Initial Setup:

VPC – CIDR: 10.0.0.0/16

Private Subnet1: 10.0.16.0/20

Private Subnet2: 10.0.32.0/24

Public subnet: 10.0.0.0/22

VMs:

|  |  |  |  |
| --- | --- | --- | --- |
| VM name | Public IP | Private IP | Type |
| HAProxy | 13.127.89.42 | 10.0.2.71 | T2.nano |
| Master-01 | 65.0.44.152 | 10.0.3.39 | T2.medium |
| Master-02 | 15.207.203.190 | 10.0.1.201 | T2.medium |
| Worker01 | 3.6.146.99 | 10.0.2.66 | T2.medium |
| Worker02 | 13.233.222.45 | 10.0.0.210 | T2.medium |

Security Group:

Master security group

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Port | Source | Purpose | Used By |
| Custom TCP | 6443 | 0.0.0.0/0 | Kube-api server | All |
| SSH | 22 | 0.0.0.0/0 |  |  |
| Custom TCP | 2379-2380 | 0.0.0.0/0 | Etcd server client API | Kube-apiserver, etcd |
|  |  |  |  |  |

Worker security group

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Type | Port | Source | Purpose | Used By |
| SSH | 22 | 0.0.0.0/0 |  |  |
| Custom TCP | 30000-32767 | 0.0.0.0/0 | Node port services | All |
|  |  |  |  |  |

K8S – 1 Master, 1 Worker

**Step 1**: Commands to be executed on all nodes (master,worker):

1. Disable swap – For kubeadm to work properly, disable swap on all the nodes:

sudo swapoff -a

(crontab -l 2>/dev/null; echo "@reboot /sbin/swapoff -a") | crontab - || true

fstab entry will make sure the swap is off on system reboots.

1. Enable IP bridges traffic on all the nodes:

cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

# sysctl params required by setup, params persist across reboots

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

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

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

net.ipv4.ip\_forward = 1

EOF

# Apply sysctl params without reboot

sudo sysctl –system

1. Installing CRIO runtime:

The basic requirement for a Kubernetes cluster is a [container runtime](https://devopscube.com/what-is-docker/). You can have any one of the following container runtimes.

* CRI-O
* Containerd
* Docker Engine (Using cri-dockerd) – deprecated

Execute below commands to install required dependencies and the latest version of CRIO.

## Install CRIO Runtime

sudo apt-get update -y

sudo apt-get install -y software-properties-common curl apt-transport-https ca-certificates gpg

sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg] https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-o.list

sudo apt-get update -y

sudo apt-get install -y cri-o

sudo systemctl daemon-reload

sudo systemctl enable crio --now

sudo systemctl start crio.service

1. Install kubelet, kubectl and kubeadm on all the node:

# Add kubernetes signing key and add kubernetes repository on nodes

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

# Use apt-get command to install kubeadm, kubelet and kubectl packages with specific version.

sudo apt-get update -y

sudo apt-get install -y kubelet="1.29.0-\*" kubectl="1.29.0-\*" kubeadm="1.29.0-\*"

sudo apt-get update -y

sudo apt-get install -y jq

sudo systemctl enable --now kubelet

sudo systemctl start kubelet

**Step 2**: Commands to be executed only on Master node:

1. Initialize kubeadm on master node to setup control plane

Here, there are 2 options:

* Master node with private ip –

If we have nodes with only private IP addresses the API server would be accessed over the private IP of the master node.

IPADDR="10.0.0.10"

NODENAME=$(hostname -s)

POD\_CIDR="192.168.0.0/16"

sudo kubeadm init **--apiserver-advertise-address**=$IPADDR --apiserver-cert-extra-sans=$IPADDR --pod-network-cidr=$POD\_CIDR --node-name $NODENAME --ignore-preflight-errors Swap

--ignore-preflight-errors Swap is actually not required as we disabled the swap initially.

* Master node with public ip –

If we are setting up a Kubeadm cluster on Cloud platforms and we need master Api server access over the Public IP of the master node server.

