Docker and Kubernetes offline installation in RHEL7

Setup Kubernetes Cluster on Red Hat Enterprise Linux 7 - Offline

Pre-requisites

- Online Machine: CentOS 7 machine with internet access
- Offline Machine: CentOS 7 machine where Docker & K8s will be installed in offline mode

The idea is to download all the dependencies in the online machine and transfer them to offline machine

Installing Docker

Online Machine

- Login as root
- Configure YUM package manager
  ```
  yum update
  yum install yum-utils device-mapper-persistent-data lvm2
  ```
- Add docker repo
  ```
  yum-config-manager --add-repo \
  https://download.docker.com/linux/centos/docker-ce.repo
  ```
- Install and enable docker
  ```
  yum install docker-ce
  systemctl start docker
  systemctl enable docker
  ```
- Check docker version
  ```
  docker version
  ```
- Create a directory to save all the docker dependencies needed for offline machine
  ```
  mkdir ~/docker_dependencies
  cd ~/docker_dependencies
  yumdownloader --assumeyes --destdir=$PWD/yum --resolve yum-utils
  yumdownloader --assumeyes --destdir=$PWD/dm --resolve device-mapper-persistent-
  data
  yumdownloader --assumeyes --destdir=$PWD/lvm2 --resolve lvm2
  yumdownloader --assumeyes --destdir=$PWD/docker-ce --resolve docker-ce
  yumdownloader --assumeyes --destdir=$PWD/se --resolve container-selinux
  ```
- Copy this folder to offline machine

Offline Machine

- Login as root
- Install all the dependencies downloaded in the previous step
  ```
  yum install -y --cacheonly --disablerepo=* docker_dependencies/yum/*.rpm
  yum install -y --cacheonly --disablerepo=* docker_dependencies/dm/*.rpm
  yum install -y --cacheonly --disablerepo=* docker_dependencies/lvm2/*.rpm
  yum install -y --cacheonly --disablerepo=* docker_dependencies/docker-ce/*.rpm
  ```

```
yum install -y --cacheonly --disablerepo=* docker_dependencies/se/*.rpm
```

- Enable and start docker daemon
  ```
  systemctl enable docker
  systemctl start docker
  systemctl status docker
  ```
- Check the installation
  ```
  docker version
  ```

Installing Kubernetes: Setting Up Offline Master Node

Online Machine

- Login as root
- Add Kubernetes yum repository

Terminal

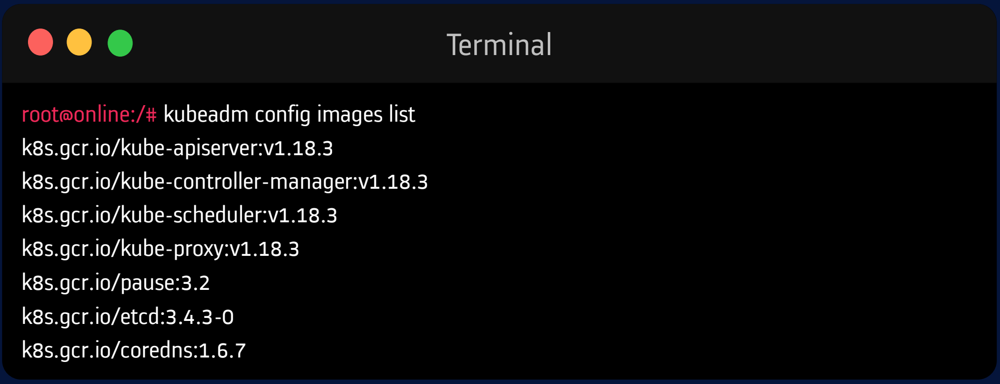
```
root@online:/# cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

- Build yum cache
  ```
  yum makecache fast
  ```
- Download Kubernetes utilities
  - Create a directory to save all the dependencies required by kubelet, kubeadm & kubectl
    ```
    mkdir ~/k8s_dependencies
    cd ~/k8s_dependencies
    yumdownloader --assumeyes --destdir=$PWD --resolve yum-utils kubeadm-1.18.* kubelet-1.18.* kubectl-1.18.* ebtables
    ```
- Copy this directory to offline machine

Offline Machine: PART 1

- Login as root
- Install all the dependencies downloaded in the previous step
  ```
  cd k8s_dependencies
  yum install -y --cacheonly --disablerepo=* $PWD/*.rpm
  ```
- We have successfully installed kubelet, kubeadm & kubectl

- Next step is to find the images required by kubeadm to bootstrap the cluster. Run the below command in the terminal to get the list of images
  kubeadm config images list



Terminal

