Kubernetes the hard way

node notes

kube-1 192.168.1.8 192.168.2.151
kube-2 192.168.1.10 192.168.2.162
kube-3 192.168.1.6 192.168.2.157
kube-4 192.168.1.7 192.168.2.154
kube-lb 192.168.1.13 192.168.2.170

Controller components

- kube-apiserver: Serves the Kubernetes API. This allows users to interact with the cluster.
- etcd: Kubernetes cluster datastore.
- kube-scheduler: Schedules pods on available worker nodes.
- kube-controller-manager: Runs a series of controllers that provide a wide range of functionality.
- cloud-controller-manager: Handles interaction with underlying cloud providers.

Worker components

- kubelet: Controls each worker node, providing the APIs that are used by the control plane to manage nodes and pods, and interacts with the container runtime to manage containers.
- kube-proxy: Manages iptables rules on the node to provide virtual network access to pods.
- Container runtime: Downloads images and runs containers. Two examples of container runtimes are Docker and containerd

Add to all nodes

```
vim /etc/hosts
192.168.1.8 kube-1.myhypervisor.ca kube-1
192.168.1.10 kube-2.myhypervisor.ca kube-2
192.168.1.6 kube-3.myhypervisor.ca kube-3
192.168.1.7 kube-4.myhypervisor.ca kube-4
192.168.1.13 kube-lb.myhypervisor.ca kube-lb
```

Install kubectl / cfssl

On local workstation

https://github.com/kelseyhightower/kubernetes-the-hard-way/blob/master/docs/02-client-tools.md
https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl

Gen Certs:

on local workstation

CA

```
{

cat > ca-config.json << EOF
{
  "signing": {
    "default": {
      "expiry": "8760h"
    },
    "profiles": {
      "kubernetes": {
        "usages": ["signing", "key encipherment", "server auth", "client auth"],
        "expiry": "8760h"
      }
    }
  }
}
EOF

cat > ca-csr.json << EOF
{
  "CN": "Kubernetes",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "Kubernetes",
      "OU": "CA",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert -initca ca-csr.json | cfssljson -bare ca

}
```

Admin Client certificate

```
{

cat > admin-csr.json << EOF
{
  "CN": "admin",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "system:masters",
      "OU": "Kubernetes The Hard Way",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  admin-csr.json | cfssljson -bare admin

}
```

Kubelet Client certificates

```
WORKER0_HOST=kube-3.myhypervisor.ca
WORKER0_IP=192.168.1.6
WORKER1_HOST=kube-4.myhypervisor.ca
WORKER1_IP=192.168.1.7

{
cat > ${WORKER0_HOST}-csr.json << EOF
{
  "CN": "system:node:${WORKER0_HOST}",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "system:nodes",
      "OU": "Kubernetes The Hard Way",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -hostname=${WORKER0_IP},${WORKER0_HOST} \
  -profile=kubernetes \
  ${WORKER0_HOST}-csr.json | cfssljson -bare ${WORKER0_HOST}

cat > ${WORKER1_HOST}-csr.json << EOF
{
  "CN": "system:node:${WORKER1_HOST}",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "system:nodes",
      "OU": "Kubernetes The Hard Way",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -hostname=${WORKER1_IP},${WORKER1_HOST} \
  -profile=kubernetes \
  ${WORKER1_HOST}-csr.json | cfssljson -bare ${WORKER1_HOST}

}
```

Controller Manager Client certificate:

```
{

cat > kube-controller-manager-csr.json << EOF
{
  "CN": "system:kube-controller-manager",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "system:kube-controller-manager",
      "OU": "Kubernetes The Hard Way",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  kube-controller-manager-csr.json | cfssljson -bare kube-controller-manager
}
```

Kube Proxy Client

```
{

cat > kube-proxy-csr.json << EOF
{
  "CN": "system:kube-proxy",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "system:node-proxier",
      "OU": "Kubernetes The Hard Way",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  kube-proxy-csr.json | cfssljson -bare kube-proxy

}
```

