

Setup K8S Cluster (Centos)

- [Build Base Image](#)
 - [Install Docker](#)
 - [Install Kubernetes](#)
 - [Pull images](#)
 - [Configure OS](#)
 - [Disable firewall](#)
 - [Fix for error: FileContent--proc-sys-net-bridge-bridge-nf-call-iptables](#)
 - [Disable SWAP](#)
- [Build the Load Balancer](#)
 - [Set Hostname](#)
 - [Update Yum](#)
 - [Disable firewall](#)
 - [Install haproxy](#)
- [Build a K8sMaster1 Node](#)
 - [Login to your Master node](#)
 - [Set the hostname](#)
 - [Generate SSH Key](#)
 - [Create kubeadm-config file](#)
 - [Initialize Master \(using Flannel\)](#)
 - [Install Flannel Network Plugin](#)
 - [Copy Certificates to Other Master Nodes](#)
- [Setup Other Master Nodes](#)
 - [Set the hostname](#)
 - [Move Certificates](#)
 - [Join the Cluster](#)
- [Setup Worker Nodes](#)
 - [Set the hostname](#)
 - [Join the Cluster](#)
- [Install Dashboard](#)
 - [Access the Dashboard by using the API proxy](#)
 - [Access the Dashboard using port forwarding](#)
 - [Sign in to the Dashboard](#)
- [Install Sample Pod](#)
- [Troubleshooting](#)
 - [Reset and start all over](#)
- [Centos Commands](#)
- [References](#)

Build Base Image

Install Docker

```
> sudo yum check-update  
  
> sudo curl -fsSL https://get.docker.com/ | sh  
  
> sudo systemctl start docker  
  
> sudo systemctl status docker  
  
> sudo systemctl enable docker  
  
> sudo usermod -aG docker $(whoami)  
  
> sudo reboot
```

Install Kubernetes

```
sudo su
```

```
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

# Set SELinux in permissive mode (effectively disabling it)
setenforce 0
sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

systemctl enable --now kubelet
```

Pull images

```
> sudo kubeadm config images pull
```

```
[config/images] Pulled k8s.gcr.io/kube-apiserver:v1.13.1
[config/images] Pulled k8s.gcr.io/kube-controller-manager:v1.13.1
[config/images] Pulled k8s.gcr.io/kube-scheduler:v1.13.1
[config/images] Pulled k8s.gcr.io/kube-proxy:v1.13.1
[config/images] Pulled k8s.gcr.io/pause:3.1
[config/images] Pulled k8s.gcr.io/etcd:3.2.24
[config/images] Pulled k8s.gcr.io/coredns:1.2.6
```

Configure OS

Disable firewall

```
> systemctl disable firewalld
> systemctl stop firewalld
> systemctl status firewalld
```

Fix for error: FileContent--proc-sys-net-bridge-bridge-nf-call-iptables

```
> vi /etc/sysctl.conf
```

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

```
> sudo sysctl -p
```

Disable SWAP

```
> swapoff -va
```

```
> vi /etc/fstab
```

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda1 during installation
UUID=e7b204f7-9f41-42d4-b55f-292990f4137a / ext4 errors=remount-ro 0 1
# swap was on /dev/sda5 during installation
#UUID=9ca9f4cb-876e-4e23-91a4-2f543b5537ac none swap sw 0 0
```

> reboot

Build the Load Balancer

Set Hostname

> sudo hostnamectl set-hostname k8slb

> sudo hostnamectl

Update Yum

> yum update

Disable firewall

```
> systemctl disable firewalld
> systemctl stop firewalld
> systemctl status firewalld
```

Install haproxy

> yum install haproxy

> vi /etc/haproxy/haproxy.cfg

/etc/haproxy/haproxy.cfg

```
global
...
defaults
...
#-----
# main frontend which proxys to the backends
#-----
frontend kubernetes
    bind *:6443
    mode tcp
    default_backend kubernetes-master-nodes

#-----
# static backend for serving up images, stylesheets and such
#-----
backend kubernetes-master-nodes
    mode tcp
    balance roundrobin
    option tcp-check
    server k8smaster1 172.20.233.181:6443 check fall 3 rise 2
    server k8smaster2 172.20.233.182:6443 check fall 3 rise 2
    server k8smaster3 172.20.233.183:6443 check fall 3 rise 2
```

```
> sudo systemctl start haproxy
> sudo systemctl enable haproxy
> sudo systemctl status haproxy
```

Verify that you can connect

```
nc -v LOAD_BALANCER_IP 6443
> nc -v 172.20.233.180 6443
```

Build a K8sMaster1 Node

Login to your Master node

```
> ssh test@172.20.233.181
```

Set the hostname

```
> sudo hostnamectl set-hostname k8smaster1
> sudo hostnamectl
```

Generate SSH Key

As test:

```
> ssh-keygen -t rsa -b 2048
```

Copy to other nodes

```
> ssh-copy-id test@172.20.233.182
```

...

