1. 規格配置

Master IP: 10.118.253.85

Node1 IP: 10.118.253.90

Node2 IP: 10.118.253.99

Node3 IP: 10.118.253.89

1. 前置作業
   1. 關閉swap(in root)

swapoff –a

vi /etc/fstab

* 1. vi /etc/hosts

(所有server都要加)

[Server ip server hostname]

10.118.253.85 cbn-k8smaster

10.118.253.90 cbn-k8sNode1

10.118.253.99 cbn-k8sNode2

* 1. Docker安裝(已安裝可跳過)(所有server都要安裝)

sudo apt-get update

sudo apt-get install -y apt-transport-https curl

sudo apt-get install docker.io –y

* 1. 啟動服務

sudo systemctl enable docker

sudo systemctl start docker

sudo systemctl status docker

1. 安裝kubectl、kubelet、kubeadm(in root)(ALL server)
   1. apt-key

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

* 1. kubernetes.list

cat <<EOF >/etc/apt/sources.list.d/kubernetes.list

deb http://apt.kubernetes.io/ kubernetes-xenial main

EOF

* 1. 安裝(所有server都要安裝)

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

* 1. 開機啟用

sudo systemctl enable kubelet

1. Master配置
   1. Profile配置

sudo vi /etc/profile

在最後加上

export KUBECONFIG=/etc/kubernetes/admin.conf

* 1. 重新加載

sudo systemctl daemon-relod

* 1. 初始化(要稍等一下)

kubeadm init --pod-network-cidr=10.244.0.0/16 --apiserver-advertise-address=10.118.253.85 --kubernetes-version=v1.20.0 --ignore-preflight-errors=swap

註：出現” Your Kubernetes control-plane has initialized successfully”表示成功

--pod-network-cidr 配置節點中的pod的可用IP Address，此為內部IP

--apiserver-advertise-address 寫master server ip

--kubernetes-version 寫kubernetes的版本



* 1. 初始化後欲使用cluster前配置(非root)

sudo cp /etc/kubernetes/admin.conf $HOME/

sudo chown $(id -u):$(id -g) $HOME/admin.conf

export KUBECONFIG=$HOME/admin.conf

* 1. 查看節點

kubectl get nodes

註：此時應為NotReady

* 1. 網路配置(root)

kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

* 1. 重新查看節點，應為Ready

kubectl get nodes



* 1. 生成token，並記下來

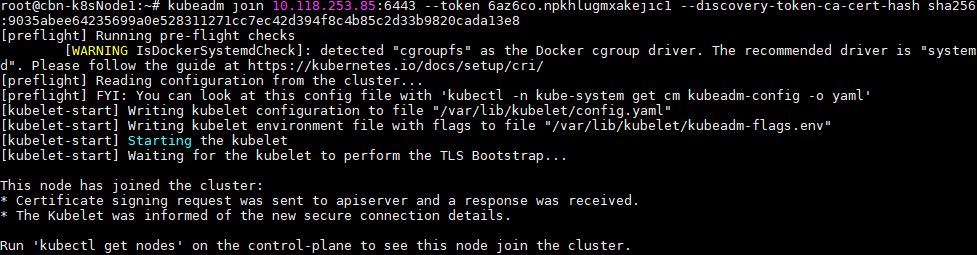
kubeadm token create --print-join-command



1. Node配置
   1. 加入Cluster(每一台node都要加)(Root)

[kubeadm join **master\_ip**:6443 --token **token\_name** --discovery-token-ca-cert-hash **token-ca-cert-hash\_name**]

kubeadm join 10.118.253.85:6443 --token 6az6co.npkhlugmxakejic1 --discovery-token-ca-cert-hash sha256:9035abee64235699a0e528311271cc7ec42d394f8c4b85c2d33b9820cada13e8

