**Install Kubernetes (K8s) Offline on CentOS 7**

[Kubernetes (K8s)](https://kubernetes.io/" \t "_blank" \o "Kubernetes (K8s)) uses a containerization platform like [Docker](https://www.docker.com/), [containerd](https://containerd.io/" \o "containerd" \t "_blank), etc. and requires a **Registry** to download and use Docker images. [Docker Hub](https://hub.docker.com/) is the global public registry that serves the purpose. However, there are situations, when we want to use Kubernetes (K8s) in a private network. In such a situation, we cannot access Docker Hub, therefore, we must configure a Private Docker Registry for our Kubernetes (K8s) cluster.

In this article, we will **install Kubernetes (K8s) offline on CentOS 7**. We are not configuring a Private Docker Registry here, but you can read our following articles to configure it by yourself.

- [Configure Secure Registry with Docker-Distribution on CentOS 7](https://ahmermansoor.blogspot.com/2019/04/configure-secure-registry-docker-distribution-centos-7.html" \o "Configure Secure Registry with Docker-Distribution on CentOS 7)  
- [Configure a Private Docker Registry on CentOS 7](https://ahmermansoor.blogspot.com/2019/03/configure-private-docker-registry-centos-7.html" \o "Configure a Private Docker Registry on CentOS 7)

For understanding Kubernetes (K8s) concepts and use it in your environment, we recommend that you should read [Kubernetes in Action](https://amzn.to/2V9o9sF" \t "_blank" \o "Kubernetes in Action) by Marko Luksa.

**System Specification:**

We have configured two CentOS 7 virtual machines.

|  |  |  |
| --- | --- | --- |
| **Hostname:** | docker-online.example.com | docker-offline.example.com |
| **Operating System:** | CentOS 7.6 | CentOS 7.6 |
| **Internet:** | Yes | No |
| **Docker Version:** | Docker CE 18.09 | Docker CE 18.09 |

**Installing Docker Offline on CentOS 7:**

We have already written a complete article [Install Docker CE on an Offline CentOS 7 Machine](https://ahmermansoor.blogspot.com/2019/02/install-docker-ce-on-offline-centos-7-machine.html" \o "Install Docker CE on an Offline CentOS 7 Machine). Therefore, it is advised that, you should follow that article to install **Docker CE** before **Kubernetes (K8s)** installation on both machines.

We are also required to install Docker CE on docker-online.example.com, because we will pull required images from **Docker Hub** using **docker** command.

Connect with docker-offline.example.com using **ssh** as **root** user.

After installing Docker CE, we must configure it to use with **Kubernetes (K8s)**.

[root@docker-offline ~]# mkdir /etc/docker

[root@docker-offline ~]# cat > /etc/docker/daemon.json << EOF

> {

> "exec-opts": ["native.cgroupdriver=systemd"],

> "log-driver": "json-file",

> "log-opts": {

> "max-size": "100m"

> },

> "storage-driver": "overlay2",

> "storage-opts": [

> "overlay2.override\_kernel\_check=true"

> ]

> }

> EOF

Restart **docker.service**.

[root@docker-offline ~]# systemctl restart docker.service

**Downloading Packages/Images for Offline Installation of Kubernetes (K8s):**

Connect with docker-online.example.com using **ssh** as **root** user.

Add **Kubernetes (K8s)** yum repository as follows:

[root@docker-online k8s]# cat > /etc/yum.repos.d/kubernetes.repo << EOF

> [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

Build yum cache.