IPADDR=$(curl ifconfig.me && echo "")

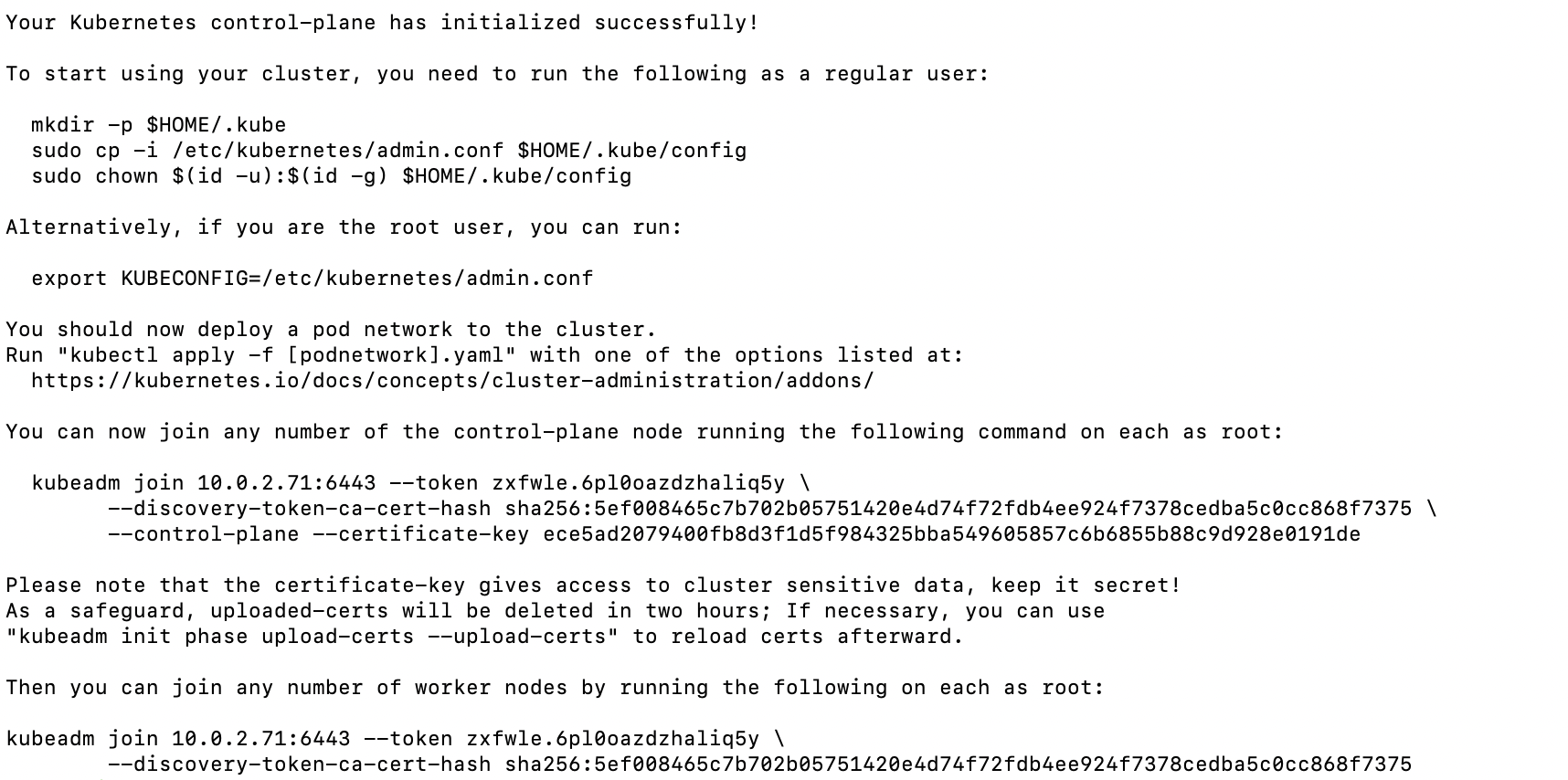
NODENAME=$(hostname -s)

POD\_CIDR="192.168.0.0/16"

sudo kubeadm init **--control-plane-endpoint**=$IPADDR --apiserver-cert-extra-sans=$IPADDR --pod-network-cidr=$POD\_CIDR --node-name $NODENAME --ignore-preflight-errors Swap

Here instead of *--apiserver-advertise-address* we use *--control-plane-endpoint* parameter for the API server endpoint.

