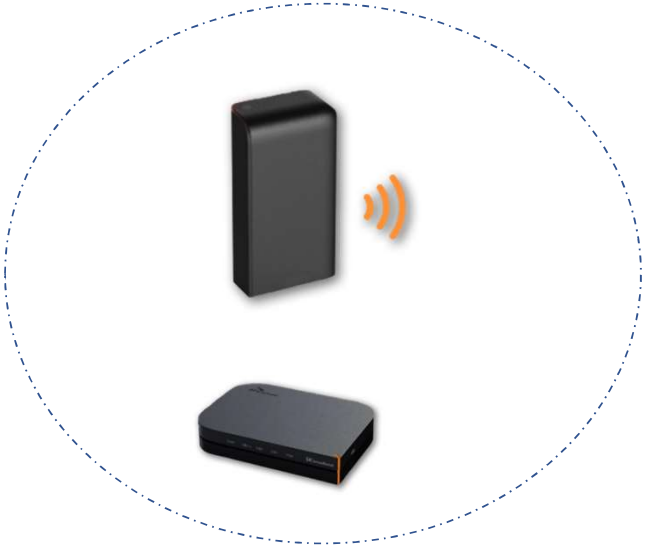
코멘토 직무 에센스 강의

Cloud Platform for Private 5G

PROFILE

現 통신장비 개발사 직무 7년차

- 광 통신 유선 장비 개발. (FTTH – PON)
- 유무선 공유기 개발
- SDN 장비 개발
- Cloud Platform



CONTENTS

Chapter. 1

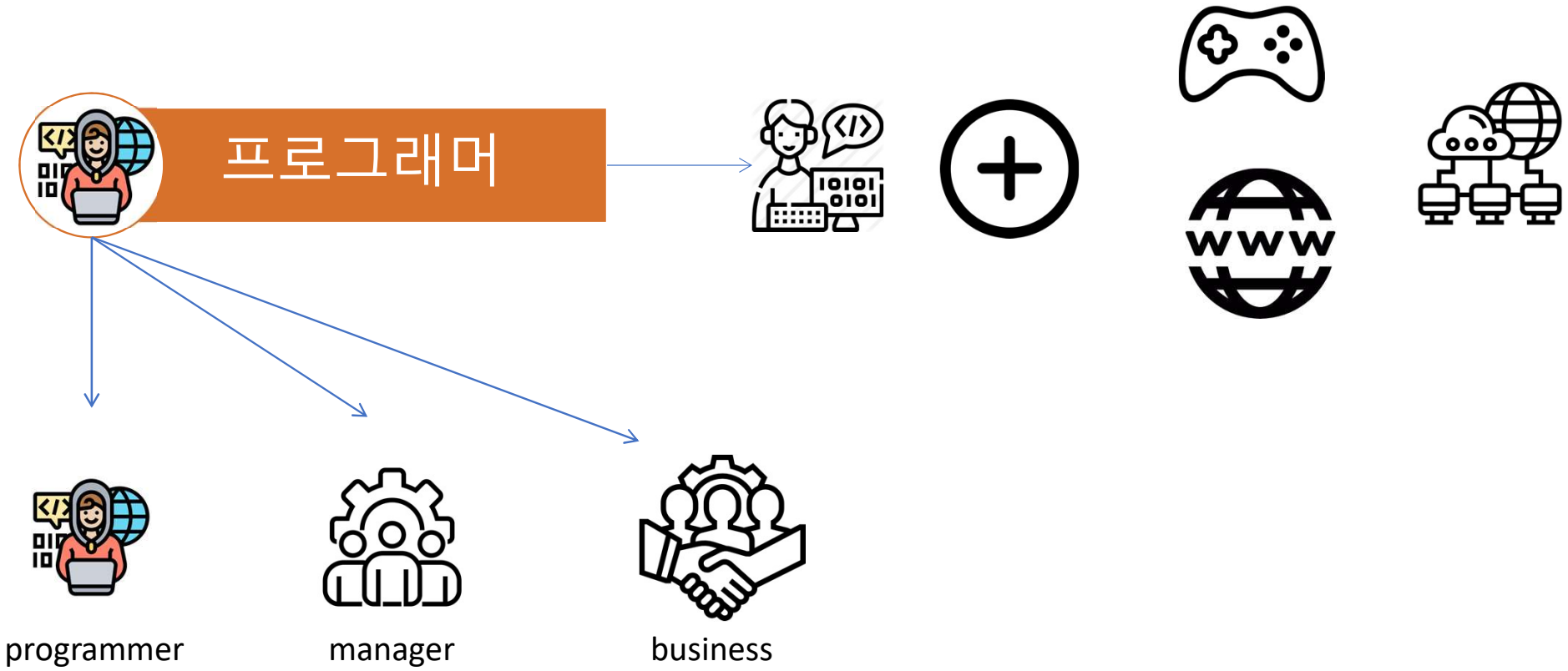
업무 일과

업무 일과

▣ 특징 및 장단점

업무 일과

▣ 특징 및 장단점



업무 일과

▣ 특징 및 장단점

네트워크 개발자

System

- Linux Kernel