```
root@online:/# kubeadm config images list
k8s.gcr.io/kube-apiserver:v1.18.3
k8s.gcr.io/kube-controller-manager:v1.18.3
k8s.gcr.io/kube-scheduler:v1.18.3
k8s.gcr.io/kube-proxy:v1.18.3
k8s.gcr.io/pause:3.2
k8s.gcr.io/etcd:3.4.3-0
k8s.gcr.io/coredns:1.6.7
```

- We will download all the above images in the online machine and transfer them back to offline machine for bootstrapping the cluster.

Online Machine

- Login as root
- It is assumed that docker is already up and running in this machine
- Pull all docker images required to setup k8s cluster
  ```
  docker pull k8s.gcr.io/kube-apiserver:v1.18.3
  docker pull k8s.gcr.io/kube-controller-manager:v1.18.3
  docker pull k8s.gcr.io/kube-scheduler:v1.18.3
  docker pull k8s.gcr.io/kube-proxy:v1.18.3
  docker pull k8s.gcr.io/pause:3.2
  docker pull k8s.gcr.io/etcd:3.4.3-0
  docker pull k8s.gcr.io/coredns:1.6.7
  ```
- We will also need the docker image of CNI. For this setup I have used Flannel as the CNI
- Find out the image needed by flannel from the below link
  https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
  It is quay.io/coreos/flannel:v0.12.0-amd64
- Download this docker image as well
  ```
  docker pull quay.io/coreos/flannel:v0.12.0-amd64
  ```
- [Optional] If required, download the images required for setting up Kubernetes dashboard and ingress controllers like nginx, traefik etc., Since offline machines doesn't have access to public internet, feel free to download your docker hub images at this stage. You can do this after the cluster setup as well.
- Save individual images as a TAR archive
  ```
  mkdir k8s_images
  cd k8s_images
  docker save k8s.gcr.io/kube-apiserver:v1.18.3 > kube-api.tar
  ```

```
docker save k8s.gcr.io/kube-controller-manager:v1.18.3 > kube-controller.tar
docker save k8s.gcr.io/kube-scheduler:v1.18.3 > kube-sched.tar
docker save k8s.gcr.io/kube-proxy:v1.18.3 > kube-proxy.tar
docker save k8s.gcr.io/pause:3.2 > pause.tar
docker save k8s.gcr.io/etcd:3.4.3-0 > etcd.tar
docker save k8s.gcr.io/coredns:1.6.7 > coredns.tar
docker save quay.io/coreos/flannel:v0.12.0-amd64 7 > flannel.tar
```

- Transfer this directory with TAR images to offline machine

Offline Machine: PART2

- Login as root
- Unpack the tar images copied from online machine by executing the following command
  docker load < <image name>.tar
  ```
  docker load < kube-api.tar
  ```
  Repeat this for all the images
- Run docker images to see all the loaded images
- Disable swap
  ```
  swapoff -a
  sed -e '/swap/s/^/#/g' -i /etc/fstab
  ```
- Switch SELinux to Permissive mode
  ```
  setenforce 0
  sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
  ```
- Ensure net.bridge.bridge-nf-call-iptables is set to 1 in your sysctl config
  ```
  cat <<EOF >  /etc/sysctl.d/k8s.conf
  net.bridge.bridge-nf-call-ip6tables = 1
  net.bridge.bridge-nf-call-iptables = 1
  EOF
  ```
- Run sysctl --system
- Configure kubectl autocompletion
  ```
  echo "source <(kubectl completion bash)" >> ~/.bashrc
  ```
- Allow Kubernetes service ports in Linux firewall
  On master nodes: firewall-cmd --permanent --add-port={6443,2379,2380,10250,10251,10252}/tcp
  On worker nodes: firewall-cmd --permanent --add-port={10250,30000-32767}/tcp

- Check kubectl version
  ```
  kubectl version
  ```
- Run the following command to set up the cluster
  kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=<IP of Master Node>