Kube Scheduler Client Certificate:

```
{

cat > kube-scheduler-csr.json << EOF
{
  "CN": "system:kube-scheduler",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "system:kube-scheduler",
      "OU": "Kubernetes The Hard Way",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  kube-scheduler-csr.json | cfssljson -bare kube-scheduler

}
```

API server

```
CERT_HOSTNAME=10.32.0.1,192.168.1.8,kube-1.myhypervisor.ca,192.168.1.10,kube-
2.myhypervisor.ca,192.168.1.13,kube-lb.myhypervisor.ca,127.0.0.1,localhost,kubernetes.default
{

cat > kubernetes-csr.json << EOF
{
  "CN": "kubernetes",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "Kubernetes",
      "OU": "Kubernetes The Hard Way",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -hostname=${CERT_HOSTNAME} \
  -profile=kubernetes \
  kubernetes-csr.json | cfssljson -bare kubernetes

}
```

service account

```
{

cat > service-account-csr.json << EOF
{
  "CN": "service-accounts",
  "key": {
    "algo": "rsa",
    "size": 2048
  },
  "names": [
    {
      "C": "CA",
      "L": "Montreal",
      "O": "Kubernetes",
      "OU": "Kubernetes The Hard Way",
      "ST": "Quebec"
    }
  ]
}
EOF

cfssl gencert \
  -ca=ca.pem \
  -ca-key=ca-key.pem \
  -config=ca-config.json \
  -profile=kubernetes \
  service-account-csr.json | cfssljson -bare service-account

}
```

scp

```
scp ca.pem kube-3.myhypervisor.ca-key.pem kube-3.myhypervisor.ca.pem root@192.168.2.157:~/
scp ca.pem kube-4.myhypervisor.ca-key.pem kube-4.myhypervisor.ca.pem root@192.168.2.154:~/

scp ca.pem ca-key.pem kubernetes-key.pem kubernetes.pem \
    service-account-key.pem service-account.pem root@192.168.2.151:~/
scp ca.pem ca-key.pem kubernetes-key.pem kubernetes.pem \
    service-account-key.pem service-account.pem root@192.168.2.162:~/
```

Kubeconfig

kubelet kubeconfig for each worker node

```
KUBERNETES_ADDRESS=192.168.1.13
for instance in kube-3.myhypervisor.ca kube-4.myhypervisor.ca; do
  kubectl config set-cluster kubernetes-the-hard-way \
    --certificate-authority=ca.pem \
    --embed-certs=true \
    --server=https://${KUBERNETES_ADDRESS}:6443 \
    --kubeconfig=${instance}.kubeconfig

  kubectl config set-credentials system:node:${instance} \
    --client-certificate=${instance}.pem \
    --client-key=${instance}-key.pem \
    --embed-certs=true \
    --kubeconfig=${instance}.kubeconfig

  kubectl config set-context default \
    --cluster=kubernetes-the-hard-way \
    --user=system:node:${instance} \
    --kubeconfig=${instance}.kubeconfig

  kubectl config use-context default --kubeconfig=${instance}.kubeconfig
done
```

Kube proxy

```
{
  kubectl config set-cluster kubernetes-the-hard-way \
    --certificate-authority=ca.pem \
    --embed-certs=true \
    --server=https://${KUBERNETES_ADDRESS}:6443 \
    --kubeconfig=kube-proxy.kubeconfig
  kubectl config set-credentials system:kube-proxy \
    --client-certificate=kube-proxy.pem \
    --client-key=kube-proxy-key.pem \
    --embed-certs=true \
    --kubeconfig=kube-proxy.kubeconfig

  kubectl config set-context default \
    --cluster=kubernetes-the-hard-way \
    --user=system:kube-proxy \
    --kubeconfig=kube-proxy.kubeconfig

  kubectl config use-context default --kubeconfig=kube-proxy.kubeconfig
}
```

