Cert-CKA-Cluster-Set-OnPremise

https://kubernetes.io/ko/docs/setup/production-environment/tools/kubeadm/install-kubeadm/
https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/
https://projectcalico.docs.tigera.io/getting-started/kubernetes/self-managed-onprem/onpremises

```
sudo killall dpkg
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

sudo swapoff /swap.img


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


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


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 enable docker
sudo systemctl daemon-reload
sudo systemctl restart docker


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.googl
```

```
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl


# Controller
sudo kubeadm init --ignore-preflight-errors=all --pod-network-cidr=192.168.0.0/16 --apiserver-ac
## user1: --apiserver-advertise-address=203.248.23.192
## user2: --apiserver-advertise-address=203.248.23.194
## user3: --apiserver-advertise-address=203.248.23.196

mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
## 필요하다면 Worker 로 복사 후 상기사항 수행

# Worker
sudo kubeadm join --token <token> <controlplane-host>:<controlplane-port> --ignore-preflight-err


# Controller
kubectl get nodes
## NotReady


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

kubectl get pods -A
kubectl get nodes -o wide
## Ready


# All
sudo vi /etc/default/kubelet
KUBELET_EXTRA_ARGS='--node-ip 203.248.23.192'
## Multi NIC 환경에서 INTERNAL-IP 가 다른 NIC 으로 설정되는 경우 직접 수동으로 수정한다.
## IP 는 각 Node 의 환경에 맞게 설정한다.

sudo systemctl daemon-reload
sudo systemctl restart kubelet

kubectl get nodes -o wide
kubectl cluster-info


# Controller
kubectl run hello --image=nginx --dry-run=client -o yaml
```

```
kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl apply -f -

kubectl get pods -o wide
```

Cert-CKA-Cluster-Set-IKS

Controller Node 1EA 는 무료다. 단, 30일마다 재생성 조치를 해야한다.

2Core / 4GB 스펙을 지원하며 30일 뒤에 자동 삭제된다.

```
kubectl describe nodes | grep Taint
## Taints:             <none>
## Controller Node 에도 Pod 배포 가능
```

IBM Cloud 검색 - Catalog - Services - Kubernetes Service - Pricing plan: Free

```
Cluster name: ibm-cluster
```

ibmcloud Command Install

```
wget https://download.clis.cloud.ibm.com/ibm-cloud-cli/2.0.2/IBM_Cloud_CLI_2.0.2_amd64.tar.gz

tar xvzf IBM_Cloud_CLI_2.0.2_amd64.tar.gz

cd Bluemix_CLI/

sudo ./install
```

ibmcloud Plugin Install

```
ibmcloud login

ibmcloud plugin install container-service

ibmcloud plugin install container-registry

ibmcloud plugin install observe-service

ibmcloud plugin list
```

ibmcloud Config Get

```
ibmcloud ks cluster ls

ibmcloud target -g Default

ibmcloud ks cluster ls

cp ~/.kube/config ~/.kube/config_ori

ibmcloud ks cluster config --cluster ibm-cluster
## Cluster Name

kubectl config current-context

kubectl config get-contexts

kubectl get nodes -o wide
```

ibmcloud Test

```
kubectl run nginx --image=nginx

kubectl get pod -o wide

kubectl run -it test --image=wayles54/debugger-alpine --rm=true -- sh

curl 172.30.250.202
```

Switch

```
kubectl config get-contexts

kubectl config use-context kubernetes-admin@kubernetes
```

Cert-CKA-Cluster-Set-Docker_Desktop

Intro

현재는 Window Home 버전에서도 Docker Desktop 사용 가능 (=WSL 지원으로 가능)

Docker Desktop 설치

```
Settings - Kubernetes - Enable Kubernetes

kubectl config get-contexts
## docker-desktop Node 확인

kubectl run hello --image=nginx

kubectl get pods

kubectl exec hello -- curl 127.0.0.1
```

```
config
```

docker-desktop 의 config 은 하기 경로에 있다.

```
/c/Users/admin-mgmt/.kube/config
```

Cert-CKA-Cluster-Set-Shell

```
Keyword
```

kubernetes auto complete

```
Auto Complete
```

```
echo '' >>~/.bashrc
echo 'source <(kubectl completion bash)' >>~/.bashrc
echo 'alias k=kubectl' >>~/.bashrc
echo 'complete -F __start_kubectl k' >>~/.bashrc

. ~/.bashrc
```

```
Exercise
```

