**DevOps Task**

**Kubernetes Task:**

Assume we have two virtual machines which one is master and the other node is worker with static ip address and master node must have 2 cpus minimum and 2G ram minimum .

Master node ip : 192.168.1.13

Worker node ip : 192.168.1.12

**First Part of task:**

In this first part we will create kubernetes cluster that is consist of two nodes (master and worker) .

**First Step on master:**

we will set a new host name for the master

**# hostnamectl set-hostname master**

And we will add hostname of master and worker with their ip in /etc/hosts to deal with hostname or ip address

**# vim /etc/hosts**

Then add this lines

192.168.1.13 master

192.168.1.12 worker

After that we go to disable swap , selinux and update firewall rules as it is a required for creating kubernetes cluster

For swap :

**# swapoff –a**

**# vim /etc/fstab**

And hash this line which is belong to swap

**/dev/mapper/centos-swap swap swap defaults 0 0**

For selinux:

**# setenforce 0**

**# vim /etc/selinux/config**

Then change parameter SELINUX=enforcing to SELINUX=disabled then we must reboot to make changes permanent

**# reboot**

After rebooting master node successfully to apply changes we must update firewall rules or disable firewalld service so

In case one (update fire wall rules):

**# Firewall-cmd --permanent --add-port=6443/tcp**

**# Firewall-cmd --permanent --add-port=2379-2380/tcp**

**# Firewall-cmd --permanent --add-port=10250/tcp**

**# Firewall-cmd --permanent --add-port=10251/tcp**

**# Firewall-cmd --permanent --add-port=10252/tcp**

**# Firewall-cmd --permanent --add-port=10255/tcp**

**# Firewall-cmd --reload**

And we must update iptables config

**# vim /etc/sysctl.d/k8s.conf**

Then add this lines for ipv4 and ipv6

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

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

Then

**# sysctl --system**

In case two (disable firewalld service)

**# systemctl disable firewalld**

**# systemctl stop firewalld**

After this step , we will install docker containerization tool as kubernetes is based on managing docker containers

So from this tutorial link <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-centos-7>

We can create docker repo and install docker package with yum tool or we can run script to install docker which automate the process of creating docker repo and install docker package

**# yum check-update**

**# curl -fsSL https://get.docker.com/ | sh**

**# systemctl start docker**

**# systemctl enable docker**

After that we create kubernets repo and install kubelet kubectl kubeadm package from yum tool

**# vim /etc/yum.repos.d/k8s.repo**

And write this lines inside this file

**[k8s]**

**name=k8s**

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

**enabled=1**

**gpgcheck=0**

Then install kubernetes packages

**# yum install kubeadm kubelet kubectl**

After installing kubermetes packages we start kubelet service

**# systemctl start kubelet**

**# systemctl enable kubelet**

Initializing Kubernetes master is a fully automated process that is managed by the **kubeadm init** command which you will run.

**# kubeadm init**

Having initialized **Kubernetes** successfully, you will need to allow your user to start using the cluster. In our case, we want to run this installation as **root** user, therefore we will go ahead and run these commands as root.

**# mkdir -p $HOME/.kube**

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

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

After finish this configuration , We run

**# kubectl get nodes**

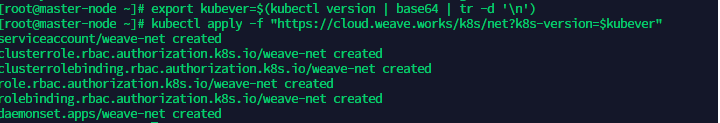
We will realize that master node is found with role master but not ready so we must setup virtual network (pod network) in the cluster



**# export kubever=$(kubectl version | base64 | tr -d '\n')**

**# kubectl apply –f** [**https://cloud.weave.works/k8s/net?k8s-version=$kubever**](https://cloud.weave.works/k8s/net?k8s-version=$kubever)

We will get this output



And now the master is ready



**Second Step on worker:**

we will set a new host name for the worker

**# hostnamectl set-hostname worker**

And we will add hostname of master and worker with their ip in /etc/hosts to deal with hostname or ip address