```
> ssh-copy-id test@172.20.233.186
```

As root:

```
> sudo su
```

```
> ssh-keygen -t rsa -b 2048
```

Copy to other nodes

```
> ssh-copy-id test@172.20.233.182
```

```
...
```

```
> ssh-copy-id test@172.20.233.186
```

Create kubeadm-config file

```
> vi kubeadm-config.yaml
```

```
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
kubernetesVersion: stable
apiServer:
  certSANs:
    - "k8slb.ott.dev.intra"
controlPlaneEndpoint: "k8slb.ott.dev.intra:6443"
networking:
  podSubnet: 10.244.0.0/16
```

Initialize Master (using Flannel)

```
> sudo kubeadm init --config=kubeadm-config.yaml
```

```
[init] Using Kubernetes version: v1.13.4
[preflight] Running pre-flight checks
...
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 k8slb.ott.dev.intra:6443 --token ktemf3.pshisb9lspt1i40i --discovery-token-ca-cert-hash sha256:
1e737466a59f00083a4ddf43c9fcf446a5b1cee8346afd1565d341fe5dee2c46
```

Record the kubeadm join command!

As your non root user:

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

Verify that your network is on the right network interface

```
kubectl get pods -o wide --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
NOMINATED	NODE						
	READINESS GATES						
kube-system	coredns-86c58d9df4-8zk5t	0/1	Pending	0	2d3h	<none>	<none>
<none>	<none>						
kube-system	coredns-86c58d9df4-tsftk	0/1	Pending	0	2d3h	<none>	<none>
<none>	<none>						
kube-system	etcd-k8master	1/1	Running	1	2d3h	10.0.3.15	k8master
<none>	<none>						
kube-system	kube-apiserver-k8master	1/1	Running	1	2d3h	10.0.3.15	k8master
<none>	<none>						
kube-system	kube-controller-manager-k8master	1/1	Running	1	2d3h	10.0.3.15	k8master
<none>	<none>						
kube-system	kube-proxy-88gdq	1/1	Running	1	2d3h	10.0.3.15	k8master
<none>	<none>						
kube-system	kube-scheduler-k8master	1/1	Running	1	2d3h	10.0.3.15	k8master
<none>	<none>						

Install Flannel Network Plugin

```
> sudo sysctl net.bridge.bridge-nf-call-iptables=1
```

```
> kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/bc79dd1505b0c8681e4e4c0d86c5cd2643275/Documentation/kube-flannel.yml
```

NOTE: See <https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/> for details on the various plugins and their setup.

Verify that all of your kubernetes pods are running

```
> kubectl get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-86c58d9df4-8zk5t	1/1	Running	0	47h
kube-system	coredns-86c58d9df4-tsftk	1/1	Running	0	47h
kube-system	etcd-k8master	1/1	Running	1	47h
kube-system	kube-apiserver-k8master	1/1	Running	1	47h
kube-system	kube-controller-manager-k8master	1/1	Running	1	47h
kube-system	kube-flannel-ds-amd64-fl5wp	1/1	Running	0	12s
kube-system	kube-proxy-88gdq	1/1	Running	1	47h
kube-system	kube-scheduler-k8master	1/1	Running	1	47h

Copy Certificates to Other Master Nodes

```
> sudo su
```

```
> vi copyCertsToMasters.sh
```

copyCertsToMasters.sh

```
USER=test # customizable
CONTROL_PLANE_IPS="172.20.233.182 172.20.233.183"
for host in ${CONTROL_PLANE_IPS}; do
    scp /etc/kubernetes/pki/ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/sa.key "${USER}"@$host:
    scp /etc/kubernetes/pki/sa.pub "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/etcd/ca.crt "${USER}"@$host:etcd-ca.crt
    scp /etc/kubernetes/pki/etcd/ca.key "${USER}"@$host:etcd-ca.key
    scp /etc/kubernetes/admin.conf "${USER}"@$host:
done
```

> chmod +x copyCertsToMasters.sh

> ./copyCertsToMasters.sh

Setup Other Master Nodes

Perform the following steps on the other master nodes (k8smaster2 and k8smaster3).

Set the hostname

> sudo su

> hostnamectl set-hostname k8smaster1

> hostnamectl

> reboot

Move Certificates

> ssh test@<ip of master node>

> sudo su

> vi moveFilesFromMaster.sh

moveFilesFromMaster.sh

