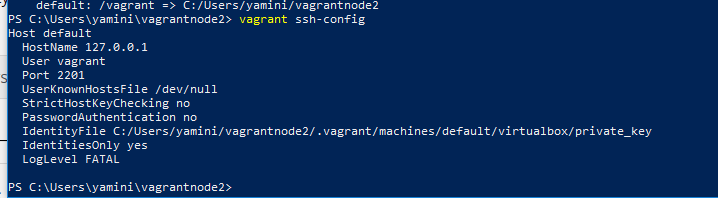
1. Download vagrant. ( vagrant version 2.1.1), prerequisite have a oraclevmbox
2. Create folder & cd to folder vagrantnode2
3. vagrant init bento/ubuntu-18.04
4. vagrant up

vagrant init – initializes a new vagrant folder

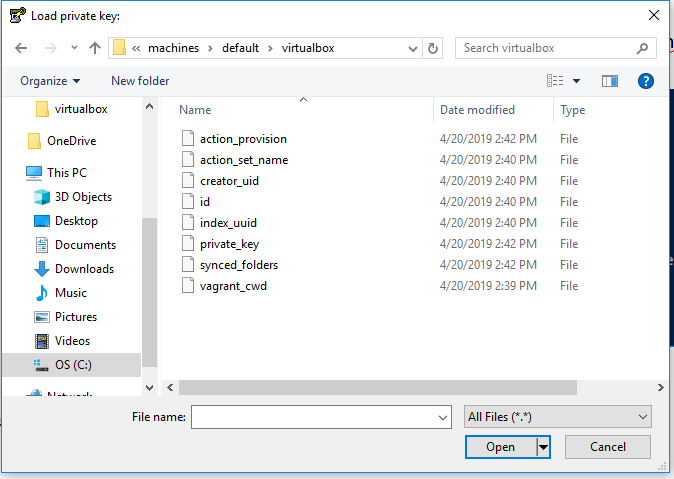
it creates vagrantfile (instructions) and .vagrant (where it has the vms specifications)

1. after box is configured, Most important config is the user and port using command vagrant ssh-config