```
kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=192.168.56.2
```

Please note 192.168.56.2 is the IP of the interface chosen for the master to listen to incoming requests(api-server). If you have multiple interfaces, choose the IP of the selected interface. Especially required when you have both dynamic and static IP interfaces and you chose to use static IP for avoiding the trouble of initiating the cluster every-time when the master IP changes on dynamic IP interface

Once above command is executed, a token of the form ‘kubeadm join 192.168.56.2:6443 --token oqp357.wexmezpiszo5zeeg \
    --discovery-token-ca-cert-hash sha256:67577a18d7abcd1815cf5086e2196dadf5ac39ab3b18fb705694fdafac84580b’ will be generated. save this token for later use to join worker nodes in the cluster

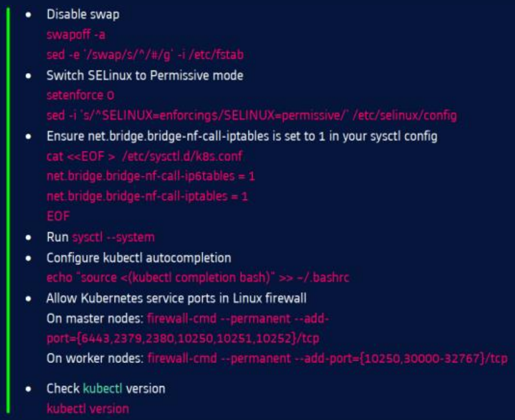
```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

{These commands will be shown in the terminal once you run kubeadm init…}

- Deploy flannel CNI
  ```
  kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
  ```
- Get list of nodes
  ```
  kubectl get nodes
  ```

Installing Kubernetes: Setting Up Offline Worker Nodes

- Worker nodes will be setup in the same way as the master
- Follow PART-1 of offline k8s installation which sets up kubelet, kubeadm & kubectl
- Follow PART-2 of offline k8s installation from disabling swap to checking kubectl version



- Disable swap
  ```
  swapoff -a
  sed -e '/swap/s/^/#/g' -i /etc/fstab
  ```
- Switch SELinux to Permissive mode
  ```
  setenforce 0
  sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
  ```
- Ensure net.bridge.bridge-nf-call-iptables is set to 1 in your sysctl config
  ```
  cat <<EOF >  /etc/sysctl.d/k8s.conf
  net.bridge.bridge-nf-call-ip6tables = 1
  net.bridge.bridge-nf-call-iptables = 1
  EOF
  ```
- Run sysctl --system
- Configure kubectl autocompletion
  ```
  echo "source <(kubectl completion bash)" >> ~/.bashrc
  ```
- Allow Kubernetes service ports in Linux firewall
  On master nodes: firewall-cmd --permanent --add-port={6443,2379,2380,10250,10251,10252}/tcp
  On worker nodes: firewall-cmd --permanent --add-port={10250,30000-32767}/tcp
- Check kubectl version
  ```
  kubectl version
  ```

Other Useful Links

Bootstrap Kubernetes cluster using Kubeadm

https://github.com/kunchalavikram1427/Kubernetes_public/blob/master/Bootstrap_K8s_Cluster_Kubeadm.pdf

Learn K8s

https://github.com/kunchalavikram1427/Kubernetes_public/blob/master/Kubernetes_Made_Easy.pdf