[Kubernetes](https://www.techrepublic.com/search/?q=kubernetes) is an open-source system used for automating the deployment, scaling, and management of containerized applications. [Kubernetes](https://kubernetes.io/) is enterprise-ready and can be installed on various platforms. I've already touched on how to get Kubernetes up and running on Ubuntu Server (see "[How to quickly install Kubernetes on Ubuntu](https://www.techrepublic.com/article/how-to-quickly-install-kubernetes-on-ubuntu/)"). This time around, I'm going to walk you through the process of setting up a Kubernetes cluster on [CentOS 7](https://www.techrepublic.com/search/?q=CentOS). This makes for an outstanding one-two punch for your containerized applications.

**What you'll need**

I'll be demonstrating with Two RHEL/[CentOS 7](https://download.cnet.com/CentOS/3000-18513_4-75206335.html) servers (at the following IP addresses):

* 10.3.3.45 k8smaster.telenor.com.pk
* 10.3.3.46 k8sworker1.telenor.com.pk

Make sure to change the IP addresses to fit your needs. You'll also need root access on all two servers. I do suggest first testing this on virtual machines, before attempting the installation on production servers.

* With that said, let's install.

## Pre-installation configuration

The first thing you want to do is configure your /etc/hosts file, so that each machine can ping one another via hostname. So on each machine, issue the su command (to change to the root user) and then edit the file with the command nano /etc/hosts. At the end of the file, append the following (again, adjusting the IP addresses to fit your needs):

10.3.3.45 k8smaster.telenor.com.pk

10.3.3.46 k8sworker1.telenor.com.pk

Save and close that file.

## Disable SELinux and swap

Now we need to disable both SELinux and swap. On all two machines, issue the following commands:

**setenforce 0**

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

Next, disable swap (on all two machines) with the following command:

**swapoff -a**

We must also ensure that swap isn't re-enabled during a reboot on each server. Open up the /etc/fstab and comment out the swap entry like this:

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

## Enable br\_netfilter

For our next trick, we'll be enabling the br\_netfilter kernel module on all two servers. This is done with the following commands:

**modprobe br\_netfilter**

**echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables**

## Installation Docker and Kubernetes on RHEL/CENTOS

|  |  |  |
| --- | --- | --- |
| SR # | RHEL 7.5 | CENTOS 7 |
| 1 | Install Docker **sudo yum install -y docker**  **sudo systemctl enable docker && sudo systemctl start docker** | Install Docker-ce It's time to install the necessary Docker tool. On all two machines, install the Docker-ce dependencies with the following command:  **yum install -y yum-utils device-mapper-persistent-data lvm2**  Next, add the Docker-ce repository with the command:  **yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo**  Install Docker-ce with the command:  yum install -y docker-ce |
| 2 | Install Kubernetes **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**  **exclude=kube\***  **EOF**  **setenforce 0**  **yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes**  **systemctl enable kubelet && systemctl start kubelet** | Install Kubernetes This is also done on all two servers. First we need to create a repository entry for yum. To do this, issue the command nano /etc/yum.repos.d/kubernetes.repo and then add the following contents:  [**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  Save and close that file. Install Kubernetes with the command:  yum install -y kubelet kubeadm kubectl Once the installation completes, reboot all two machines. As soon as each machine has rebooted, log back in and su to the root user. |

## Cgroup changes

Now we need to ensure that both Docker-ce and Kubernetes belong to the same control group (cgroup). By default, Docker should already belong to cgroupfs (you can check this with the command docker info | grep -i cgroup). To add Kubernetes to this, issue the command:

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

Restart the systemd daemon and the kubelet service with the commands:

**systemctl daemon-reload**

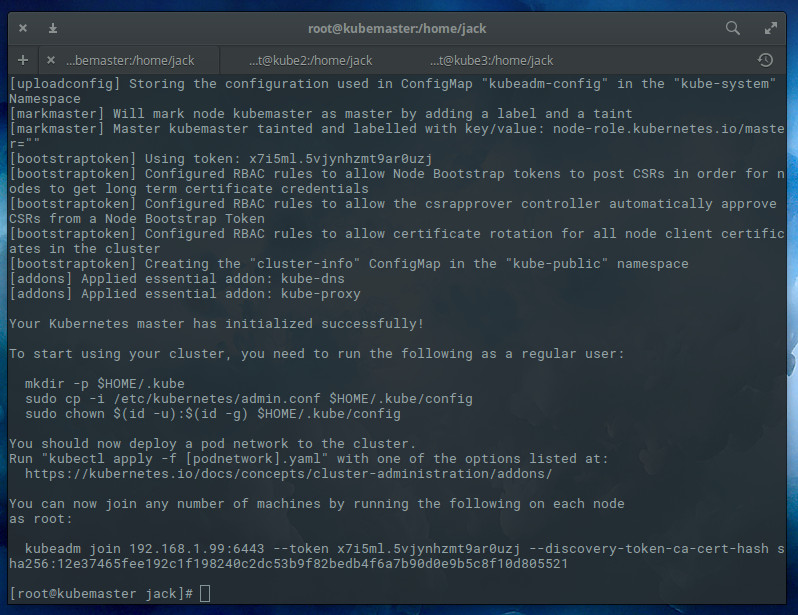
**systemctl restart kubelet**

## Initialize the Kubernetes cluster (At Master Node)

We're now ready to initialize the Kubernetes cluster. This is done on kubemaster (and only on that machine). On kubemaster, issue the command (again, adjusting the IP addresses to fit your needs):

**kubeadm init --apiserver-advertise-address=192.168.164.159 --pod-network-cidr=172.30.99.0/16 --service-cidr=172.31.99.0/16**

When this completes (it'll take anywhere from 30 seconds to 5 minutes), the output should include the joining command for your nodes (**Figure A**).



**Figure A**

The output of the Kubernetes cluster initialization command.

**ADD Nodes into cluster (At worker node)**

* Once that completes, head over to **worker nodes 1** and issue the command (adjusting the IP address to fit your needs):
* Where TOKEN and DISCOVERY\_TOKEN are the tokens displayed after the initialization command completes.

**Master Token for joining Worker Node 1:**

**[root@mytelenor4 ~]# kubeadm join 10.3.3.46:6443 --token 6mgkbd.7peeopbqbls74nqf --discovery-token-ca-cert-hash sha256:37fca968267f88903563901b40795ebc5dfafc47020905299d1f964f41124ae1**

Where **TOKEN** and **DISCOVERY\_TOKEN** are the tokens displayed after the initialization command completes.

## Configuring Kubernetes (At Master Node)

Before Kubernetes can be used, we must take care of a bit of configuration. Issue the following two commands (to create a new. kube configuration directory, copy the necessary configuration file, and give the file the proper ownership):

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

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

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

## Deploy flannel network

Now we must deploy the flannel network to the cluster with the command:

* **#download kube-flannel.yml file**

**curl -O https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml**

* **# change the ip 10.244.0.0/16 to "172.32.99.0/16” in kube-flannel file**

**nano kube-flannel.yml**

**kubectl apply -f kube-flannel.yml**

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

## Checking your nodes

Once the deploy command completes, you should be able to see both nodes on the master, by issuing the command **kubectl get nodes** (**Figure B**).

## Figure B

## All ready

Congratulations, you now have a Kubernetes cluster ready for pods. I'll be demonstrating how to deploy your first pod when next we visit the Kubernetes topic. Until then, happy clustering!

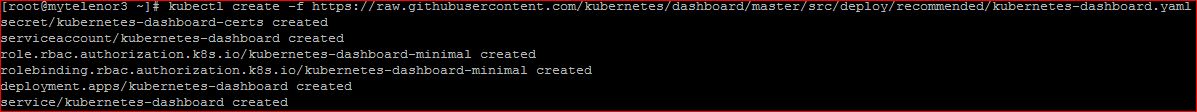
# Kubernetes dashboard:

**Deploying the Dashboard UI:**

The Dashboard UI is not deployed by default. To deploy it, run the following command:

[root@mytelenor3 ~]#**kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml**

It will take few seconds to complete the installation.



