2022.12.07 CKA(Certified Kubernetes Administrator) 시험 응시 소감

2022.06.25 부터 바뀐 시험환경 (PSI Secure Browser) 로 바뀌었다고 하는데 금번에 시험을 치루면서 환경에 대한 소감을 정리 하였습니다;

관련 업데이트 URL :
https://training.linuxfoundation.org/bridge-migration-2021/

시험 준비 – 최소 시험 시간 30분 전 입실 필수

CKA 는 원격 전용 시험이며 Schedule 된 시간 30분 전에
TAKE EXAM 버튼이 활성화 됩니다.

시험 시작 전 세팅 준비하는 시간으로 실제 시험시간에 카운팅되지는 않으나 늦어질수록 실제 시험 시간이 늦어지므로 늦게 끝나게 됩니다.

 # 로그인 및 시험 등록은 해당 URL 에서 이루어 집니다.
https://trainingportal.linuxfoundation.org/learn/course/certified-kubernetes-administrator-cka/exam/exam

TAKE EXAM 을 누른 후 PSI 전용 브라우저를 설치할 수 있습니다.
클라이언트 환경에 Cotana, 원격(zoon, bluejean) 툴, VPN 등의 프로세스가 메모리에 떠 있을 경우 프로그램이 실행되지 않으므로 프로세스 종료해야 합니다
실행한 후 브라우저의 기능적인 신분증 촬영, 안면 촬영 등의 신원 확인 절차를 진행합니다.

이후 시험 감독관이 CHAT 으로 지시하는 내용들을 잘 따르면 됩니다.
 # 시험감독관은 저의 경우 다음의 상황을 요구했습니다
- 시험실 전체를 비추기
- 책상 밑 아래 비추기
- 책상에 노트북(데스크탑) 외 물건 치우기
(저의 경우 무선 마우스와, HDMI 용 집게를 치우라고 했습니다
마우스는 CHAT 으로 설명하고 허락 받았습니다)
- 셔츠 걷어서 팔, 손바닥(앞/뒤) 비춰보기
- 마스크 벗어서 보이지 않도록 칠판 스크린 뒤에 감추기

점검이 끝나면 협조에 고맙다고 하고 라벨이 달린 물병은 소지할 수 없음을 추가 안

내합니다.

시험관이 원격에서 승인 시, 화면에 시험 랩이 나타나게 됩니다.
그 때부터 시간이 실제로 적용됩니다.

시험 환경 – 느림

저의 경우 회사 사무실에서 무선 와이파이로 진행했습니다.
외부 모니터는 지원하지 않기에 14인치 노트북을 사용하였는데
화면이 많이 작은데에다 해외망 원격 랩 시뮬이라 그런지 반응이 많이 느려 고생을 했습니다.

시험 진행시간은 2시간입니다.
문제는 총 17문제이며, 문제 내 2-3가지 연속된 과정을 물어볼 경우
부분점수가 배정됩니다.

단답형 문제는 2~4점
연속형 문제는 6~12점까지 배점이 매우 큽니다.
Flag 버튼이 있어서 문제를 순서대로 풀 필요는 없습니다.

시험 환경 URL :
https://docs.linuxfoundation.org/tc-docs/certification/tips-cka-and-ckad#adjusting-font-and-windows-in-the-examui

복사&붙여넣기

브라우저의 가상 실습 랩 안에서 Firefox 브라우저를 이용한
Kubernetes.io 의 Document 서치만 허용되는데
북마크 사용할 수 없는데에다 반응이 원활하지 않으므로
사전에 YAML 템플릿 위치 등을 숙지해 두는 것을 권장합니다.

복사-붙여넣기는 Ctrl+Shift+C , V 로 가능하다고 하지만
마우스 오른쪽으로 직접 Copy&Paste 를 권장합니다.

번역 & 플러그인

PSI 브라우저의 정해진 가상 랩 안에서 실행되는 것으로
크롬과는 달리 기본으로 번역 기능이 지원되지 않는 Firefox에
번역 플러그인을 설치할 수 없습니다.
문제의 영어 형식은 간단하기 때문에 크게 문제되지는 않을 것입니다.

시험관과는 CHAT 으로만 소통이 가능합니다.
화장실을 가기 전 반드시 전달하고 가야 하며 시험시간은 그대로 흘러갑니다.