- Boot Loader

Wire

- L2
- L3
- Protocol

Wireless

- WIFI
- Radio

업무 일과

▣ 특징 및 장단점

클라우드 업무

System

Operator

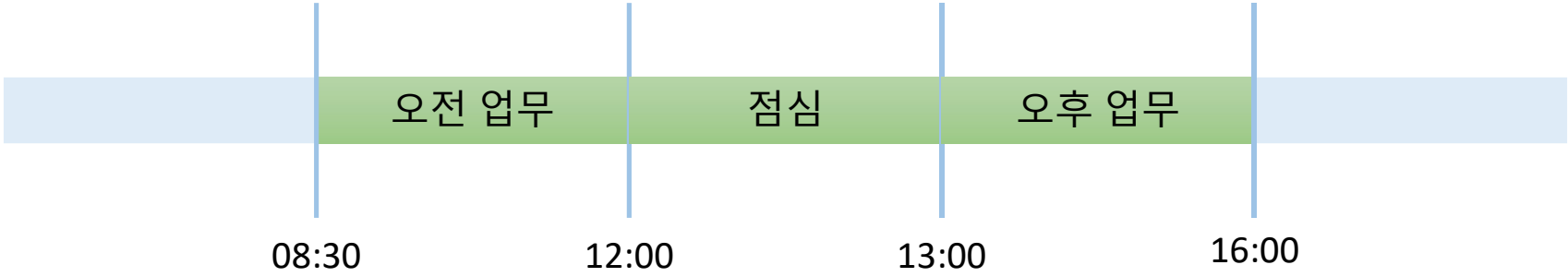
Manager

Programmer

업무 일과

▣ 하루 일과

- 주 40시간, 하루 8시간 근무
- 사내 업무 6시간 30분, **자율 근무 1시간 30분**



▣ 그 외

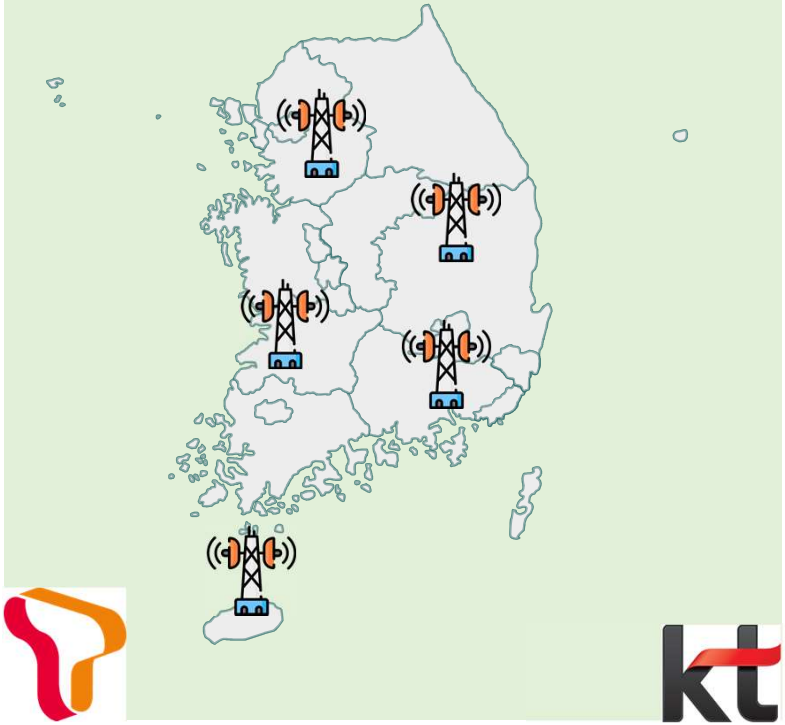
Chapter. 2

담당 업무

