: 100% (16619/16619), done.
remote: Compressing objects: 100% (6927/6927), done.
remote: Total 16619 (delta 8230), reused 16484 (delta 8195), pack-reused 0
Receiving objects: 100% (16619/16619), 36.10 MiB | 28.09 MiB/s, done.
Resolving deltas: 100% (8230/8230), done.
root@K8sMaster1:~# cd cri-dockerd
root@K8sMaster1:~/cri-dockerd# mkdir bin
root@K8sMaster1:~/cri-dockerd# go build -o bin/cri-dockerd
root@K8sMaster1:~/cri-dockerd#
```

root $ mkdir -p /usr/local/bin

root $ install -o root -g root -m 0755 bin/cri-dockerd /usr/local/bin/cri-dockerd

root $ cp -a packaging/systemd/* /etc/systemd/system

root $ sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service

root $ systemctl daemon-reload

root $ systemctl enable cri-docker.service

root $ systemctl enable --now cri-docker.socket

```
root@K8sMaster1:~/cri-dockerd# mkdir -p /usr/local/bin
root@K8sMaster1:~/cri-dockerd# install -o root -g root -m 0755 bin/cri-dockerd /usr/local/bin/cri-dockerd
root@K8sMaster1:~/cri-dockerd# cp -a packaging/systemd/* /etc/systemd/system
root@K8sMaster1:~/cri-dockerd# sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.serv
root@K8sMaster1:~/cri-dockerd# systemctl daemon-reload
root@K8sMaster1:~/cri-dockerd# systemctl enable cri-docker.service
Created symlink /etc/systemd/system/multi-user.target.wants/cri-docker.service → /etc/systemd/system/cri-docker.service.
root@K8sMaster1:~/cri-dockerd# systemctl enable --now cri-docker.socket
Created symlink /etc/systemd/system/sockets.target.wants/cri-docker.socket → /etc/systemd/system/cri-docker.socket.
root@K8sMaster1:~/cri-dockerd#
```

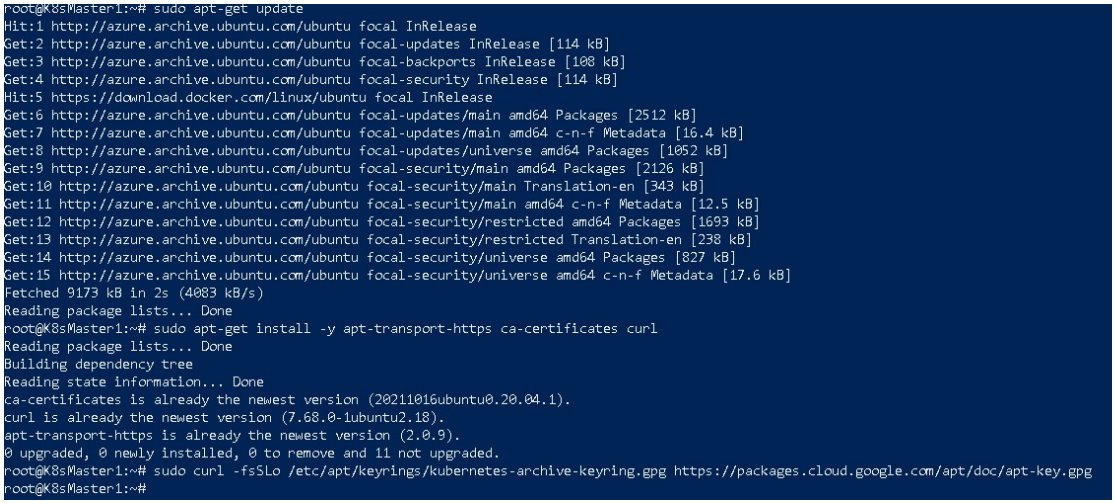
4. Installing kubeadm, kubelet and kubectl (all nodes as root)

root $ cd ~

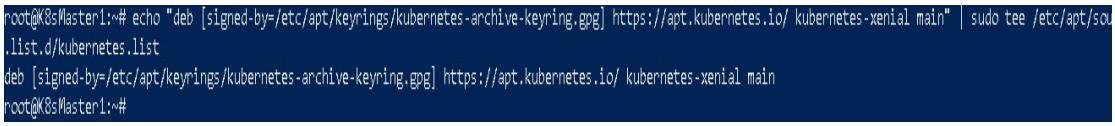
root $ sudo apt-get update

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

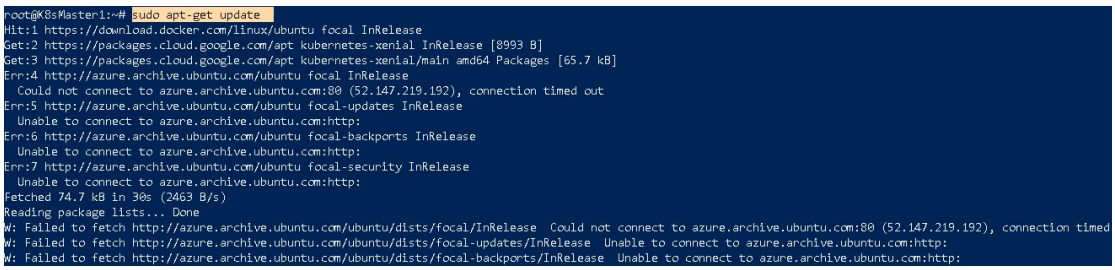
root $ sudo curl -fsSLo /etc/apt/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg



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



root $ sudo apt-get update



root $ sudo apt-get install -y kubelet kubeadm kubectl