[root@docker-online ~]# yum makecache fast

Loaded plugins: fastestmirror

Loading mirror speeds from cached hostfile

epel/x86\_64/metalink | 6.7 kB 00:00

\* base: mirrors.ges.net.pk

\* epel: mirror1.ku.ac.th

\* extras: mirrors.ges.net.pk

\* updates: mirrors.ges.net.pk

base | 3.6 kB 00:00

docker-ce-nightly | 3.5 kB 00:00

docker-ce-stable | 3.5 kB 00:00

extras | 3.4 kB 00:00

kubernetes/signature | 454 B 00:00

Retrieving key from https://packages.cloud.google.com/yum/doc/yum-key.gpg

Importing GPG key 0xA7317B0F:

Userid : "Google Cloud Packages Automatic Signing Key <gc-team@google.com>"

Fingerprint: d0bc 747f d8ca f711 7500 d6fa 3746 c208 a731 7b0f

From : https://packages.cloud.google.com/yum/doc/yum-key.gpg

Is this ok [y/N]: y

Retrieving key from https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

kubernetes/signature | 1.4 kB 00:06 !!!

updates | 3.4 kB 00:00

kubernetes/primary | 47 kB 00:03

kubernetes 342/342

Metadata Cache Created

Create a directory to download required **Kubernetes (K8s)** packages.

[root@docker-online ~]# mkdir ~/k8s

[root@docker-online k8s]# cd ~/k8s

Download Kubernetes (K8s) packages using **yumdownloader**.

[root@docker-online k8s]# yumdownloader --resolve kubelet kubeadm kubectl

Loaded plugins: fastestmirror

Loading mirror speeds from cached hostfile

\* base: mirrors.ges.net.pk

\* epel: kartolo.sby.datautama.net.id

\* extras: mirrors.ges.net.pk

\* updates: mirrors.ges.net.pk

--> Running transaction check

---> Package kubeadm.x86\_64 0:1.14.1-0 will be installed

--> Processing Dependency: kubernetes-cni >= 0.7.5 for package: kubeadm-1.14.1-0.x86\_64

--> Processing Dependency: cri-tools >= 1.11.0 for package: kubeadm-1.14.1-0.x86\_64

---> Package kubectl.x86\_64 0:1.14.1-0 will be installed

---> Package kubelet.x86\_64 0:1.14.1-0 will be installed

--> Processing Dependency: socat for package: kubelet-1.14.1-0.x86\_64

--> Processing Dependency: conntrack for package: kubelet-1.14.1-0.x86\_64

--> Running transaction check

---> Package conntrack-tools.x86\_64 0:1.4.4-4.el7 will be installed

--> Processing Dependency: libnetfilter\_cttimeout.so.1(LIBNETFILTER\_CTTIMEOUT\_1.1)(64bit) for package: conntrack-tools-1.4.4-4.el7.x86\_64

--> Processing Dependency: libnetfilter\_cttimeout.so.1(LIBNETFILTER\_CTTIMEOUT\_1.0)(64bit) for package: conntrack-tools-1.4.4-4.el7.x86\_64

--> Processing Dependency: libnetfilter\_cthelper.so.0(LIBNETFILTER\_CTHELPER\_1.0)(64bit) for package: conntrack-tools-1.4.4-4.el7.x86\_64

--> Processing Dependency: libnetfilter\_queue.so.1()(64bit) for package: conntrack-tools-1.4.4-4.el7.x86\_64

--> Processing Dependency: libnetfilter\_cttimeout.so.1()(64bit) for package: conntrack-tools-1.4.4-4.el7.x86\_64

--> Processing Dependency: libnetfilter\_cthelper.so.0()(64bit) for package: conntrack-tools-1.4.4-4.el7.x86\_64

---> Package cri-tools.x86\_64 0:1.12.0-0 will be installed

---> Package kubernetes-cni.x86\_64 0:0.7.5-0 will be installed

---> Package socat.x86\_64 0:1.7.3.2-2.el7 will be installed

--> Running transaction check

---> Package libnetfilter\_cthelper.x86\_64 0:1.0.0-9.el7 will be installed

---> Package libnetfilter\_cttimeout.x86\_64 0:1.0.0-6.el7 will be installed

---> Package libnetfilter\_queue.x86\_64 0:1.0.2-2.el7\_2 will be installed

--> Finished Dependency Resolution

(1/10): conntrack-tools-1.4.4-4.el7.x86\_64.rpm | 186 kB 00:03

warning: /root/k8s/53edc739a0e51a4c17794de26b13ee5df939bd3161b37f503fe2af8980b41a89-cri-tools-1.12.0-0.x86\_64.rpm: Header V4 RSA/SHA512 Signature, key ID 3e1ba8d5: NOKEY