kube-controller-manager

```
{
  kubectl config set-cluster kubernetes-the-hard-way \
    --certificate-authority=ca.pem \
    --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.pem \
    --client-key=kube-controller-manager-key.pem \
    --embed-certs=true \
    --kubeconfig=kube-controller-manager.kubeconfig

  kubectl config set-context default \
    --cluster=kubernetes-the-hard-way \
    --user=system:kube-controller-manager \
    --kubeconfig=kube-controller-manager.kubeconfig

  kubectl config use-context default --kubeconfig=kube-controller-manager.kubeconfig
}
```

kube-scheduler

```
{
  kubectl config set-cluster kubernetes-the-hard-way \
    --certificate-authority=ca.pem \
    --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.pem \
    --client-key=kube-scheduler-key.pem \
    --embed-certs=true \
    --kubeconfig=kube-scheduler.kubeconfig

  kubectl config set-context default \
    --cluster=kubernetes-the-hard-way \
    --user=system:kube-scheduler \
    --kubeconfig=kube-scheduler.kubeconfig

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

admin

```
{
  kubectl config set-cluster kubernetes-the-hard-way \
    --certificate-authority=ca.pem \
    --embed-certs=true \
    --server=https://127.0.0.1:6443 \
    --kubeconfig=admin.kubeconfig

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

  kubectl config set-context default \
    --cluster=kubernetes-the-hard-way \
    --user=admin \
    --kubeconfig=admin.kubeconfig

  kubectl config use-context default --kubeconfig=admin.kubeconfig
}
```

SCP

```
scp kube-3.myhypervisor.ca.kubeconfig kube-proxy.kubeconfig root@192.168.2.157:~/
scp kube-4.myhypervisor.ca.kubeconfig kube-proxy.kubeconfig root@192.168.2.154:~/
scp admin.kubeconfig kube-controller-manager.kubeconfig kube-scheduler.kubeconfig root@192.168.2.151:~/
scp admin.kubeconfig kube-controller-manager.kubeconfig kube-scheduler.kubeconfig root@192.168.2.162:~/
```

Generating the Data Encryption Config

```

ENCRYPTION_KEY=$(head -c 32 /dev/urandom | base64)

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

scp

```
scp encryption-config.yaml root@192.168.2.151:~/
scp encryption-config.yaml root@192.168.2.162:~/
```

Creating the etcd Cluster

on both controller nodes

```
wget -q --show-progress --https-only --timestamping \
  "https://github.com/etcd-io/etcd/releases/download/v3.3.10/etcd-v3.3.10-linux-amd64.tar.gz"
tar -xvf etcd-v3.3.10-linux-amd64.tar.gz
mv etcd-v3.3.10-linux-amd64/etcd* /usr/local/bin/
mkdir -p /etc/etcd /var/lib/etcd
cp ca.pem kubernetes-key.pem kubernetes.pem /etc/etcd/
```

run on controller node1

```
ETCD_NAME=kube-1.myhypervisor.ca
INTERNAL_IP=192.168.1.8
INITIAL_CLUSTER=kube-1.myhypervisor.ca=https://192.168.1.8:2380,kube-
2.myhypervisor.ca=https://192.168.1.10:2380
```

run on controller node2

```
ETCD_NAME=kube-2.myhypervisor.ca
INTERNAL_IP=192.168.1.10
INITIAL_CLUSTER=kube-1.myhypervisor.ca=https://192.168.1.8:2380,kube-
```

```
2.myhypervisor.ca=https://192.168.1.10:2380
```

on both controller nodes

```
cat << EOF | tee /etc/systemd/system/etcd.service
[Unit]
Description=etcd
Documentation=https://github.com/coreos

[Service]
ExecStart=/usr/local/bin/etcd \\
  --name ${ETCD_NAME} \\
  --cert-file=/etc/etcd/kubernetes.pem \\
  --key-file=/etc/etcd/kubernetes-key.pem \\
  --peer-cert-file=/etc/etcd/kubernetes.pem \\
  --peer-key-file=/etc/etcd/kubernetes-key.pem \\
  --trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-trusted-ca-file=/etc/etcd/ca.pem \\
  --peer-client-cert-auth \\
  --client-cert-auth \\
  --initial-advertise-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-peer-urls https://${INTERNAL_IP}:2380 \\
  --listen-client-urls https://${INTERNAL_IP}:2379,https://127.0.0.1:2379 \\
  --advertise-client-urls https://${INTERNAL_IP}:2379 \\
  --initial-cluster-token etcd-cluster-0 \\
  --initial-cluster ${INITIAL_CLUSTER} \\
  --initial-cluster-state new \\
  --data-dir=/var/lib/etcd
Restart=on-failure
RestartSec=5

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

enable and start service

