필기(20241203) - Ubuntu Kubernetes 실습

기초 작업

`Kubernetes 서버를 구축하기 위해 ubuntu 가상 머신 3개를 생성 후 호스트 네임을 지정`

마스터노드(k8s-control-node) 는 CPU가 최소 2개 이상 필요합니다.

---

1. Set Hostname & Update Hosts File

`Configure the master and worker nodes' hostnames and update the hosts file for network communication.`

- 각 머신에 host name 지정

```
# 메인 머신
`sudo hostnamectl set-hostname "k8s-control-node"      // Master Node

# 머신 2
`sudo hostnamectl set-hostname "k8s-worker01-node"    // Worker Node 1

# 머신 3
`sudo hostnamectl set-hostname "k8s-worker02-node"    // Worker Node 2
```

- 3개의 머신의 hosts 파일 수정

```
# 호스트 파일 수정
vi /etc/hosts

# 머신들의 dns 입력 (모든 머신에 동일하게 추가)
192.168.2.135  k8s-control-node
192.168.2.136  k8s-worker01-node
192.168.2.137 k8s-worker02-node
```

---

2 Disable Swap & Load Kernel Modules

`Disable swap memory and configure kernel modules like overlay and br_netfilter for Kubernetes.`

- 3개의 머신에 스왑 및 오버레이 설정 변경

```
# 3개의 머신에 스왑 기능 정지 및 부팅시 스왑 기능 정지
sudo swapoff -a && sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

# 3개의 머신에 bernetes에 대한 overlay 및 br_netfilter와 같은 커널 모듈을 구성
sudo modprobe overlay && sudo modprobe br_netfilter
```

- 3개의 머신에 conf 파일 수정

```
sudo tee /etc/modules-load.d/k8s.conf << EOF
overlay
br_netfilter
EOF

sudo tee /etc/sysctl.d/kubernetes.conf << EOF
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
EOF
```

- 3개의 머신에 sysctl 설정

```
sudo sysctl --system
```

---

3. Install Containerd

`Install and configure containerd with SystemdCgroup to manage container runtime.`

- 3개의 머신에 Containerd 설치

```
# 의존성 설치
sudo apt install -y curl gnupg2 software-properties-common apt-transport-https
ca-certificates

#
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmour -o /etc/apt/trusted.gpg.d/containerd.gpg

#
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"

#
sudo apt update && sudo apt install containerd.io -y

#
containerd config default | sudo tee /etc/containerd/config.toml

#
sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g'
/etc/containerd/config.toml

# 완료 후 contatilnerd 재시작
sudo systemctl restart containerd
```

---

4. Add Kubernetes Package Repository

`Download and configure the Kubernetes package repository for Ubuntu 24.04.`

```
# v1.30 버젼 gpg 키값 추가 (버젼 명을 변경해서 다른 버젼을 설치가능)
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo gpg
--dearmor -o /etc/apt/keyrings/k8s.gpg

# v1.31 버젼 저장소 추가 (1.31이 최신 버젼)
echo 'deb [signed-by=/etc/apt/keyrings/k8s.gpg]
https://pkgs.k8s.io/core:/stable:/v1.31/deb/ /' | sudo tee
/etc/apt/sources.list.d/k8s.list
```

5. Install Kubernetes Components

`Install Kubeadm, Kubelet, and Kubectl on all nodes to manage your Kubernetes cluster.`

```
#
sudo apt update

#
sudo apt install kubelet kubeadm kubectl -y
```

6. Initialize the Kubernetes Cluster

`Initialize the control plane node using Kubeadm.`
`On the master node, run following set of commands.`
이 작업은 '마스터노드' 에서만 적용하면 됩니다.