Public key for 53edc739a0e51a4c17794de26b13ee5df939bd3161b37f503fe2af8980b41a89-cri-tools-1.12.0-0.x86\_64.rpm is not installed

(2/10): 53edc739a0e51a4c17794de26b13ee5df939bd3161b37f503f | 4.2 MB 00:21

(3/10): 9e1af74c18311f2f6f8460dbd1aa3e02911105bfd455416381 | 8.7 MB 00:45

(4/10): 5c6cb3beb5142fa21020e2116824ba77a2d1389a3321601ea5 | 9.5 MB 00:42

(5/10): libnetfilter\_cthelper-1.0.0-9.el7.x86\_64.rpm | 18 kB 00:02

(6/10): libnetfilter\_cttimeout-1.0.0-6.el7.x86\_64.rpm | 18 kB 00:03

(7/10): libnetfilter\_queue-1.0.2-2.el7\_2.x86\_64.rpm | 23 kB 00:03

(8/10): socat-1.7.3.2-2.el7.x86\_64.rpm | 290 kB 00:02

(9/10): 548a0dcd865c16a50980420ddfa5fbccb8b59621179798e6dc | 10 MB 00:46

(10/10): e1e8f430609698d7ec87642179ab57605925cb9aa48d406da | 23 MB 01:27

List downloaded files.

[root@docker-online k8s]# ls

53edc739a0e51a4c17794de26b13ee5df939bd3161b37f503fe2af8980b41a89-cri-tools-1.12.0-0.x86\_64.rpm

548a0dcd865c16a50980420ddfa5fbccb8b59621179798e6dc905c9bf8af3b34-kubernetes-cni-0.7.5-0.x86\_64.rpm

5c6cb3beb5142fa21020e2116824ba77a2d1389a3321601ea53af5ceefe70ad1-kubectl-1.14.1-0.x86\_64.rpm

9e1af74c18311f2f6f8460dbd1aa3e02911105bfd455416381e995d8172a0a01-kubeadm-1.14.1-0.x86\_64.rpm

conntrack-tools-1.4.4-4.el7.x86\_64.rpm

e1e8f430609698d7ec87642179ab57605925cb9aa48d406da97dedfb629bebf2-kubelet-1.14.1-0.x86\_64.rpm

libnetfilter\_cthelper-1.0.0-9.el7.x86\_64.rpm

libnetfilter\_cttimeout-1.0.0-6.el7.x86\_64.rpm

libnetfilter\_queue-1.0.2-2.el7\_2.x86\_64.rpm

socat-1.7.3.2-2.el7.x86\_64.rpm

Download Docker images from **Docker Hub**, as required by Kubernetes (K8s) for node initialization.