```
systemctl daemon-reload
systemctl enable etcd
systemctl start etcd
systemctl status etcd
```

check if working

```
ETCDCTL_API=3 etcdctl member list \
  --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/etcd/ca.pem \
  --cert=/etc/etcd/kubernetes.pem \
  --key=/etc/etcd/kubernetes-key.pem
```

if typo during config remove data in /var/lib/etcd/* and restart service (rm -rf /var/lib/etcd/*)

kubernetes controller bin

on both controllers

```
mkdir -p /etc/kubernetes/config

wget -q --show-progress --https-only --timestamping \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-apiserver" \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-controller-manager" \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-scheduler" \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kubectl"

chmod +x kube-apiserver kube-controller-manager kube-scheduler kubectl

mv kube-apiserver kube-controller-manager kube-scheduler kubectl /usr/local/bin/
```

Kubernetes API Server

on both controllers

```
sudo mkdir -p /var/lib/kubernetes/

sudo cp ca.pem ca-key.pem kubernetes-key.pem kubernetes.pem \
  service-account-key.pem service-account.pem \
  encryption-config.yaml /var/lib/kubernetes/

wget -q --show-progress --https-only --timestamping \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-apiserver" \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-controller-manager"
\
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-scheduler" \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kubectl"



wget -q --show-progress --https-only --timestamping \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-apiserver" \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-controller-manager"
\
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-scheduler" \
  "https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kubectl"

  service-account-key.pem service-account.pem \
  encryption-config.yaml /var/lib/kubernetes/
```

controller 1

```
INTERNAL_IP=192.168.1.8
CONTROLLER0_IP=192.168.1.8
CONTROLLER1_IP=192.168.1.10



#### controller 2
INTERNAL_IP=192.168.1.10
```

```
CONTROLLER0_IP=192.168.1.8
CONTROLLER1_IP=192.168.1.10

#### both
cat << EOF | 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=${INTERNAL_IP} \\
  --allow-privileged=true \\
  --apiserver-count=3 \\
  --audit-log-maxage=30 \\
  --audit-log-maxbackup=3 \\
  --audit-log-maxsize=100 \\
  --audit-log-path=/var/log/audit.log \\
  --authorization-mode=Node,RBAC \\
  --bind-address=0.0.0.0 \\
  --client-ca-file=/var/lib/kubernetes/ca.pem \\
  --enable-admission-
plugins=Initializers,NamespaceLifecycle,NodeRestriction,LimitRanger,ServiceAccount,DefaultStorageClass,Resou
rceQuota \\
  --enable-swagger-ui=true \\
  --etcd-cafile=/var/lib/kubernetes/ca.pem \\
  --etcd-certfile=/var/lib/kubernetes/kubernetes.pem \\
  --etcd-keyfile=/var/lib/kubernetes/kubernetes-key.pem \\
  --etcd-servers=https://$CONTROLLER0_IP:2379,https://$CONTROLLER1_IP:2379 \\
  --event-ttl=1h \\
  --experimental-encryption-provider-config=/var/lib/kubernetes/encryption-config.yaml \\
  --kubelet-certificate-authority=/var/lib/kubernetes/ca.pem \\
  --kubelet-client-certificate=/var/lib/kubernetes/kubernetes.pem \\
  --kubelet-client-key=/var/lib/kubernetes/kubernetes-key.pem \\
  --kubelet-https=true \\
  --runtime-config=api/all \\
  --service-account-key-file=/var/lib/kubernetes/service-account.pem \\
  --service-cluster-ip-range=10.32.0.0/24 \\
  --service-node-port-range=30000-32767 \\
  --tls-cert-file=/var/lib/kubernetes/kubernetes.pem \\
  --tls-private-key-file=/var/lib/kubernetes/kubernetes-key.pem \\
  --v=2 \\
  --kubelet-preferred-address-types=InternalIP,InternalDNS,Hostname,ExternalIP,ExternalDNS
Restart=on-failure
RestartSec=5