점수

총점 100점에서 **66점(2/3) 이상 획득시 Pass** 입니다.
24시간 이내에 메일 및 결과 확인할 수 있게 되어 있습니다.
저의 경우에는 정확히 24시간 뒤 결과가 회신되었습니다.

`Fail` 시 `Reschedule` 기회를 한번 더 준다고 합니다.

바로바로 해결하기 어려운 문제는 `Flag` 찍어놓고 다른 문제부터 빠르게 푸는 것을 권장합니다.
반응도 느리고 망설이면 생각보다 시간이 부족했습니다.

기타

시험 접수 시 이름을 한글로 했을 경우
실제 자격증에 이름이 표기가 되지 않을 수 있습니다.
여권/신용카드에 기재된 영문 네임과 동일하게 접수 권장드립니다.

Kubernetes 1.24 + cri-docker Installation (kubeadm)

우분투에서 Kubernetes 최신 버전 설치하는 방법입니다.
18.04 에서 시험되었으나 상위버전에서도 차이는 없습니다.

Controll Node (node01) , Worker Node (node02 ,node03) 3EA 구성입니다.
모든 설치는 root 가 아닌 `user(worker)`을 사용하며 `sudo` 를 사용합니다

(Pre-Option) 설치 패키지 캐쉬 초기화

```
sudo kill -9 $(lsof -t /var/lib/dpkg/lock)
sudo kill -9 $(lsof -t /var/lib/apt/lists/lock)
sudo kill -9 $(lsof -t /var/cache/apt/archives/lock)
sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
sudo rm /var/lib/dpkg/lock
sudo dpkg --configure -a
```

(Require) SWAPOFF

```
# SWAP-On 시 퍼포먼스 이슈가 있어 일반적으로 제거
sudo swapoff /swap.img
sudo sed -i -e '/swap.img/d' /etc/fstab
```

(유의사항) Container Runtime Install

k8s 1.24 (2022/05) 이후
k8s 에서 기본적으로 내부 연결 지원해주던 dockershim 이 제거되어
cri-docker 추가 설치하여 도커를 k8s에 연결하는 작업이 필요

Pre-Setting – All node

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo systemctl enable --now docker && sudo systemctl status
docker --no-pager
sudo usermod -aG docker worker
sudo docker container ls

# cri-docker Install
VER=$(curl                                                  -s
https://api.github.com/repos/Mirantis/cri-dockerd/releases/lat
est|grep tag_name | cut -d '"' -f 4|sed 's/v//g')
echo $VER
wget
https://github.com/Mirantis/cri-dockerd/releases/download/v${V
ER}/cri-dockerd-${VER}.amd64.tgz
tar xvf cri-dockerd-${VER}.amd64.tgz
sudo mv cri-dockerd/cri-dockerd /usr/local/bin/

# cri-docker Version Check
cri-dockerd --version

wget
https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/
packaging/systemd/cri-docker.service
wget
https://raw.githubusercontent.com/Mirantis/cri-dockerd/master/
packaging/systemd/cri-docker.socket
```

```
sudo mv cri-docker.socket cri-docker.service /etc/systemd/system/
sudo sed -i -e 's,/usr/bin/cri-dockerd,/usr/local/bin/cri-dockerd,' /etc/systemd/system/cri-docker.service

sudo systemctl daemon-reload
sudo systemctl enable cri-docker.service
sudo systemctl enable --now cri-docker.socket

# cri-docker Active Check
sudo systemctl restart docker && sudo systemctl restart cri-docker
sudo systemctl status cri-docker.socket --no-pager

# Docker cgroup Change Require to Systemd
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

sudo systemctl restart docker && sudo systemctl restart cri-docker
sudo docker info | grep Cgroup

# Kernel Forwarding
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
```

Packages Install – All node

```
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates curl
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/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

# Update 해야 패키지 인식함
sudo apt-get update

# k8s 설치
sudo apt-get install -y kubelet kubeadm kubectl

# 버전 확인하기
kubectl version --short

Client Version: v1.24.3
Kustomize Version: v4.5.4

# 버전 고정하기
sudo apt-mark hold kubelet kubeadm kubectl
```

k8s Init – Controller Node (Node01)

```
# Controller Node
sudo kubeadm config images pull --cri-socket unix:///run/cri-dockerd.sock

sudo kubeadm init --ignore-preflight-errors=all --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=211.115.207.136 --cri-socket /var/run/cri-dockerd.sock
## --apiserver-advertise-address=203.248.23.161 -> Controller 서버 IP.
```