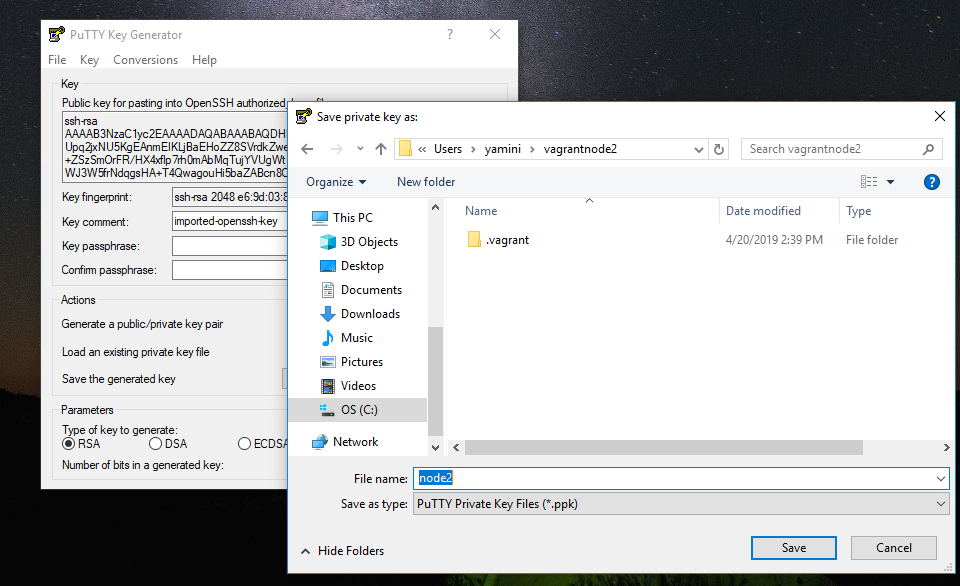


1. using putty to login, Create a private using puttygen from

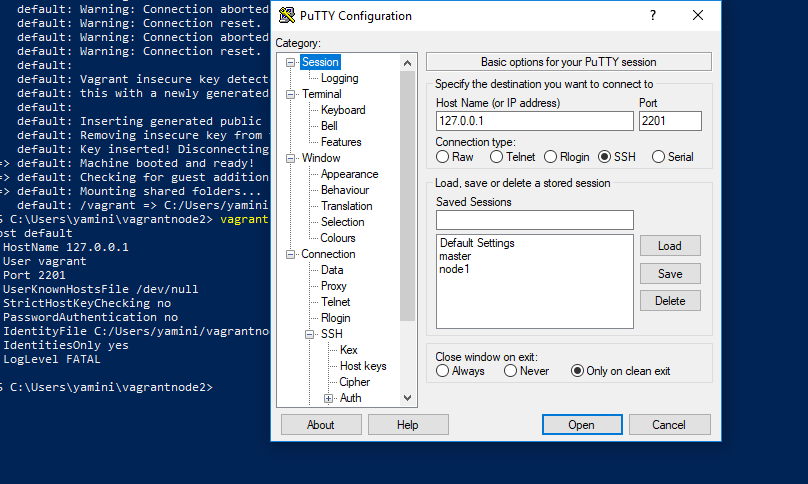
path: C:\Users\yamini\vagrantnode2\.vagrant\machines\default\virtualbox and use all files to view privatekey



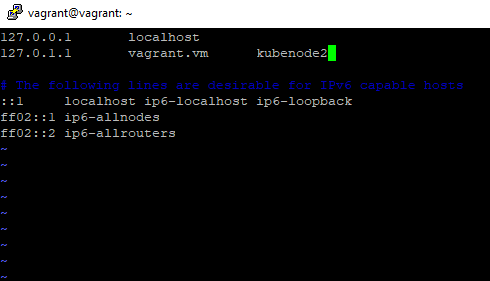
1. After loading save privatekey



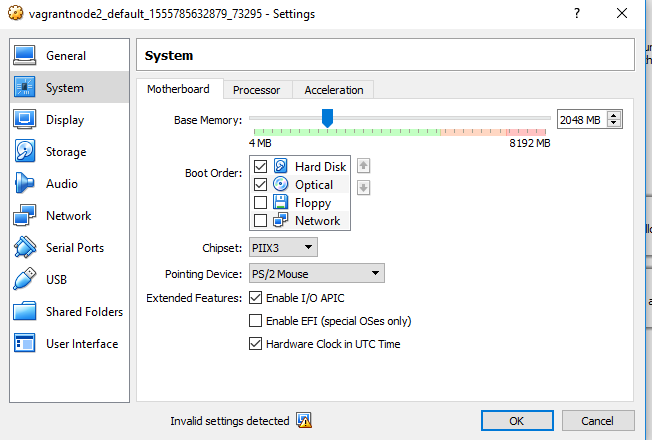
1. Use putty to create a session. Most important is the port and address from vagrant ssh-config and also data where username and also authfile to load ppk



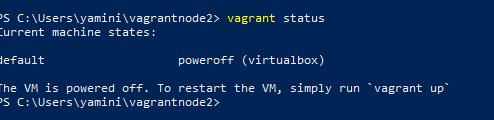
1. Change the hostname kubenode2 in /etc/hostname and /etc/hosts & restart using sudo poweroff or sudo reboot



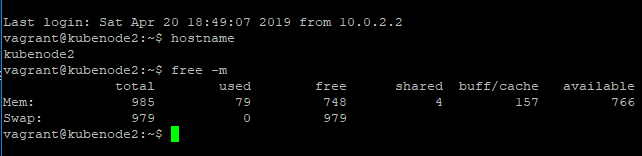
1. Change the memory settings



1. Checking status of box.



1. Checking whether vm hostname is changed:



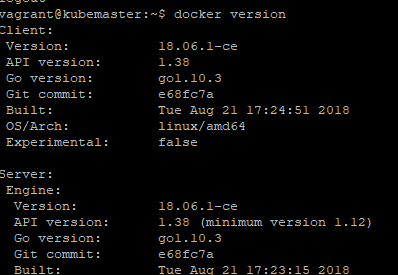
1. Installing Docker and kubernetes
2. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
3. sudo add-apt-repository \

"deb [arch=amd64] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) \

stable"

1. sudo apt-get update
2. sudo apt-get install -y docker-ce=18.06.1~ce~3-0~ubuntu
3. sudo apt-mark hold docker-ce (to lock down the version)
4. Please make sure the daemon is running by checking permissions for /var/run/docker.sock and using docker ps



1. Kubernetes:
2. curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
3. cat << EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF

1. sudo apt-get update
2. sudo apt-get install -y kubelet=1.12.7-00 kubeadm=1.12.7-00 kubectl=1.12.7-00
3. sudo apt-mark hold kubelet kubeadm kubectl

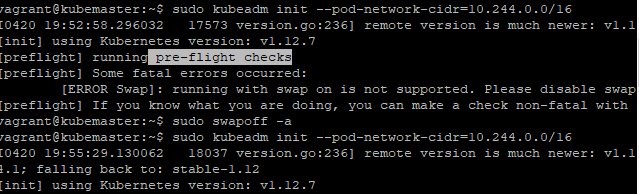
15 . After performing or adding we need to bootstrap first to have pod networking which is used in flannel

