**Section A: Kubernetes For CentOS 8:**

**STEP 1: Install this on All Master & Worker Nodes:**

*vim /etc/hosts*

*10.0.15.10      k8s-master  
10.0.15.21      node01  
10.0.15.22      node02*

**Disable SELinux**

*setenforce 0  
sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux*

### Enable br\_netfilter Kernel Module

*modprobe br\_netfilter  
echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables*

### Disable SWAP

*swapoff -a*

And then edit the '/etc/fstab' file.

*vim /etc/fstab*

Comment the swap line UUID

### Install Docker CE

Install the latest version of Docker-ce from the docker repository.

Install the package dependencies for docker-ce.

*yum install -y yum-utils device-mapper-persistent-data lvm2*

Add the docker repository to the system and install docker-ce using the yum command.

*yum-config-manager --add-repo* [*https://download.docker.com/linux/centos/docker-ce.repo*](https://download.docker.com/linux/centos/docker-ce.repo)

*yum erase podman buildah*

*yum install -y docker-ce*

### Install Kubernetes

Add the kubernetes repository to the centos 7 system by running the following command.

*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*

Now install the kubernetes packages kubeadm, kubelet, and kubectl using the yum command below.

*yum install -y kubelet kubeadm kubectl*

**RESTART all the Master & Worker Nodes**

Log in again to the server and start the services, docker and kubelet.

*systemctl start docker && systemctl enable docker  
systemctl start kubelet && systemctl enable kubelet*

### Change the cgroup-driver

We need to make sure the docker-ce and kubernetes are using same 'cgroup'.

Check docker cgroup using the docker info command.

*docker info | grep -i cgroup*

sed -i 's/cgroup-driver=systemd/cgroup-driver=cgroupfs/g' /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf

Reload the systemd system and restart the kubelet service.

*systemctl daemon-reload  
systemctl restart kubelet*

## Step 2 - Kubernetes Cluster Initialization

Only on MASTER NODE:

*kubeadm init --apiserver-advertise-address=<<Master Node IP Address>> --pod-network-cidr=10.244.0.0/16*

You will get something like this, Save them in a notepad or something

kubeadm join 192.168.1.40:6443 --token 4ucn56.tso7hrudxkjzo0y8 \

--discovery-token-ca-cert-hash sha256:c46506ebaec0f8a1062c4fec8c85c4566ff30ded24dc6868d75341d2a85c21b6

Create new '.kube' configuration directory and copy the configuration 'admin.conf'.

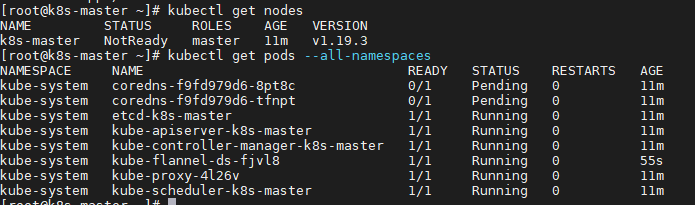
*mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config*

Next, deploy the flannel network to the kubernetes cluster using the kubectl command.

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

Wait for a minute and then check kubernetes node and pods using commands below.

*kubectl get nodes  
kubectl get pods --all-namespaces*



## Step 3 - Adding node01 and node02 to the Cluster

In this step, we will add node01 and node02 to join the 'k8s' cluster.

Connect to the node01 server and run the kubeadm join command as we copied on the top.

Run this in all the nodes:

“firewall-cmd --add-port=6443/tcp --permanent”

firewall-cmd --reload

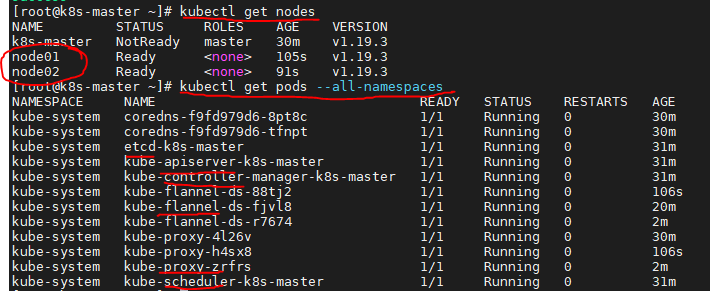
Run this only in worker nodes:

kubeadm join 192.168.1.40:6443 --token 4ucn56.tso7hrudxkjzo0y8 \

--discovery-token-ca-cert-hash sha256:c46506ebaec0f8a1062c4fec8c85c4566ff30ded24dc6868d75341d2a85c21b6

To Test Go to MASTER Node:

*kubectl get nodes  
kubectl get pods --all-namespaces*



You will see all these are running

## Step 4 - Testing Create First Pod

*kubectl create deployment nginx --image=nginx*

*kubectl describe deployment nginx*

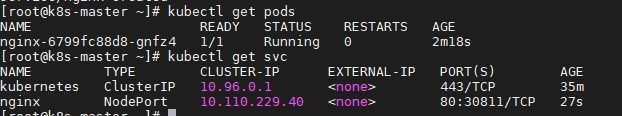
Next, we will expose the nginx pod accessible via the internet. And we need to create new service NodePort for this.