[root@docker-online ~]# docker pull k8s.gcr.io/kube-apiserver:v1.14.1

v1.14.1: Pulling from kube-apiserver

346aee5ea5bc: Pull complete

7f0e834d5a94: Pull complete

Digest: sha256:bb3e3264bf74cc6929ec05b494d95b7aed9ee1e5c1a5c8e0693b0f89e2e7288e

Status: Downloaded newer image for k8s.gcr.io/kube-apiserver:v1.14.1

[root@docker-online ~]# docker pull k8s.gcr.io/kube-controller-manager:v1.14.1

v1.14.1: Pulling from kube-controller-manager

346aee5ea5bc: Already exists

f4db69ee8ade: Pull complete

Digest: sha256:5279e0030094c0ef2ba183bd9627e91e74987477218396bd97a5e070df241df5

Status: Downloaded newer image for k8s.gcr.io/kube-controller-manager:v1.14.1

[root@docker-online ~]# docker pull k8s.gcr.io/kube-scheduler:v1.14.1

v1.14.1: Pulling from kube-scheduler

346aee5ea5bc: Already exists

b88909b8f99f: Pull complete

Digest: sha256:11af0ae34bc63cdc78b8bd3256dff1ba96bf2eee4849912047dee3e420b52f8f

Status: Downloaded newer image for k8s.gcr.io/kube-scheduler:v1.14.1

[root@docker-online ~]# docker pull k8s.gcr.io/kube-proxy:v1.14.1

v1.14.1: Pulling from kube-proxy

346aee5ea5bc: Already exists

1e695dec1fee: Pull complete

100690d61cf6: Pull complete

Digest: sha256:44af2833c6cbd9a7fc2e9d2f5244a39dfd2e31ad91bf9d4b7d810678db738ee9

Status: Downloaded newer image for k8s.gcr.io/kube-proxy:v1.14.1

[root@docker-online ~]# docker pull k8s.gcr.io/pause:3.1

3.1: Pulling from pause

67ddbfb20a22: Pull complete

Digest: sha256:f78411e19d84a252e53bff71a4407a5686c46983a2c2eeed83929b888179acea

Status: Downloaded newer image for k8s.gcr.io/pause:3.1

[root@docker-online ~]# docker pull k8s.gcr.io/etcd:3.3.10

3.3.10: Pulling from etcd

90e01955edcd: Pull complete

6369547c492e: Pull complete

bd2b173236d3: Pull complete

Digest: sha256:17da501f5d2a675be46040422a27b7cc21b8a43895ac998b171db1c346f361f7

Status: Downloaded newer image for k8s.gcr.io/etcd:3.3.10

[root@docker-online ~]# docker pull k8s.gcr.io/coredns:1.3.1

1.3.1: Pulling from coredns

e0daa8927b68: Pull complete

3928e47de029: Pull complete

Digest: sha256:02382353821b12c21b062c59184e227e001079bb13ebd01f9d3270ba0fcbf1e4

Status: Downloaded newer image for k8s.gcr.io/coredns:1.3.1

List Docker images.

[root@docker-online ~]# docker image ls -a

REPOSITORY TAG IMAGE ID CREATED SIZE

k8s.gcr.io/kube-proxy v1.14.1 20a2d7035165 2 weeks ago 82.1MB

k8s.gcr.io/kube-apiserver v1.14.1 cfaa4ad74c37 2 weeks ago 210MB

k8s.gcr.io/kube-controller-manager v1.14.1 efb3887b411d 2 weeks ago 158MB

k8s.gcr.io/kube-scheduler v1.14.1 8931473d5bdb 2 weeks ago 81.6MB

k8s.gcr.io/coredns 1.3.1 eb516548c180 3 months ago 40.3MB

k8s.gcr.io/etcd 3.3.10 2c4adeb21b4f 4 months ago 258MB

k8s.gcr.io/pause 3.1 da86e6ba6ca1 16 months ago 742kB

Export Kubernetes (K8s) related Docker images to individual **tar** files.

[root@docker-online ~]# docker save k8s.gcr.io/kube-apiserver:v1.14.1 > ~/k8s/kube-apiserver.tar

[root@docker-online ~]# docker save k8s.gcr.io/kube-controller-manager:v1.14.1 > ~/k8s/kube-controller-manager.tar

[root@docker-online ~]# docker save k8s.gcr.io/kube-scheduler:v1.14.1 > ~/k8s/kube-scheduler.tar

[root@docker-online ~]# docker save k8s.gcr.io/kube-proxy:v1.14.1 > ~/k8s/kube-proxy.tar

[root@docker-online ~]# docker save k8s.gcr.io/pause:3.1 > ~/k8s/pause.tar

[root@docker-online ~]# docker save k8s.gcr.io/etcd:3.3.10 > ~/k8s/etcd.tar

[root@docker-online ~]# docker save k8s.gcr.io/coredns:1.3.1 > ~/k8s/coredns.tar

List tar files.

[root@docker-online ~]# ls ~/k8s/\*.tar

/root/k8s/coredns.tar /root/k8s/kube-proxy.tar

/root/k8s/etcd.tar /root/k8s/kube-scheduler.tar

/root/k8s/kube-apiserver.tar /root/k8s/pause.tar

/root/k8s/kube-controller-manager.tar

Download **Flannel** Network script.