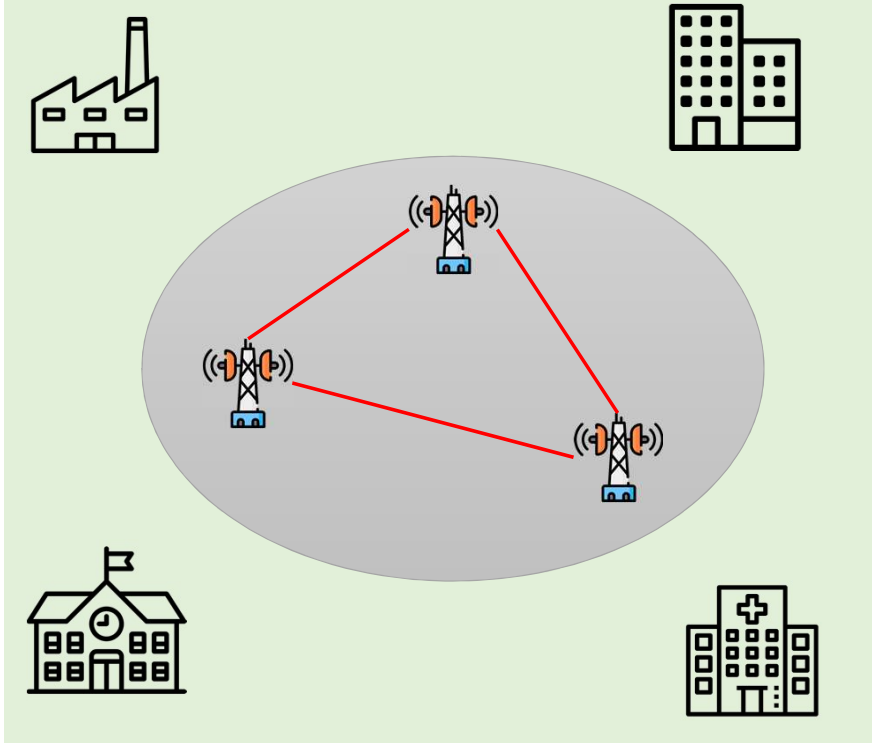
담당 업무

▣ **Private 5G** vs Public 5G

Public 5G

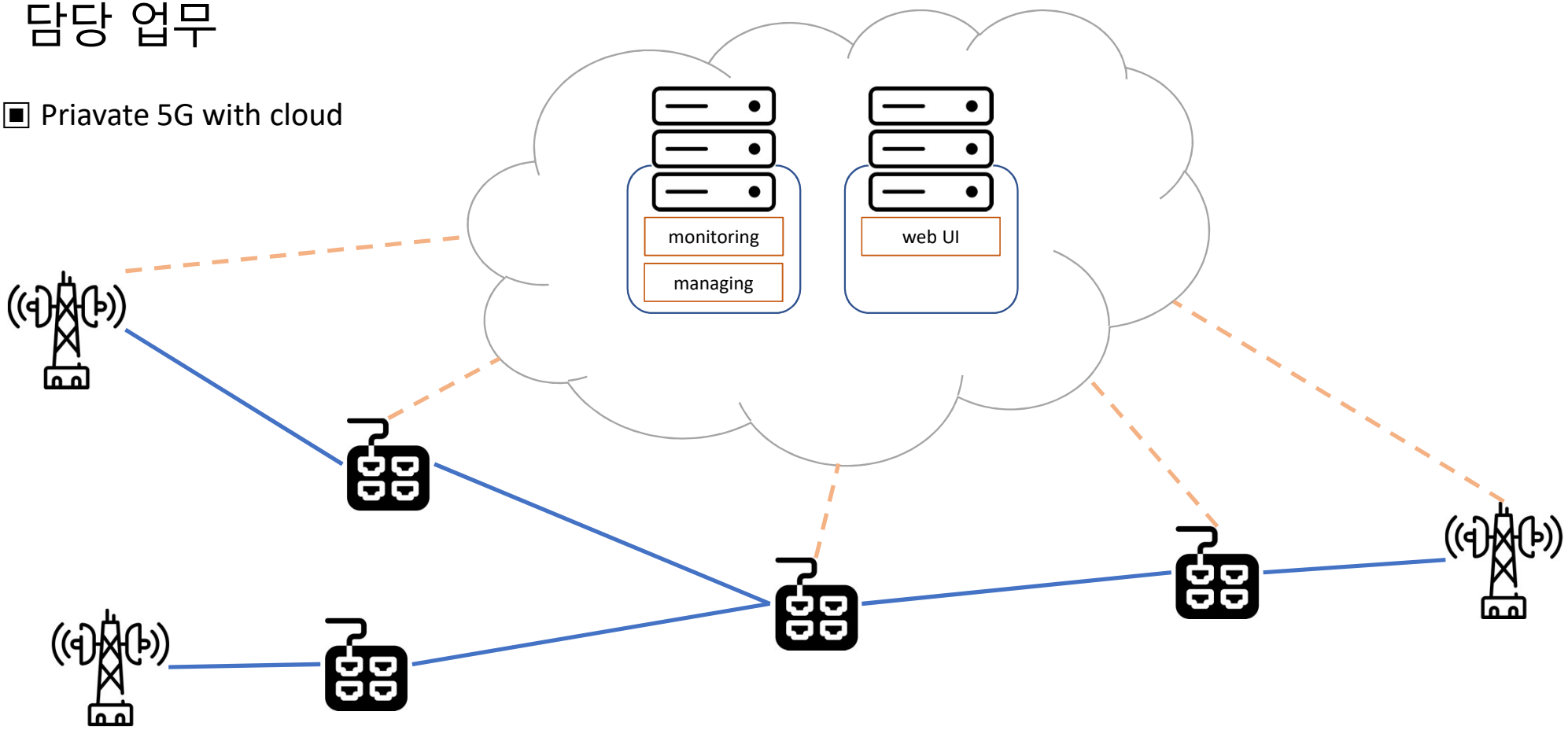


Private 5G



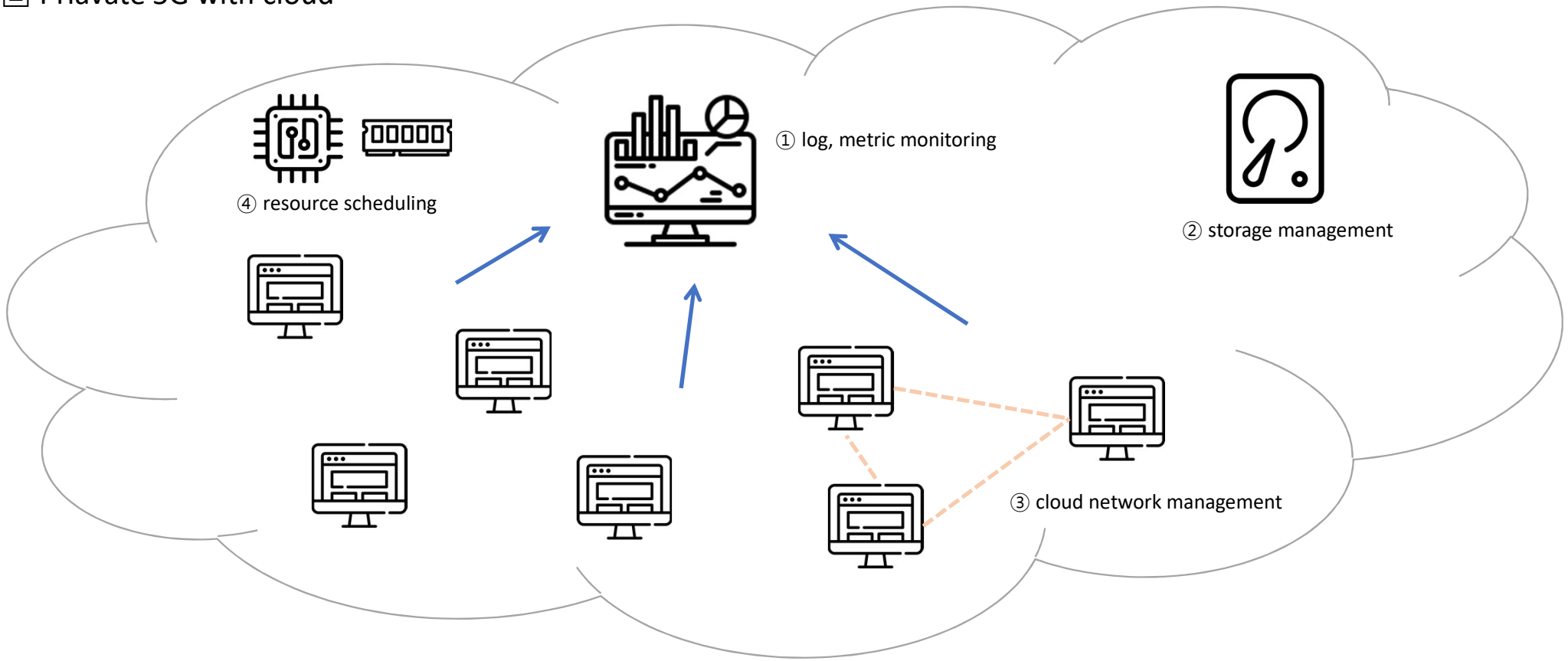
담당 업무

▣ Priavate 5G with cloud



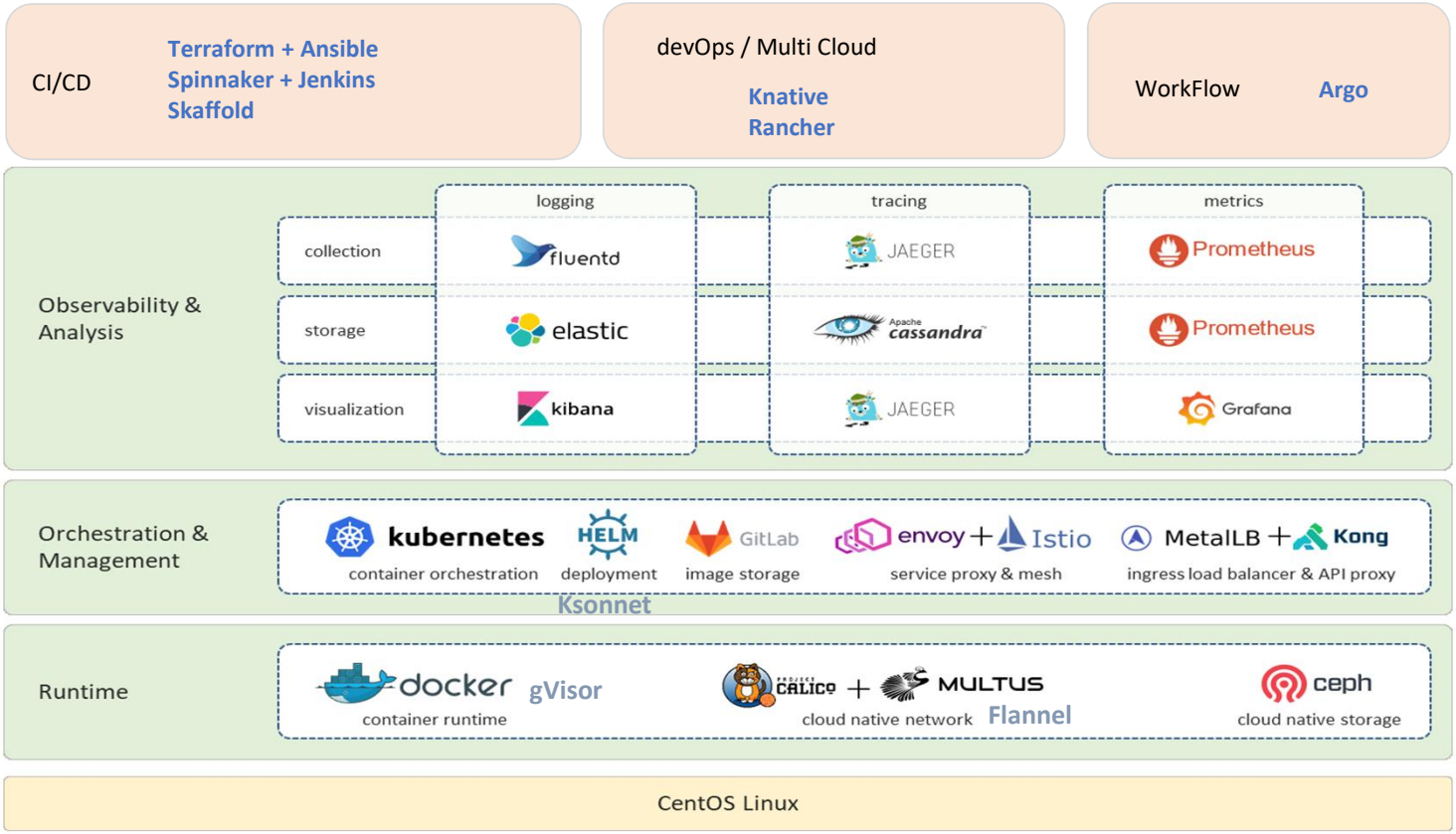
담당 업무

▣ Priavate 5G with cloud



담당 업무

▣ Priavate 5G with cloud



담당 업무

▣ 필요한 역량 (private 5G)