```
# 설치 시 출력되는 join Command 를 복사해둔다.
kubeadm join 211.115.207.136:6443 --token r8gfco.6s7f60dns4vgwcc0 \
                        --discovery-token-ca-cert-hash sha256:7b0f82be076748e67f8615eab0b86a61317bac397f94b2921810231ab14afdcc

# kubeadm 을 root 처럼 사용하기 위한 추가 설정
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Token 기억이 안 날 경우
kubeadm token create --print-join-command

# 서비스 확인 ( Ready , Running )
kubectl get nodes -o wide
kubectl get pod -A

# Worker Node에서 kubectl 을 사용하려면 admin.conf 를 복사한다
sudo scp /etc/kubernetes/admin.conf worker@node02:/home/worker/admin.conf
sudo scp /etc/kubernetes/admin.conf worker@node03:/home/worker/admin.conf
```

Node Join – Worker Node (Node02 , Node03)

```
# Worker Node Join. --cri-docker sorket 수동 지정해서 가입해야 한다. init 시 나온 토큰값을 참고한다. 기억이 안 날 경우 커맨드로 확인한다

sudo kubeadm join --token <token> <controlplane-host>:<controlplane-port> --ignore-preflight-errors=all --cri-socket unix:///var/run/cri-dockerd.sock
# sudo kubeadm join [Controller IP]:6443 --token 3nvrgw.33k750dq9klm5omi --discovery-token-ca-cert-hash sha256:b10158bcea37aae0e92ed6b68b4dd1e8213623cc7d406e77eef55fe6196fe346 --cri-socket /var/run/cri-dockerd.sock

# Join
sudo kubeadm join [Controller IP]:6443 --token r8gfco.6s7f60dns4vgwcc0 \
                        --discovery-token-ca-cert-hash
```

```
sha256:7b0f82be076748e67f8615eab0b86a61317bac397f94b2921810231
ab14afdcc --cri-socket /var/run/cri-dockerd.sock

# Check
systemctl status kubelet

# worker 계정에 kubectl 사용 환경변수 추가
cd ~
mkdir -p $HOME/.kube
sudo cp -i ./admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

추가 후 node 에서도 kubectl get nodes -o wide 등을 사용할 수 있게 됨

Pod Network Install – Controller Node (Node01)

```
# 예전 버전에는 CNI Plugin 을 셋업하지 않으면 CoreDNS 가 Pending 이었는데
지금은 기본 kube-system-proxy 로도 구성이 잘 되는 것 같다.

# CNI Mode 확인하기
# Check - kube-proxy 이름 확인한다. kube-proxy 는 Node 마다 존재하므로 3개일 것이다.
kubectl get pod -A
kubectl logs -f pod/kube-proxy-[name] -n kube-system

# Log 로 확인 시 - Linux 커널의 iptables Mode 인 것으로 확인되었다.
kubectl logs -f pod/kube-proxy-6dqp8 -n kube-system | grep
mode
I0805 06:32:48.444077       1 server_others.go:578] "Unknown
proxy mode, assuming iptables proxy" proxyMode=""
I0805 06:32:48.469028        1 server_others.go:213] "kube-
proxy running in dual-stack mod
" ipFamily=IPv4

## Pod Network Plugin Install
다양한 네트워크 플러그인이 있으며, 해당 CKA 과정 글에서는 Callico 의 Plugin 을 사용한다.
Calico Network는 IPIP , VXLAN 등을 지원하는 IPVS 모델이다

kubectl                    create                    -f
https://projectcalico.docs.tigera.io/manifests/tigera-operator
```

```
.yaml
curl
https://projectcalico.docs.tigera.io/manifests/custom-resource
s.yaml -O
kubectl create -f custom-resources.yaml
kubectl get pods -A

## Control-Plane ( Controller ) Node 일 경우 Taint (해당 노드 생성
금지) 영향으로 calico-controller-pod가 Pending 걸릴 수 있는데 다음과 같
이 실행

## Taint Check
kubectl describe node node01 | grep Taints

kubectl taint nodes --all node-role.kubernetes.io/master-
kubectl taint nodes node01 node-role.kubernetes.io/control-
plane:NoSchedule-

## PoD check
kubectl get pod -A

#kubectl describe pods/calico-kube-controllers-657d56796-xxxxx
-n calico-system

## Calicoctl 로 Status 확인하기
cd /usr/local/bin
sudo                          curl                          -L
https://github.com/projectcalico/calico/releases/download/v3.2
3.3/calicoctl-linux-amd64 -o calicoctl
sudo chmod +x calicoctl

## CNI Type Check
calicoctl get ippool -o wide

## Block Check
sudo calicoctl ipam show --show-blocks

## BGP Protocol Check
sudo calicoctl node status

## Node Endpoint Check
calicoctl get workloadendpoint -A
```

