Lab. How to Install Kubernetes on Ubuntu 22.04 LTS

1. Kubernetes Cluster 구성 전 요구사항 확인
    1)쿠버네티스가 지원하는 배포판 여부 확인 (대부분 Debian, Redhat 계열 배포판 지원)
    2)2GB 이상 Memory
        $ free
    3)2 CPUs or more
        $ lscpu

2. Master node hostname 변경(예)
    $ hostname
    $ sudo hostnamectl set-hostname master
    $ hostname
    $ sudo vim /etc/hosts

    172.31.6.113 master
    172.31.4.31   worker1
    172.31.13.34 worker2
    ---

3. IP 주소 확인
    $ ip address

4. Master Node Instance Setting
    1)[이름 및 태그] : userxx-master
    2)[Application and OS Images(Amazon Machine Image) : Ubuntu 22.04 LTS 64-bit
    3)[인스턴스 유형] : t2.medium
    4)[키 페어(로그인)] > [새 키 페어 생성] : userxx-node-key
        -userxx-node-key.pem 파일 C:/temp에 저장

    5)[네트워크 설정] > [편집]
        -[서브넷] : ap-northeast-2a
        -[퍼블릭 IP 자동 할당] : 활성화
        -[방화벽(보안 그룹)] > [보안 그룹 생성] : userxx-kube-sg

    6)[스토리지 구성] : 30 GiB

5. Master Node 설치 후
    1)Docker Engine 설치
        -Add Docker's official GPG key:
            $ sudo apt-get update
            $ sudo apt-get install ca-certificates curl gnupg
            $ sudo install -m 0755 -d /etc/apt/keyrings
            $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
            $ sudo chmod a+r /etc/apt/keyrings/docker.gpg

        -Add the repository to Apt sources:
            $ echo ₩
                "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu ₩
                $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | ₩
                sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
            $ sudo apt-get update

        -To install the latest version, run:
            $ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

-Docker 서비스 등록 실행
$ sudo systemctl enable docker
$ sudo systemctl start docker

2)kubelet 적절한 동작을 위해서 swap 사용 중지 설정
$ sudo swapoff -a
$ sudo sed -i '/swap/s/^/#/' /etc/fstab

3)노드간 통신을 위한 iptables에 브릿지 관련 설정 추가
$ cat << EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF

$ cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

$ sudo sysctl --system

4)방화벽 비활성
$ sudo ufw disable

5)Installing kubeadm, kubelet and kubectl (Debian-based distributions)
-https 기반 레파지토리 사용 지원 패키지 설치
$ sudo apt-get update
$ sudo apt-get install -y apt-transport-https ca-certificates curl

-구글 클라우드의 공개 사이닝 키를 다운로드
$ sudo -i
# curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | gpg --dearmor -o /etc/apt/keyrings/kubernetes-archive-keyring.gpg

-쿠버네티스 apt 레파지토리 추가
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

-kubelet, kubeadm, kubectl 설치 후 버전을 고정
# exit
$ sudo apt update

※만일 오류가 발생하면
Err:2 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
The following signatures couldn't be verified because the public key is not available: NO_PUBKEY B53DC80D13EDEF05

※이런 에러 나면 아래 코드 쿠버네티스 gpg 추가 중 apt-update 시 에러 발생하면 쿠버네티스 GPG key 추가
$ sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://dl.k8s.io/apt/doc/apt-key.gpg
$ echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list
$ sudo apt-get update -y

$ sudo apt-get install -y kubelet kubeadm kubectl
$ sudo apt-mark hold kubelet kubeadm kubectl

6)Configuring the kubelet cgroup driver
-cGroups이 컨테이너 런타임과 kubelet에 의해 제어될 수 있도록 설정
$ sudo mkdir /etc/docker

```
        $ 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
```

7)쿠버네티스 서비스 등록 및 재시작 수행

```
        $ sudo systemctl daemon-reload
        $ sudo systemctl restart kubelet
```