```
USER=test # customizable
mkdir -p /etc/kubernetes/pki/etcd
mv /home/${USER}/ca.crt /etc/kubernetes/pki/
mv /home/${USER}/ca.key /etc/kubernetes/pki/
mv /home/${USER}/sa.pub /etc/kubernetes/pki/
mv /home/${USER}/sa.key /etc/kubernetes/pki/
mv /home/${USER}/front-proxy-ca.crt /etc/kubernetes/pki/
mv /home/${USER}/front-proxy-ca.key /etc/kubernetes/pki/
mv /home/${USER}/etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt
mv /home/${USER}/etcd-ca.key /etc/kubernetes/pki/etcd/ca.key
mv /home/${USER}/admin.conf /etc/kubernetes/admin.conf
```

> chmod +x moveFilesFromMaster.sh

> ./moveFilesFromMaster.sh

Join the Cluster

> sudo su

```
> kubeadm join k8slb.ott.dev.intra:6443 --token ktemf3.pshisb9lspt1i40i --discovery-token-ca-cert-hash sha256:1e737466a59f00083a4ddf43c9fcf446a5b1cee8346afd1565d341fe5dee2c46 --experimental-control-plane
```

Notice the addition of the `--experimental-control-plane` flag. This flag automates joining this control plane node to the cluster.

As your non root user:

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

Verify that the node has joined the cluster:

From the k8smaster1 under the test account issue the following command:

```
> kubectl get nodes
```

Setup Worker Nodes

Perform the following steps on all of the worker nodes (k8sworker1, k8sworker2, k8sworker3).

Set the hostname

```
> sudo su
> hostnamectl set-hostname k8sworker1
> hostnamectl
> reboot
```

Join the Cluster

From the worker nodes issue the following command. If your token has expired from you may need to create a new one.

```
> sudo su
> kubeadm join k8slb.ott.dev.intra:6443 --token ktemf3.pshisb9lspt1i40i --discovery-token-ca-cert-hash sha256:1e737466a59f00083a4ddf43c9fcf446a5b1cee8346afd1565d341fe5dee2c46
```

Verify that the nodes have been added by issuing the following command at the master:

```
> kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
k8smaster1	Ready	master	27m	v1.13.4
k8smaster2	Ready	master	8m43s	v1.13.4
k8smaster3	Ready	master	7m56s	v1.13.4
k8sworker1	Ready	<none>	71s	v1.13.4
k8sworker2	Ready	<none>	63s	v1.13.4
k8sworker3	Ready	<none>	25s	v1.13.4

Install Dashboard

From the master node:

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

secret/kubernetes-dashboard-certs created
serviceaccount/kubernetes-dashboard created
role.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
rolebinding.rbac.authorization.k8s.io/kubernetes-dashboard-minimal created
deployment.apps/kubernetes-dashboard created
service/kubernetes-dashboard created
```

Create a dashboard admin user:

vi dashboard-adminuser.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kube-system
```

> kubectl apply -f dashboard-adminuser.yaml

```
serviceaccount/admin-user created
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
```

Get the access token

```
> kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep admin-user | awk '{print $1}')
```

```
Name:          admin-user-token-s6x6d
Namespace:     kube-system
Labels:        <none>
Annotations:   kubernetes.io/service-account.name: admin-user
                kubernetes.io/service-account.uid: 7f55a022-1a05-11e9-a8cb-6c3be541582b

Type:  kubernetes.io/service-account-token

Data
====
namespace:  11 bytes
token:       eyJhbGciOiJIUzU1IiwiaWUiOiJrdWJlLXN5
eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJ2ZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlLXN5
c3RlbnSiOiJmVmVmbV0ZXMuaW8vc2Vydm1jZWZjY291bnQvc2VjcmV0Lm5hbWUiOiJhZGlpb11lc2VyLXRva2VuLXM2eDZkIiwia3ViZXJ2ZXRl
cy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFmFkblwLXVzZXIiLCJrdWJlcm5ldGVzLm1vL3NlcnZpY2VhY2NvdW50
L3NlcnZpY2UtYWNjb3VudC5laWQioiI3ZjU1YTAYMi0xYTA1LTExZTktYThjYi02YzNiZTU0MTU4MmIiLCJ2ZW50Iiwia3VudC9uYW1lc3BhY2UiOiJzeXN0ZW06c2Vydm1jZWZj
Y291bnQ6a3ViZS1zeXN0ZW06YWRtaW4tdXNlcjI9LjBtoYxamkG_dww0sht85C_txkXr6v0rwiSzwkyGT1livj9MIgrvBG0eF044VOihhWygjv-
lJQOpbFk3TnbDE4QWUr9aFyiQZo_qQGgqSj_NBKWuDw3N0Ktjk_siB_3Zb4tOCGe9i0iJ3zwYxDlh5eAD_-YiY-
yU8QKY85pJPbTqR4USWiHh4Saj4HEg0TM0EUIToCqF-
u9qEF0Y5ldc17VW3NKECEIfarP4x893yzYRnRrbk8ItgJqqln6oQ2AH49dow28fUiGyUCculdazXrOr0A_j-Lg-
5KaVt6dXIU5SMMLrhzrdrmrK9akbEP3AdKRERvNosVR_afWdOo8lByA
ca.crt:     1025 bytes
```