Common

- linux 기본 지식
- OSI 7 계층
- Database

Cloud

- Docker
- Kubernetes
- go, python
- gRPC, Kafka

5G

- Embeded
- Radio interface
- C, C++

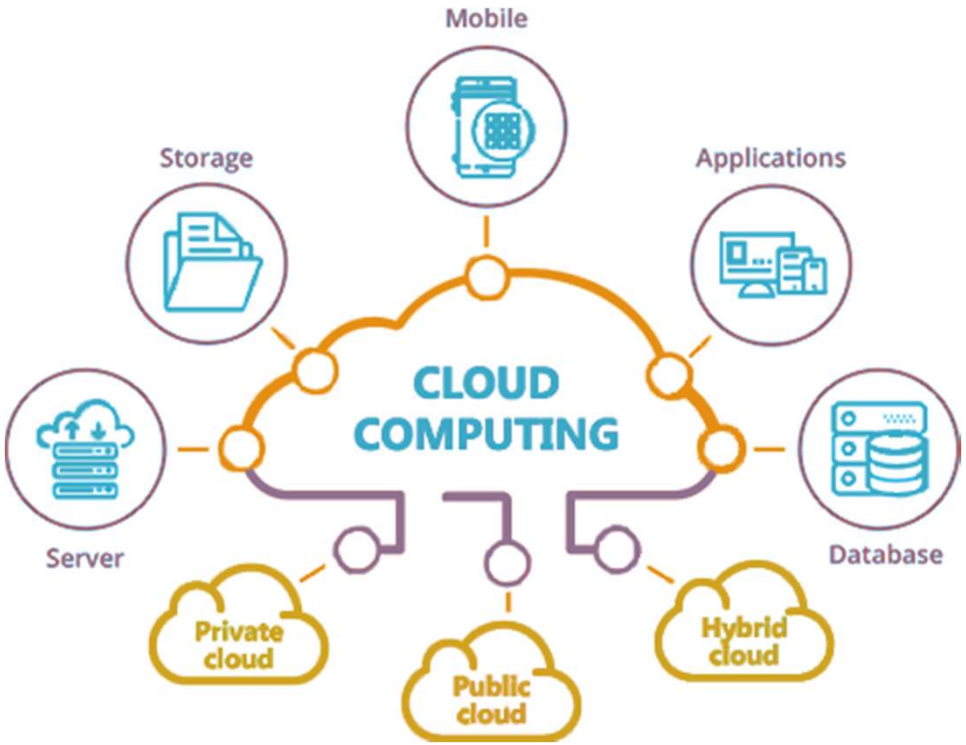
Web UI

- java
- javascript, html

Chapter. 3

Cloud Computing

Cloud Computing

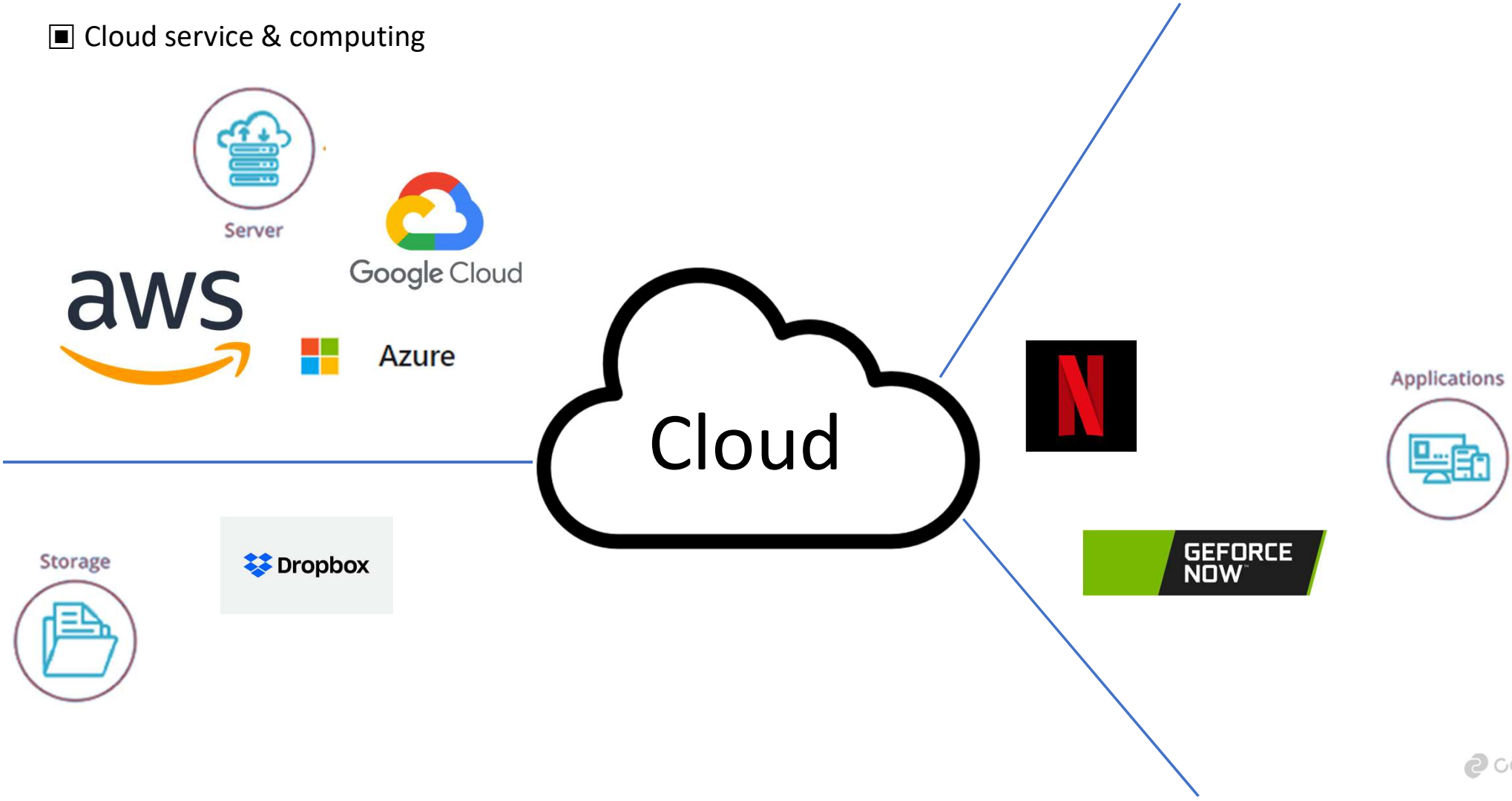


▣ Cloud Computing이란?

- server, storage, database, software등을 네트워크를 통해 별도의 작업 없이 제공하는 것.