Run the kubectl command below.

*kubectl create service nodeport nginx --tcp=80:80*

To test again:

*kubectl get pods  
kubectl get svc*



Using the nginx port give in “kubectl get svc” command “curl node02:30811” try you will get something like below:

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Welcome to nginx!</title>**

**<style>**

**body {**

**width: 35em;**

**margin: 0 auto;**

**font-family: Tahoma, Verdana, Arial, sans-serif;**

**}**

**</style>**

**</head>**

**<body>**

**<h1>Welcome to nginx!</h1>**

**<p>If you see this page, the nginx web server is successfully installed and**

**working. Further configuration is required.</p>**

**<p>For online documentation and support please refer to**

**<a href="http://nginx.org/">nginx.org</a>.<br/>**

**Commercial support is available at**

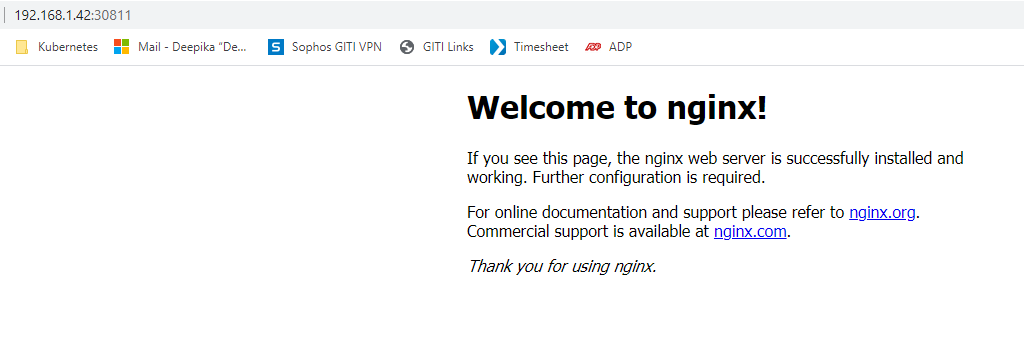
**<a href="http://nginx.com/">nginx.com</a>.</p>**

**<p><em>Thank you for using nginx.</em></p>**

**</body>**

**</html>**

Now access from the web browser.



**To get all IPAddress of the PODS:**

kubectl get pods -n kube-system

kubectl get service --all-namespaces

**To get all pods running:**

kubectl get pods --all-namespaces

**Section B: HELM Installation**

**Install Helm and Helm CLi for Providing Prometheus and Grafana**

*# curl -fsSL -o get\_helm.sh https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3*

*# chmod 700 get\_helm.sh*

*# ./get\_helm.sh*

*<<Once you installed Helm you can install any applications like below>>*

# *sudo ansible-galaxy collection install community.kubernetes*

# *sudo ansible-galaxy collection install community.grafana*

**Section C: ELK Installation**

Followed this link exactly for the ELK installation

https://medium.com/@tharangarajapaksha/elk-stack-in-k8s-cluster-13bb509185e0

**To find namespace so that you edit the existing configuration:**

**Method 1:**

kubectl get namespace

kubectl get  svc --namespace <<namespace>>

​

*first use the above command to get the services on the logging namespace*

​

 kubectl edit  svc <<Service name>> --namespace <<namespace>>

:q (to exit without saving, :wq (for save n exit)

If you want to use nano editor:

KUBE\_EDITOR="nano" kubectl edit svc logstash --namespace logging

KUBE\_EDITOR="nano" kubectl edit svc <<Service name>> --namespace <<namespace>>

**Method 2:**

Or if you deployed your changes using a configuration file:

Change your changes in that .yaml file then apply the below command:

kubectl apply -f logstash-deployment.yaml

**To find the configuration files**:

kubectl get namespace

*Syntax:* kubectl get configmap -n <<namespace>>

kubectl get configmap -n logging

To edit Conf file:

kubectl edit configmap logstash-pipeline --namespace logging

(or)

KUBE\_EDITOR="nano"  kubectl edit configmap logstash-pipeline --namespace logging

**How to test ELK environments in K8:**

kubectl get pods --all-namespaces

kubectl get namespace

kubectl exec -it <<POD Name>> bash -n <<name space>>

**ex:** kubectl exec -it elasticsearch-master-0 bash -n logging