Kube API service Installation

```
#!/bin/bash
# Move the kube-apiserver binary to /usr/bin directory #
cd /root/binaries/kubernetes/server/bin/
cp kube-apiserver /usr/local/bin/

# move to the certificate directory #
cd /root/certificates

# configuration for the kube-api server #
cat <<EOF | sudo tee api.conf
[req]
req_extensions = v3_req
distinguished_name = req_distinguished_name
[req_distinguished_name]
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = kubernetes
DNS.2 = kubernetes.default
DNS.3 = kubernetes.default.svc
DNS.4 = kubernetes.default.svc.cluster.local
IP.1 = 127.0.0.1
IP.2 = 192.168.193.111
IP.3 = 10.32.0.1
EOF

# genrate the Certifiacate for the Kube=API server #
openssl genrsa -out kube-api.key 2048
openssl req -new -key kube-api.key -subj "/CN=kube-apiserver" -out kube-api.csr -config api.conf
openssl x509 -req -in kube-api.csr -CA ca.crt -CAkey ca.key -CAcreateserial  -out kube-api.crt -extensions v3_req -extfile api.conf
-days 1000

# Generate Certificate for Service Account: #
openssl genrsa -out service-account.key 2048
openssl req -new -key service-account.key -subj "/CN=service-accounts" -out service-account.csr
openssl x509 -req -in service-account.csr -CA ca.crt -CAkey ca.key -CAcreateserial  -out service-account.crt -days 100

# Copy the certificate files to /var/lib/kubernetes directory #
mkdir /var/lib/kubernetes
cp etcd.crt etcd.key ca.crt kube-api.key kube-api.crt service-account.crt service-account.key /var/lib/kubernetes

# Creating Encryption key and Configuration #
ENCRYPTION_KEY=$(head -c 32 /dev/urandom | base64)

# create the yaml file for the encryption #
```

```
cat > encryption-at-rest.yaml <<EOF
kind: EncryptionConfig
apiVersion: v1
resources:
  - resources:
      - secrets
    providers:
      - aescbc:
          keys:
            - name: key1
              secret: ${ENCRYPTION_KEY}
      - identity: {}
EOF

# copy yaml file to /var/lib/kubernetes directory #
cp encryption-at-rest.yaml /var/lib/kubernetes/encryption-at-rest.yaml

# create the Systemd service #
cat <<EOF | sudo tee /etc/systemd/system/kube-apiserver.service
[Unit]
Description=Kubernetes API Server
Documentation=https://github.com/kubernetes/kubernetes

[Service]
ExecStart=/usr/local/bin/kube-apiserver \
--advertise-address=192.168.193.111 \
--allow-privileged=true \
--authorization-mode=Node,RBAC \
--client-ca-file=/var/lib/kubernetes/ca.crt \
--enable-admission-plugins=NamespaceLifecycle,NodeRestriction,LimitRanger,ServiceAccount,DefaultStorageClass,ResourceQuota \
--enable-bootstrap-token-auth=true \
--etcd-cafile=/var/lib/kubernetes/ca.crt \
--etcd-certfile=/var/lib/kubernetes/etcd.crt \
--etcd-keyfile=/var/lib/kubernetes/etcd.key \
--etcd-servers=https://127.0.0.1:2379 \
--kubelet-certificate-authority=/var/lib/kubernetes/ca.crt \
--kubelet-client-certificate=/var/lib/kubernetes/kube-api.crt \
--kubelet-client-key=/var/lib/kubernetes/kube-api.key \
--kubelet-https=true \
--service-account-key-file=/var/lib/kubernetes/service-account.crt \
--service-cluster-ip-range=10.32.0.0/24 \
--tls-cert-file=/var/lib/kubernetes/kube-api.crt \
--tls-private-key-file=/var/lib/kubernetes/kube-api.key \
--requestheader-client-ca-file=/var/lib/kubernetes/ca.crt \
--service-node-port-range=30000-32767 \
--service-account-issuer=kubernetes.default.svc \
--service-account-signing-key-file=/var/lib/kubernetes/service-account.key \
--audit-log-maxage=30 \
--audit-log-maxbackup=3 \
```

```
--audit-log-maxsize=100 \
--audit-log-path=/var/log/kube-api-audit.log \
--bind-address=0.0.0.0 \
--event-ttl=1h \
--encryption-provider-config=/var/lib/kubernetes/encryption-at-rest.yaml \
--v=2
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF

# start the kubeapi server #

systemctl start kube-apiserver
systemctl status kube-apiserver
systemctl enable kube-apiserver
```