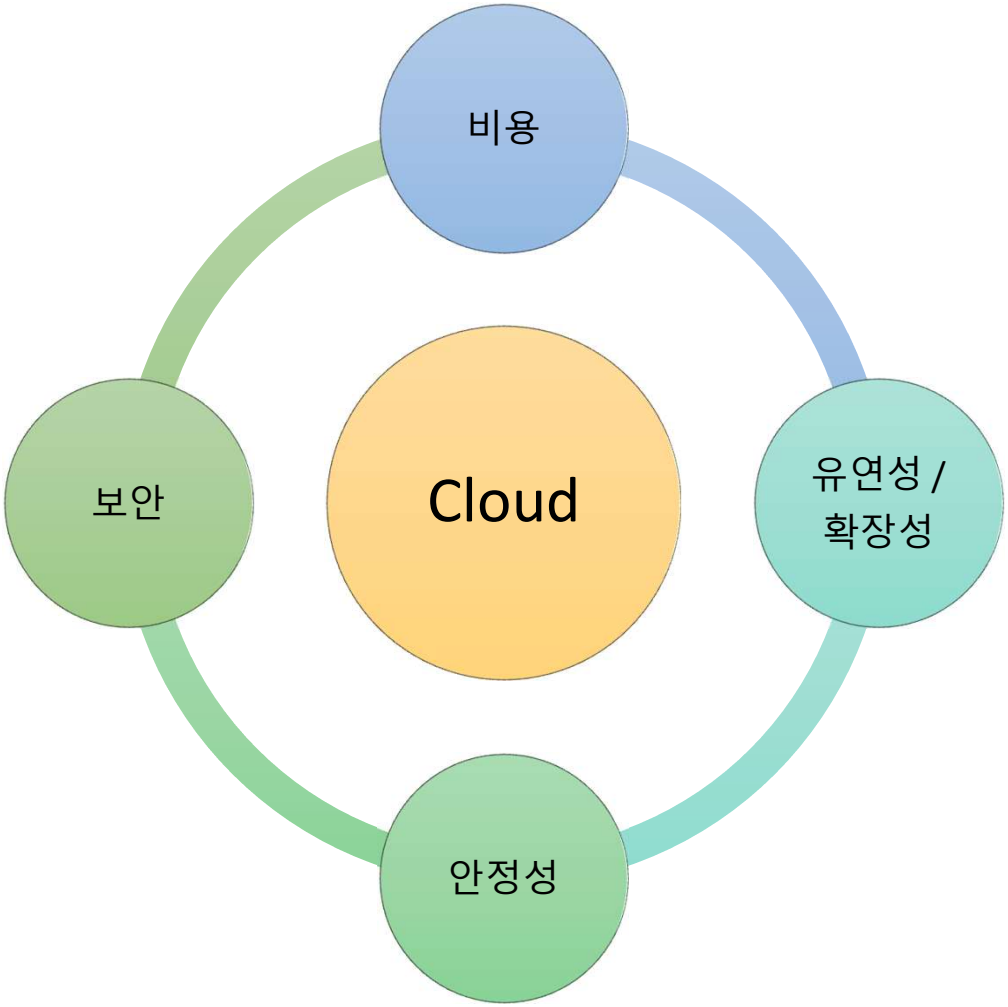
Cloud Computing

▣ Cloud service & computing



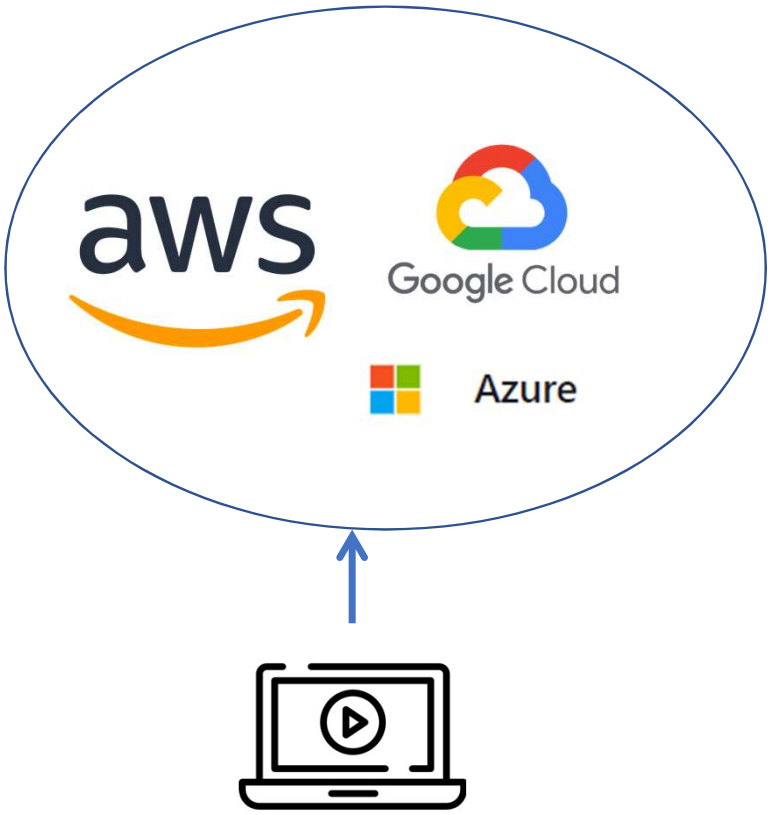
Cloud Computing

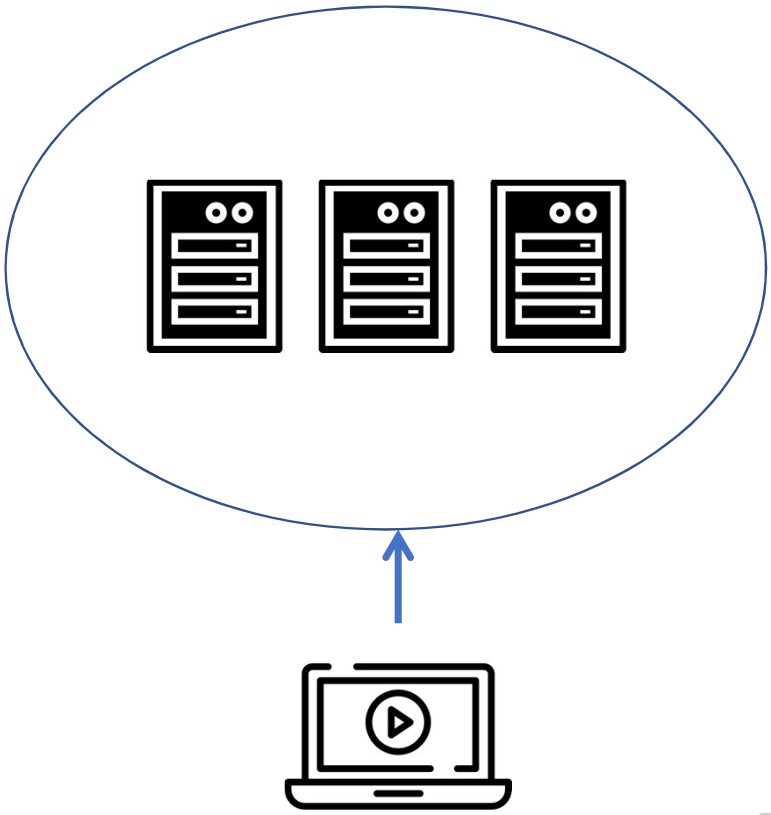
▣ Feature



Cloud Computing

▣ Public & Private(on – premise)