8)Master Node Poweroff

```
        $ sudo poweroff
```

6. 보안그룹에 인바운드 규칙 추가

1)[유형] : 모든 ICMP - IPv4
2)[소스] : 0.0.0.0/0
3)[유형] : 사용자 지정 TCP
4)[포트 범위] : 6443
5)[소스] : 0.0.0.0/0

7. Master Node를 AMI 이미지 생성하기

1)userxx-master 체크 후 [작업] > [이미지 및 템플릿] > [이미지 생성]
2)[이미지 생성] 페이지에서,
-[이미지 이름] : userxx-kube-img
-[이미지 생성] 버튼 클릭

8. 새로 생성한 AMI로 Worker1, Worker2 Node 생성하기

1)페이지 좌측메뉴 중 [이미지] > [AMI]에서
2)[AMI 이름]에서 userxx-kube-img를 체크하고 [AMI로 인스턴스 시작] 버튼 클릭
3)[인스턴스 시작] 페이지에서
-[이름 및 태그] : userxx-worker1
-[인스턴스 유형] : t2.micro
-[키 페어(로그인)] : 목록에서 userxx-node-key 선택
-[네트워크 설정] > [편집]
--[서브넷] : ap-northeast-2a
--[퍼블릭 IP 자동 할당] : 활성화
--[방화벽(보안 그룹) > [기본 보안 그룹 선택]
--[일반 보안 그룹] : 목록에서 userxx-kube-sg 선택
-[인스턴스 시작] 버튼 클릭

4)동일하게 userxx-worker2 생성
5)중지되어 있는 userxx-master 인스턴스 시작
6)이렇게 하면 각 개인 당 userxx-master, userxx-worker1, userxx-worker2 모두 3개의 인스턴스가 생성됨.

9. 각 노드 연결 후

1)각 노드의 [인스턴스 상태]가 모두 "실행 중"이고, [상태 검사]가 "2/2개 검사 통과"가 나온 후
2)Tabby에서 각각의 노드를 Public IP로 연결하여 접속할 것
3)Master Node hostname 변경

```
        $ sudo -i
        # hostnamectl set-hostname master
        # logout
        $ logout
```

<--- 다시 연결
hostname

4)Worker1 Node hostname 변경
$ sudo -i
# hostnamectl set-hostname worker1
# logout
$ logout
<--- 다시 연결
hostname

5)Worker2 Node hostname 변경
$ sudo -i
# hostnamectl set-hostname worker2
# logout
$ logout
<--- 다시 연결
hostname

6)/etc/hosts 파일 수정하기
-모든 노드의 /etc/hosts를 편집기를 사용하여 각 노드의 Private IP를 지정할 것
-예:
127.0.0.1 localhost
172.31.11.157 master
172.31.1.114 worker1
172.31.5.83 worker2

7)서로 ping test
-Master Node에서
ping -c 4 worker1
ping -c 4 worker2

-Worker1 Node에서
ping -c 4 master
ping -c 4 worker2

-Worker2 Node에서
ping -c 4 master
ping -c 4 worker1

10. kubeadm init 명령을 통해서 클러스터를 생성
※반드시 Master Node에서만 실행할 것
1) kubeadm을 통한 네트워크 설정 잡기
$ sudo kubeadm init

.....
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.11.157:6443 --token r2tr6p.xfshcw4p0tvlb7lk ₩
--discovery-token-ca-cert-hash
sha256:e4c7135caafaed67fecd404280ddfcc176b4f474072c90ae70bda8d4d08443f8

2)클러스터 조인 토큰 복사 및 저장
-Windows + r > notepad (실행) > join_token.txt

※kubeadm init시
I0118 11:59:03.624346 3087 version.go:256] remote version is much newer: v1.29.1; falling back to: stable-1.28
[init] Using Kubernetes version: v1.28.6
[preflight] Running pre-flight checks
error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR CRI]: container runtime is not running: output: time="2024-01-18T11:59:04Z" level=fatal msg="validate service connection: CRI v1 runtime API is not implemented for endpoint ₩"unix:///var/run/containerd/containerd.sock₩": rpc error: code = Unimplemented desc = unknown service runtime.v1.RuntimeService"
, error: exit status 1
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...`
To see the stack trace of this error execute with --v=5 or higher

