Setup K8S Cluster (NAT Network)

Build Base VM

VirtualBox VM Settings

Create a VM with one network interfaces:

- NAT Network

Base Memory:

- 2048 MB

HD Size

- 10 GB

Audio

- Disabled

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

Login and get IP address:

> ifconfig

```
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:fb:43:44
          inet addr:10.0.2.6  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fefb:4344/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:370 errors:0 dropped:0 overruns:0 frame:0
          TX packets:256 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:410213 (410.2 KB)  TX bytes:23216 (23.2 KB)

...
```

Update apt-get

> sudo su

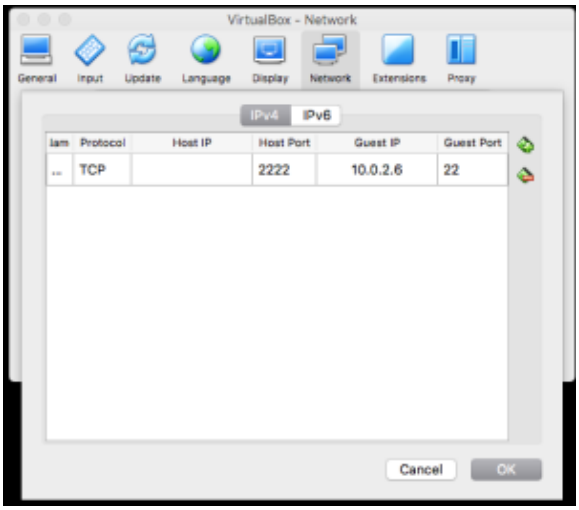
> apt-get update

Install openssh (if not already installed)

> apt-get install openssh-server

Setup port forwarding

Virtual Box VM Preferences Network Edit Nat Network Port Forwarding



Now you can ssh into the virtual machine from your host

> ssh test@10.0.2.6

Install Docker

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

Install Curl

> sudo 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 kmaster, 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	10.0.2.100
k8worker1	10.0.2.101
k8worker2	10.0.2.102

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 nat network 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 10.0.2.100
        netmask 255.255.255.0
        network 10.0.2.0
        broadcast 10.0.2.255
        gateway 10.0.2.1
        dns-nameservers 10.0.2.1 8.8.8.8
```

Disable SWAP

> sudo swapoff -va

> sudo 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 10.0.2.100 --pod-network-cidr 192.168.0.0/16

```
[init] Using Kubernetes version: v1.13.1
[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'
[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] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [k8master kubernetes kubernetes.default kubernetes.
default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.0.2.100]
[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 [k8master localhost] and IPs [10.0.2.100 127.0.0.1 ::1]
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [k8master localhost] and IPs [10.0.2.100 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-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
[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"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc
/kubernetes/manifests". This can take up to 4m0s
[apiclient] All control plane components are healthy after 20.009880 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace
[kubelet] Creating a ConfigMap "kubelet-config-1.13" in namespace kube-system with the configuration for the
kubelets in the cluster
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API object "k8master"
as an annotation
[mark-control-plane] Marking the node k8master as control-plane by adding the label "node-role.kubernetes.io
/master=''"
[mark-control-plane] Marking the node k8master as control-plane by adding the taints [node-role.kubernetes.io
/master:NoSchedule]
[bootstrap-token] Using token: i25b7g.0rowxx40128kpf0n
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get
long term certificate credentials
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a
Node Bootstrap Token
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client certificates in the
cluster
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

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 10.0.2.100:6443 --token i25b7g.0rowxx40128kpf0n --discovery-token-ca-cert-hash sha256:
92b2711cb1d1f7da7c6536991321a4c5224e05490da8ef07a7512372cddc9223
```

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   10.0.2.100   k8master
<none>           <none>
kube-system   kube-apiserver-k8master             1/1     Running   1          2d3h   10.0.2.100   k8master
<none>           <none>
kube-system   kube-controller-manager-k8master    1/1     Running   1          2d3h   10.0.2.100   k8master
<none>           <none>
kube-system   kube-proxy-88gdq                    1/1     Running   1          2d3h   10.0.2.100   k8master
<none>           <none>
kube-system   kube-scheduler-k8master             1/1     Running   1          2d3h   10.0.2.100   k8master
<none>           <none>
```

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