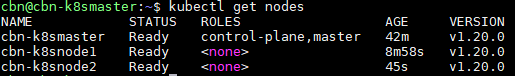


註：如要確認token(會在生成後24小時後失效)：

kubeadm token list

* 1. 回到master server確認

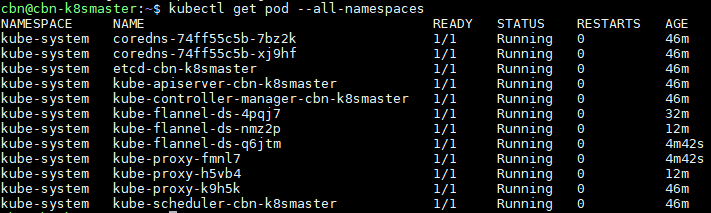
kubectl get nodes



註：確定STATUS都READY即表示node加入k8s cluster

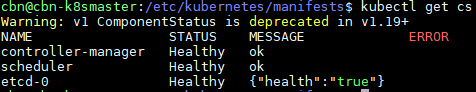
註：如後續須要加入新node server，須重複2.前置作業再進行5-A

kubectl get pod --all-namespaces

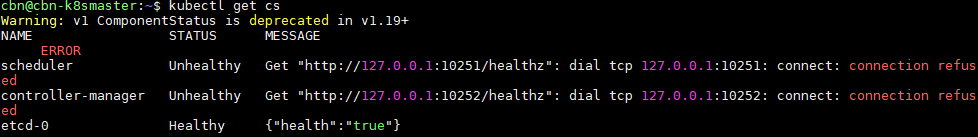


註：若部分服務沒有正常啟動，一般原因為缺少Images，需手動下載

1. 其他確認
   1. kubectl get cs



註：若出現如下scheduler與controller-manager出現error



修改以下兩份文件，並將裡面的--port=0註釋起來，如此可開啟10251與10252 port，再次檢查kubectl get cs應可呈現healthy

sudo vi /etc/kubernetes/manifests/kube-scheduler.yaml

sudo vi /etc/kubernetes/manifests/kube-controller-manager.yaml

* **如果master斷電或重開機，k8s如果未重啟**

1. sudo systemctl status kubelet.service (確認是否active)
2. 如1.不是active，運行sudo systemctl start kubelet.service
3. sudo systemctl status docker.service (同上1.2.步驟確認)
4. kubectl get node

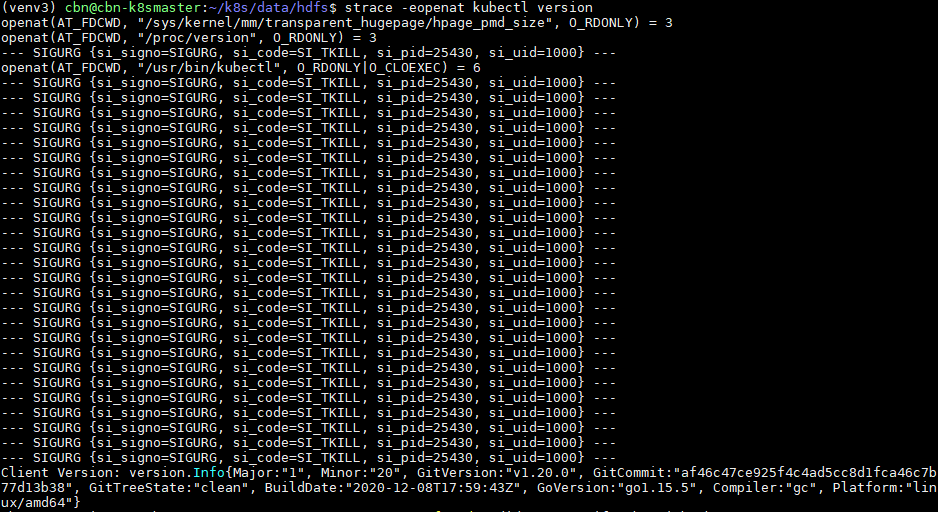
如發生The connection to the server [master IP]:6443 was refused - did you specify the right host or port?運行以下

1. sudo -i (進入root)

swapoff -a (關掉swap)

