

Create Kubernetes cluster with Kubeadm

Including a master node and a worker node

Create virtual machines

- - It is needed to create two virtual machines (one for each node)
- - Ubuntu 20.04 TLS
- - Need to create two virtual machines with the normal process

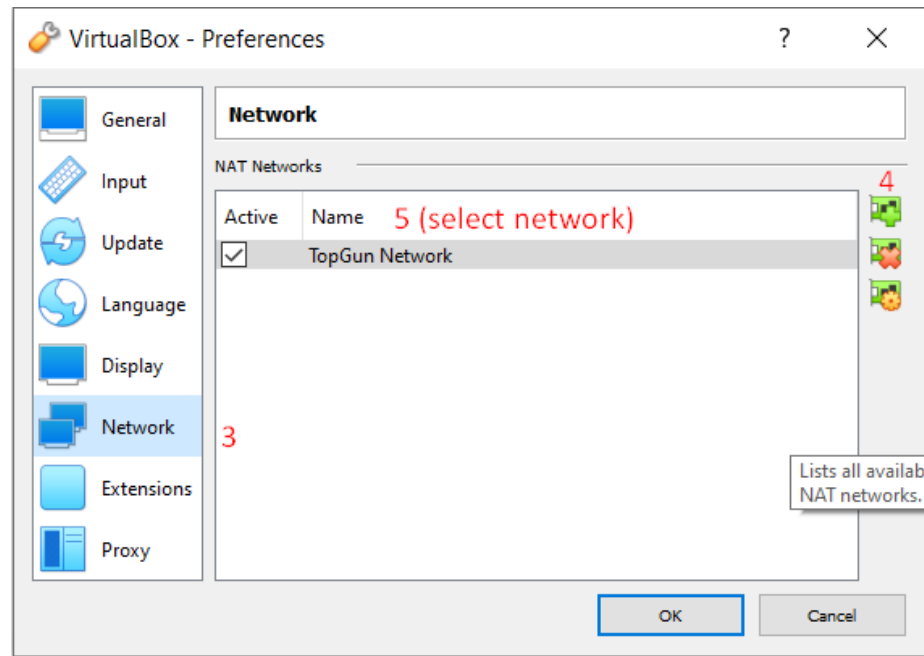
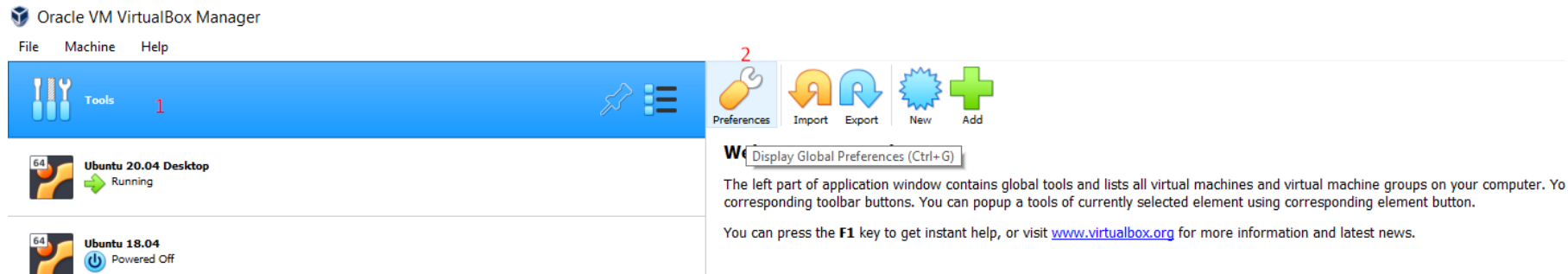
Role	IP	OS	RAM	CPU
Master	192.168.100.4*	Ubuntu 20.04 TLS	2Gb	2
Worker	192.168.100.5*	Ubuntu 20.04 TLS	1Gb	1

- After creating VM, they must be started. Then, you use the Ubuntu 20.04 TLS ISO file to put the OS.
- Install Ubuntu with minimum requirements, nothing else needed. Give the VM names like master, worker, or easy identifiable names, as they will be much used in future.

