Kubernetes 설치 및 서버구축

구축 목적

: 도커의 기능을 이용하여 컨테이너를 동적으로 관리 및 자동화 하기 위하여(오케스트레이션)

- kubeadm : 클러스터를 부트스트랩하는 명령이다.
- kubelet : 클러스터의 모든 머신에서 실행되는 pod와 container시작과 같은 작업을 수행하는 컴포넌트이다.
- kubectl : 클러스터와 통신하기 위한 커맨드 라인 유틸리티이다.

Kubernetes 환경 구축

구축 전!!

```
* 설치버전 : Docker  18.06.2.ce
           kubernetes  1.15.5
```

1. 설치에 앞서 Master 서버 1대와 Worker 서버 2대를 만들고 시작한다.

2. 각 서버 모두 Docker를 설치해야한다.

- 링크 : [Docker 설치][link]
    - 포스팅과 다른 Docker 18.06.2.ce 버전으로 설치해야함!
      [link]:https://github.com/Dawon2/Docker/blob/main/Docker%20%EC%84%A4%EC%B9%98%20%EB%B0%8F%20%EC%BB%A8%ED%85%8C%EC%9D%B4%EB%84%88%20%EC%82%AC%EC%9A%A9%EB%B2%95.md

3. 방화벽 포트 추가 설정

- Master 노드 : 6443, 2379-2380, 10250, 10251, 10252
- Worker 노드 : 10250, 30000-32767
- pod network add-on (Master, Worker) : 179

---

1. 노드 구성

```
# hostnamectl set-hostname master
# hostnamectl set-hostname worker1
# hostnamectl set-hostname worker2
( 각 노드별 호스트네임 변경 )

# vi /etc/hosts
-----------------------
192.168.37.131 master
192.168.37.132 worker1
```

```
192.168.37.133 worker2
-----------------------
( 모든 노드에 /etc/hosts에 해당 내용 추가 )
```

2. Kubernetes 설치

```
# cat << EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
( 쿠버네티스 레포지토리 추가 )

# vi /etc/selinux/config
-----------------------
7 SELINUX=permissive
-----------------------
( permissive로 변경 )

# sestatus

# yum -y install --disableexcludes=kubernetes kubeadm-1.15.5-0.x86_64 kubectl-
1.15.5-0.x86_64 kubelet-1.15.5-0.x86_64
( 1.15.5 버전으로 설치 ) - 설치중 yum 오류 발생 시 아래 내용 참조

# systemctl enable kubelet
# systemctl start kubelet

# cat << EOF >  /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
# sysctl --system
( 컨테이너의 네트워크 패킷이 호스트머신의 iptables 설정에 따라 제어되도록 하는 설정 )

# swapoff -a
# sed s/\\/dev\\/mapper\\/centos-swap/#\ \\/dev\\/mapper\\/centos-swap/g -i
/etc/fstab
( swap이 off 되어 있지 않으면 init 단계에서 오류 발생 )

# kubeadm init --pod-network-cidr=172.16.0.0/16
( 쿠버네티스 초기화 - pod에 172.16 대역 부여 )
-> 정상적으로 init 된 후에 맨 마지막줄에 kubeadm join 명령어 줄 복사

# mkdir -p $HOME/.kube
# cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
```

```
# chown $(id -u):$(id -g) $HOME/.kube/config
( root 계정을 이용해서 kubectl을 실행하기 위한 환경 변수를 설정 )
```

3. pod network add-on 설치

> Pod들이 통신하기 위해 CNI를 기반으로 만들어진 시스템

```
-Master 노드-

# kubectl get pods -n kube-system
NAME                              READY   STATUS    RESTARTS   AGE
coredns-576cbf47c7-2fkrr          0/1     Pending   0          71s
coredns-576cbf47c7-bmns4          0/1     Pending   0          71s
( coredns가 Pending 상태임을 확인 )

<flannel>
# kubectl edit cm coredns -n kube-system
-----------
24 #loop
-----------
( loop 부분 주석 처리 )

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


<Calico>
# curl -O https://docs.projectcalico.org/v3.9/manifests/calico.yaml
# sed s/192.168.0.0\\/16/172.16.0.0\\/16/g -i calico.yaml
# kubectl apply -f calico.yaml

> 설정 후 잠시 기다렸다가 다시 확인해보면 Running 상태로 변경 확인
```