Kube Api service install in the Master Node

```
root@k8master:~# systemctl status  kube-apiserver.service
● kube-apiserver.service - Kubernetes API Server
   Loaded: loaded (/etc/systemd/system/kube-apiserver.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2021-01-10 13:34:42 UTC; 3min 33s ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 20484 (kube-apiserver)
    Tasks: 10 (limit: 4915)
   CGroup: /system.slice/kube-apiserver.service
           └─20484 /usr/local/bin/kube-apiserver --advertise-address=192.168.193.111 --allow-privileged=true
```

Controller installation on master

```
#!/bin/bash
# Generate Certificates #
cd /root/certificates

openssl genrsa -out kube-controller-manager.key 2048
openssl req -new -key kube-controller-manager.key -subj "/CN=system:kube-controller-manager" -out kube-controller-manager.csr
openssl x509 -req -in kube-controller-manager.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out kube-controller-manager.crt -days
1000

# Generating KubeConfig #

cp /root/binaries/kubernetes/server/bin/kubectl /usr/local/bin

kubectl config set-cluster kubernetes-from-scratch \
    --certificate-authority=ca.crt \
    --embed-certs=true \
    --server=https://127.0.0.1:6443 \
    --kubeconfig=kube-controller-manager.kubeconfig
kubectl config set-cluster kubernetes-from-scratch \
    --certificate-authority=ca.crt \
    --embed-certs=true \
    --server=https://127.0.0.1:6443 \
    --kubeconfig=kube-controller-manager.kubeconfig

kubectl config set-credentials system:kube-controller-manager \
    --client-certificate=kube-controller-manager.crt \
    --client-key=kube-controller-manager.key \
    --embed-certs=true \
    --kubeconfig=kube-controller-manager.kubeconfig

kubectl config set-context default \
    --cluster=kubernetes-from-scratch \
    --user=system:kube-controller-manager \
    --kubeconfig=kube-controller-manager.kubeconfig
kubectl config use-context default --kubeconfig=kube-controller-manager.kubeconfig

# Copying the files to kubernetes directory #

cp kube-controller-manager.crt kube-controller-manager.key kube-controller-manager.kubeconfig ca.key /var/lib/kubernetes/

# Configuring SystemD service file #

cat <<EOF | sudo tee /etc/systemd/system/kube-controller-manager.service
[Unit]
Description=Kubernetes Controller Manager
Documentation=https://github.com/kubernetes/kubernetes

[Service]
```

```
ExecStart=/usr/local/bin/kube-controller-manager \\
--address=0.0.0.0 \\
--service-cluster-ip-range=10.32.0.0/24 \\
--cluster-cidr=10.200.0.0/16 \\
--kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \\
--authentication-kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \\
--authorization-kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \\
--leader-elect=true \\
--cluster-signing-cert-file=/var/lib/kubernetes/ca.crt \\
--cluster-signing-key-file=/var/lib/kubernetes/ca.key \\
--root-ca-file=/var/lib/kubernetes/ca.crt \\
--service-account-private-key-file=/var/lib/kubernetes/service-account.key \\
--use-service-account-credentials=true \\
--v=2
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
# copy file to controller manager #

cp /root/binaries/kubernetes/server/bin/kube-controller-manager /usr/local/bin
systemctl start kube-controller-manager
systemctl status kube-controller-manager
systemctl enable kube-controller-manager
```

Controller service install in the Master Node

```
root@k8master:~# systemctl status kube-controller-manager
● kube-controller-manager.service - Kubernetes Controller Manager
   Loaded: loaded (/etc/systemd/system/kube-controller-manager.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2021-01-10 13:53:14 UTC; 4min 20s ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 20590 (kube-controller)
    Tasks: 8 (limit: 4915)
   CGroup: /system.slice/kube-controller-manager.service
           └─20590 /usr/local/bin/kube-controller-manager --address=0.0.0.0 --service-cluster-ip-range=10.32.0.0/24
```