**Command to check kubernetes-dashboard service:**

[root@mytelenor3 ~]#kubectl -n kube-system get service kubernetes-dashboard



**CONFIGURING DASHBOARD:**

By default, the dashboard will install with minimum user role privileges.To access the dashboard with full administrative permission, create a YAML file named dashboard-admin.yaml.

[root@mytelenor3 ~]#vi dashboard-admin.yaml

* Add the below coding in the YAML file and save it.

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: kubernetes-dashboard

labels:

k8s-app: kubernetes-dashboard

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

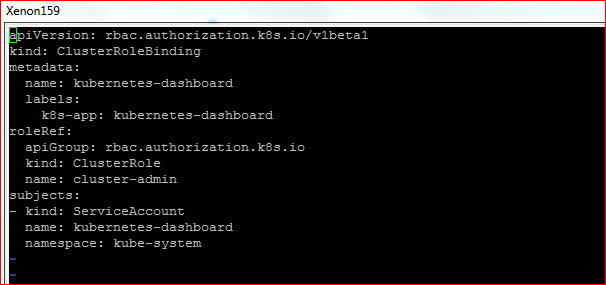
name: cluster-admin

subjects:

- kind: ServiceAccount

name: kubernetes-dashboard

namespace: kube-system



* Apply the full admin privileges to dashboard service account using the dashboard-admin YAML file.

[root@mytelenor3 ~]#kubectl create -f dashboard-admin.yaml

It will take few seconds to apply the settings.

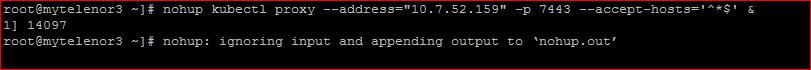
**D:\Xenon Project\create clusterrolebinding.JPG**

**Enabling KUBE PROXY:**

From the master server, execute the below command to run the kubernetes proxy command in the background.

[root@mytelenor3 ~]#nohup kubectl proxy --address="10.7.52.159" -p 7443 --accept-hosts='^\*$' &

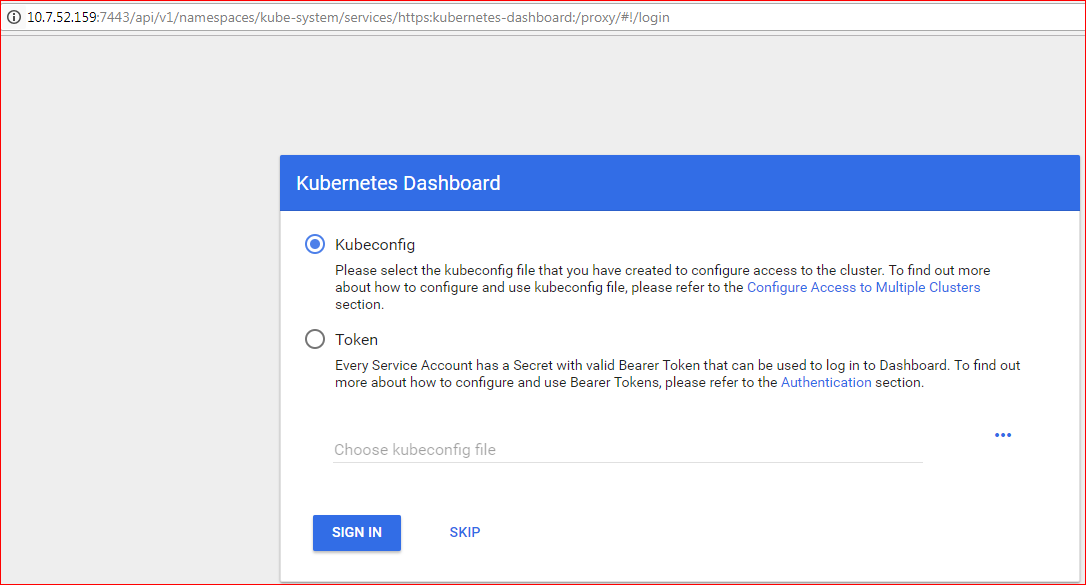
The proxy address has been added successfully.