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

Kubernetes Controller Manager

on both controllers

```
cp kube-controller-manager.kubeconfig /var/lib/kubernetes/

cat << EOF | 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 \\
  --cluster-cidr=10.200.0.0/16 \\
  --cluster-name=kubernetes \\
  --cluster-signing-cert-file=/var/lib/kubernetes/ca.pem \\
  --cluster-signing-key-file=/var/lib/kubernetes/ca-key.pem \\
  --kubeconfig=/var/lib/kubernetes/kube-controller-manager.kubeconfig \\
  --leader-elect=true \\
  --root-ca-file=/var/lib/kubernetes/ca.pem \\
  --service-account-private-key-file=/var/lib/kubernetes/service-account-key.pem \\
  --service-cluster-ip-range=10.32.0.0/24 \\
  --use-service-account-credentials=true \\
  --v=2
Restart=on-failure
RestartSec=5

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

Kubernetes Scheduler

on both controllers

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

cat << EOF | tee /etc/kubernetes/config/kube-scheduler.yaml
apiVersion: kubescheduler.config.k8s.io/v1alpha1
kind: KubeSchedulerConfiguration
clientConnection:
  kubeconfig: "/var/lib/kubernetes/kube-scheduler.kubeconfig"
leaderElection:
  leaderElect: true
EOF

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

[Service]
ExecStart=/usr/local/bin/kube-scheduler \\
  --config=/etc/kubernetes/config/kube-scheduler.yaml \\
  --v=2
Restart=on-failure
RestartSec=5

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

```
EOF
```

Enable services

on both controllers

```
systemctl daemon-reload
systemctl enable kube-apiserver kube-controller-manager kube-scheduler
systemctl start kube-apiserver kube-controller-manager kube-scheduler
```

check status

on both controllers

```
systemctl status kube-apiserver kube-controller-manager kube-scheduler
kubectl get componentstatuses --kubeconfig admin.kubeconfig
```

Set up RBAC for Kubelet Authorization

On controller 1

Create a role with the necessary permissions:

```
cat << EOF | kubectl apply --kubeconfig admin.kubeconfig -f -
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  annotations:
    rbac.authorization.kubernetes.io/autoupdate: "true"
  labels:
    kubernetes.io/bootstrapping: rbac-defaults
  name: system:kube-apiserver-to-kubelet
rules:
  - apiGroups:
      - ""
    resources:
      - nodes/proxy
      - nodes/stats
      - nodes/log
      - nodes/spec
      - nodes/metrics
    verbs:
      - "*"
EOF
```

Bind the role to the kubernetes user:

```
cat << EOF | kubectl apply --kubeconfig admin.kubeconfig -f -
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: system:kube-apiserver
  namespace: ""
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:kube-apiserver-to-kubelet
subjects:
  - apiGroup: rbac.authorization.k8s.io
    kind: User
    name: kubernetes
```

```
EOF
```

Setting up a Kube API Frontend Load Balancer

on LB

```
sudo apt-get install -y nginx
sudo systemctl enable nginx
sudo mkdir -p /etc/nginx/tcpconf.d
sudo vi /etc/nginx/nginx.conf
```

Add the following to the end of nginx.conf:

```
include /etc/nginx/tcpconf.d/*;
```

Set up some environment variables for the lead balancer config file:

```
CONTROLLER0_IP=192.168.1.8
CONTROLLER1_IP=192.168.1.10
```

Create the load balancer nginx config file:

```
cat << EOF | sudo tee /etc/nginx/tcpconf.d/kubernetes.conf
stream {
    upstream kubernetes {
        server $CONTROLLER0_IP:6443;
        server $CONTROLLER1_IP:6443;
    }

    server {
        listen 6443;
        listen 443;
        proxy_pass kubernetes;
    }
}
EOF
```

Reload the nginx configuration:

```
sudo nginx -s reload
```

You can verify that the load balancer is working like so:

```
curl -k https://localhost:6443/version
```

Install bin on worker

on both worker nodes

```
sudo apt-get -y install socat conntrack ipset

wget -q --show-progress --https-only --timestamping \
  https://github.com/kubernetes-sigs/cri-tools/releases/download/v1.13.0/critest-v1.13.0-linux-amd64.tar.gz \
  https://storage.googleapis.com/kubernetes-the-hard-way/runsc \
  https://github.com/opencontainers/runc/releases/download/v1.0.0-rc6/runc.amd64 \
  https://github.com/containernetworking/plugins/releases/download/v0.7.4/cni-plugins-amd64-v0.7.4.tgz \
  https://github.com/containerd/containerd/releases/download/v1.2.2/containerd-1.2.2.linux-amd64.tar.gz \
  https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kubectl \
  https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kube-proxy \
  https://storage.googleapis.com/kubernetes-release/release/v1.13.0/bin/linux/amd64/kubelet

sudo mkdir -p \
  /etc/cni/net.d \
  /opt/cni/bin \
  /var/lib/kubelet \
  /var/lib/kube-proxy \
  /var/lib/kubernetes \
  /var/run/kubernetes

chmod +x kubectl kube-proxy kubelet runc.amd64 runsc
sudo mv runc.amd64 runc
sudo mv kubectl kube-proxy kubelet runc runsc /usr/local/bin/
sudo tar -xvf critest-v1.13.0-linux-amd64.tar.gz -C /usr/local/bin/
sudo tar -xvf cni-plugins-amd64-v0.7.4.tgz -C /opt/cni/bin/
sudo tar -xvf containerd-1.2.2.linux-amd64.tar.gz -C /
```

Install containerd

on both worker nodes

Create the containerd config.taml:

```
sudo mkdir -p /etc/containerd/

cat << EOF | sudo tee /etc/containerd/config.toml
[plugins]
  [plugins.cri.containerd]
    snapshotter = "overlayfs"
    [plugins.cri.containerd.default_runtime]
      runtime_type = "io.containerd.runtime.v1.linux"
      runtime_engine = "/usr/local/bin/runc"
      runtime_root = ""
    [plugins.cri.containerd.untrusted_workload_runtime]
      runtime_type = "io.containerd.runtime.v1.linux"
      runtime_engine = "/usr/local/bin/runsc"
      runtime_root = "/run/containerd/runsc"
EOF
```

Create the containerd unit file:

```
cat << EOF | sudo tee /etc/systemd/system/containerd.service
[Unit]
Description=containerd container runtime
Documentation=https://containerd.io
After=network.target

[Service]
ExecStartPre=/sbin/modprobe overlay
ExecStart=/bin/containerd
Restart=always
RestartSec=5
Delegate=yes
KillMode=process
OOMScoreAdjust=-999
LimitNOFILE=1048576
LimitNPROC=infinity
LimitCORE=infinity

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

Config kublelet

worker1

```
HOSTNAME=kube-3.myhypervisor.ca
sudo mv ${HOSTNAME}-key.pem ${HOSTNAME}.pem /var/lib/kubelet/
sudo mv ${HOSTNAME}.kubeconfig /var/lib/kubelet/kubeconfig
sudo mv ca.pem /var/lib/kubernetes/
```

worker2

```
HOSTNAME=kube-4.myhypervisor.ca
sudo mv ${HOSTNAME}-key.pem ${HOSTNAME}.pem /var/lib/kubelet/
sudo mv ${HOSTNAME}.kubeconfig /var/lib/kubelet/kubeconfig
sudo mv ca.pem /var/lib/kubernetes/
```