scheduler installation on master

```
#!/bin/bash

cd /root/certificates/

# Generate Certificates#

openssl genrsa -out kube-scheduler.key 2048
openssl req -new -key kube-scheduler.key -subj "/CN=system:kube-scheduler" -out kube-scheduler.csr
openssl x509 -req -in kube-scheduler.csr -CA ca.crt -CAkey ca.key -CAcreateserial  -out kube-scheduler.crt -days 1000

# Generate Kubeconfig file#

{
  kubectl config set-cluster kubernetes-from-scratch \
    --certificate-authority=ca.crt \
    --embed-certs=true \
    --server=https://127.0.0.1:6443 \
    --kubeconfig=kube-scheduler.kubeconfig

  kubectl config set-credentials system:kube-scheduler \
    --client-certificate=kube-scheduler.crt \
    --client-key=kube-scheduler.key \
    --embed-certs=true \
    --kubeconfig=kube-scheduler.kubeconfig

  kubectl config set-context default \
    --cluster=kubernetes-from-scratch \
    --user=system:kube-scheduler \
    --kubeconfig=kube-scheduler.kubeconfig

  kubectl config use-context default --kubeconfig=kube-scheduler.kubeconfig
}

# Copy the scheduler kubeconfig #

cp kube-scheduler.kubeconfig /var/lib/kubernetes/

# Configuring SystemD service #
cat <<EOF | sudo tee /etc/systemd/system/kube-scheduler.service
[Unit]
Description=Kubernetes Scheduler
Documentation=https://github.com/kubernetes/kubernetes

[Service]
ExecStart=/usr/local/bin/kube-scheduler \\
  --kubeconfig=/var/lib/kubernetes/kube-scheduler.kubeconfig \\
  --authentication-kubeconfig=/var/lib/kubernetes/kube-scheduler.kubeconfig \\
  --authorization-kubeconfig=/var/lib/kubernetes/kube-scheduler.kubeconfig \\
```

```
  --bind-address=127.0.0.1 \\
  --leader-elect=true
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF

# copy the schedular bin file #
cp /root/binaries/kubernetes/server/bin/kube-scheduler /usr/local/bin
systemctl start kube-scheduler
systemctl enable kube-scheduler
systemctl status kube-scheduler
```

Scheduler service install in the Master Node

```
root@k8master:~# systemctl enable kube-scheduler
Created symlink /etc/systemd/system/multi-user.target.wants/kube-scheduler.service → /etc/systemd/system/kube-scheduler.service.
root@k8master:~# systemctl status kube-scheduler
● kube-scheduler.service - Kubernetes Scheduler
   Loaded: loaded (/etc/systemd/system/kube-scheduler.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2021-01-10 14:03:18 UTC; 53s ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 20721 (kube-scheduler)
    Tasks: 9 (limit: 4915)
   CGroup: /system.slice/kube-scheduler.service
           └─20721 /usr/local/bin/kube-scheduler --kubeconfig=/var/lib/kubernetes/kube-scheduler.kubeconfig --authentication-kubeconfig=/
```

Validate the cluster

```
#!/bin/bash
#Step 1. Generate Certificate for Administrator User #
cd /root/certificates
openssl genrsa -out admin.key 2048
openssl req -new -key admin.key -subj "/CN=admin/O=system:masters" -out admin.csr
openssl x509 -req -in admin.csr -CA ca.crt -CAkey ca.key -CAcreateserial  -out admin.crt -days 1000

 # Step 2. Create KubeConfig file #

{
  kubectl config set-cluster kubernetes-from-scratch \
    --certificate-authority=ca.crt \
    --embed-certs=true \
    --server=https://192.168.193.111:6443 \
    --kubeconfig=admin.kubeconfig

  kubectl config set-credentials admin \
    --client-certificate=admin.crt \
    --client-key=admin.key \
    --embed-certs=true \
    --kubeconfig=admin.kubeconfig

  kubectl config set-context default \
    --cluster=kubernetes-from-scratch \
    --user=admin \
    --kubeconfig=admin.kubeconfig

  kubectl config use-context default --kubeconfig=admin.kubeconfig
}
# Step 3: Verify Cluster Status #

kubectl get componentstatuses --kubeconfig=admin.kubeconfig
cp /root/certificates/admin.kubeconfig ~/.kube/config
kubectl get componentstatuses
```

Service Validation status

```
NAME                 STATUS    MESSAGE             ERROR
scheduler            Healthy   ok
controller-manager   Healthy   ok
etcd-0               Healthy   {"health":"true"}
```

Worker node configuration

```
#!/bin/bash

# make directory #

mkdir /root/binaries
 cd /root/binaries

# download the package on worker node #

wget https://dl.k8s.io/v1.20.0/kubernetes-node-linux-amd64.tar.gz

#unzip tha package on worker node #

tar -zxvf kubernetes-node-linux-amd64.tar.gz

##copy kubectl and kubelet #

cd  /root/binaries/kubernetes/node/bin/
cp kube-proxy kubectl kubelet /usr/local/bin

#set the IP forwarding#
sysctl -w net.ipv4.conf.all.forwarding=1

#install the docker #
wget https://download.docker.com/linux/ubuntu/gpg
apt-key add gpg
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common -y
sudo apt-get install -y \
  containerd.io=1.2.13-2 \
  docker-ce=5:19.03.11~3-0~ubuntu-$(lsb_release -cs) \
  docker-ce-cli=5:19.03.11~3-0~ubuntu-$(lsb_release -cs)

  systemctl enable docker
```