[root@docker-online ~]# cd ~/k8s

[root@docker-online k8s]# wget https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

--2019-04-27 20:09:16-- https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.8.133

Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.8.133|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 12306 (12K) [text/plain]

Saving to: âkube-flannel.ymlâ

100%[======================================>] 12,306 --.-K/s in 0.006s

2019-04-27 20:09:18 (2.11 MB/s) - âkube-flannel.ymlâ saved [12306/12306]

We have successfully downloaded all required files for **Kubernetes (K8s)** installation.

Transfer the directory ~/k8s from docker-online to docker-offline.

**Configuring Prerequisites for Kubernetes (K8s):**

Connect with docker-offline.example.com using **ssh** as **root** user.

Set Kernel parameter as required by Kubernetes (K8s).

[root@docker-offline ~]# cat > /etc/sysctl.d/kubernetes.conf << EOF

> net.ipv4.ip\_forward = 1

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

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

> EOF

Reload Kernel parameter configuration files.

[root@docker-offline ~]# modprobe br\_netfilter

[root@docker-offline ~]# sysctl --system

\* Applying /usr/lib/sysctl.d/00-system.conf ...

net.bridge.bridge-nf-call-ip6tables = 0

net.bridge.bridge-nf-call-iptables = 0

net.bridge.bridge-nf-call-arptables = 0

\* Applying /usr/lib/sysctl.d/10-default-yama-scope.conf ...

kernel.yama.ptrace\_scope = 0

\* Applying /usr/lib/sysctl.d/50-default.conf ...

kernel.sysrq = 16

kernel.core\_uses\_pid = 1

net.ipv4.conf.default.rp\_filter = 1

net.ipv4.conf.all.rp\_filter = 1

net.ipv4.conf.default.accept\_source\_route = 0

net.ipv4.conf.all.accept\_source\_route = 0

net.ipv4.conf.default.promote\_secondaries = 1

net.ipv4.conf.all.promote\_secondaries = 1

fs.protected\_hardlinks = 1

fs.protected\_symlinks = 1

\* Applying /etc/sysctl.d/99-sysctl.conf ...

\* Applying /etc/sysctl.d/kubernetes.conf ...

net.ipv4.ip\_forward = 1

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

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

\* Applying /etc/sysctl.conf ...

Turn off Swap for Kubernetes (K8s) installation.

[root@docker-offline ~]# swapoff -a

[root@docker-offline ~]# sed -e '/swap/s/^/#/g' -i /etc/fstab

Allow Kubernetes (K8s) service ports in Linux firewall.

For Master node

[root@docker-offline ~]# firewall-cmd --permanent --add-port={6443,2379,2380,10250,10251,10252}/tcp

success

For Worker nodes

[root@docker-offline ~]# firewall-cmd --permanent --add-port={10250,30000-32767}/tcp

success

Reload firewall configurations.

[root@docker-offline ~]# firewall-cmd --reload

success

Switch SELinux to **Permissive** mode using following commands.

[root@docker-offline ~]# setenforce 0

[root@docker-offline ~]# sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config

**Installing Kubernetes (K8s) Offline on CentOS 7:**

Now, install Kubernetes (K8s) packages from ~/k8s directory using **rpm** command.

[root@docker-offline ~]# rpm -ivh --replacefiles --replacepkgs ~/k8s/\*.rpm

warning: 53edc739a0e51a4c17794de26b13ee5df939bd3161b37f503fe2af8980b41a89-cri-tools-1.12.0-0.x86\_64.rpm: Header V4 RSA/SHA512 Signature, key ID 3e1ba8d5: NOKEY

Preparing... ################################# [100%]

Updating / installing...

1:socat-1.7.3.2-2.el7 ################################# [ 10%]

2:libnetfilter\_queue-1.0.2-2.el7\_2 ################################# [ 20%]

3:libnetfilter\_cttimeout-1.0.0-6.el################################# [ 30%]

4:libnetfilter\_cthelper-1.0.0-9.el7################################# [ 40%]

5:conntrack-tools-1.4.4-4.el7 ################################# [ 50%]

6:kubelet-1.14.1-0 ################################# [ 60%]

7:kubernetes-cni-0.7.5-0 ################################# [ 70%]

8:kubectl-1.14.1-0 ################################# [ 80%]

9:cri-tools-1.12.0-0 ################################# [ 90%]

10:kubeadm-1.14.1-0 ################################# [100%]

Enable bash completion for **kubectl**.