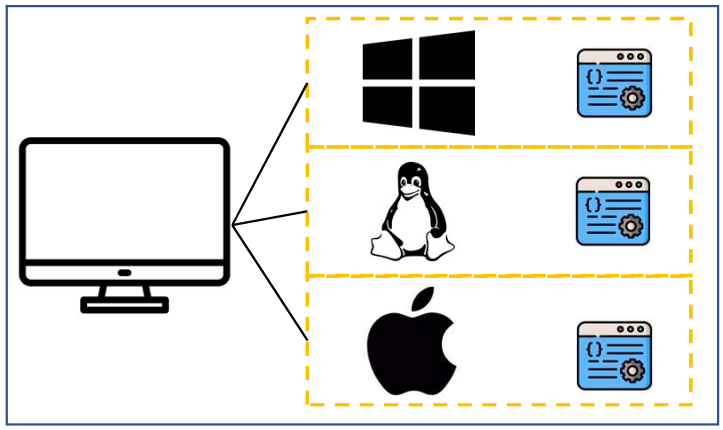




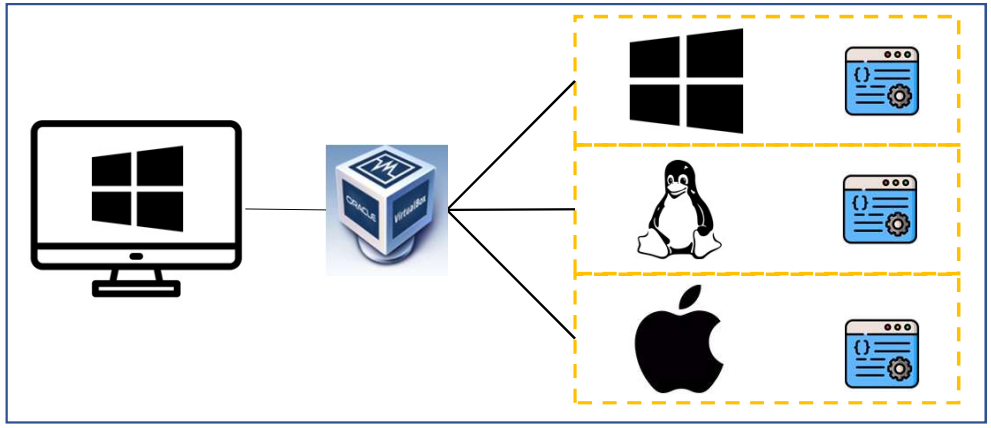
Chapter. 4

Kubernetes

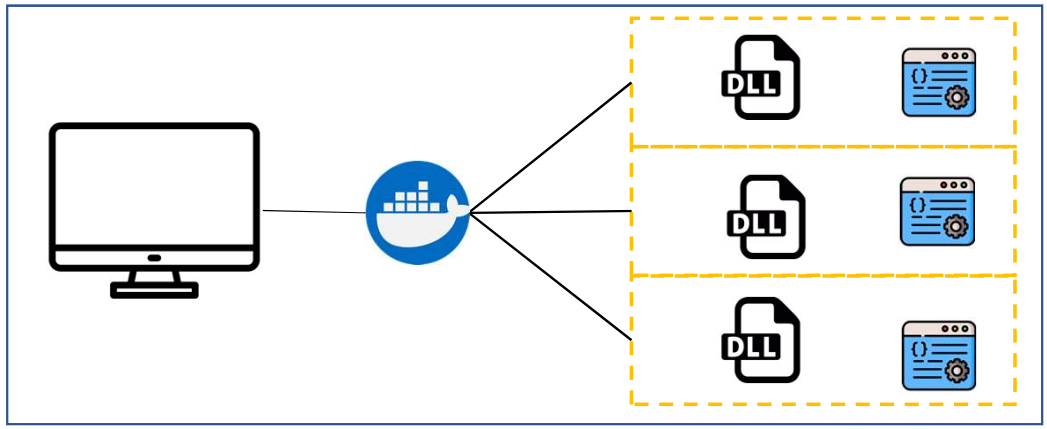
Kubernetes



Traditional

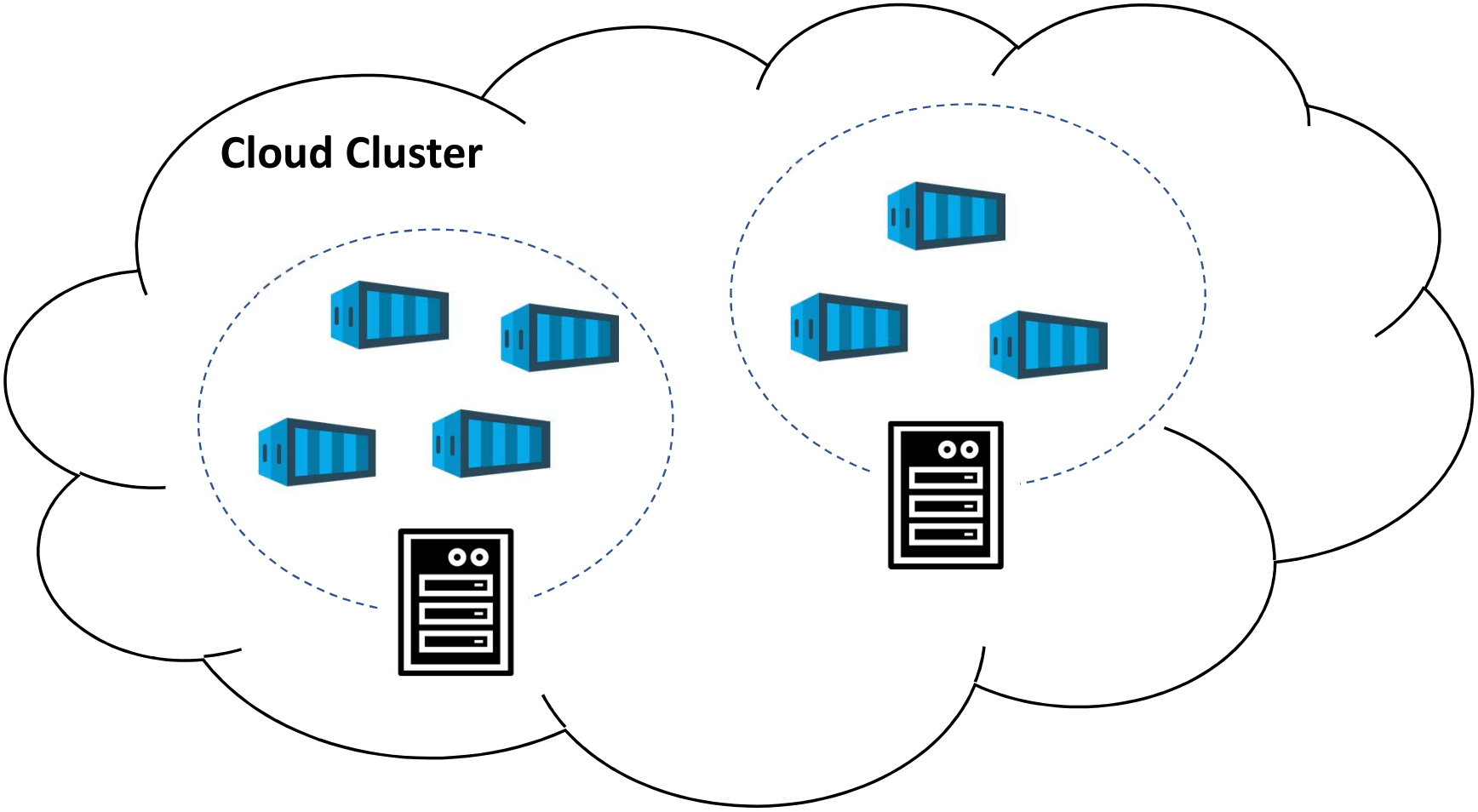


Virtualized



Container

Kubernetes



Kubernetes

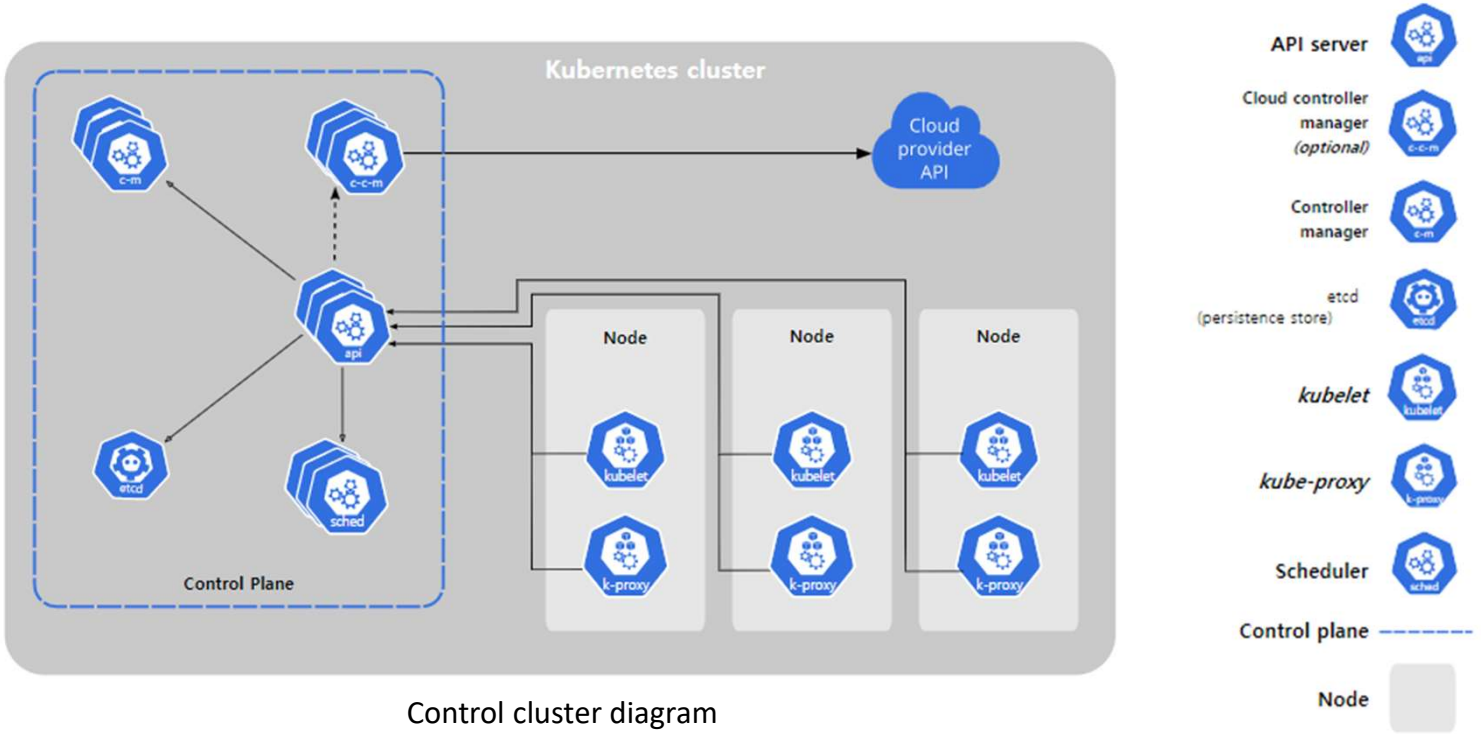
orchestration

Auto roll back

Self healing

configuration management

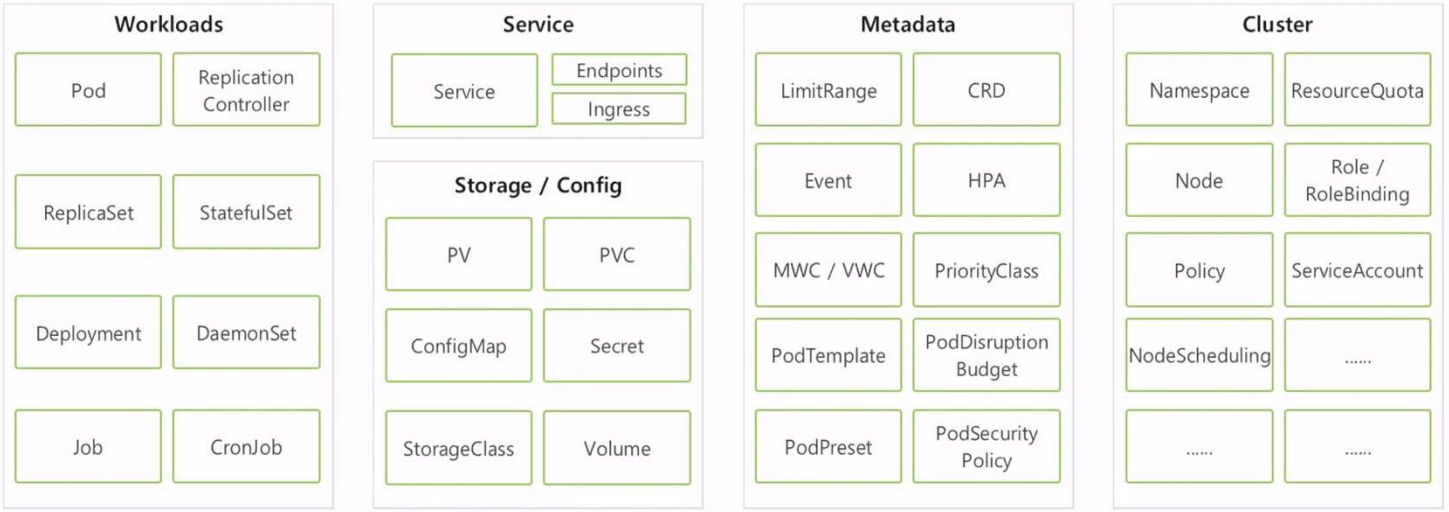