Access the Dashboard by using the API proxy

From your local machine:

```
> ssh -L 8001:127.0.0.1:8001 <USER>@<IP>
```

```
> kubectl proxy
```

Browse to:

<http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>

Access the Dashboard using port forwarding

From your local machine:

```
> ssh -L 8443:127.0.0.1:8443 <USER>@<IP>
```

```
> kubectl port-forward $(kubectl -n kube-system get pods |grep kubernetes-dashboard |awk '{print $1}') 8443:8443 --namespace=kube-system
```

Browse to:

<https://localhost:8443>

Sign in to the Dashboard

Sign in using the token previously retrieved.

Kubernetes Dashboard

☐ Kubeconfig

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the [Configure Access to Multiple Clusters](#) section.

☒ Token

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the [Authentication](#) section.

[Enter token](#)

.....

SIGN IN

SKIP

The screenshot shows the Kubernetes Dashboard Overview page. The left sidebar contains a navigation menu with the following items: Overview (selected), Namespaces, Nodes, Persistent Volumes, Roles, Storage Classes, Workloads (Cron Jobs, Daemon Sets, Deployments, Jobs, Pods, Replica Sets, Replication Controllers, Stateful Sets), Discovery and Load Balancing (Ingresses, Services), Config and Storage (Config Maps, Persistent Volume Claims, Secrets). The main content area is divided into two sections: 'Discovery and Load Balancing' and 'Config and Storage'. The 'Discovery and Load Balancing' section contains a 'Services' table with one entry: 'kubernetes' (component: provider.ku, Cluster IP: 10.96.0.1, Internal endpoints: kubernetes:..., Age: 6 hours). The 'Config and Storage' section contains a 'Secrets' table with one entry: 'default-token-nnmbm' (Type: kubernetes.io/service-account..., Age: 6 hours).

Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
kubernetes	component: provider.ku	10.96.0.1	kubernetes:...	-	6 hours

Name	Type	Age
default-token-nnmbm	kubernetes.io/service-account...	6 hours

Install Sample Pod

```
> vi nginx-example.yaml
```

```

apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 3 # tells deployment to run 3 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - port: 80
      nodePort: 31080
      name: nginx

```

> kubectl apply -f nginx-example.yaml

> kubectl get services

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	5h32m
nginx	NodePort	10.99.190.114	<none>	80:31080/TCP	2m46s

From the above we can see that the nginx service is exposed on port 31080.

Verify by issuing the following command:

> curl http://<NODE_IP>:31080

<http://172.20.233.184:31080/>

Troubleshooting

Reset and start all over

> sudo kubeadm reset

Centos Commands

Cmd	Description
hostname -I	Get the ip address
ip addr	ifconfig

References

Reference	URL
Install and configure a multi-master Kubernetes cluster with kubeadm	https://blog.inkubate.io/install-and-configure-a-multi-master-kubernetes-cluster-with-kubeadm/
High Availability Cluster	https://kubernetes.io/docs/setup/independent/high-availability/
Building a Kubernetes Cluster	https://medium.com/@KevinHoffman/building-a-kubernetes-cluster-in-virtualbox-with-ubuntu-22cd338846dd
Cluster Networking	https://kubernetes.io/docs/concepts/cluster-administration/networking/
Creating a single master cluster with kubeadm	https://kubernetes.io/docs/setup/independent/create-cluster-kubeadm/
Kubernetes Network Plugins	https://kubedex.com/kubernetes-network-plugins/
Flannel	https://github.com/coreos/flannel#flannel
Dashboard	https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/#using-dashboard
Adding Users to CentOS	https://www.rosehosting.com/blog/how-to-create-a-sudo-user-on-centos-7/