1. Prerequisites:
   2GB RAM, 2 CPUs,

2. Ensure unique MAC address:

```
devops@devops-VirtualBox:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:58:da:07 brd ff:ff:ff:ff:ff:ff
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:1f:03:46 brd ff:ff:ff:ff:ff:ff
```

3. Ensure unique Product uuid

```
devops@devops-VirtualBox:~$ sudo cat /sys/class/dmi/id/product_uuid
[sudo] password for devops:
04ec4899-6462-f14d-b6a8-4d8a1a8abc9d
```

4. Ensure unique hostname for the machine:
   Note: This may require you to start a new session for hostname to reflect in command prompt.

```
devops@devops-VirtualBox:~$ sudo hostnamectl set-hostname master

devops@master:~$ hostname
master
devops@master:~$ cat /etc/hostname
master
```

5. Set Static ip address in Ubuntu 18.04:

```
devops@master:~$ sudo netplan generate

devops@master:~$ sudo ls /etc/netplan/
01-network-manager-all.yaml

devops@master:~$ sudo cat /etc/netplan/01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.56.120/24, ]
    enp0s8:
      dhcp4: yes
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
```

```
devops@master:~$ sudo netplan apply
devops@master:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
state UP group default qlen 1000
    link/ether 08:00:27:58:da:07 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.120/24 brd 192.168.56.255 scope global
noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe58:da07/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
state UP group default qlen 1000
    link/ether 08:00:27:1f:03:46 brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.15/24 brd 10.0.3.255 scope global dynamic
noprefixroute enp0s8
       valid_lft 86340sec preferred_lft 86340sec
    inet6 fe80::a00:27ff:fe1f:346/64 scope link
       valid_lft forever preferred_lft forever

devops@master:~$ sudo cat /etc/hosts
127.0.0.1   localhost
127.0.1.1   devops-VirtualBox
192.168.56.120  master

devops@master:~$ ping master
PING master (192.168.56.120) 56(84) bytes of data.
64 bytes from master (192.168.56.120): icmp_seq=1 ttl=64 time=0.021
ms
64 bytes from master (192.168.56.120): icmp_seq=2 ttl=64 time=0.025
ms
^C
--- master ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1011ms
rtt min/avg/max/mdev = 0.021/0.023/0.025/0.002 ms
```

6. Letting iptables see bridged traffic:

```
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

devops@master:~$ cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
> br_netfilter
> EOF
[sudo] password for devops:
br_netfilter

devops@master:~$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
> net.bridge.bridge-nf-call-ip6tables = 1
> net.bridge.bridge-nf-call-iptables = 1
> EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1

devops@master:~$ sudo sysctl --system
* Applying /etc/sysctl.d/10-console-messages.conf ...
kernel.printk = 4 4 1 7
* Applying /etc/sysctl.d/10-ipv6-privacy.conf ...
net.ipv6.conf.all.use_tempaddr = 2
net.ipv6.conf.default.use_tempaddr = 2
* Applying /etc/sysctl.d/10-kernel-hardening.conf ...
kernel.kptr_restrict = 1
* Applying /etc/sysctl.d/10-link-restrictions.conf ...
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
* Applying /etc/sysctl.d/10-magic-sysrq.conf ...
kernel.sysrq = 176
* Applying /etc/sysctl.d/10-network-security.conf ...
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.tcp_syncookies = 1
* Applying /etc/sysctl.d/10-ptrace.conf ...
kernel.yama.ptrace_scope = 1
* Applying /etc/sysctl.d/10-zeropage.conf ...
vm.mmap_min_addr = 65536
* Applying /usr/lib/sysctl.d/50-default.conf ...
net.ipv4.conf.all.promote_secondaries = 1
net.core.default_qdisc = fq_codel
* Applying /etc/sysctl.d/99-sysctl.conf ...
* Applying /etc/sysctl.d/k8s.conf ...
* Applying /etc/sysctl.conf ...
```

7. Check for availability of following ports:

```
Install "netstat" if not found using the following command.
devops@master:~$ sudo apt-get install net-tools

devops@master:~$ netstat --listen | grep 6443
devops@master:~$ netstat --listen | grep 2379
devops@master:~$ netstat --listen | grep 2380
devops@master:~$ netstat --listen | grep 1025[0-2]
devops@master:~$ netstat --listen | grep 3[0-2][0-7][0-6][0-7]
```

8. Install Docker:

```
   a. Install packages to allow apt to use a repository over HTTPS

sudo apt-get update && sudo apt-get install -y \
  apt-transport-https ca-certificates curl software-properties-common
gnupg2
```

b. Add Docker's official GPG key:

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

c. Add the Docker apt repository:

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

d. Install Docker CE

```
sudo apt-get update && sudo apt-get install -y \
  containerd.io=1.2.13-2 \
  docker-ce=5:19.03.11~3-0~ubuntu-$(lsb_release -cs) \
  docker-ce-cli=5:19.03.11~3-0~ubuntu-$(lsb_release -cs)
```

e. Create /etc/docker

```
sudo mkdir /etc/docker
```

f. Set up the Docker daemon

```
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
```

g. Create /etc/systemd/system/docker.service.d

```
sudo mkdir -p /etc/systemd/system/docker.service.d
```

h. Restart Docker

```
sudo systemctl daemon-reload
sudo systemctl restart docker
```

i. If you want the docker service to start on boot, run the following command:

```
sudo systemctl enable docker
```

j. Check for completion of Docker installation.

```
devops@master:~$ docker --version
Docker version 19.03.11, build 42e35e61f3
```

k. To allow docker to be used as non-root user:

Create the docker group.

```
$ sudo groupadd docker
```

Add your user to the docker group.

```
$ sudo usermod -aG docker $USER
```

Log out and log back in so that your group membership is re-evaluated.

9. Install Kubernetes:

```
sudo apt-get update && sudo apt-get install -y apt-transport-https
curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
apt-key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl
```

Check for Kubernetes Installation:

```
devops@master:~$ kubeadm version
kubeadm version: &version.Info{Major:"1", Minor:"20",
GitVersion:"v1.20.2",
GitCommit:"faecb196815e248d3ecfb03c680a4507229c2a56",
GitTreeState:"clean", BuildDate:"2021-01-13T13:25:59Z",
GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
```