**# vim /etc/hosts**

Then add this lines

192.168.1.13 master

192.168.1.12 worker

After that we go to disable swap , selinux and update firewall rules as it is a required for creating kubernetes cluster

For swap :

**# swapoff –a**

**# vim /etc/fstab**

And hash this line which is belong to swap

**/dev/mapper/centos-swap swap swap defaults 0 0**

**For selinux :**

**# setenforce 0**

**# vim /etc/selinux/config**

Then change parameter SELINUX=enforcing to SELINUX=disabled then we must reboot to make changes permanent

**# reboot**

After rebooting master node successfully to apply changes we must update firewall rules or disable firewalld service so

In case one (update fire wall rules):

**# Firewall-cmd --permanent --add-port=6443/tcp**

**# Firewall-cmd --permanent --add-port=2379-2380/tcp**

**# Firewall-cmd --permanent --add-port=10250/tcp**

**# Firewall-cmd --permanent --add-port=10251/tcp**

**# Firewall-cmd --permanent --add-port=10252/tcp**

**# Firewall-cmd --permanent --add-port=10255/tcp**

**# Firewall-cmd --reload**

And we must update iptables config

**# vim /etc/sysctl.d/k8s.conf**

Then add this lines for ipv4 and ipv6

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

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

Then

**# sysctl --system**

In case two (disable firewalld service)

**# systemctl disable firewalld**

**# systemctl stop firewalld**

After this step , we will install docker containerization tool as kubernetes is based on managing docker containers

So from this tutorial link <https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-centos-7>

We can create docker repo and install docker package with yum tool or we can run script to install docker which automate the process of creating docker repo and install docker package

**# yum check-update**

**# curl -fsSL https://get.docker.com/ | sh**

**# systemctl start docker**

**# systemctl enable docker**

After that we create kubernets repo and install kubelet kubectl kubeadm package from yum tool

**# vim /etc/yum.repos.d/k8s.repo**

And write this lines inside this file

**[k8s]**

**name=k8s**

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

**enabled=1**

**gpgcheck=0**

Then install kubernetes packages

**# yum install kubeadm kubelet kubectl**

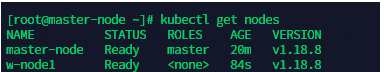
After installing kubermetes packages we start kubelet service

**# systemctl start kubelet**

**# systemctl enable kubelet**

Now We can join the cluster with worker node through

**# kubeadm join 192.168.1.13:6443 --token 848gwg.mpe76povky8qeqvu --discovery-token-ca-cert-hash sha256:f0a16f51dcc077da9e41f01bdcbc465343668f36d55f41250c570a2be8321eac**



For this point :

**Access to kubernetes API is restricted to number of IP addresses**

After investigation , I found that in all kubernetes configuration file , there is parameter is called **server** which **server:** [**https://192.168.1.13:6443**](https://192.168.1.13:6443)

This is mean that access to api is restricted to master ip 192.168.1.13 on port 6443 .

 Kubernetes Manifest files are the heart of Kubernetes. These plaintext configuration files describe how a pod's containers should be run and connected to other objects, such as services or replication controllers .

**For Second Part (Kubernetes Task):**

● Deploy nginx ingress - https://github.com/kubernetes/ingress-nginx/.

At the first we install git at our master node

**# yum install git**

After installing git , we download the content of nginx ingress from url

<https://github.com/kubernetes/ingress-nginx/>.

**# git clone** [**https://github.com/kubernetes/ingress-nginx/**](https://github.com/kubernetes/ingress-nginx/)

Then deploy nginx ingress from our yaml file

**#** **kubectl apply -f ingress-nginx/deploy/static/provider/cloud/deploy.yaml**

**# kubectl get pods --namespace=ingress-nginx**



Anther solution for this point

● Deploy nginx ingress - https://github.com/kubernetes/ingress-nginx/.

**# kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.1.1/deploy/static/provider/baremetal/deploy.yaml**

● Deploy "Juice Shop" application, you can use official docker image from ‘bkimminich/juice-shop’.