4. kubeadm join

> Worker 노드가 되어 Master 노드와 연결하기 위하여 등록

```
-Worker 노드-

# kubeadm join 172.17.120.243:6443 --token w2zi5z.rsdpqftvorhll3rr \
    --discovery-token-ca-cert-hash
sha256:23678857db6f67c361b2c1be9d5c8a8acccb30b98ca4b38fe481737ba45821be
( 각 Worker 노드에 위에서 init 후에 발급받은 join 명령어 붙여넣기 )

-Master 노드-

# kubectl get nodes
( Worker 노드 정상 등록 확인 )
```

5. Kubernetes Dashboard 설치

```
# mkdir ~/certs
# cd ~/certs

# openssl genrsa -des3 -passout pass:x -out dashboard.pass.key 2048
# openssl rsa -passin pass:x -in dashboard.pass.key -out dashboard.key
# openssl req -new -key dashboard.key -out dashboard.csr
# openssl x509 -req -sha256 -days 365 -in dashboard.csr -signkey dashboard.key -
out dashboard.crt
( 인증서 생성 )

# kubectl create secret generic kubernetes-dashboard-certs --from-file=$HOME/certs
-n kube-system
# kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/recommen
ded/kubernetes-dashboard.yaml
( HTTPS를 사용하는 Recommended setup 방식으로 설치 )

# kubectl edit service kubernetes-dashboard -n kube-system
-----------------------------------
     20   clusterIP: 10.109.168.90
     21   externalTrafficPolicy: Cluster
     22   ports:
     23   - nodePort: 31055
     24     port: 443
     25     protocol: TCP
     26     targetPort: 8443
     27   selector:
     28     k8s-app: kubernetes-dashboard
     29   sessionAffinity: None
     30   type: NodePort
     31 status:
     32   loadBalancer: {}
-----------------------------------
( 대시보드 설정 변경 ->   23 - nodePort: 31055
                          30   type: NodePort  )

# kubectl get service -n kube-system
( kubernetes-dashboard -> NodePort로 변경, Port 번호 확인 )

# kubectl create serviceaccount cluster-admin-dashboard-sa
# kubectl create clusterrolebinding cluster-admin-dashboard-sa --
clusterrole=cluster-admin --serviceaccount=default:cluster-admin-dashboard-sa
( 대시보드 계정 생성 )

# kubectl get secret $(kubectl get serviceaccount cluster-admin-dashboard-sa -o
jsonpath="{.secrets[0].name}") -o jsonpath="{.data.token}" | base64 --decode
( 대시보드 로그인 시 필요한 토큰 생성 -> 복사해놓기 )
```

- https://[노드 IP]:31055 로 접속!

-> 위에서 복사한 토큰으로 대시보드 계정 로그인 확인!

---

* 각종 에러 해결 모음

```
Unable to connect to the server: x509: ~~
-> export KUBECONFIG=/etc/kubernetes/admin.conf

~~ "CustomResourceDefinition" in version "apiextensions.k8s.io/v1beta1"
-> sed -i 's/v1beta1/v1/g' calico.yaml

One of the configured repositories failed (Kubernetes) - yum 에러
-> # gpgcheck=0
   # repo_gpgcheck=0

NotReady
-> 위의 flannel loop 주석처리 후 실행

ContainerCreating
-> kubectl describe pods , journalctl -f -u kubelet.service 등으로 상태 확인 후
   그래도 해결 불가 시에 kubeadm reset 후 재구성

대시보드 Pod - CrashLoopBackoff
-> # kubectl get pods -o wide -n kube-system 명령어로 대시보드 파드의 node가 master
가 아닌 worker노드로 설정되있는 경우
   # kubectl delete pods kubernetes-dashboard-7d75c474bb-7stzd --grace-period=0 --
force -n kube-system 명령어로 파드 삭제 후 master 노드로 재배정

[WARNING Hostname]:hostname "호스트명" could not be reached
-> vi /etc/hosts 파일의 호스트네임 잘 들어갔는지 다시 확인 - # hostname

[WARNING IsDockerSystemdCheck]: detected "cgroupfs"
# cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

# systemctl daemon-reload
# systemctl restart docker

# docker info | grep -i cgroup
 Cgroup Driver: systemd

리셋
# kubeadm reset
```

```
# systemctl restart kubelet

토큰 재 확인
# kubeadm token create --print-join-command

***
```

참조

- https://zunoxi.tistory.com/42
- https://www.cubrid.com/blog/3820603
- https://nirsa.tistory.com/292
- https://toridori.tistory.com/186
- https://minkukjo.github.io/study/docs/kubernetes/15steps/03-kubernetes-architecture/