On successful kubeadm initialization, we get an output with *kubeconfig file*location and the **join command with the token** as shown below. Copy that and save it to the file. we will need it for **joining the worker node to the master**.



A

B

C

Use the **commands (A) from the output**to create the kubeconfig in master so that we can use kubectl to interact with cluster API.

Now, verify the kubeconfig by executing the following kubectl command to list all the pods in the kube-system namespace.

kubectl get po -n kube-system

We should see the following output. You will see the two **Coredns pods in a pending state.** It is the expected behavior. Once we **install the network plugin**, it will be in a running state.

1. Install calico network plugin for Pod Networking

Kubeadm does not configure any network plugin. We need to install a network plugin of our choice for kubernetes pod networking and enable network policy.

Command to install calico network plugin:

kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

Once installed, we can check the coreDNS pods would be in running state.

**Step 3**: Executing the JOIN command on worker node:

1. If we missed copying the join command, execute the following command in the **master node** to recreate the token with the join command.

kubeadm token create --print-join-command

1. Paste the worker node join commands with “sudo” and execute

Using the join commands, we can add additional master node.

Misc:

Execute below command to label a worker node

kubectl label node <node-name> node-role.kubernetes.io/worker=worker

Ref:

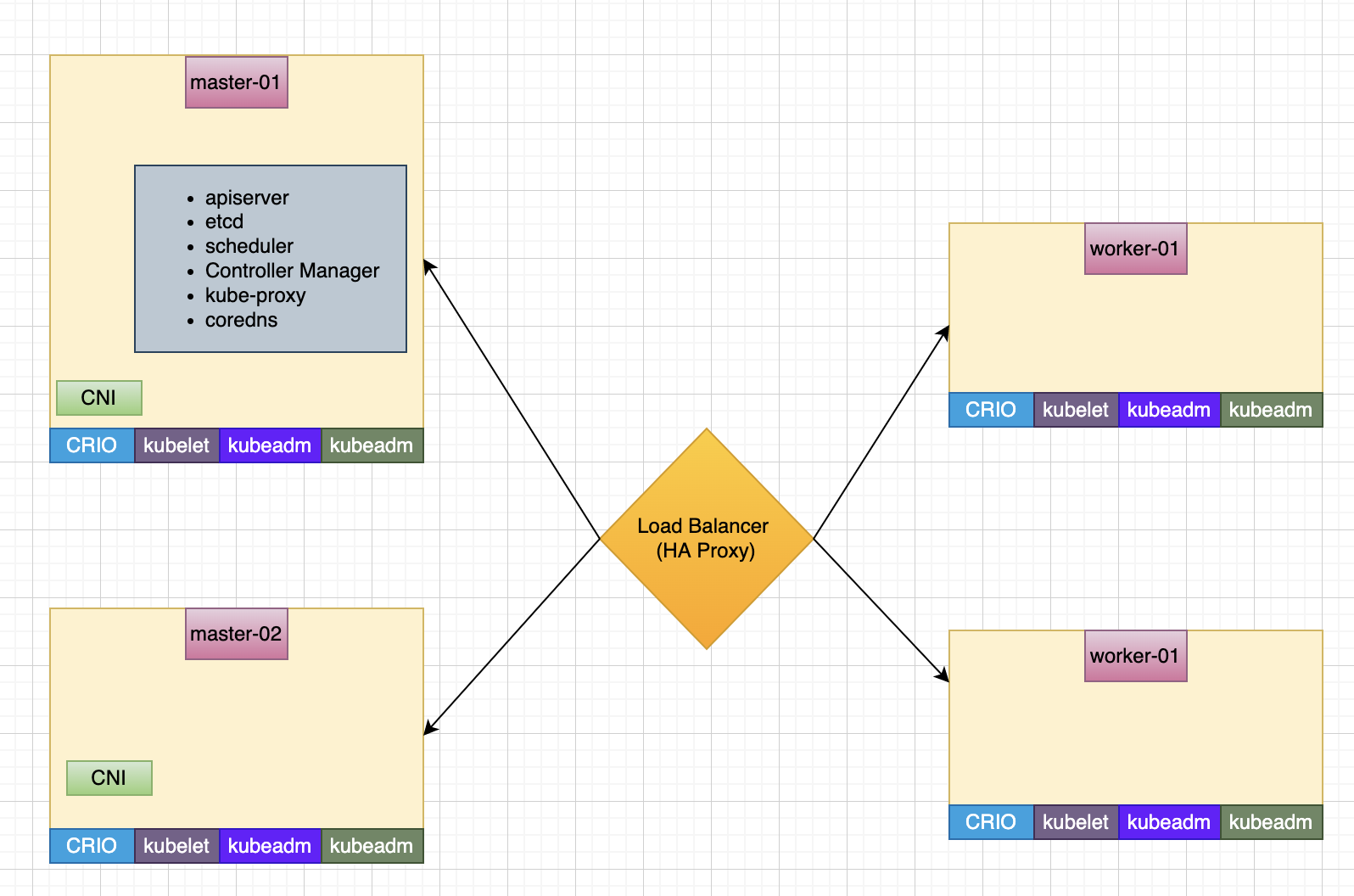
<https://www.youtube.com/watch?v=xX52dc3u2HU>