Rejoin or Reset

(Trouble) 설치 중 에러 발생시 클린 삭제 후 재 설치를 진행한다

```
# All Node
sudo systemctl stop kubelet
sudo kubeadm reset -f --cri-socket /var/run/cri-dockerd.sock

sudo rm -rf ~/.kube
sudo rm -rf /root/.kube
sudo rm -rf /var/lib/etcd
sudo rm -rf /etc/kubernetes
```

Calico Network 만 삭제하고 싶다면

```
# ALL Node
kubectl delete -f custom-resources.yaml
kubectl delete -f tigera-operator.yaml
sudo rm -rf /var/run/calico/
sudo rm -rf /var/lib/calico/
sudo rm -rf /etc/cni/net.d/
sudo rm -rf /var/lib/cni/
sudo rm -rf /opt/cni
sudo reboot
```

[CKA] #2. Pod – 1

[CKA] #2. Pod (1)

※ 본 글은 작성 후에도 기재 순서의 변경 및 내용 업데이트 예정

CKA 시험은 오픈북 형식으로 크롬에서 몇 가지 URL 를 허용해준다.
대부분의 커맨드 및 설명은 공식문서에 있으나, 시험 중에 바로바로 찾으려면 평소에 숙달이 되어야 한다
주요 명령어의 북마크 등을 미리 설정해 놓는 것도 좋을 것 같다.

Pod ?

Kubernetes 에서 최소 배포 단위
최소 옵션으로 생성 시, Pod 라는 하나의 공간 안에 서비스 Container 와 Pod의 Network 정보를 가지고 있는 /pause 컨테이너로 기본적인 생성이 된다.
즉 Pod 라는 최소 단위 안에는, 컨테이너가 한 개 이상 및 볼륨 등의 다른 서비스도 존재할 수 있다.
외부에서는 Pod 의 Network 만 인지하고 있겠지만, /pause 가 내부 컨테이너들을 포트로 구분하는 역할을 해준다.

Pod Create

실행 전에 yaml 형식의 사전 테스트 모드인 dry run 을 항상 실행해보는 것이 좋다

```
$ kubectl run hello --image=nginx --dry-run=client -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: hello
  name: hello
spec:
  containers:
  - image: nginx
    name: hello
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
```

실제 실행은 위 명령어를 | (파이프)를 사용하여 실제 적용(apply) 의 - 표준 입력으로 넘겨 사용 하는 형식도 권장한다

```
kubectl run hello2 --image=nginx --dry-run=client -o yaml |
kubectl apply -f -
kubectl get pods -o wide
```

nodeName 을 추가하여 원하는 노드에 pod 생성이 가능하다.
이 방식은 Taint 의 스케줄을 무시한다.
yaml 파일에 nodeName 을 추가 지정해 주면 된다.

```
$ kubectl run hi --image=nginx --dry-run=client -o yaml  >
```

```
hi.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: hi
  name: hi
spec:
  containers:
  - image: nginx
    name: hi
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  nodeName: user-controller      ## Node 기입
status: {}

# 정의된 YAML 을 이용한 Pod 생성
$ kubectl create -f hi.yaml
$ kubectl get pods -o wide
NAME                     READY   STATUS    RESTARTS   AGE
IP               NODE              NOMINATED NODE   READINESS
GATES
hello-776c774f98-894tt   1/1     Running   0          18h
192.168.153.193   user-worker       <none>           <none>
hi                       1/1     Running   0          3m10s
192.168.136.5     user-controller   <none>           <none>
```