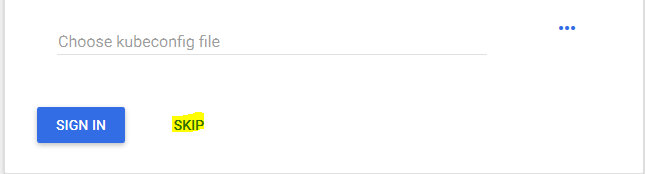
**ACCESSING THE DASHBOARD:**

* Open the web browser from your local PC and access the below URL.

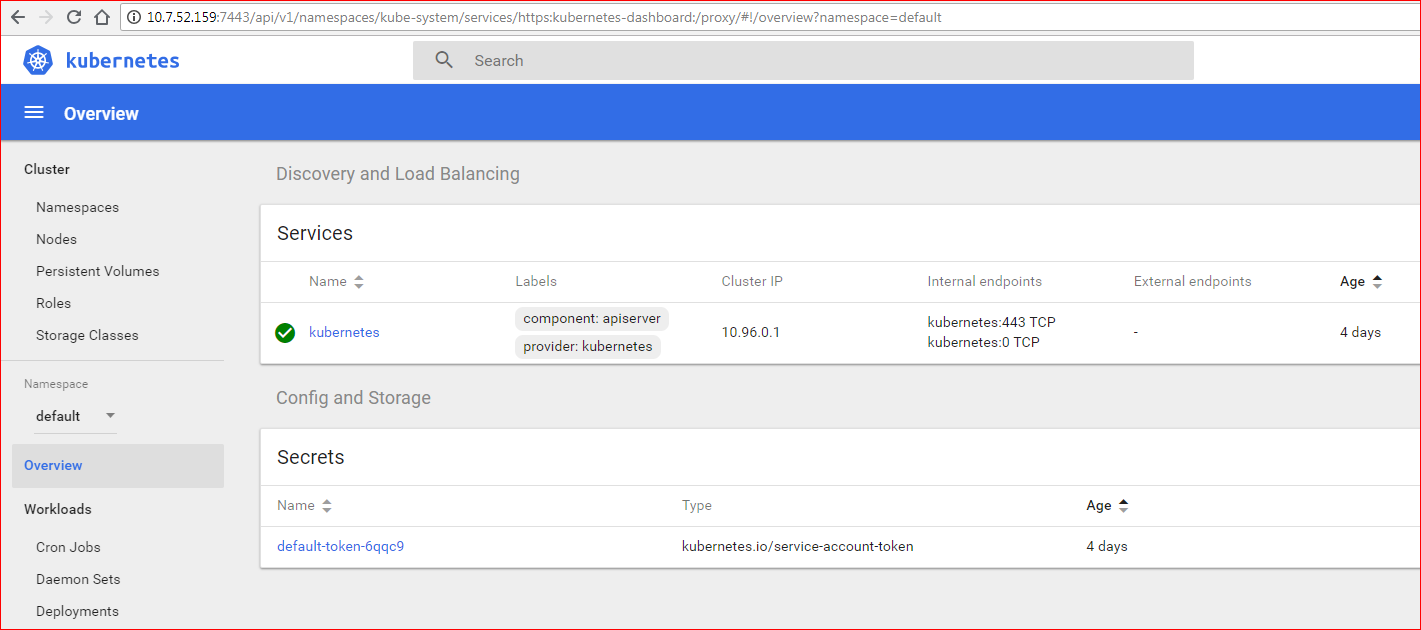
<http://10.3.3.45:7443/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>



* We can able to access the dashboard through the kubeconfig file or bearer token. We have already provided full admin access to dashboard service account. So just click on SKIP option to access the dashboard.