**# kubectl run juice-shop --image=bkimminich/juice-shop**



And to create deployment from bkimminich/juice-shop

**# vim juice-shop-deployment.yaml**

Then write this lines :

**kind: Deployment**

**apiVersion: apps/v1**

**metadata:**

**name: juice-shop**

**spec:**

**template:**

**metadata:**

**labels:**

**app: juice-shop**

**spec:**

**containers:**

**- name: juice-shop**

**image: bkimminich/juice-shop**

**selector:**

**matchLabels:**

**app: juice-shop**

After that run command

**# kubectl create -f juice-shop-deployment.yaml**

**# kubectl get pods**

This pod from the new deployment



● Expose “Juice Shop” application inside the cluster using the a service.

We create yaml file for service

**# vim juice-shop-service.yaml**

Then write the lines below

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: juice-shop service**

**spec:**

**type: ClusterIP**

**selector:**

**app: juice-shop**

**ports:**

**- name: http**

**port: 8000**

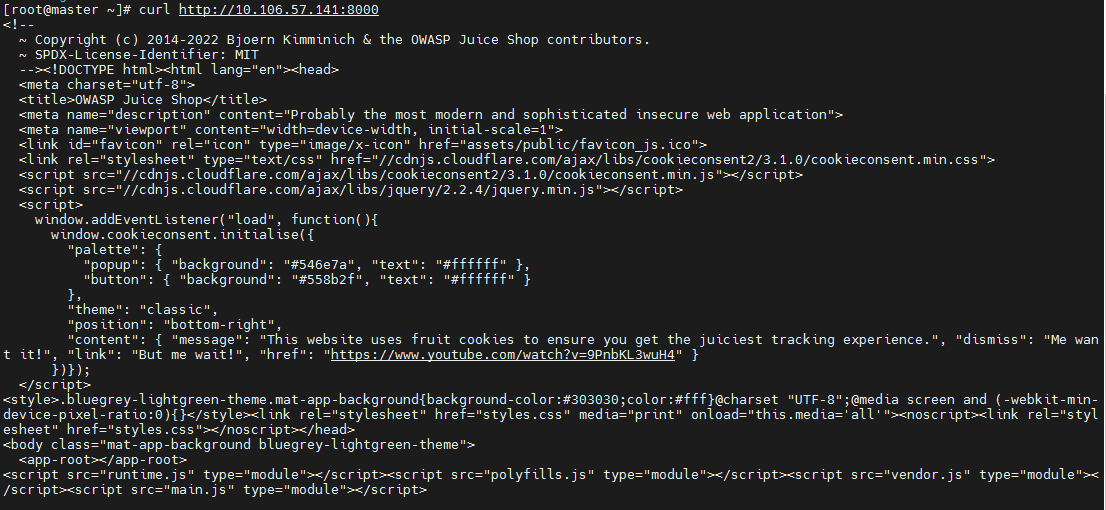
**targetPort: 3000**

**# kubectl create -f juice-shop-service.yaml**



To check ClusterIP

**# curl** [**http://10.106.57.141:8000**](http://10.106.57.141:8000)



● Expose “Juice Shop” application to outside the cluster using the nginx ingress.

We create yaml file for service

**# vim juice-shop-nginx.yaml**

Then write the lines below

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: juice-shop1**

**spec:**

**type: NodePort**

**ports:**

**- port: 80**

**nodePort: 30080**

**name: nginx**

**selector:**

**name: juice-shop1**

**# kubectl create -f juice-shop-nginx.yaml**



Anther solution for point

● Expose “Juice Shop” application to outside the cluster using the nginx ingress

**# vim ingress.yaml**

**apiVersion: networking.k8s.io/v1**

**kind: Ingress**

**metadata:**

**name: hello-world**

**namespace: default**

**annotations:**

**spec:**

**ingressClassName: nginx**

**rules:**

**- host: "hello.me.io"**

**http:**

**paths:**

**- pathType: Prefix**

**path: "/"**

**backend:**

**service:**

**name: juice-shop**

**port:**

**number: 8080**

**# kubectl create -f ingress.yaml**