Pod status 확인하기

```
$ kubectl describe pod hi

# Pod 에 직접 접근해보기. -- kubectl 의 arg 와의 옵션 구분을 위해 -- 를 사용이 권장되는 추세이다
$ kubectl exec -it hi -- /bin/bash
root@hi:/#
```

어플리케이션 Pod 배포하기

```
# 다중 수량 옵션인 --replicas 를 지정할 수 있다. deployment 옵션이 커맨드에 들어간다.
$ kubectl create deployment web --image=nginx --replicas=3
deployment.apps/web created
$ kubectl get pods -o wide | grep -i web
web-76b56fd968-c2pk9     1/1     Running   0          11s   192.168.153.217   user-worker       <none>           <none>
web-76b56fd968-chr4w     1/1     Running   0          11s   192.168.136.6     user-controller   <none>           <none>
web-76b56fd968-mmdfn     1/1     Running   0          11s   192.168.153.218   user-worker       <none>           <none>
```

Pod log 확인하기

```
# Pod info 확인
kubectl describe pod hi

# Pod log 확인
kubectl logs hi

# journal Log(kubelet) 확인
sudo journalctl -u kubelet

# 하위 레벨 Log 확인.
hi POD 내의 Container 탐색.
/pause 컨테이너가 Pod의 포워딩 네트워크 정보. nignx 컨테이너가 서비스 컨테이너이다
$ sudo docker ps -a | grep -i hi
d507d8b298c3   nginx                             "/docker-entrypoint.…"       26 hours ago       Up 26 hours   k8s_hi_hi_default_6a1464a1-0fea-4ff8-a5c6-426afe281173_0
d8fb1a992247   k8s.gcr.io/pause:3.6              "/pause"                     26 hours ago       Up 26 hours   k8s_POD_hi_default_6a1464a1-0fea-4ff8-a5c6-426afe281173_0

# nginx Service Container info
$ sudo docker inspect d507d8b298c3
$ sudo docker logs d507d8b298c3
```

```
# Pod Network info
$ sudo docker inspect d8fb1a992247
$ sudo docker logs d8fb1a992247

# Container 레벨에서 접근해보기. 서비스 컨테이너만 접근 가능하다
$ sudo docker exec -it d507d8b298c3 ls
bin   docker-entrypoint.d   home   media  proc  sbin  tmp
boot  docker-entrypoint.sh  lib    mnt    root  srv   usr
dev   etc                   lib64  opt    run   sys   var

$ sudo docker exec -it d507d8b298c3 /bin/bash
root@hi:/#

# 컨테이너 내에 Net 관련 명령어가 없어 불편하다면 nsenter 를 사용해본다.
ip 명령이 없어서 확인이 어렵다
$ $ sudo docker exec -it d507d8b298c3 ip addr
OCI runtime exec failed: exec failed: container_linux.go:380:
starting container process caused: exec: "ip": executable file
not found in $PATH: unknown

컨테이너 PID 추출한다
$ sudo docker inspect --format '{{ .State.Pid }}' d507d8b298c3
1244489

nsenter 를 사용하여 해당 PID 를 대상으로 클라이언트(pod namespace가 위치
한) 레벨의 명령어 실행이 가능하다.
$ sudo nsenter -t 1244489 -n ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state
UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: tunl0@NONE: <NOARP> mtu 1480 qdisc noop state DOWN group
default qlen 1000
    link/ipip 0.0.0.0 brd 0.0.0.0
4: eth0@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1480 qdisc
noqueue state UP group default
     link/ether c6:3d:04:5d:80:82 brd ff:ff:ff:ff:ff:ff link-
netnsid 0
    inet 192.168.136.5/32 scope global eth0
       valid_lft forever preferred_lft forever
```

Pod delete

```
# 일반 Pod 삭제
$ kubectl delete hi

# 어플리케이션 Pod Delete. replicas 로 다중 배포한 모두가 삭제된다.
$ kubectl delete deployment web
deployment.apps "web" deleted
$ kubectl get pods -o wide | grep -i web
-
```

참고 자료

Pod : https://kubernetes.io/docs/concepts/workloads/pods/
Pod Networking : https://www.digitalocean.com/community/tutorials/how-to-inspect-kubernetes-networking