[root@docker-offline ~]# source <(kubectl completion bash)

[root@docker-offline ~]# kubectl completion bash > /etc/bash\_completion.d/kubectl

Import the **tar** files of docker images into Docker.

[root@docker-offline ~]# docker load < ~/k8s/coredns.tar

fb61a074724d: Loading layer 479.7kB/479.7kB

c6a5fc8a3f01: Loading layer 40.05MB/40.05MB

Loaded image: k8s.gcr.io/coredns:1.3.1

[root@docker-offline ~]# docker load < ~/k8s/kube-proxy.tar

5ba3be777c2d: Loading layer 43.88MB/43.88MB

0b8d2e946c93: Loading layer 3.403MB/3.403MB

8b9a8fc88f0d: Loading layer 36.69MB/36.69MB

Loaded image: k8s.gcr.io/kube-proxy:v1.14.1

[root@docker-offline ~]# docker load < ~/k8s/etcd.tar

8a788232037e: Loading layer 1.37MB/1.37MB

30796113fb51: Loading layer 232MB/232MB

6fbfb277289f: Loading layer 24.98MB/24.98MB

Loaded image: k8s.gcr.io/etcd:3.3.10

[root@docker-offline ~]# docker load < ~/k8s/kube-scheduler.tar

e04ef32df86e: Loading layer 39.26MB/39.26MB

Loaded image: k8s.gcr.io/kube-scheduler:v1.14.1

[root@docker-offline ~]# docker load < ~/k8s/kube-apiserver.tar

97f70f3a7a0c: Loading layer 167.6MB/167.6MB

Loaded image: k8s.gcr.io/kube-apiserver:v1.14.1

[root@docker-offline ~]# docker load < ~/k8s/pause.tar

e17133b79956: Loading layer 744.4kB/744.4kB

Loaded image: k8s.gcr.io/pause:3.1

[root@docker-offline ~]# docker load < ~/k8s/kube-controller-manager.tar

d8ca6e1aa16e: Loading layer 115.6MB/115.6MB

Loaded image: k8s.gcr.io/kube-controller-manager:v1.14.1

Now, we have all the required Docker images in local registry. Therefore, we can configure this CentOS 7 machine either as **Kubernetes (K8s)** master or worker node.

We don't have configured any master node. Therefore, we have to configure a master node first.

[root@docker-offline ~]# kubeadm init

I0427 20:36:47.088327 18015 version.go:96] could not fetch a Kubernetes version from the internet: unable to get URL "https://dl.k8s.io/release/stable-1.txt": Get https://dl.k8s.io/release/stable-1.txt: net/http: request canceled while waiting for connection (Client.Timeout exceeded while awaiting headers)

I0427 20:36:47.090078 18015 version.go:97] falling back to the local client version: v1.14.1

[init] Using Kubernetes version: v1.14.1

[preflight] Running pre-flight checks

[WARNING Firewalld]: firewalld is active, please ensure ports [6443 10250] are open or your cluster may not function correctly

[WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'

[preflight] Pulling images required for setting up a Kubernetes cluster

[preflight] This might take a minute or two, depending on the speed of your internet connection

[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'

[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"

[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"

[kubelet-start] Activating the kubelet service

[certs] Using certificateDir folder "/etc/kubernetes/pki"

[certs] Generating "front-proxy-ca" certificate and key

[certs] Generating "front-proxy-client" certificate and key

[certs] Generating "etcd/ca" certificate and key

[certs] Generating "etcd/server" certificate and key

[certs] etcd/server serving cert is signed for DNS names [docker-offline.example.com localhost] and IPs [192.168.116.159 127.0.0.1 ::1]