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

Cert-CKA-Cluster-Set-Taint

Taint 기본 정책

```
kubectl get nodes

kubectl describe nodes user-controller

kubectl describe nodes user-controller | grep Taint
## Taints:             node-role.kubernetes.io/master:NoSchedule

kubectl describe nodes user-worker | grep Taint
## Taints:             <none>

kubectl create deployment hello --image=nginx --replicas=3

kubectl get pods -o wide
# 모두 user-worker 노드에 스케줄링 되었음을 확인할 수 있다.
## NAME                     READY   STATUS    RESTARTS   AGE   IP                NODE          N
## hello-776c774f98-5xjj4   1/1     Running   0          12s   192.168.153.238   user-worker   <
## hello-776c774f98-t2ghw   1/1     Running   0          12s   192.168.153.239   user-worker   <
## hello-776c774f98-tl874   1/1     Running   0          12s   192.168.153.237   user-worker   <

kubectl delete deployment hello
```

Taint 정책 제거

```
kubectl taint nodes user-controller node-role.kubernetes.io/master:NoSchedule-

kubectl describe nodes user-controller | grep Taint
## Taints:             <none>

kubectl create deployment hello --image=nginx --replicas=3

kubectl get pods -o wide
# 이제 user-controller 노드에도 스케줄링 되었음을 확인할 수 있다.
## NAME                     READY   STATUS    RESTARTS   AGE   IP                NODE
## hello-776c774f98-cr2tg   1/1     Running   0          9s    192.168.136.14    user-controller
## hello-776c774f98-fhqnr   1/1     Running   0          9s    192.168.153.240   user-worker
## hello-776c774f98-kk86m   1/1     Running   0          9s    192.168.153.241   user-worker
```

Cert-CKA-Cluster-Component-ControlPlane

kube-apiserver

```
kubectl get pods -A | grep kube-apiserver
```

```
sudo cat /etc/kubernetes/manifests/kube-apiserver.yaml
```

`kube-scheduler`

Pod 스케줄링

```
sudo cat /etc/kubernetes/manifests/kube-scheduler.yaml
```

`kube-controller-manager`

Control plane component that runs controller processes.

Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

```
sudo cat /etc/kubernetes/manifests/kube-controller-manager.yaml
```

`etcd`

```
sudo cat /etc/kubernetes/manifests/etcd.yaml
```

Cert-CKA-Cluster-Component-Node

`kubelet`

An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

`kube-proxy`

kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.

Cert-CKA-Cluster-Component-Controller-Type

Types of Controllers

```
ReplicaSet
Deployment
DaemonSet
StatefulSet
Job
CronJob
```

Cert-CKA-Cluster-Component-Resource-Type

K8S 에서 지원하는 모든 Resource 를 출력한다.

```
kubectl api-resources
## NAME / SHORTNAMES / APIVERSION / NAMESPACED / KIND
## NAMESPACED: Scope 개념
```

Cert-CKA-Cluster-Kubectl

K8S 클러스터를 제어하는 명령어

Cert-CKA-Cluster-Kubectl-Imperative

명령 방식

```
kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl apply -f -

kubectl delete pod hello
```

Cert-CKA-Cluster-Kubectl-Declarative

선언 방식

```
kubectl run hello --image=nginx --dry-run=client -o yaml | tee ~/test.yml
# apiVersion: v1
# kind: Pod
# metadata:
#   creationTimestamp: null
#   labels:
#     run: hello
#   name: hello
# spec:
#   containers:
#   - image: nginx
#     name: hello
#     resources: {}
#   dnsPolicy: ClusterFirst
#   restartPolicy: Always
# status: {}

kubectl apply -f ~/test.yml

kubectl delete -f ~/test.yml
```

Cert-CKA-Cluster-Kubectl-Command-explain

```
kubectl explain pod
kubectl explain pod.spec
kubectl explain pod.spec.containers
kubectl explain pod.spec.containers.name
kubectl explain pod.spec.containers.image

kubectl explain pod --recursive | grep -A6 -B6 -i runasuser
```

Cert-CKA-Cluster-Kubectl-Command-get

```
kubectl get all
## 모든 Resource 조회
```

Cert-CKA-Cluster-Kubectl-Command-create

기 생성된 자원에 대해서 미동작한다.

```
kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl create -f -
kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl create -f -
## Error from server (AlreadyExists): error when creating "STDIN": pods "hello" already exists
```