<https://devopscube.com/setup-kubernetes-cluster-kubeadm/>

https://github.com/LondheShubham153/kubestarter/blob/main/kubeadm\_installation.md

K8S – Multi Master Setup

For multimaster setup, we need a load balancer. Here we would use HAProxy as load balancer.



**HAProxy Setup**:

1. Use apt-get command to install HAProxy package as below.

apt update && apt install -y haproxy

1. Configure HAProxy

Configure HAProxy to forward the traffic across masternodes. Edit /etc/haproxy/haproxy.cfg to define the backend and frontend configuration. Append the below information:

frontend kubernetes

bind <private ip of HA Proxy>:6443

option tcplog

mode tcp

default\_backend kubernetes-master-nodes

backend kubernetes-master-nodes

mode tcp

balance roundrobin

option tcp-check

server kubernetes-master1 <privateip of master01>:6443 check fall 3 rise 2

server kubernetes-master2 <privateip of master02>:6443 check fall 3 rise 2

1. Use systemctl command to start and enable the HAProxy:

$sudo systemctl start haproxy

$sudo systemctl enable haproxy

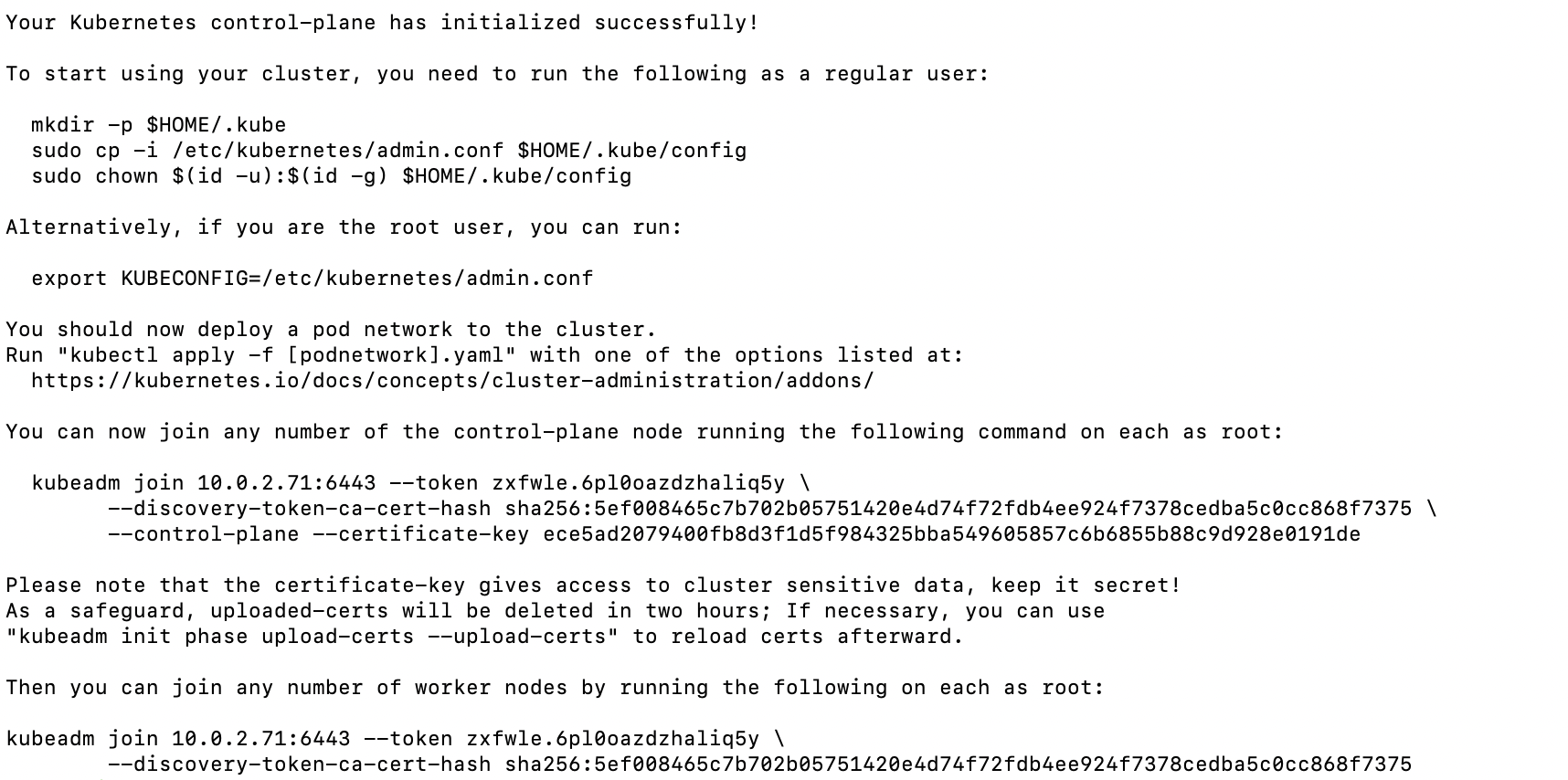
**kubeadm init Setup (only on 1 master node)**:

IPADDR=<public ip of loadbalancer>

POD\_CIDR="192.168.0.0/16"

sudo kubeadm init **--control-plane-endpoint**=$IPADDR –uploadcerts –apiserver-advertise-address <privare address master01> --pod-network-cidr=$POD\_CIDR

On successful kubeadm initialization, we get an output with *kubeconfig file*location and the **join command with the token** as shown below. Copy that and save it to the file. we will need it for **joining the worker node to the master**.



A

B

C

Once Kubernetes cluster initialization is completed, Copy the join marked B to join any number of the control-plane node. We need to append

-–apiserver-advertise-address <privare address of the node>

Copy the other join token marked C to any number of worker nodes.

Ref:

<https://www.youtube.com/watch?v=SueeqeioyKY>

<https://www.learnitguide.net/2021/10/kubernetes-multi-master-setup-with.html>