[certs] Generating "apiserver-etcd-client" certificate and key

[certs] Generating "etcd/peer" certificate and key

[certs] etcd/peer serving cert is signed for DNS names [docker-offline.example.com localhost] and IPs [192.168.116.159 127.0.0.1 ::1]

[certs] Generating "etcd/healthcheck-client" certificate and key

[certs] Generating "ca" certificate and key

[certs] Generating "apiserver" certificate and key

[certs] apiserver serving cert is signed for DNS names [docker-offline.example.com kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs [10.96.0.1 192.168.116.159]

[certs] Generating "apiserver-kubelet-client" certificate and key

[certs] Generating "sa" key and public key

[kubeconfig] Using kubeconfig folder "/etc/kubernetes"

[kubeconfig] Writing "admin.conf" kubeconfig file

[kubeconfig] Writing "kubelet.conf" kubeconfig file

[kubeconfig] Writing "controller-manager.conf" kubeconfig file

[kubeconfig] Writing "scheduler.conf" kubeconfig file

[control-plane] Using manifest folder "/etc/kubernetes/manifests"

[control-plane] Creating static Pod manifest for "kube-apiserver"

[control-plane] Creating static Pod manifest for "kube-controller-manager"

[control-plane] Creating static Pod manifest for "kube-scheduler"

[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"

[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods from directory "/etc/kubernetes/manifests". This can take up to 4m0s

[apiclient] All control plane components are healthy after 27.538706 seconds

[upload-config] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace

[kubelet] Creating a ConfigMap "kubelet-config-1.14" in namespace kube-system with the configuration for the kubelets in the cluster

[upload-certs] Skipping phase. Please see --experimental-upload-certs

[mark-control-plane] Marking the node docker-offline.example.com as control-plane by adding the label "node-role.kubernetes.io/master=''"

[mark-control-plane] Marking the node docker-offline.example.com as control-plane by adding the taints [node-role.kubernetes.io/master:NoSchedule]

[bootstrap-token] Using token: 6e4ntu.a5r1md9vuqex4pe8

[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles

[bootstrap-token] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials

[bootstrap-token] configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token

[bootstrap-token] configured RBAC rules to allow certificate rotation for all node client certificates in the cluster

[bootstrap-token] creating the "cluster-info" ConfigMap in the "kube-public" namespace

[addons] Applied essential addon: CoreDNS

[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane 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/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 192.168.116.159:6443 --token 6e4ntu.a5r1md9vuqex4pe8 \

--discovery-token-ca-cert-hash sha256:19f4d9f6d433cc12addb70e2737c629213777deed28fa5dcc33f9d05d2382d5b

Execute suggested script.

[root@docker-offline ~]# mkdir -p $HOME/.kube

[root@docker-offline ~]# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

[root@docker-offline ~]# sudo chown $(id -u):$(id -g) $HOME/.kube/config

Start and enable **kubelet.service**.

[root@docker-offline ~]# systemctl enable kubelet.service

Created symlink from /etc/systemd/system/multi-user.target.wants/kubelet.service to /usr/lib/systemd/system/kubelet.service.

[root@docker-offline ~]# systemctl start kubelet.service

Add **Flannel** network.

[root@docker-offline ~]# kubectl apply -f ~/k8s/kube-flannel.yml

podsecuritypolicy.extensions/psp.flannel.unprivileged created

clusterrole.rbac.authorization.k8s.io/flannel created

clusterrolebinding.rbac.authorization.k8s.io/flannel created

configmap/kube-flannel-cfg created

daemonset.extensions/kube-flannel-ds-amd64 created

daemonset.extensions/kube-flannel-ds-arm64 created

daemonset.extensions/kube-flannel-ds-arm created

daemonset.extensions/kube-flannel-ds-ppc64le created

daemonset.extensions/kube-flannel-ds-s390x created

List nodes in Kubernetes (K8s) cluster.

[root@docker-offline ~]# kubectl get nodes

NAME STATUS ROLES AGE VERSION

docker-offline.example.com NotReady master 5m9s v1.14.1

We have successfully installed Kubernetes (K8s) offline on CentOS 7.