Setup K8S Cluster

Build Base VM

VirtualBox VM Settings

Create a VM with two network interfaces:

1. Host Only Network
2. NAT

Base Memory:

- 2048 MB

HD Size

- 10 GB

Audio

- Disabled

Install Ubuntu or Centos and enable/install openssh if available.

Enable both network interfaces

If you are installing Ubuntu server it will enable a primary network interface.

For VirtualBox VMs we are using 2 network interfaces:

- a host only network for accessing the host from our machine without having to setup port forwarding. (primary)
- a NAT network used for accessing the internet

>ifconfig -a

Add the missing interface to your interfaces config file and reboot

> vi /etc/network/interfaces

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).


source /etc/network/interfaces.d/*


# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp0s3
iface enp0s3 inet dhcp

auto enp0s8
iface enp0s8 inet dhcp
```

> reboot

Login and get IP address:

> ifconfig

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:56:82:00
          inet addr:192.168.56.3  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe56:8200/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:170 errors:0 dropped:0 overruns:0 frame:0
          TX packets:112 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:18488 (18.4 KB)  TX bytes:19156 (19.1 KB)

enp0s8    Link encap:Ethernet  HWaddr 08:00:27:f0:a2:f5
          inet addr:10.0.3.15  Bcast:10.0.3.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fef0:a2f5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:329 errors:0 dropped:0 overruns:0 frame:0
          TX packets:141 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:389613 (389.6 KB)  TX bytes:10690 (10.6 KB)


...
```

Update apt-get

> sudo su

> apt-get update

Install openssh (if not already installed)

> apt-get install openssh-server

Now you can ssh into the virtual machine on the host-only network from your host

> ssh test@192.168.56.3

Install Docker

```
> sudo su
```

```
> apt-get install -y docker.io
```

Install Curl

> apt-get install -y apt-transport-https curl

Install Kubernetes

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc/apt/sources.list.d/kubernetes.
list
sudo apt-get update
sudo apt-get install -y kubectl kubelet kubeadm
sudo apt-mark hold kubelet kubeadm kubectl
```

Pull images

> kubeadm config images pull

```
[config/images] Pulled k8s.gcr.io/kube-apiserver:v1.13.1
[config/images] Pulled k8s.gcr.io/kube-controller-manager:v1.13.1
[config/images] Pulled k8s.gcr.io/kube-scheduler:v1.13.1
[config/images] Pulled k8s.gcr.io/kube-proxy:v1.13.1
[config/images] Pulled k8s.gcr.io/pause:3.1
[config/images] Pulled k8s.gcr.io/etcd:3.2.24
[config/images] Pulled k8s.gcr.io/coredns:1.2.6
```

Now clone (full clone) this VM with names:

- k8master
- k8worker1
- k8worker2

For the k8master, set the CPU cores to 2.

Setup Networking on VMs

On the VMs that we have defined, lets get them configured.

VM	Ip Address
k8master	192.168.56.100
k8worker1	192.168.56.101
k8worker2	192.168.56.102

Set Hostname

> sudo vi /etc/hostname

```
k8master
```

> sudo vi /etc/hosts

```
127.0.0.1       localhost
127.0.1.1       k8master

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Set IP address

Set a static ip address for our host-only interface (enp0s3)

> sudo vi /etc/network/interfaces

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).


source /etc/network/interfaces.d/*

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto enp0s3
iface enp0s3 inet static
        address 192.168.56.100
        netmask 255.255.255.0
        network 192.168.56.0
        broadcast 192.168.56.255


auto enp0s8
iface enp0s8 inet dhcp
```

Disable SWAP

> swapoff -va

> vi /etc/fstab

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda1 during installation
UUID=e7b204f7-9f41-42d4-b55f-292990f4137a /               ext4    errors=remount-ro 0   1
# swap was on /dev/sda5 during installation
#UUID=9ca9f4cb-876e-4e23-91a4-2f543b5537ac none            swap    sw              0   0
```

> reboot

Repeat for all VMs

Initialize Master

> sudo kubeadm init --apiserver-advertise-address 192.168.56.100 --pod-network-cidr 192.168.0.0/16

```
...
Your Kubernetes master 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

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/

You can now join any number of machines by running the following on each node
as root:

kubeadm join 192.168.56.100:6443 --token 69sqqp.yelc6ct7o3v3uoqp --discovery-token-ca-cert-hash sha256:
03b55f52661338d761e8dd68203b738f3e126428cda239db81c2723a7bccba83
```

Record the kubeadm join command!

