Installation of HA Proxy Server

1. edit the /etc/hosts file for hostname resolution.

csbpubbpswrker01.csb.co.in 10.51.15.105
csbpubbpswrker02.csb.co.in 10.51.15.106
csbpubbpsmastr01.csb.co.in 10.51.15.107
csbpubbpsmastr02.csb.co.in 10.51.15.104
csbpubbpshaprxy.csb.co.in 10.51.15.60

2. install the haproxy package.

dnf install haproxy -y

3. create the backup of haproxy configuration directory.

cp -r /etc/haproxy /tmp/haproxy_backup

4. create the haproxy.cfg under /etc/haproxy with below configuration and save it.

```
global
    log /dev/log  local0 warning
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

   stats socket /var/lib/haproxy/stats

defaults
  log global
  option  httplog
```

```
  option  dontlognull
        timeout connect 5000
        timeout client 50000
        timeout server 50000


frontend kube-apiserver
  bind *:6443
  mode tcp
  option tcplog
  default_backend kube-apiserver


backend kube-apiserver
    mode tcp
    option tcplog
    option tcp-check
    balance roundrobin
    default-server inter 10s downinter 5s rise 2 fall 2 slowstart 60s maxconn 250 maxqueue 256 weight 100
    server  csbpubbpsmastr01 10.51.15.107:6443  check # Replace the IP address with your own.
    server  csbpubbpsmastr02 10.51.15.104:6443  check # Replace the IP address with your own.
```

7. restart and enable the haproxy service.

systemctl start haproxy
systemctl enable haproxy

Kubernetes Installation in Offline Server

This documents will be use to install and configure the kubernetes V1.26.1 it is standard operation procedure to install and configure the kubernetes where internet connectivity is not available.

Pre-installation steps: - Below steps need to perform in all the master / worker nodes.

1. Apply the latest patches.
   dnf update –y

2. Turn off the swap.
   swapoff –a
   vi /etc/fstab and comment the line that has swap written

3. Edit the hostname
   Vi /etc/hostname

4. Edit /etc/hosts to add hostname and IP address on all nodes

   vi /etc/hosts
   10.51.15.105 csbpubbpswrker01.csb.co.in
   10.51.15.106 csbpubbpswrker02.csb.co.in
   10.51.15.107 csbpubbpsmastr01.csb.co.in
   10.51.15.104 csbpubbpsmastr02.csb.co.in
   10.51.15.60 csbpubbpshaprxy.csb.co.in

5. stop the disable the firewall.

   systemctl stop firewalld && systemctl disable firewalld

6. disable the selinux.

   open /etc/selinux/config and edit like below
   SELINUX=disabled

7. install the iproute package

   dnf install -y iproute-tc

Installation of kubernetes -

1. download the docker, containerd, kubelet, kubeadm, kubectl binary in the online server and transfer it on offline server (Worker and master node)

   i) execute the below command in the online server to download the required binary.

   yumdownloader --assumeyes --destdir=<your_rpm_dir> --resolve yum-utils kubeadm-1.26.1 kubelet-1.26.1 kubectl-1.26.1 containerd docker

ii) transfer the files which are downloaded in previous step in the online server. Execute the below command to install the kubernetes.

yum install -y --cacheonly --disablerepo=* <your_rpm_dir>/*.rpm

2. Download the kube-proxy, kube-apiserver, kube-controlle-manager, pause, etcd, coredns, kube-proxy images in the online server and upload in the offline server (worker and master node)

i) execute the below command in the offline server (master server) to check the required images which are needed to pull in the online server.

kubeadm config images list

ii) Pull and save the docker images in the online server :-

docker pull registry.k8s.io/kube-apiserver:v1.26.1
docker pull registry.k8s.io/kube-controller-manager:v1.26.1
docker pull registry.k8s.io/kube-scheduler:v1.26.1
docker pull registry.k8s.io/kube-proxy:v1.26.1
docker pull registry.k8s.io/pause:3.9
docker pull registry.k8s.io/etcd:3.5.6-0
docker pull registry.k8s.io/coredns/coredns:v1.9.3

docker save registry.k8s.io/kube-apiserver:v1.26.1 > kube-apiserver.tar
docker save registry.k8s.io/kube-controller-manager:v1.26.1 > kube-controlle-manager.tar
docker save registry.k8s.io/kube-scheduler:v1.26.1 > kube-scheduler.tar
docker save registry.k8s.io/kube-proxy:v1.26.1 > kube-proxy.tar
docker save registry.k8s.io/pause:3.9 > pause.tar
docker save registry.k8s.io/etcd:3.5.6-0 > etcd.tar
docker save registry.k8s.io/coredns/coredns:v1.9.3 > coredns.tar

iii) upload all the saved images in the master / worker nodes. execute the below command to load the images in the k8s namespace of the containerd.

ctr -n k8s.io images import kube-apiserver.tar
ctr -n k8s.io images import kube-controlle-manager.tar
ctr -n k8s.io images import kube-scheduler.tar
ctr -n k8s.io images import kube-proxy.tar
ctr -n k8s.io images import etcd.tar
ctr -n k8s.io images import coredns.tar
ctr -n k8s.io images import pause3_6.tar
ctr -n k8s.io images import pause.tar

iv) verify the loaded images by executing the below command.