Kubernetes



Control cluster diagram

Kubernetes

▣ Object



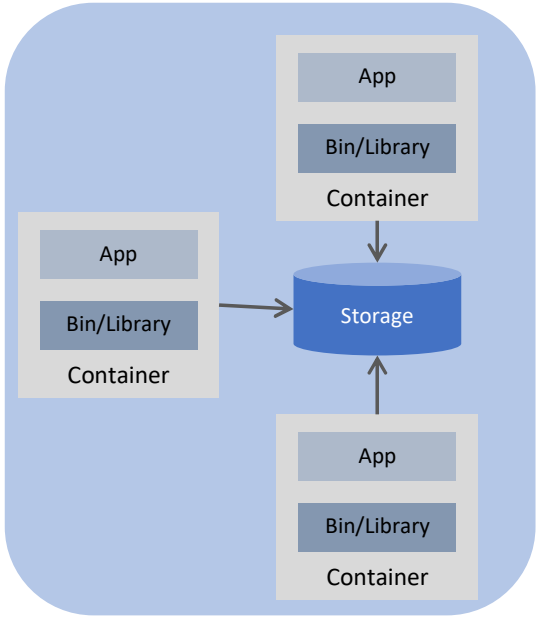

```
apiVersion: v1

kind: Pod

metadata:
  name: nginx

spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 8090
```

nginx-pod.yaml

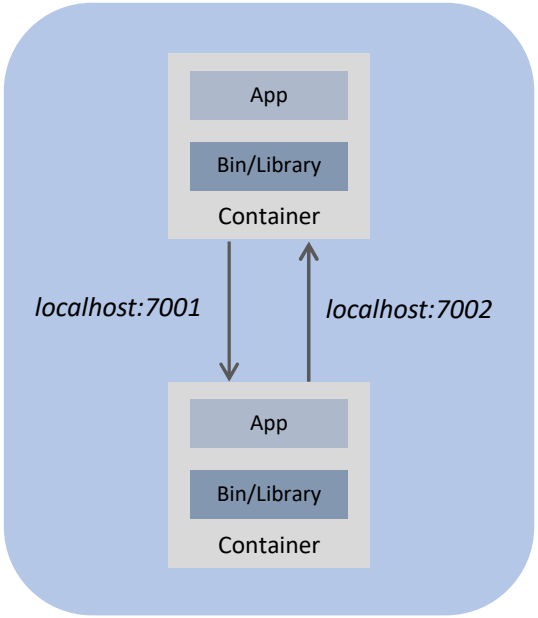
Kubernetes

▣ Pod

- Kubernetes에서 가장 기본적인 배포 단위로, 컨테이너를 포함하는 단위



⦿ Disk volume를 서로 공유