* Now we will able to view the homepage of the kubernetes dashboard.



# Troubleshooting

**---Reset kubernetes---**

sudo kubeadm reset

sudo systemctl daemon-reload

##sudo iptables -F && sudo iptables -t nat -F && sudo iptables -t mangle -F && sudo iptables -X

**----initialize----**

**#at master**

kubeadm init --apiserver-advertise-address=192.168.164.159 --pod-network-cidr=172.30.99.0/16 --service-cidr=172.31.99.0/16

**#at Worker nodes**

kubeadm join 192.168.164.159:6443 --token 3020mj.vw4hep3zqr4wpyjx --discovery-token-ca-cert-hash sha256:722cfc7c7cb401fcc6c1ba3fe0baf64466cbbe55eac8f8d587191ad93bea59bb

**#at master**

mkdir -p $HOME/.kube

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

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

kubectl get pods

kubectl get nodes

kubectl get pods -n kube-system

#download kube-flannel.yml file

curl -O https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

## change the ip 10.244.0.0/16 to "172.32.99.0/16" in kube-flannel file

nano kube-flannel.yml

kubectl apply -f kube-flannel.yml

kubectl get pods -n kube-system

###dasboard

kubectl create -f https://raw.githubusercontent.com/kubernetes/dashboard/master/src/deploy/recommended/kubernetes-dashboard.yaml

kubectl get pods -n kube-system

##dashboard-admin

#nano dashboard-admin.yaml

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: kubernetes-dashboard

labels:

k8s-app: kubernetes-dashboard

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: kubernetes-dashboard

namespace: kube-system

# run proxy

nohup kubectl proxy --address="192.168.164.159" -p 7443 --accept-hosts='^\*$' &

#and access the page in browser chrome

http://192.168.164.159:7443/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/

###run deployment test

kubectl run nginx --image=docker.io/nginx:latest --port=80 --replicas=1

kubectl expose deployment nginx --port=80 --type=NodePort

#check the status

kubectl get pods -o wide

**# to check status at master node**

kubectl get pods -n kube-system

kubectl describe pod -n kube-system coredns-78fcdf6894-5vwfl

kubectl describe node k8smaster.telenor.com.pk

kubectl logs -n kube-system kubernetes-dashboard-6948bdb78-xg592

kubectl get po -o wide --all-namespaces

kubectl get deployment -n kube-system

kubectl get Role -n kube-system

kubectl get secret -n kube-system

kubectl get serviceaccount -n kube-system

kubectl get rolebinding -n kube-system

##Delete all dashboard pods

kubectl get secret,sa,role,rolebinding,services,deployments --namespace=kube-system | grep dashboard

kubectl delete deployment kubernetes-dashboard --namespace=kube-system

kubectl delete service kubernetes-dashboard --namespace=kube-system

kubectl delete role kubernetes-dashboard-minimal --namespace=kube-system

kubectl delete rolebinding kubernetes-dashboard-minimal --namespace=kube-system

kubectl delete sa kubernetes-dashboard --namespace=kube-system

kubectl delete secret kubernetes-dashboard-certs --namespace=kube-system

kubectl delete secret kubernetes-dashboard-key-holder --namespace=kube-system

**#if the node showing SchedulingDisabled**

**[root@kubemaster ~]# kubectl get nodes**

NAME STATUS ROLES AGE VERSION

kubemaster.localdomain Ready master 61d v1.11.1

kubenode1.localdomain Ready,SchedulingDisabled <none> 61d v1.11.1

**#kubernetes disable scheduling**

kubectl uncordon kubenode1.localdomain

**#enable scheduling on working node**

kubectl cordon kubenode1.localdomain