Generate Kubelet Certificate for Worker Node using Master Node

RUN THE COMMAND ON MASTER NODE

```
#|/bin/bash
# Generate Kubelet Certificate #
cd /root/certificates


# configurate for kubelet service in master node #

cat > openssl-cka-worker.cnf <<EOF
[req]
req_extensions = v3_req
distinguished_name = req_distinguished_name
[req_distinguished_name]
[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names
[alt_names]
DNS.1 = cka-worker1
IP.1 = 192.168.193.112
EOF

#genrate the certificate#

openssl genrsa -out cka-worker1.key 2048
openssl req -new -key cka-worker1.key -subj "/CN=system:node:cka-worker1/O=system:nodes" -out cka-worker1.csr -config openssl-cka-
worker.cnf
openssl x509 -req -in cka-worker1.csr -CA ca.crt -CAkey ca.key -CAcreateserial  -out cka-worker1.crt -extensions v3_req -extfile
openssl-cka-worker.cnf -days 1000

# genrate the kubeproxy service #

openssl genrsa -out kube-proxy.key 2048
openssl req -new -key kube-proxy.key -subj "/CN=system:kube-proxy" -out kube-proxy.csr
openssl x509 -req -in kube-proxy.csr -CA ca.crt -CAkey ca.key -CAcreateserial  -out kube-proxy.crt -days 1000


# copy genrate certificate to /tmp folder on worker node manually change authentication according to your
# environment #
scp kube-proxy.crt kube-proxy.key cka-worker1.crt cka-worker1.key ca.crt vijay@192.168.193.112:/tmp
```

Configure the certificate for kubelet

RUN THE COMMAND ON Worker Node

```
#!/bin/bash
swapoff -a
mkdir /root/certificates

# move the certifiacte to directory #
cd /tmp
cp kube-proxy.crt kube-proxy.key cka-worker1.crt cka-worker1.key ca.crt /root/certificates
mkdir /var/lib/kubernetes
mkdir /var/lib/kubelet
cp ca.crt /var/lib/kubernetes/
cp ca.crt /var/lib/kubelet/
mv cka-worker1.crt  cka-worker1.key  kube-proxy.crt  kube-proxy.key /var/lib/kubelet/

#genrate the kubelet configuration #
cat <<EOF | sudo tee /var/lib/kubelet/kubelet-config.yaml
kind: KubeletConfiguration
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    enabled: true
  x509:
    clientCAFile: "/var/lib/kubernetes/ca.crt"
authorization:
  mode: Webhook
clusterDomain: "cluster.local"
clusterDNS:
  - "10.32.0.10"
runtimeRequestTimeout: "15m"
EOF

# Generate Systemd service file for kubelet #

cat <<EOF | sudo tee /etc/systemd/system/kubelet.service
[Unit]
Description=Kubernetes Kubelet
Documentation=https://github.com/kubernetes/kubernetes
After=docker.service
Requires=docker.service

[Service]
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --image-pull-progress-deadline=2m \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --tls-cert-file=/var/lib/kubelet/cka-worker1.crt \\
  --tls-private-key-file=/var/lib/kubelet/cka-worker1.key \\
```

```
  --network-plugin=cni \\
  --register-node=true \\
  --v=2 \\
  --cgroup-driver=systemd \\
  --runtime-cgroups=/systemd/system.slice \\
  --kubelet-cgroups=/systemd/system.slice
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF

# Generate the Kubeconfig file for Kubelet #
cd /var/lib/kubelet

{
  kubectl config set-cluster kubernetes-from-scratch \
    --certificate-authority=ca.crt \
    --embed-certs=true \
    --server=https://192.168.193.111:6443 \
    --kubeconfig=cka-worker1.kubeconfig

  kubectl config set-credentials system:node:cka-worker1 \
    --client-certificate=cka-worker1.crt \
    --client-key=cka-worker1.key \
    --embed-certs=true \
    --kubeconfig=cka-worker1.kubeconfig

  kubectl config set-context default \
    --cluster=kubernetes-from-scratch \
    --user=system:node:cka-worker1 \
    --kubeconfig=cka-worker1.kubeconfig

  kubectl config use-context default --kubeconfig=cka-worker1.kubeconfig
}

mv cka-worker1.kubeconfig kubeconfig
```