```
root@K8sMaster1:~# sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  conntrack cri-tools ebtables kubernetes-cni socat
Suggested packages:
  nftables
The following NEW packages will be installed:
  conntrack cri-tools ebtables kubeadm kubectl kubelet kubernetes-cni socat
0 upgraded, 8 newly installed, 0 to remove and 11 not upgraded.
Need to get 85.9 MB of archives.
After this operation, 329 MB of additional disk space will be used.
Get:1 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 cri-tools amd64 1.26.0-00 [18.9 MB]
Get:2 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubernetes-cni amd64 1.2.0-00 [27.6 MB]
Get:3 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubelet amd64 1.27.1-00 [18.7 MB]
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubectl amd64 1.27.1-00 [10.2 MB]
Get:5 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 kubeadm amd64 1.27.1-00 [9928 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu focal/main amd64 conntrack amd64 1:1.4.5-2 [30.3 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu focal/main amd64 ebtables amd64 2.0.11-3build1 [80.3 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu focal/main amd64 socat amd64 1.7.3.3-2 [323 kB]
Fetched 85.9 MB in 7s (11.8 MB/s)
Selecting previously unselected package conntrack.
(Reading database ... 58955 files and directories currently installed.)
Preparing to unpack .../0-conntrack_1%3a1.4.5-2_amd64.deb ...
Unpacking conntrack (1:1.4.5-2) ...
Selecting previously unselected package cri-tools.
Preparing to unpack .../1-cri-tools_1.26.0-00_amd64.deb ...
```

root $ sudo apt-mark hold kubelet kubeadm kubectl

```
root@K8sMaster1:~# sudo apt-get install -y kubelet kubeadm kubectl
Reading package lists... Done
Building dependency tree
Reading state information... Done
kubeadm is already the newest version (1.27.1-00).
kubectl is already the newest version (1.27.1-00).
kubelet is already the newest version (1.27.1-00).
0 upgraded, 0 newly installed, 0 to remove and 11 not upgraded.
```

5. Run the next commands as root only in Master {master-node} node to setup cri-socket (as root)

root@Master-node:~# kubeadm init --pod-network-cidr "10.244.0.0/16" --cri-socket "unix:///var/run/cri-dockerd.sock"

```
root@K8sMaster1:~# kubeadm init --pod-network-cidr "10.244.0.0/16" --cri-socket "unix:///var/run/cri-dockerd.sock"
[init] Using Kubernetes version: v1.27.1
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
W0426 09:45:39.355990   22984 images.go:80] could not find officially supported version of etcd for Kubernetes v1.27.1, falling back to the nearest etcd version (3.
0)
W0426 09:45:50.535907   22984 checks.go:835] detected that the sandbox image "registry.k8s.io/pause:3.6" of the container runtime is inconsistent with that used by
adm. It is recommended that using "registry.k8s.io/pause:3.9" as the CRI sandbox image.
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [k8smaster1 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IP
0.96.0.1 10.0.0.4]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [k8smaster1 localhost] and IPs [10.0.0.4 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [k8smaster1 localhost] and IPs [10.0.0.4 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Starting the kubelet
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
```

6. Copy "kubeadmin join <>" command along with the specific TOKEN from the output of above command for joining the other Worker Nodes to this Master Node.

we can find it at the end of the STDOUT of above command.

