**Lab 01 : Package and Dependencies**

**Ref : https://kubernetes.io/docs/setup/independent/install-kubeadm/**

**Step 1 : Install docker**

# yum install docker

systemctl enable docker && systemctl start docker

cat <<EOF > /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

EOF

cat /etc/yum.repos.d/kubernetes.repo

setenforce 0

 Disabling SELinux by running setenforce 0 is required to allow containers to access the host filesystem, which is required by pod networks for example. You have to do this until SELinux support is improved in the kubelet.

getenforce

yum install -y kubelet kubeadm kubectl

systemctl enable kubelet && systemctl start kubelet

systemctl disable firewalld && systemctl stop firewalld

kubeadm init --skip-preflight-checks --token-ttl 0

Notes :

init] Using Kubernetes version: v1.8.0

[init] Using Authorization modes: [Node RBAC]

[preflight] Skipping pre-flight checks

[kubeadm] WARNING: starting in 1.8, tokens expire after 24 hours by default (if you require a non-expiring token use --token-ttl 0)

[certificates] Generated ca certificate and key.

[certificates] Generated apiserver certificate and key.

[certificates] apiserver serving cert is signed for DNS names [kubeadmin2 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 10.128.0.5]

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

[certificates] Generated sa key and public 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"

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

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

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

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

[controlplane] Wrote Static Pod manifest for component kube-apiserver to "/etc/kubernetes/manifests/kube-apiserver.yaml"

[controlplane] Wrote Static Pod manifest for component kube-controller-manager to "/etc/kubernetes/manifests/kube-controller-manager.yaml"

[controlplane] Wrote Static Pod manifest for component kube-scheduler to "/etc/kubernetes/manifests/kube-scheduler.yaml"

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

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

init] This often takes around a minute; or longer if the control plane images have to be pulled.

[apiclient] All control plane components are healthy after 50.002664 seconds

[uploadconfig] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace

[markmaster] Will mark node kubeadmin2 as master by adding a label and a taint

[markmaster] Master kubeadmin2 tainted and labelled with key/value: node-role.kubernetes.io/master=""

[bootstraptoken] Using token: ba0689.104b8534bac9ff71

[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] Creating the "cluster-info" ConfigMap in the "kube-public" namespace

[addons] Applied essential addon: kube-dns

[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

To start using your cluster, you need to run (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:

http://kubernetes.io/docs/admin/addons/

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

as root:

kubeadm join --token ba0689.104b8534bac9ff71 10.128.0.5:6443 --discovery-token-ca-cert-hash sha256:7d56e6ca5462e5613a95db86165799cd6f971c3a8e5cf4d4e2db8fdb51a3f48c

echo $HOME

mkdir -p $HOME/.kube

ls $HOME ls -la

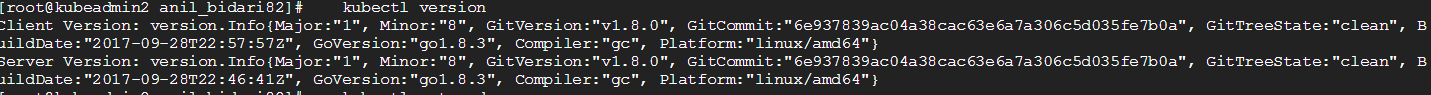
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