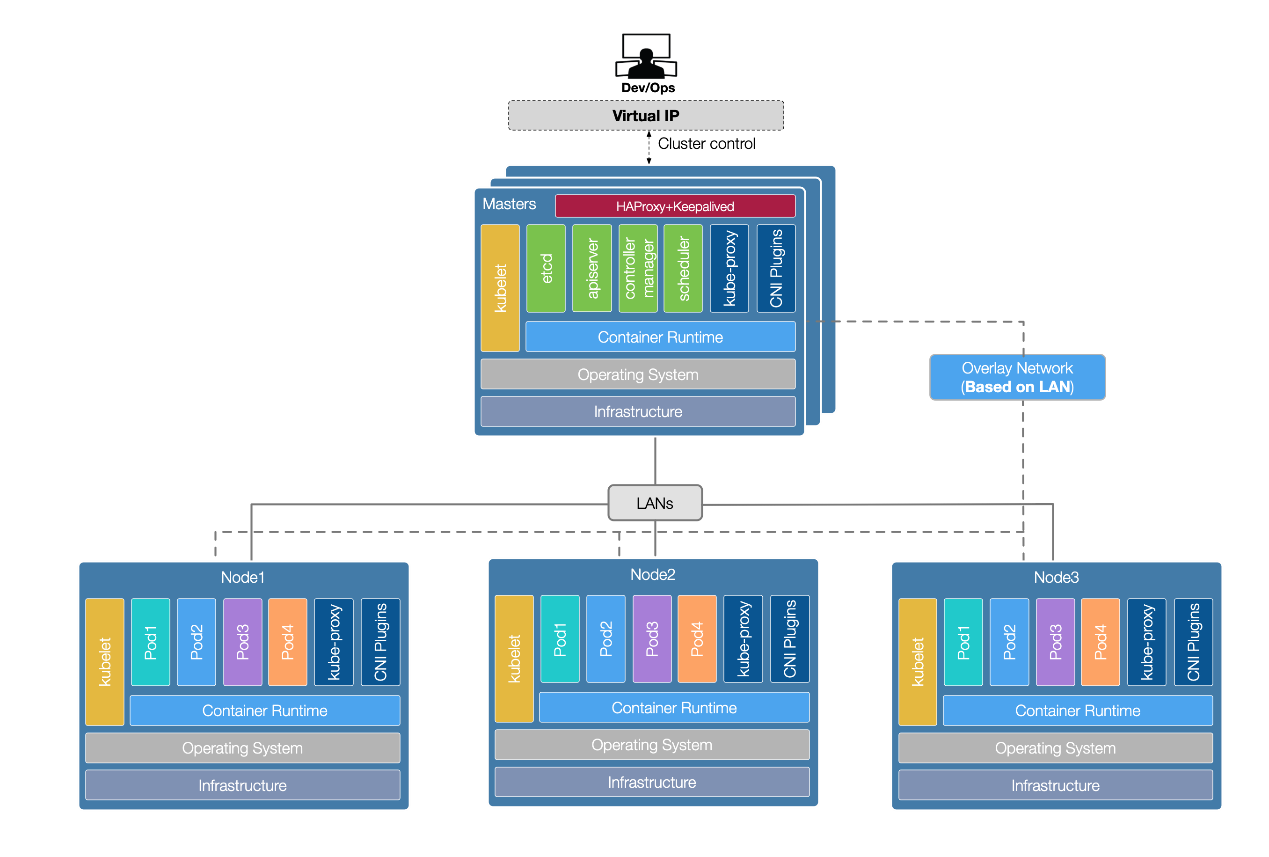
exit (離開root)

1. strace -eopenat kubectl version



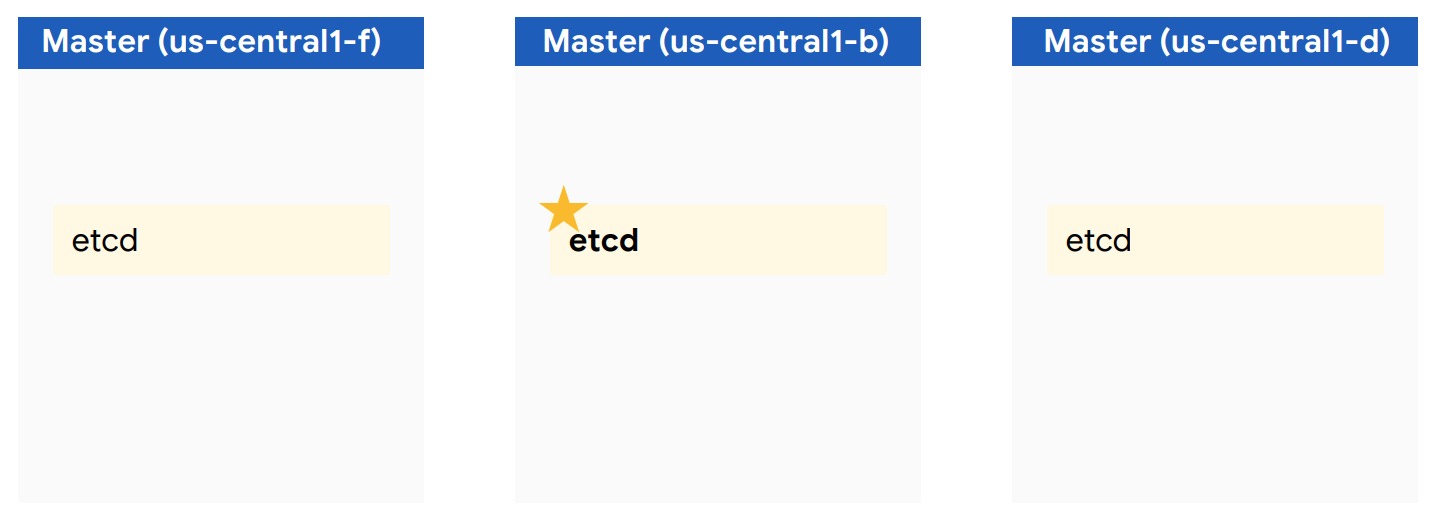
稍等一下即可開啟kubelet

* 補充



* etcd: 透過多節點的 etcd 實例組成叢集，並利用 Raft 演算法，來選取一個領導者(Leader)處理需要叢集共識的所有客戶端的請求(Request)。另外由於 Raft 演算法關析，還需要注意叢集的故障容許度(Failure Tolerance)。