On masternode:

sudo kubeadm init --pod-network-cidr=10.244.0.0/16

Issue like swap space may come in: which shows preflight checks

Then run sudo swapoff –a and kubeadm init



1. kubeadm is now working and you can check how many versions are created

vagrant@kubemaster:~$ sudo kubeadm init --pod-network-cidr=10.244.0.0/16

I0420 19:55:29.130062 18037 version.go:236] remote version is much newer: v1.1 4.1; falling back to: stable-1.12

[init] using Kubernetes version: v1.12.7

[preflight] running pre-flight checks

[preflight/images] Pulling images required for setting up a Kubernetes cluster

[preflight/images] This might take a minute or two, depending on the speed of yo ur internet connection

[preflight/images] You can also perform this action in beforehand using 'kubeadm config images pull'

[kubelet] Writing kubelet environment file with flags to file "/var/lib/kubelet/ kubeadm-flags.env"

[kubelet] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"

[preflight] Activating the kubelet service

[certificates] Generated etcd/ca certificate and key.

[certificates] Generated etcd/peer certificate and key.

[certificates] etcd/peer serving cert is signed for DNS names [kubemaster localh ost] and IPs [10.0.2.15 127.0.0.1 ::1]

[certificates] Generated etcd/healthcheck-client certificate and key.

[certificates] Generated etcd/server certificate and key.

[certificates] etcd/server serving cert is signed for DNS names [kubemaster loca lhost] and IPs [127.0.0.1 ::1]

[certificates] Generated apiserver-etcd-client certificate and key.

[certificates] Generated ca certificate and key.

[certificates] Generated apiserver certificate and key.

[certificates] apiserver serving cert is signed for DNS names [kubemaster kubern etes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.lo cal] and IPs [10.96.0.1 10.0.2.15]

[certificates] Generated apiserver-kubelet-client certificate and key.

[certificates] Generated front-proxy-ca certificate and key.

[certificates] Generated front-proxy-client certificate and key.

[certificates] valid certificates and keys now exist in "/etc/kubernetes/pki"

[certificates] Generated sa key and public key.

[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"

[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"

[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager. conf"

[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"

[controlplane] wrote Static Pod manifest for component kube-apiserver to "/etc/k ubernetes/manifests/kube-apiserver.yaml"

[controlplane] wrote Static Pod manifest for component kube-controller-manager t o "/etc/kubernetes/manifests/kube-controller-manager.yaml"

[controlplane] wrote Static Pod manifest for component kube-scheduler to "/etc/k ubernetes/manifests/kube-scheduler.yaml"

[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/m anifests/etcd.yaml"

[init] waiting for the kubelet to boot up the control plane as Static Pods from directory "/etc/kubernetes/manifests"

[init] this might take a minute or longer if the control plane images have to be pulled

[apiclient] All control plane components are healthy after 45.053620 seconds

[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in t he "kube-system" Namespace

[kubelet] Creating a ConfigMap "kubelet-config-1.12" in namespace kube-system wi th the configuration for the kubelets in the cluster

[markmaster] Marking the node kubemaster as master by adding the label "node-rol e.kubernetes.io/master=''"

[markmaster] Marking the node kubemaster as master by adding the taints [node-ro le.kubernetes.io/master:NoSchedule]

[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to t he Node API object "kubemaster" as an annotation

[bootstraptoken] using token: grigg6.vpwkstrafnm2hk34

[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CS Rs in order for nodes to get long term certificate credentials

[bootstraptoken] configured RBAC rules to allow the csrapprover controller autom atically approve CSRs from a Node Bootstrap Token

[bootstraptoken] configured RBAC rules to allow certificate rotation for all nod e client certificates in the cluster

[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" name space

[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 172.31.22.47:6443 --token 77af8b.qc4m1ix1v3rm3irj --discovery-token-ca-cert-hash sha256:2bcc453d7f2ba7a9529a4d0f091d3bcea90a75ebb23f4e12b758126355aff4d7

Simple issues and resolutions:

1. When not working in user add the home kube Config:

mkdir -p $HOME/.kube

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

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

1. How many namespaces / projects in openshift

Command : kubectl get namespaces

NAME STATUS AGE

default Active 17m

kube-public Active 17m

kube-system Active 17m

1. to find the IP address of the API server:

kubectl get pods --all-namespaces -o wide

1. label applied to the etcd pod on the master node.

kubectl get pods --all-namespaces --show-labels -o wide

component=etcd,tier=control-plane

1. if we forget to copy kubeadm join command

sudo kubeadm token create --print-join-command