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

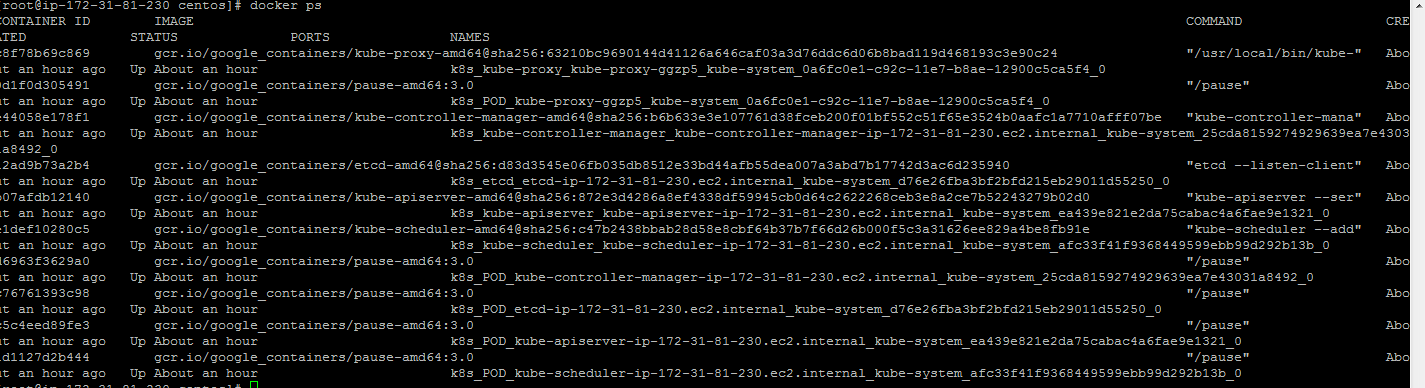
export KUBECONFIG=$HOME/.kube/config



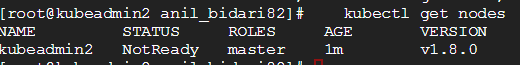
kubectl version



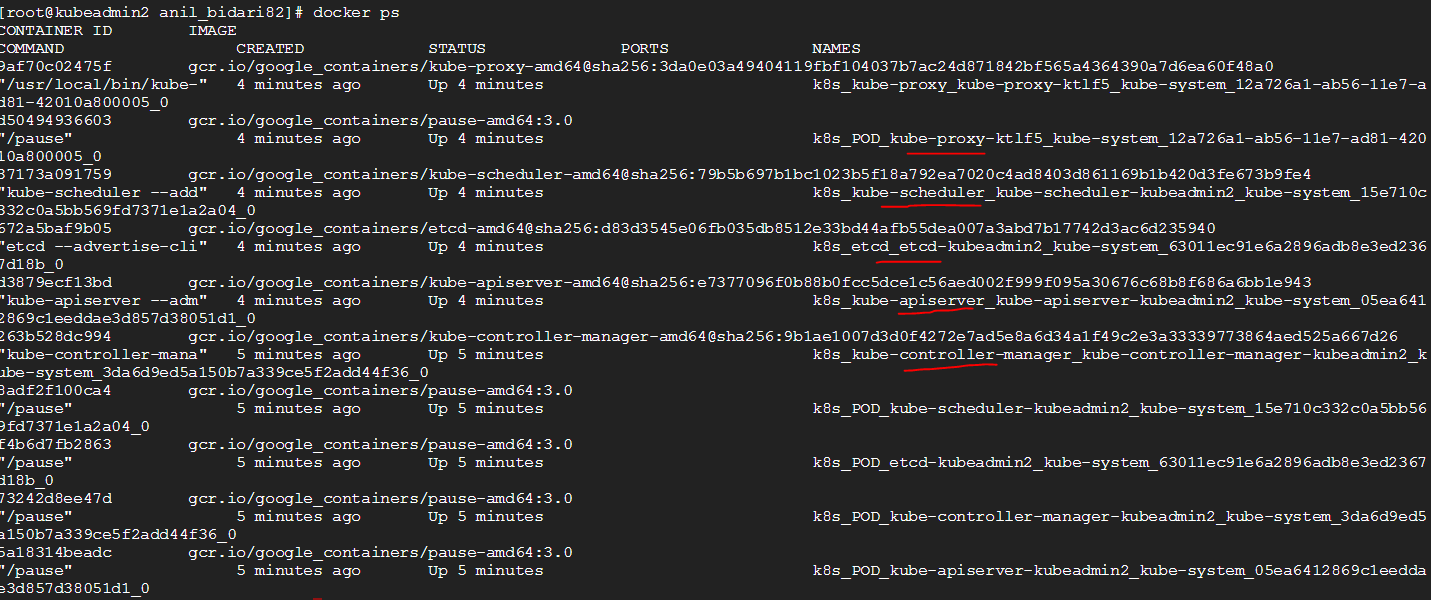
docker ps



kubectl get nodes



docker ps



kubectl apply -f https://git.io/weave-kube-1.6

kubectl get namespaces

kubectl get pods --all-namespaces