* The Ips are assigned automatically by the DHCP after we create a virtual box network (in next slide)

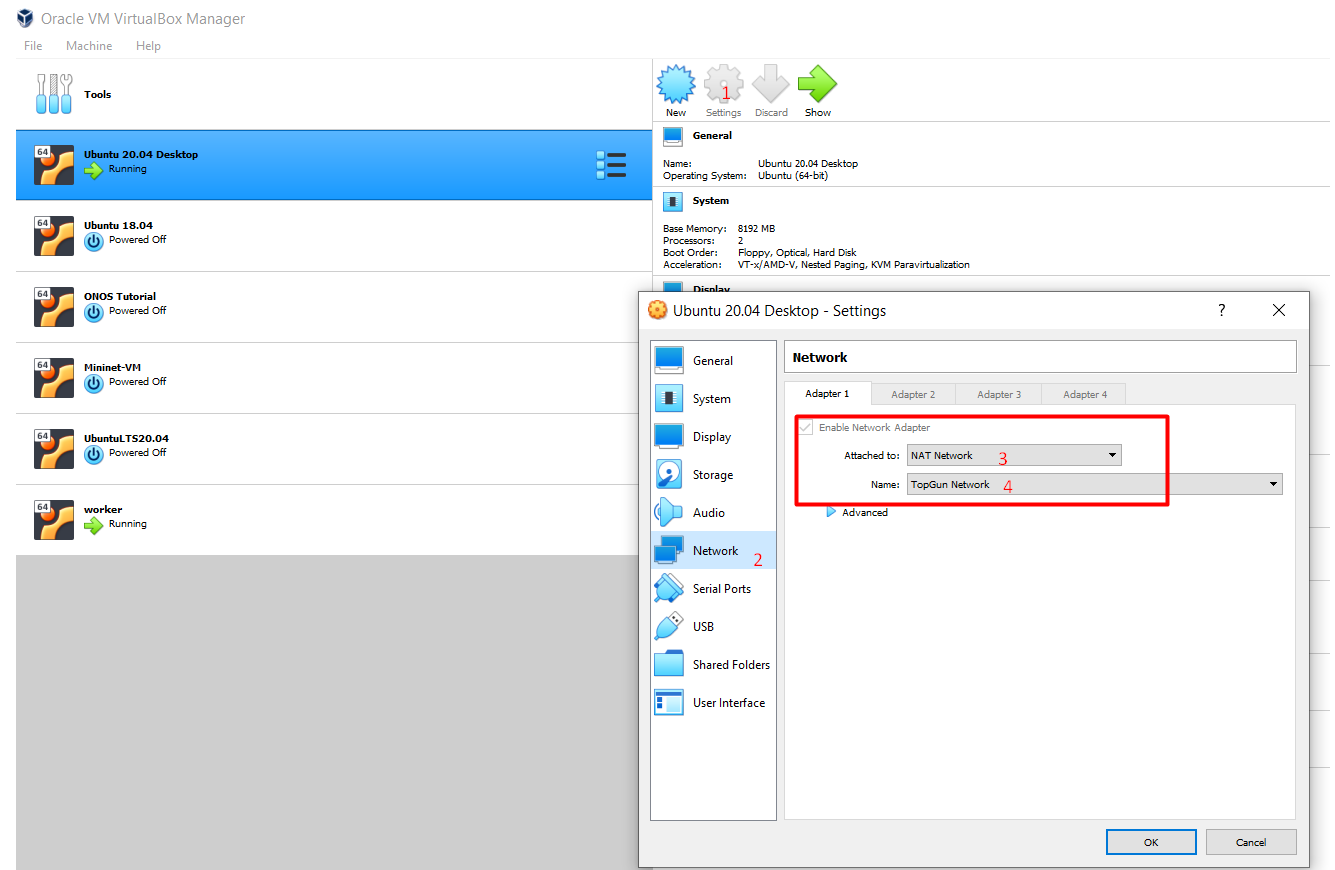
Create network for virtual machines

- In virtual Box, a network must be created for the VMs to talk among them



Create network for virtual machines (2)