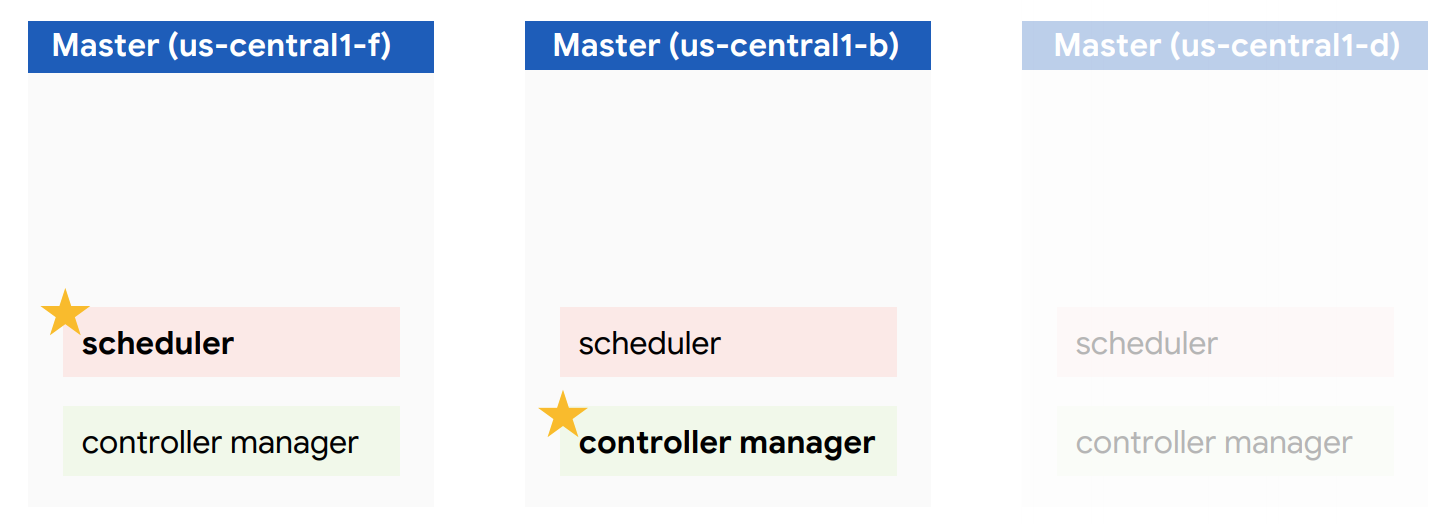
計算故障容許節點數為(N/2)+1，其中 N 為叢集大小。



* API server: 每個 API server 會與本地端的 etcd 溝通，並接收來至客戶端與其他元件的 API 請求。由於 API server 屬於 Active-Active 架構，因此每個 API server 在叢集中都處於可用狀態。



* controllers, scheduler: 這些元件採用 Lease 機制來從所有實例中選取一個作為領導者，並由領導者處理監聽對應的 API 資源來完成功能，因此整個叢集只會有一個擁有完整功能，除非原本領導的節點發生故障，才會尤其它接手。



* kubelet: 由於 kubelet 只能設定跟一個 API server 的端點(Endpoint)，但為了達到某個 API server 故障時，還能夠繼續正常執行的需求，我們需要提供一個虛擬 IP(VIP, Virtual IP)，以及負載平衡器(Load Balancer)來讓 kubelet 能夠存取多個 API server，一方面利用負載平衡器的機制來分散工作負載到所有 API server 上。

另外由於 API servers 需要提供 VIP 與負載平衡器，因此必須在所有主節點上額外安裝以下元件來達到需求。

* Keepalived: 基於 VRRP 協定來實現高可靠架構，所有主節點會基於此元件舉出一個 VIP 來作為存取 API 的端點，這主要是確保其他元件連接 API 時，不會因為某個主節點中斷而無法存取。
* HAProxy: 與 API server 一樣，為 Active-Active 架構，因此每個主節點都可以存取作為 Proxy 的 IP 與 Port。