Create the kubelet config file:

```
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.pem"
authorization:
  mode: Webhook
clusterDomain: "cluster.local"
clusterDNS:
  - "10.32.0.10"
runtimeRequestTimeout: "15m"
tlsCertFile: "/var/lib/kubelet/${HOSTNAME}.pem"
tlsPrivateKeyFile: "/var/lib/kubelet/${HOSTNAME}-key.pem"
EOF
```

Create the kubelet unit file:

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

[Service]
ExecStart=/usr/local/bin/kubelet \\
  --config=/var/lib/kubelet/kubelet-config.yaml \\
  --container-runtime=remote \\
  --container-runtime-endpoint=unix:///var/run/containerd/containerd.sock \\
  --image-pull-progress-deadline=2m \\
  --kubeconfig=/var/lib/kubelet/kubeconfig \\
  --network-plugin=cni \\
  --register-node=true \\
  --v=2 \\
  --hostname-override=${HOSTNAME} \\
  --allow-privileged=true
Restart=on-failure
RestartSec=5

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

Config kube-proxy

Both workers

You can configure the kube-proxy service like so. Run these commands on both worker nodes:

```
sudo mv kube-proxy.kubeconfig /var/lib/kube-proxy/kubeconfig
```

Create the kube-proxy config file:

```
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
```

Create the kube-proxy unit 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
```

Now you are ready to start up the worker node services! Run these:

```
sudo systemctl daemon-reload
sudo systemctl enable containerd kubelet kube-proxy
sudo systemctl start containerd kubelet kube-proxy
```

Check the status of each service to make sure they are all active (running) on both worker nodes:

```
sudo systemctl status containerd kubelet kube-proxy
```

Finally, verify that both workers have registered themselves with the cluster. Log in to one of your control nodes and run this:

on controller node

```
kubectl get nodes
```

kubectl on workstation

```

ssh -L 6443:localhost:6443 root@192.168.2.170


kubectl config set-cluster kubernetes-the-hard-way \
  --certificate-authority=ca.pem \
  --embed-certs=true \
  --server=https://localhost:6443


kubectl config set-credentials admin \
  --client-certificate=admin.pem \
  --client-key=admin-key.pem


kubectl config set-context kubernetes-the-hard-way \
  --cluster=kubernetes-the-hard-way \
  --user=admin


kubectl config use-context kubernetes-the-hard-way
```

test commands

```
kubectl get pods
kubectl get nodes
kubectl version
```

Setup worker networking

on both worker nodes

```
sudo sysctl net.ipv4.conf.all.forwarding=1
```

Install Weave Net

on local workstation

```
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"
```

check if running

```
kubectl get pods -n kube-system
```

create an Nginx deployment with 2 replicas

```
cat << EOF | kubectl apply -f -
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      run: nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
EOF
```

create a service for that deployment so that we can test connectivity to services

```
kubectl expose deployment/nginx
```

start up another pod. We will use this pod to test our networking

```
kubectl run busybox --image=radial/busyboxplus:curl --command -- sleep 3600
POD_NAME=$(kubectl get pods -l run=busybox -o jsonpath="{.items[0].metadata.name}")
```

get the IP addresses of our two Nginx pods

```
kubectl get ep nginx
```

ouput looks like this

```
NAME    ENDPOINTS                     AGE
nginx   10.200.0.2:80,10.200.128.1:80   14s
```

check that busybox pod can connect to the Nginx pods on both of those IP addresses

```
kubectl exec $POD_NAME -- curl 10.200.0.2
kubectl exec $POD_NAME -- curl 10.200.128.1
```

Delete pods

```
kubectl delete deployment busybox
kubectl delete deployment nginx
kubectl delete svc nginx
```

Install kube-dns

from workstation or controller node

```
kubectl create -f https://storage.googleapis.com/kubernetes-the-hard-way/kube-dns.yaml
```

Verify that the kube-dns pod starts up correctly

```
kubectl get pods -l k8s-app=kube-dns -n kube-system
```

Revision #3
Created 1 February 2019 18:59:45 by Dave
Updated 10 February 2019 19:24:17 by Dave