- 초기화 작업

```
# k8s-control-node 는 마스터 노드 이름
sudo kubeadm init --control-plane-endpoint=k8s-control-node
```

- 만약 CPU 수가 2개 이상이 아니여서 에러가 발생한 경우

```
# CPU 갯수 무시하고 초기화 작업
sudo kubeadm init --control-plane-endpoint=k8s-control-node --ignore-
preflight-errors=NumCPU
```

- 만약에 초기화 작업을 잘못해서 처음부터 해야할 경우

```
# kubeadm 리셋
sudo kubeadm reset

# 남은 파일들 삭제
sudo rm -rf /etc/kubernetes/
sudo rm -rf ~/.kube
```

```
sudo rm -rf /var/lib/etcd

# 다시 초기화 작업 시작 (CPU 무시)
sudo kubeadm init --control-plane-endpoint=k8s-control-node --ignore-
preflight-errors=NumCPU
```

- 작업 후 로그 상단에 successfully 가 뜨는지 확인

```
Your Kubernetes control-plane has initialized succes
sfully!
```

- 초기화 작업 이후 입력

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

---

7. Join Worker Nodes

`Add worker nodes to your Kubernetes cluster using the token generated during initialization.`

이 작업은 두개의 '작업 노드' 에서 만 적용 하면됩니다.

- 노드를 연결 (마스터 노드 콘솔에서 써야할 명령어와 키값을 모두 보여주니 그걸 복사해서 붙여 넣는게 더 좋습니다.)

```
kubeadm join k8s-control-node:6443 --token 9lvl11.s0
weixgb5rgygpui \
        --discovery-token-ca-cert-hash sha256:8afbb5
bb3715eb689179c9e5bdfbb82a2f908042ea37c7754d2e5a75d1
60b2a2
```

```
# xxx 에 마스터 노드에서 출력된 키값을 입력
sudo kubeadm join k8s-master-noble:6443 --token
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

# ex (콘솔 창에 명령어와 키값이 모두 출력되니 복사 붙여넣기)
kubeadm join k8s-control-node:6443 --token 9lvl11.s0weixgb5rgygpui \
```

```
        --discovery-token-ca-cert-hash
sha256:8afbb5bb3715eb689179c9e5bdfbb82a2f908042ea37c7754d2e5a75d160b2a2
```

`Now head back to the master node and run kubectl get nodes command to verify the status of worker nodes.`

- 연결된 노드를 확인 (마스터 노드)

```
# 연결된 노드 확인
kubectl get nodes
```

- kubectl get nodes 명령어 입력 결과

```
root@k8s-control-node:/home/ygh# kubectl get nodes
NAME                STATUS   ROLES           AGE    VERSION
k8s-control-node    Ready    control-plane   133m   v1.31.3
k8s-worker01-node   Ready    <none>          123m   v1.31.3
k8s-worker02-node   Ready    <none>          123m   v1.31.3
root@k8s-control-node:/home/ygh#
```

`3 node(STATUS NotReady) 로 뜨는건 정상`

---

8. Install Calico Network Plugin

- 네트워크 플러그인 설치

```
kubectl create -f
https://raw.githubusercontent.com/projectcalico/calico/v3.28.2/manifests/calic
o.yaml
```

`After the successful installation of calico, nodes status will change to Ready in a minute or two.`

9. Test Kubernetes Installation

`Create and expose an NGINX deployment to verify the setup.`

- NGINX 서비스를 등록하여 테스트

```
kubectl create ns demo-app
```

```
kubectl create deployment nginx-app --image nginx --replicas 2 --namespace
demo-app

kubectl get deployment -n demo-app

kubectl get pods -n demo-app

kubectl expose deployment nginx-app -n demo-app --type NodePort --port 80
```

- 동작 서비스 확인 하기

```
# 동작중인 앱을 확인 (포트도 확인)
kubectl get svc -n demo-app

# 등록 후 2개의 worker node id 중 아무 노드로 접속
http://[workerNodIp]:[위에서 확인한 포트]/
```

- kubectl get svc -n demo-app 명령어 실행 결과