Verification at the worker node

```
● kubelet.service - Kubernetes Kubelet
   Loaded: loaded (/etc/systemd/system/kubelet.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2021-01-10 20:10:03 UTC; 10min ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 15449 (kubelet)
    Tasks: 0 (limit: 4915)
   CGroup: /system.slice/kubelet.service
           └─15449 /usr/local/bin/kubelet --config=/var/lib/kubelet/kubelet-config.yaml -
```

Verification at the master Node

```
root@k8master:~/certificates# kubectl get nodes
NAME          STATUS     ROLES    AGE   VERSION
cka-worker1   NotReady   <none>   12m   v1.20.0
root@k8master:~/certificates#
```

Still not ready due to Networking elements

Configure the Kube-proxy in worker node

RUN THE COMMAND ON Worker Node

```
#!/bin/bash
# create the directory run all command from wher you ca.crt file is stored #

mkdir /var/lib/kube-proxy
 cd  /root/certificates/

# Step 2: Generate KubeConfig file#

{
  kubectl config set-cluster kubernetes-from-scratch \
    --certificate-authority=ca.crt \
    --embed-certs=true \
    --server=https://192.168.193.111:6443 \
    --kubeconfig=kube-proxy.kubeconfig

  kubectl config set-credentials system:kube-proxy \
    --client-certificate=kube-proxy.crt \
    --client-key=kube-proxy.key \
    --embed-certs=true \
    --kubeconfig=kube-proxy.kubeconfig

  kubectl config set-context default \
    --cluster=kubernetes-from-scratch \
    --user=system:kube-proxy \
    --kubeconfig=kube-proxy.kubeconfig

  kubectl config use-context default --kubeconfig=kube-proxy.kubeconfig
}
mv kube-proxy.kubeconfig /var/lib/kube-proxy/kubeconfig

# Step 3: Generate kube-proxy configuration file #

cd /var/lib/kube-proxy
cat <<EOF | sudo tee /var/lib/kube-proxy/kube-proxy-config.yaml
kind: KubeProxyConfiguration
apiVersion: kubeproxy.config.k8s.io/v1alpha1
clientConnection:
  kubeconfig: "/var/lib/kube-proxy/kubeconfig"
mode: "iptables"
clusterCIDR: "10.200.0.0/16"
EOF
# Step 4: Create kube-proxy service file:#

cat <<EOF | sudo tee /etc/systemd/system/kube-proxy.service
[Unit]
Description=Kubernetes Kube Proxy
Documentation=https://github.com/kubernetes/kubernetes
```

```
[Service]
ExecStart=/usr/local/bin/kube-proxy \\
  --config=/var/lib/kube-proxy/kube-proxy-config.yaml
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
EOF
```

Kube-proxy service status

```
root@cka-worker1:~# systemctl status kube-proxy.service
● kube-proxy.service - Kubernetes Kube Proxy
   Loaded: loaded (/etc/systemd/system/kube-proxy.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2021-01-10 17:44:44 UTC; 16h ago
     Docs: https://github.com/kubernetes/kubernetes
 Main PID: 4986 (kube-proxy)
    Tasks: 5 (limit: 4915)
   CGroup: /system.slice/kube-proxy.service
           └─4986 /usr/local/bin/kube-proxy --config=/var/lib/kube-proxy/kube-proxy-config.yaml
```

Network Addon on the worker node

RUN THE COMMAND ON worker node

```
#!/bin/bash

cd /tmp
wget
https://github.com/containernetworking/plugins/releases/download/v0.8.6/cni-plugins-linux-amd64-v0.8.6.t
gz

mkdir -p \
  /etc/cni/net.d \
  /opt/cni/bin \
  /var/run/kubernetes

  mv cni-plugins-linux-amd64-v0.8.6.tgz /opt/cni/bin
  cd /opt/cni/bin
  tar -xzvf cni-plugins-linux-amd64-v0.8.6.tgz
```

Weave Network Addon on the master node

RUN THE COMMAND ON Master Node

kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')&env.IPALLOC_RANGE=10.200.0.0/16"

```
root@k8master:~# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')&env.IPALLOC_RAN
GE=10.200.0.0/16"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
```

Kubectl Get nodes

```
root@k8master:~# kubectl get nodes
NAME          STATUS   ROLES    AGE   VERSION
cka-worker1   Ready    <none>   13h   v1.20.0
root@k8master:~#
root@k8master:~#
root@k8master:~#
```