위와 같은 에러가 발생한다면
Ubuntu 20.04 / containerd.io 1.3.7 이상에서 발생하는 문제로 아래의 코드를 실행하자.
$ sudo rm /etc/containerd/config.toml
$ sudo systemctl restart containerd
$ sudo kubeadm init
---

3)root 사용자 클러스터의 API 접근 인증 설정
-kubeconfig 파일의 위치를 root 사용자의 KUBECONFIG 쉘 환경변수에 추가
$ sudo -i
# vi ~/.bashrc
--제일 마지막 줄에 다음 한 줄 추가
export KUBECONFIG=/etc/kubernetes/admin.conf
--저장
source ~/.bashrc
echo $KUBECONFIG

4)Pod 통신을 위한 CNI(Container Network Interface) 기반 Pod 네트워크 추가
-설치되기 전에는 클러스터 DNS (CoreDNS)가 시작되지 않음
refer to : https://docs.projectcalico.org/getting-started/kubernetes/self-managed-onprem/onpremises
wget https://raw.githubusercontent.com/projectcalico/calico/v3.25.0/manifests/calico.yaml
kubectl apply -f calico.yaml

※6443 에러시
$ containerd config default | tee /etc/containerd/config.toml
$ sed -i 's/SystemdCgroup = false/SystemdCgroup = true/g' /etc/containerd/config.toml
$ service containerd restart
$ service kubelet restart

5)pod network 애드온 설치
$ sudo kubectl apply -f
https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.yaml

11. Worker node 구성(only worker)

1)join token 관리자의 권한으로 붙여넣기

$ sudo kubeadm join 172.31.11.157:6443 --token r2tr6p.xfshcw4p0tvlb7lk ₩
--discovery-token-ca-cert-hash
sha256:e4c7135caafaed67fecd404280ddfcc176b4f474072c90ae70bda8d4d08443f8

※워커 노드 설정시 join시 아래와 같은 에러 나면
워커 노드 조인 시 [ERROR CRI]: container runtime is not running 에러가 발생하는 경우
/etc/containerd/config.toml 파일에서
disabled_plugins 항목에서 CRI 제거한 뒤 혹은 주석처리 한 뒤
$ sudo systemctl restart containerd

---

2)다시 sudo kubeadm join 172.31.11.157:6443 --token r2tr6p.xfshcw4p0tvlb7lk ₩
--discovery-token-ca-cert-hash
sha256:e4c7135caafaed67fecd404280ddfcc176b4f474072c90ae70bda8d4d08443f8

12. Master에서 노드 확인

1) 노드 연결 확인

```
# kubectl get nodes -o wide
NAME      STATUS   ROLES           AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE
KERNEL-VERSION   CONTAINER-RUNTIME
master    Ready    control-plane   12m     v1.28.2   172.31.11.157   <none>        Ubuntu 22.04.3 LTS
6.2.0-1017-aws   containerd://1.6.27
worker1   Ready    <none>          4m38s   v1.28.2   172.31.1.114    <none>        Ubuntu 22.04.3 LTS
6.2.0-1017-aws   containerd://1.6.27
worker2   Ready    <none>          2m50s   v1.28.2   172.31.5.83     <none>        Ubuntu 22.04.3 LTS
6.2.0-1017-aws   containerd://1.6.27

//또는
# kubectl get nodes
NAME      STATUS   ROLES           AGE     VERSION
master    Ready    control-plane   11m     v1.28.2
worker1   Ready    <none>          3m38s   v1.28.2
worker2   Ready    <none>          110s    v1.28.2

# kubectl get pods --all-namespaces
   -STATUS가 모두 Running이 될때까지 확인
```