Cert-CKA-Cluster-Kubectl-Command-apply

기 생성된 자원에 대해서 변경사항을 업데이트 한다.

kubectl edit 와 같이 변경 가능한 값은 제한된다.

```
kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl apply -f -
## pod/hello created

kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl apply -f -
## pod/hello configured

kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl apply -f -
## restartPolicy: Never
## * spec: Forbidden: pod updates may not change fields other than `spec.containers[*].image`, `
```

Cert-CKA-Cluster-Kubectl-Command-edit

기생성된 Resource 를 변경할 수 있다. 단, 변경 가능한 값은 제한된다.

```
kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl apply -f -

kubectl edit pod hello

kubectl edit pod hello
## image: httpd
## 자동으로 Pod 삭제 후 재생성된다.

kubectl edit pod hello
## restartPolicy: Never
## * spec: Forbidden: pod updates may not change fields other than `spec.containers[*].image`, `
```

Cert-CKA-Cluster-Resource-Namespace

Cert-CKA-Cluster-Resource-Namespace-Explain

```
kubectl explain namespace
```

Cert-CKA-Cluster-Resource-Namespace-CRUD

```
# Create
kubectl create namespace test
kubectl create namespace test --dry-run=client -o yaml

# Get
kubectl get namespaces
kubectl get namespaces -n default
kubectl get namespace test
kubectl get namespace test -o yaml

# Describe
kubectl describe namespace test

# Delete
kubectl delete namespace test
```

Cert-CKA-Cluster-Resource-Pod

Cert-CKA-Cluster-Resource-Pod-Explain

```
kubectl explain pod
```

Cert-CKA-Cluster-Resource-Pod-CRUD

```
# Create
kubectl run hello --image=nginx
kubectl run hello --image=nginx --dry-run=client -o yaml

# Get
kubectl get pods
kubectl get pods -n default
kubectl get pod hello
kubectl get pod hello -o yaml

# Describe
kubectl describe pod hello

# Delete
kubectl delete pod hello
```

Cert-CKA-Cluster-Resource-Pod-Domain

```
kubectl run hello --image=nginx --dry-run=client -o yaml | kubectl apply -f -

kubectl get pods -o wide
## 192.168.153.244

curl 244-153-168-192.default.pod.cluster.local
## could not be resoved... why?

kubectl get svc -n kube-system
## kube-dns: 10.96.0.10

dig @10.96.0.10 192-168-153-244.default.pod.cluster.local
## ;; ANSWER SECTION:
## 192-168-153-244.default.pod.cluster.local. 30 IN A 192.168.153.244

kubectl run -it --rm --restart=Never curl --image=curlimages/curl sh
## --rm: 터미널 로그아웃 후 Pod 삭제

cat /etc/resolv.conf
## nameserver 10.96.0.10
## search default.svc.cluster.local svc.cluster.local cluster.local

curl 192-168-153-244.default.pod.cluster.local
```

K8S Cluster 에 생성되는 Resource Pod 는 모두 Domain Name 을 가지고 있다. 다만 해당 Domain Name 에 IP 가 포함되므로 향후 이중화 구성에서 해당 Endpoint 사용은 어렵다.
-> Resource Service

```
# Pod Domain Name
dig @10.96.0.10 192-168-153-244.default.pod.cluster.local

# Service Domain Name
dig @10.96.0.10 service_name-svc.default.svc.cluster.local
```

Cert-CKA-Cluster-Resource-Replicaset

Cert-CKA-Cluster-Resource-Replicaset-Explain

```
kubectl explain replicaset
```

Cert-CKA-Cluster-Resource-Replicaset-CRUD

```
# Create
cat << EOF | kubectl apply -f -
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: hello-rs
  labels:
    app: hello-rs
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-pod
  template:
    metadata:
      labels:
        app: hello-pod
    spec:
      containers:
      - name: hello-container
        image: nginx
EOF

# Get
kubectl get replicaset
kubectl get replicaset -n default
kubectl get replicaset replicaset_name
kubectl get replicaset replicaset_name -o yaml

# Describe
kubectl describe replicaset replicaset_name

# Delete
kubectl delete replicaset replicaset_name
```