- Need to access the settings of each VM (two in our case, which will be master node and worker node), and select in the network settings NAT network (in our case, TopGun)



Steps for both VM to work before installing Kubernetes

- Execute commands:
 - `sudo apt install net-tools` -> Install ifconfig commands and etc
 - `sudo apt-get install openssh-server` -> Install to allow to do ssh connections among VMs if needed (probably needed in future for testing purposes).
- Check IP of VM
 - Execute command `ifconfig` -> As a result, it gives several interfaces. The one with name `enp0s3` is the IP of the VM.

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 192.168.100.4 netmask 255.255.255.0 broadcast 192.168.100.255
        inet6 fe80::ee0d:dcf:94d6:6d05 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:41:17:d1 txqueuelen 1000 (Ethernet)
        RX packets 65581 bytes 17491855 (17.4 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 62131 bytes 23522189 (23.5 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 1142186 bytes 222766812 (222.7 MB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 1142186 bytes 222766812 (222.7 MB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Test Ips, SSH

- - Try a SSH connection from one VM (master) to the other (worker), as a last step to check that the network is working fine
 - Execute command `ssh user@ip*`
 - **Ej. ssh mastervm@192.168.100.4*
- In this moment, everything is ready to create Kubernetes cluster with Kubeadm.

Steps for both VM (Master and worker)

- Log in as root user (every command needs to be executed as root): `sudo su -`
- Disable Firewall: `ufw disable`
- Disable swap: `swapoff -a; sed -i '/swap/d' /etc/fstab`
- Update sysctl settings for Kubernetes networking: `cat >>/etc/sysctl.d/kubernetes.conf<<EOF`

```
net.bridge.bridge-nf-call-ip6tables = 1
```

```
net.bridge.bridge-nf-call-iptables = 1
```

```
EOF
```

```
sysctl --system
```

Install Docker Engine

- `apt install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common`
- `curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -`
- `add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"`
- `apt update`
- `apt install -y docker-ce=5:19.03.10~3-0~ubuntu-focal containerd.io`

Install Kubernetes

- `curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -`
- `echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" > /etc/apt/sources.list.d/kubernetes.list`
- `apt update && apt install -y kubeadm=1.18.5-00 kubelet=1.18.5-00 kubectl=1.18.5-00`
- ** Already installs kubectl, kubeadm and kubelet*

Steps for only Master node (1)

command

Always with root permission, with command `sudo su -` or putting `sudo` in front of every

- Initialize Kubernetes : `sudo kubeadm init --pod-network-cidr=192.168.0.0/16`

- Ejecutar los comandos que vienen a continuacion:

```
mkdir -p $HOME/.kube
```

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

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

- **Download Calico,yaml** file from website <https://docs.projectcalico.org/manifests/calico.yaml>
- **Edit the Calico.yaml file.** Since the BGP has a method that the pods connect to the first IP that is available and working, it can lead to errors when you have many interfaces in a machine. To solve this, we need to edit the Calico.yaml file and change the following values corresponding to section *auto-detect the BGP IP address*. Instead, the type (or name) should be “**IP_AUTODETECTION_METHOD**” and the value selected is “**can-reach=172.16.0.4**” which is the IP of my master node. That means, BGP will choose only an IP for the node that can connect to my master node.

```
3847 |
3848 | # Auto-detect the BGP IP address.
3849 | - name: IP
3850 | value: "autodetect"
```

Value by default: BAD

```
# Auto-detect the BGP IP address.
- name: IP_AUTODETECTION_METHOD
  value: "can-reach=172.16.0.4"
```

Personalized value: GOOD

Steps for only Master node (2)

command

Always with root permission, with command `sudo su -` or putting `sudo` in front of every

