You may continue to install Kubernetes on the vm you already installed docker. If you are installing this on a different machine make sure docker is already installed.

Part 1

Installing kubeadm, kubelet, and kubectl:

1. Update the apt package index:

```
sudo apt-get update
```

2. Install packages needed to use the Kubernetes apt repository:

```
sudo apt-get install -y apt-transport-https ca-certificates curl vim git
```

3. Download the public signing key for the Kubernetes package repositories:

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

4. Add the Kubernetes apt repository:

```
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

Update the apt package index again:

```
sudo apt-get update
```

5. Install kubelet, kubeadm, and kubectl:`

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

6. Pin installed versions of kubelet, kubeadm, and kubectl to prevent them from being accidentally updated:

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

7. Check installed versions:

```
kubectl version --client
```

```
kubeadm version
```

Disable Swap Space

8. Disable all swaps from /proc/swaps.

```
sudo swapoff -a
```

`sudo sed -i.bak -r 's/(.+ swap .+)/#\1/' /etc/fstab`

9. Check if swap has been disabled by running the free command.

`free -h`

Install Container runtime

10. Configure persistent loading of modules

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

11. Load at runtime

`sudo modprobe overlay`

`sudo modprobe br_netfilter`

12. Ensure sysctl params are set

```
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
```

13. Reload configs

`sudo sysctl --system`

14. Install required packages

`sudo apt install -y containerd.io`

15. Configure containerd and start service

`sudo mkdir -p /etc/containerd`

`sudo containerd config default | sudo tee /etc/containerd/config.toml`

16. Configuring a cgroup driver

Both the container runtime and the kubelet have a property called "cgroup driver", which is essential for the management of cgroups on Linux machines.

`sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml`

17. Restart containerd

`sudo systemctl restart containerd`

`sudo systemctl enable containerd`

`systemctl status containerd`

Initialize control plane

18. Make sure that the br_netfilter module is loaded:

`lsmod | grep br_netfilter`

Output should similar to:

```
br_netfilter           22256  0
bridge                151336  1 br_netfilter
```

19. Enable kubelet service.

`sudo systemctl enable kubelet`

20. Pull container images (it will take some time):

`sudo kubeadm config images pull --cri-socket /run/containerd/containerd.sock`

21. Bootstrap the endpoint. Here we use 10.244.0.0/16 as the pod network:

`sudo kubeadm init --pod-network-cidr=10.244.0.0/16 --cri-socket /run/containerd/containerd.sock`

You will see Your Kubernetes control-plane has initialized successfully!

You need to save the kubeadm join token string on a text document for future use.

22. To start the cluster, you need to run the following as a regular user (For this scenario we will only use a single master):

`mkdir -p $HOME/.kube`

`sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config`

`sudo chown $(id -u):$(id -g) $HOME/.kube/config`

23. Check cluster info:

`kubectl cluster-info`

24. Install a simple network plugin.

`wget https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml`

`kubectl apply -f kube-flannel.yml`

25. Check the plugin is working

`kubectl get pods -n kube-flannel`

26. Confirm master node is ready: (If you see the status as Notready, give it a around 10mins)

```
kubectl get nodes -o wide
```

27. On a master/ control node to query the nodes you can use:

```
kubectl get nodes
```

28. Connecting Worker nodes:
    a. Install docker on both worker nodes as per the guidelines here (Upto step 6).
    b. Install Kubernetes on each worker nodes. Follow above steps 1 to 20.
    c. Use previously saved kubeadm join token string to connect the workers. Run it with **sudo**.

       [OPTIONAL] If you forgot the token, run the following on your master to create a new token:

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

       But remember, this will create a new token.

    d. After all workers are connected, check status of the cluster: Here after all commands should be run on the master.

       ```
       kubectl get nodes -o wide
       ```