---

[CKA] #1. 쿠버네티스 설치하기

제목 : [CKA] #1. 쿠버네티스 설치하기

Kubenertes 설치 방식

```
# 쿠버네티스는 여러가지 방식으로 설치할 수 있으나 CKA 과정을 고려한하여
공식 오픈문서의 kubeadm 방식으로 설치를 진행하였다.

구성 서버 ( VM )
Controller Server  : 1EA
Worker Server : 1EA

OS
Ubuntu 20.04 Server Minimal
```

```
# SWAP 은 기본적으로 해제하는 것을 권장하고 있다.
sudo swapoff /swap.img
sudo sed -i -e '/swap.img/d' /etc/fstab

# 호스트네임을 설정한다. 모든 작업은 일반 계정(regular user)에서 sudo 를 사용하여 권한을 빌려 실행하는 것을 권장한다.
sudo hostnamectl set-hostname controller
sudo hostnamectl set-hostname worker
```

Traffic Setup

```
# 컨테이너 런타임(예: Docker), kube-proxy에서 사용하는 브릿지 네트워크의 트래픽을 확인할 수 있도록 iptables 커널 옵션을 지정한다.

## Container / Worker
이 모듈을 로드하면, 브릿지를 통과하는 트래픽이 netfilter(iptables) 를 거치도록 활성화된다.
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
```

Container Runtime 설치

```
# 쿠버네티스의 기본 단위인 POD 에서 컨테이너를 구동하는 런타임은 여러 종류가 존재하지만 CKA 과정 상 Docker 를 설치하였다.
쿠버네티스는 외부 추가 서비스/모듈의 일관성을 보장하지는 않는다.

## Controller / Worker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

## Check
sudo docker -v
sudo docker ps -a
```

cgroup 드라이버 설정

cgroup 은 런타임 프로세스의 리소스를 제약할 수 있는 관리자이다.
설치한 OS의 cgroup 기본 관리자인 systemd 와, docker, kubelet 의 cgroupfs 관리가 별개로 작동하는 혼란을 방지하기 위해 systemd 로 통합해 주는 과정이다.

```
## Controller / Worker
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

## Docker enable && restart
sudo systemctl enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker

## Docker 의 cgroup driver 확인 시, 기본 cgroupfs 에서 systemd 로 변경된 것을 확인할 수 있다.
sudo docker info | grep -i cgroup

 Cgroup Driver: systemd
 Cgroup Version: 1
```

쿠버네티스 패키지 설치

레포지토리 구성 후 kebe 관련 관리/명령줄 패키지를 설치한다. 일반적으로 모든 서버에 설치한다.
에러 발생시 한줄씩 진행한다.

```
## Controller / Worker
sudo apt-get update
sudo apt-get install -y apt-transport-https ca-certificates
```

```
curl
sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/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
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Kube 컨트롤 노드의 InitIalize.

Controller Node 의 시작을 위해 init 하는 과정이다. 옵션을 통해 구성을 지정할 수 있다.

--cri-socket : 지정하지 않을 경우 kubeadm이 socket 으로 내부에 설치된 컨테이너 런타임을 자동 감지하여 구성한다. 다중으로 설치되어 있으면 에러가 발생한다.
--pod-network-cidr : pod 간 통신할 수 있는 network 를 지정한다. 내부 CoreDNS Service 의 조건이다.
--apiserver-advertise-address=<ip-address> : 지정하지 않으면 Controller 의 기본 네트워크 인터페이스를 사용하여 API서버를 선언한다.

Controller. 일반적으로 컨트롤러 자신의 공인IP 를 통해 API 서버임을 알린다 (Advertise)

```
sudo kubeadm init --ignore-preflight-errors=all --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=203.248.23.192
```

정상적으로 init 되면 하단에 아래의 메세지가 나온다.

1) 설명과 같이, 일반 유저(regular user) + sudo 권한으로 cluster를 사용하고 있다면, 아래의 환경 변수를 실행한다.

```
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
```

```
## Check
kubectl get nodes
NAME                STATUS     ROLES                  AGE
VERSION
user1-controller    NotReady   control-plane,master   6m28s
v1.23.5
```

2) pod network 를 사용하기 위해 Network Plugin 을 설치할 수 있다.