```
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 10.0.0.4:6443 --token 89jgdh.kfffu1djk7mcvehd \
        --discovery-token-ca-cert-hash sha256:d8c52eb5f7324edc99a3e755d30b7b1a73289334295d4671d6dcb794fc613c83
```

7. After executing the above command you will get the following steps mkdir,sudo sp,sudo chown run them as a regular user

--> Your Kubernetes control-plane has initialized successfully!

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

root@Master-node:~# exit

master@Master-node:~$ mkdir -p $HOME/.kube

master@Master-node:~$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

master@Master-node:~$ sudo chown $(id -u):$(id -g) $HOME/.kube/config

```
azureuser@K8sMaster1: $ pwd
/home/azureuser
azureuser@K8sMaster1: $ mkdir -p $HOME/.kube
azureuser@K8sMaster1: $ ls
azureuser@K8sMaster1: $ ls -l
total 0
azureuser@K8sMaster1: $ ls -al
total 36
drwxr-xr-x 5 azureuser azureuser 4096 Apr 26 09:54 .
drwxr-xr-x 3 root      root      4096 Apr 26 08:49 ..
-rw------- 1 azureuser azureuser   25 Apr 26 09:26 .bash_history
-rw-r--r-- 1 azureuser azureuser  220 Feb 25  2020 .bash_logout
-rw-r--r-- 1 azureuser azureuser 3771 Feb 25  2020 .bashrc
drwx------ 2 azureuser azureuser 4096 Apr 26 08:50 .cache
drwxrwxr-x 2 azureuser azureuser 4096 Apr 26 09:54 .kube
-rw-r--r-- 1 azureuser azureuser  807 Feb 25  2020 .profile
drwx------ 2 azureuser azureuser 4096 Apr 26 08:49 .ssh
-rw-r--r-- 1 azureuser azureuser    0 Apr 26 08:59 .sudo_as_admin_successful
azureuser@K8sMaster1: $ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
azureuser@K8sMaster1: $ sudo chown $(id -u):$(id -g) $HOME/.kube/config
azureuser@K8sMaster1: $
```

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

master@Master-node:~$ export KUBECONFIG=/etc/kubernetes/admin.conf

8. Install Flannel to start (only master node)

master@Master-node:~$ kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml

```
azureuser@K8sMaster1: $ kubectl apply -f https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml
namespace/kube-flannel created
serviceaccount/flannel created
clusterrole.rbac.authorization.k8s.io/flannel created
clusterrolebinding.rbac.authorization.k8s.io/flannel created
configmap/kube-flannel-cfg created
daemonset.apps/kube-flannel-ds created
azureuser@K8sMaster1: $
```

9. Check whether your node (master-node) is running or not

master@Master-node:~$ kubectl get nodes -w

(-w for watch, watches any changes)

```
azureuser@K8sMaster1:~$ kubectl get nodes -w
NAME          STATUS   ROLES           AGE   VERSION
k8smaster1    Ready    control-plane   16m   v1.27.1
```

KUBEADM is installed on the MASTER NODE Successfully !!

10. Go to Worker-node 01 and 02 and execute

worker@worker-node-01:~$ su - root

root@worker-node-01:~# export KUBECONFIG=/etc/kubernetes/admin.conf

root@worker-node-01:~# kubeadm join 10.0.0.4:6443 --token pozx3l.mh39vjz1rnc7rc1t --cri-socket "unix:///var/run/cri-dockerd.sock" --discovery-token-ca-cert-hash

sha256:04ddbd0439b039f8b189c9cb1334bf8a46a0856a5dcf9247b006900a994fddd6

```
root@Node1:~# kubeadm join 10.0.0.4:6443 --token 89jgdh.kfffu1djk7mcvehd \
> --cri-socket "unix:///var/run/cri-dockerd.sock" \
> --discovery-token-ca-cert-hash sha256:d8c52eb5f7324edc99a3e755d30b7b1a73289334295d4671d6dcb794fc613c83
[preflight] Running pre-flight checks
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

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.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.

root@Node1:~#
```

11. Go to your Master Node & run below command to check if the Node is visible in the list.

master@Master-node:~$ kubelet get pods -w

```
^Cazureuser@K8sMaster1:~$ kubectl get nodes -w
NAME          STATUS   ROLES           AGE   VERSION
k8smaster1    Ready    control-plane   51m   v1.27.1
node1         Ready    <none>          20s   v1.27.1
node1         Ready    <none>          31s   v1.27.1
```

Our NODE1 is in READY state !!!