As your non root user:

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

Verify that your network is on the right network interface

```
kubectl get pods -o wide --all-namespaces

NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE    IP          NODE
NOMINATED NODE   READINESS GATES
kube-system   coredns-86c58d9df4-8zk5t          0/1     Pending   0          2d3h   <none>      <none>
<none>           <none>
kube-system   coredns-86c58d9df4-tsftk          0/1     Pending   0          2d3h   <none>      <none>
<none>           <none>
kube-system   etcd-k8master                     1/1     Running   1          2d3h   10.0.3.15   k8master
<none>           <none>
kube-system   kube-apiserver-k8master           1/1     Running   1          2d3h   10.0.3.15   k8master
<none>           <none>
kube-system   kube-controller-manager-k8master  1/1     Running   1          2d3h   10.0.3.15   k8master
<none>           <none>
kube-system   kube-proxy-88gdq                  1/1     Running   1          2d3h   10.0.3.15   k8master
<none>           <none>
kube-system   kube-scheduler-k8master           1/1     Running   1          2d3h   10.0.3.15   k8master
<none>           <none>
```

The Ip should not be 10.0.3.xxx

Install Flannel Network Plugin

> kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Verify that all of your kubernetes pods are running

> kubectl get pods --all-namespaces

```
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   coredns-86c58d9df4-8zk5t          1/1     Running   0          47h
kube-system   coredns-86c58d9df4-tsftk          1/1     Running   0          47h
kube-system   etcd-k8master                     1/1     Running   1          47h
kube-system   kube-apiserver-k8master           1/1     Running   1          47h
kube-system   kube-controller-manager-k8master  1/1     Running   1          47h
kube-system   kube-flannel-ds-amd64-fl5wp       1/1     Running   0          12s
kube-system   kube-proxy-88gdq                  1/1     Running   1          47h
kube-system   kube-scheduler-k8master           1/1     Running   1          47h
```

Join Worker Nodes

User kubeadm join to join the cluster.

> kubeadm join 192.168.56.100:6443 --token 69sqqp.yelc6ct7o3v3uoqp --discovery-token-ca-cert-hash sha256:03b55f52661338d761e8dd68203b738f3e126428cda239db81c2723a7bccba83

Verify it is all working

From the master node:

```
sudo kubectl get nodes

NAME        STATUS   ROLES    AGE     VERSION
k8master    Ready    master   47h     v1.13.1
k8worker1   Ready    <none>   12m     v1.13.1
k8worker2   Ready    <none>   6m12s   v1.13.1


kubectl get pods --all-namespaces

NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
kube-system   coredns-86c58d9df4-8zk5t          1/1     Running   2          47h
kube-system   coredns-86c58d9df4-tsftk          1/1     Running   2          47h
kube-system   etcd-k8master                     1/1     Running   3          47h
kube-system   kube-apiserver-k8master           1/1     Running   3          47h
kube-system   kube-controller-manager-k8master  1/1     Running   3          47h
kube-system   kube-flannel-ds-amd64-fl5wp       1/1     Running   3          25m
kube-system   kube-flannel-ds-amd64-k26xv       1/1     Running   0          5m4s
kube-system   kube-flannel-ds-amd64-ncg64       1/1     Running   1          11m
kube-system   kube-proxy-88gdq                  1/1     Running   3          47h
kube-system   kube-proxy-b6m4d                  1/1     Running   0          5m4s
kube-system   kube-proxy-nxwmh                  1/1     Running   1          11m
kube-system   kube-scheduler-k8master           1/1     Running   3          47h
```

Now deploy something and verify it all works.

Install Some Example Pods

```
> kubectl create -f https://kubernetes.io/examples/application/deployment.yaml
> kubectl get pods

NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-76bf4969df-hkmjp   1/1     Running   0          2m18s
nginx-deployment-76bf4969df-x7f9h   1/1     Running   0          2m18s
```

Install Dashboard

From the master node:

```
> sudo su


> kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/master/aio/deploy/recommended
/kubernetes-dashboard.yaml


secret/kubernetes-dashboard-certs created
serviceaccount/kubernetes-dashboard created
role.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
deployment.apps/kubernetes-dashboard created
service/kubernetes-dashboard created


> kubectl proxy
```

From your local machine:

> ssh -L 8001:127.0.0.1:8001 test@192.168.56.100

Browse to:

....

References

Reference	URL
Building a Kuburnetes Cluster	https://medium.com/@KevinHoffman/building-a-kubernetes-cluster-in-virtualbox-with-ubuntu-22cd338846dd
Cluster Networking	https://kubernetes.io/docs/concepts/cluster-administration/networking/
Flannel	https://github.com/coreos/flannel#flannel
Dashboard	https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/#using-dashboard