```
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/
```

```
## Pod Network 통신이 구성되지 않으면 CoreDNS 가 작동하지 않는
다(Pending)
kubectl get pods --all-namespaces
NAMESPACE     NAME                                READY
STATUS    RESTARTS   AGE
kube-system   coredns-64897985d-9sj9j             0/1
Pending   0          12m
kube-system   coredns-64897985d-zfl8q             0/1
Pending   0          12m
kube-system   etcd-user1-controller               1/1
Running   0          12m
kube-system   kube-apiserver-user1-controller     1/1
Running   0          12m
kube-system   kube-controller-manager-user1-controller   1/1
Running   0          12m
kube-system   kube-proxy-g5xdv                    1/1
Running   0          12m
kube-system   kube-scheduler-user1-controller     1/1
Running   0          12m
```

```
## Pod Network Plugin Install
```
다양한 네트워크 플러그인이 있으며, 해당 CKA 과정 글에서는 Callico 의 Plugin 을 사용한다.

```
curl
https://projectcalico.docs.tigera.io/manifests/calico.yaml -O
kubectl apply -f calico.yaml
```

```
kubectl get nodes

## Check
```
플러그인 설치 후, coredns의 status 가 Running 되는지 체크한다.

```
kubectl get pods --all-namespaces
NAMESPACE     NAME                                       READY
STATUS     RESTARTS   AGE
kube-system   calico-kube-controllers-56fcbf9d6b-bnxz5   0/1
Pending    0          20s
kube-system   calico-node-khp2h                          0/1
Init:2/3   0          20s
kube-system   coredns-64897985d-9sj9j                    0/1
Pending    0          22m
kube-system   coredns-64897985d-zfl8q                    0/1
Pending    0          22m
kube-system   etcd-user1-controller                      1/1
Running    0          22m
```

Multi NIC 를 가지는 환경에서 INTERNAL-IP 설정

다수의 네트워크가 존재하는 서버의 경우 K8S 에서 첫 번째 NIC 의 IP 가 INTERNAL-IP 로 자동 설정된다.
해당 INTERNAL-IP 를 Init 시 설정한
kubeadm --apiserver-advertise-address 와 동일한 IP 대역을 사용하도록 수동 설정한다.
```
# INTERNAL-IP 가 10.0.2.15 ( Calico Network Default ) 로 되어 있다
$ kubectl get nodes -o wide
NAME                STATUS   ROLES                  AGE
VERSION     INTERNAL-IP     EXTERNAL-IP     OS-IMAGE
KERNEL-VERSION      CONTAINER-RUNTIME
user-controller    Ready     control-plane,master    44h
v1.23.5   10.0.2.15      <none>         Ubuntu 20.04.1 LTS
5.4.0-64-generic   docker://20.10.14
user-worker        Ready    <none>                  44h
v1.23.5   10.0.2.15      <none>         Ubuntu 20.04.1 LTS
5.4.0-64-generic   docker://20.10.14

# Controller.  상황에 따라 수동 설정한다
cat << EOF | sudo tee /etc/default/kubelet
KUBELET_EXTRA_ARGS='--node-ip $(hostname -I | cut -d ' ' -f2)'
```

```
EOF
sudo systemctl daemon-reload
sudo systemctl restart kubelet
kubectl cluster-info

# Worker.  상황에 따라 수동 설정한다
cat << EOF | sudo tee /etc/default/kubelet
KUBELET_EXTRA_ARGS='--node-ip $(hostname -I | cut -d ' ' -f2)'
EOF
sudo systemctl daemon-reload
sudo systemctl restart kubelet

# Check 시 Internal-IP 가 advertise 인터페이스로 변경되었다.
$ kubectl get nodes -o wide
NAME               STATUS   ROLES                  AGE
VERSION    INTERNAL-IP       EXTERNAL-IP    OS-IMAGE
KERNEL-VERSION     CONTAINER-RUNTIME
user-controller    Ready     control-plane,master    45h
v1.23.5   203.248.23.214   <none>        Ubuntu 20.04.1 LTS
5.4.0-64-generic   docker://20.10.14
user-worker        Ready    <none>                  44h
v1.23.5   203.248.23.215   <none>        Ubuntu 20.04.1 LTS
5.4.0-64-generic   docker://20.10.14
```

Worker 에서 Controller 에 Join

Then you can join any number of worker nodes by running the following on each as root:
root 로 실행하라고 안내하고 있으나
일반 유저로 Worker 에서 kebeadm을 실행하고 싶다면 Controller 의 /etc/kebenertes/admin.conf 파일을 Worker 로 복제 후 동일 환경 구성을 해준다.