- Once Calico.yaml file is edited, execute command: `kubectl apply -f calico.yaml` (file is supposed to be in same directory where command is executed)
- After a couple minutes, execute command: `watch kubectl get pods -n calico-system` (if OK, every pod is running with status 1/1)
- Create Cluster join command (to create worker nodes for the master node): `kubeadm token create --print-join-command`
- It will generate something like this (it is the command needed to be executed in worker node to join to a master):

```
root@crubio-VirtualBox:~# kubeadm token create --print-join-command
W0826 14:56:29.153292 25558 configset.go:202] WARNING: kubeadm cannot validate component configs for API groups [kubernetes.k8s.io kubeproxy.config.k8s.io]
kubeadm join 192.168.100.4:6443 --token rvgh09.qscd9fhj63pank57 --discovery-token-ca-cert-hash sha256:037f47e5d98e7a6cb01be1c98403fdb896bbf400c1b3369148991b6c18e5f1b5
```

Steps for each worker node

- Copy that command in yellow (result of previous slide) and execute in each node (Virtual Machine) that you want to join the cluster.

```
root@crubio-VirtualBox:~# kubeadm token create --print-join-command
W0826 14:56:29.153292 25558 configset.go:202] WARNING: kubeadm cannot validate component configs for API groups [kubelet.config.k8s.io kubeproxy.config.k8s.io]
kubeadm join 192.168.100.4:6443 --token rvgh09.qscd9fhj63pank57 --discovery-token-ca-cert-hash sha256:037f47e5d98e7a6cb01be1c98403fdb896bbf400c1b3369148991b6c18e5f1b5
```

- Expected result is a success message saying that “Execute kubectl get nodes in your control-plane node (master node)” to view nodes in the cluster.

Checking that cluster is up

- Only from the master node, executing the command `kubectl get nodes`, the output should be:

```
root@crubio-VirtualBox:~# kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
crubio-virtualbox    Ready     master   87m   v1.18.5
worker               Ready     <none>   81m   v1.18.5
worker2              Ready     <none>   44m   v1.18.5
root@crubio-VirtualBox:~#
```

- If everything is good, after joining two nodes to the cluster, the command `kubectl get pods -n kube-system -o wide` should look like this:

```
root@crubio-VirtualBox:~# kubectl get pods -n kube-system -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP              NODE                NOMINATED NODE   READINESS GATES
calico-kube-controllers-746f9d75cb-9j8zq  1/1    Running   1           81m   192.168.63.70   crubio-virtualbox   <none>           <none>
calico-node-6hklg                      1/1    Running   1           81m   172.16.0.4      crubio-virtualbox   <none>           <none>
calico-node-7rskg                      1/1    Running   1           77m   172.16.0.5      worker               <none>           <none>
calico-node-mhcps                      1/1    Running   1           41m   172.16.0.6      worker2              <none>           <none>
coredns-66bff467f8-q55kj               1/1    Running   1           83m   192.168.63.68   crubio-virtualbox   <none>           <none>
coredns-66bff467f8-xh58w               1/1    Running   1           83m   192.168.63.69   crubio-virtualbox   <none>           <none>
etcd-crubio-virtualbox                 1/1    Running   1           83m   172.16.0.4      crubio-virtualbox   <none>           <none>
kube-apiserver-crubio-virtualbox        1/1    Running   1           83m   172.16.0.4      crubio-virtualbox   <none>           <none>
kube-controller-manager-crubio-virtualbox 1/1    Running   1           83m   172.16.0.4      crubio-virtualbox   <none>           <none>
kube-proxy-5qtcx                       1/1    Running   1           41m   172.16.0.6      worker2              <none>           <none>
kube-proxy-w6g9h                       1/1    Running   1           83m   172.16.0.4      crubio-virtualbox   <none>           <none>
kube-proxy-x5brk                       1/1    Running   1           77m   172.16.0.5      worker               <none>           <none>
kube-scheduler-crubio-virtualbox        1/1    Running   1           83m   172.16.0.4      crubio-virtualbox   <none>           <none>
root@crubio-VirtualBox:~#
```