ctr -n k8s.io images list

v) alternatively, to load the images in the docker, please execute the below command.

```
docker load < kube-apiserver.tar
docker load < kube-controlle-manager.tar
docker load < kube-scheduler.tar
docker load < kube-proxy.tar
docker load < pause.tar
docker load < etcd.tar
docker load < coredns.tar
```

3. Kubernetes cluster initialization: - execute below command in the master nodes only.

```
kubeadm init --control-plane-endpoint 10.51.15.60:6443 --upload-certs --pod-network-cidr=192.168.0.0/16 --kubernetes-version=v1.26.1  --image-repository=csbpubbpshaprxy.csb.co.in
```

```
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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 the control-plane node running the following command on each as root:

  kubeadm join 10.51.15.60:6443 --token 4kfode.lcgj0n74fslynwup \
        --discovery-token-ca-cert-hash sha256:c76157f034c7e319f4b9d386b3f073391e40455c46734115d3647ef8e58fa22f \
        --control-plane --certificate-key d8793d6c82bf00bdf43eb5eb29dffce9913297fb4533f4bc7692b7ab22f98b20

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

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

kubeadm join 10.51.15.60:6443 --token 4kfode.lcgj0n74fslynwup \
--discovery-token-ca-cert-hash sha256:c76157f034c7e319f4b9d386b3f073391e40455c46734115d3647ef8e58fa22f

Installation and configuration of Docker repository server

1. Create the directory structure.

   mkdir images_storage
   cd images_storage
   mkdir certificate = for storing the TLS / SSL certificate
   mkdir images = for storing the images
2. Take the below certificate and arrange it on below manner. create the domain.crt. placed the domain.key and domain.crt in /image_storage/certificate

   i) domain.cer
   ii) Intermediate.cer
   iii) Root.cer

   Domain.crt

   domain.key

3. Execute the below command to deploy the local Docker registry -

docker run -d --restart=always --name registry -v /images_storage/images:/var/lib/registry -v "$(pwd)"/certificate:/certificate -e REGISTRY_HTTP_ADDR=0.0.0.0:443 -e REGISTRY_HTTP_TLS_CERTIFICATE=/certificate/domain.crt -e REGISTRY_HTTP_TLS_KEY=/certificate/domain.key -p 443:443 registry:2

4. docker ps –a = to check the running container

5. follow the below steps to push / pull the images.

   docker tag image:tag csbpubbpshaprxy.csb.co.in/image:tag
   docker push csbpubbpshaprxy.csb.co.in/image:tag
   docker pull csbpubbpshaprxy.csb.co.in/image:tag

installation and configuration of Calico CNI

1. download and save the below images in online server
docker pull docker.io/calico/cni:v3.26.1
docker pull docker.io/calico/node:v3.26.1
docker pull docker.io/calico/kube-controllers:v3.26.1
docker pull docker.io/calico/typha:v3.26.1

docker save docker.io/calico/cni:v3.26.1 > calico_cni.tar
docker save docker.io/calico/node:v3.26.1 > calico_node.tar
docker save docker.io/calico/kube-controllers:v3.26.1 > kube-controllers.tar
docker save docker.io/calico/typha:v3.26.1 > calico_typha.tar

2. transfer the images in offline server and load it.
docker load < calico_cni.tar
docker load < calico_node.tar
docker load < kube-controllers.tar
docker load < calico_typha.tar

3. tag and push the images in the locally hosted repository

docker tag calico/typha:v3.26.1 csbpubbpshaprxy.csb.co.in/calico_typha:v3.26.1
docker push csbpubbpshaprxy.csb.co.in/calico_typha:v3.26.1

docker tag calico/kube-controllers:v3.26.1 csbpubbpshaprxy.csb.co.in/calico_kube-controllers:v3.26.1
docker push csbpubbpshaprxy.csb.co.in/calico_kube-controllers:v3.26.1

docker tag calico/node:v3.26.1 csbpubbpshaprxy.csb.co.in/calico_node:v3.26.1
docker push csbpubbpshaprxy.csb.co.in/calico_node:v3.26.1

docker tag calico/cni:v3.26.1 csbpubbpshaprxy.csb.co.in/calico_cni:v3.26.1
docker push csbpubbpshaprxy.csb.co.in/calico_cni:v3.26.1

4. Download the Calico networking manifest for the Kubernetes API datastore in online server and transfer it to offline server.

   curl https://raw.githubusercontent.com/projectcalico/calico/v3.26.3/manifests/calico-typha.yaml -o calico.yaml

5. Modify the image in calico.yaml file so that images can be pulled from locally hosted repository.

   csbpubbpshaprxy.csb.co.in/calico_typha:v3.26.1
   csbpubbpshaprxy.csb.co.in/calico_kube-controllers:v3.26.1
   csbpubbpshaprxy.csb.co.in/calico_node:v3.26.1
   csbpubbpshaprxy.csb.co.in/calico_cni:v3.26.1

6. Execute the below command to deploy the calico CNI.

   Kubectl create –f calico.yaml