```
# Controller
sudo          scp          /etc/kubernetes//admin.conf
vagrant@203.248.23.193:/home/vagrant/admin.conf

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

```
kubeadm join 203.248.23.192:6443 --token wy11vq.bk2rze7g9lilg2d9 \
                        --discovery-token-ca-cert-hash sha256:f7bc17bb974c804821b21427d500cb96615f66c1fd88cb53c023d8b2c598d3f7
```

오류가 발생할 경우 ignore 옵션을 추가해 본다

```
sudo kubeadm join 203.248.23.192:6443 --token wy11vq.bk2rze7g9lilg2d9 --ignore-preflight-errors=all --discovery-token-ca-cert-hash sha256:f7bc17bb974c804821b21427d500cb96615f66c1fd88cb53c023d8b2c598d3f7

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Check

위의 환경 설정을 했을 경우 Worker 에서도 일반 유저로 pod 확인이 가능하다

```
kubectl get nodes
NAME                STATUS   ROLES                  AGE   VERSION
user1-controller    Ready    control-plane,master   33m   v1.23.5
user1-worker        Ready    <none>                 84s   v1.23.5

kubectl get pods --all-namespaces
NAMESPACE     NAME                                       READY   STATUS    RESTARTS   AGE
kube-system   calico-kube-controllers-56fcbf9d6b-bnxz5   1/1     Running   0          11m
kube-system   calico-node-khp2h                          1/1     Running   0          11m
kube-system   calico-node-skdjl                          1/1     Running   0          2m3s
```

```
kube-system   coredns-64897985d-9sj9j                          1/1
Running   0          33m
kube-system   coredns-64897985d-zfl8q                          1/1
Running   0          33m
kube-system   etcd-user1-controller                            1/1
Running   0          33m
kube-system   kube-apiserver-user1-controller                  1/1
Running   0          33m
kube-system   kube-controller-manager-user1-controller         1/1
Running   0          33m
kube-system   kube-proxy-g5xdv                                 1/1
Running   0          33m
kube-system   kube-proxy-m6ztf                                 1/1
Running   0          2m3s
kube-system   kube-scheduler-user1-controller                  1/1
Running   0          33m
```

(Trouble) 설치 중 에러 발생시 클린 삭제 후 재 설치를 진행한다

```
# All Node
sudo systemctl stop kubelet
sudo kubeadm reset -f

sudo rm -rf ~/.kube
sudo rm -rf /root/.kube
sudo rm -rf /var/lib/etcd
```

Network Plugin Status

앞서서 Pod Network 를 관리하는 애드-온 플러그인으로 Calico 를 설치하였는데, 간단한 Status 명령을 확인할 수 있다.
별도 명령어(calicoctl) 설치도 가능하고, Kubectl 의 옵션 플러그인으로 설치할 수도 있는데 전자로 진행했다.

```
# Host 의 명령어 타입으로 설치
$ cd /usr/local/bin
$ sudo curl -L https://github.com/projectcalico/calico/releases/download/v3.22.1/calicoctl-linux-amd64 -o calicoctl
$ sudo chmod +x calicoctl
```

```
# Check
$ calicoctl ipam show --show-blocks
+----------+--------------------+-----------+------------+----
----------+
| GROUPING |        CIDR        | IPS TOTAL | IPS IN USE |
IPS FREE   |
+----------+--------------------+-----------+------------+----
----------+
| IP Pool  | 192.168.0.0/16     |     65536 | 8 (0%)     |
65528 (100%) |
| Block    | 192.168.136.0/26   |        64 | 3 (5%)     | 61
(95%)     |
| Block    | 192.168.153.192/26 |        64 | 5 (8%)     | 59
(92%)     |
+----------+--------------------+-----------+------------+----
----------+
```

Kubernetes Auto Complation

```
# 주요 명령어의 alias 및 Tab 키를 사용한 자동완성을 권장한다
echo '' >>~/.bashrc
echo 'source <(kubectl completion bash)' >>~/.bashrc
echo 'alias k=kubectl' >>~/.bashrc
echo 'complete -F __start_kubectl k' >>~/.bashrc

. ~/.bashrc

# Check
## Tab 자동완성 확인
k get nodes -o wide
kubectl get nodes -o wide
```