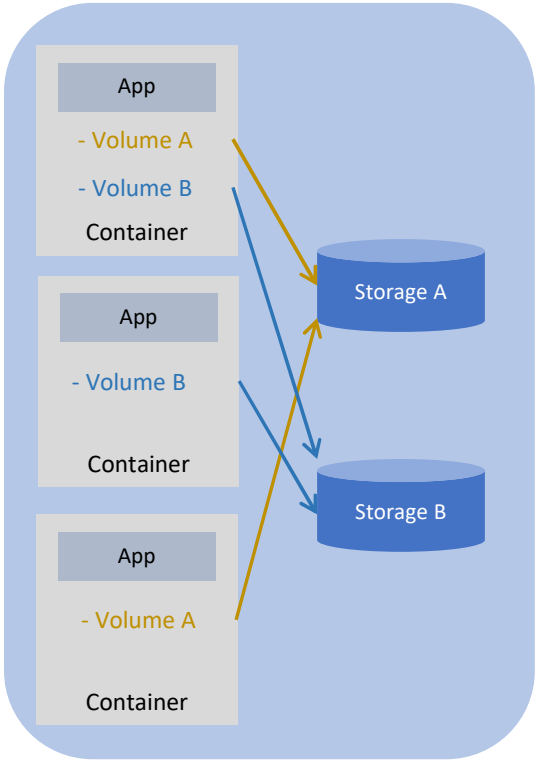
⦿ IP와 Port를 공유

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 containers:
 - name: nginx
   image: nginx:1.7.9
   ports:
   - containerPort: 8090
 containers:
 - name: database
```

Kubernetes

▣ Volume

- 컨테이너가 삭제 될 때 유실되는 로컬 데이터를 위해 제공되는 오브젝트 (Pod에 종속)
- Container가 삭제나 리셋에 상관없이 영구적으로 저장되는 스토리지의 형태
- 일반적인 로컬디스크 외에 AWS EBS, GCP Persistent등 클라우드디스크등 다양한 Volume을 지원



- Volume의 종류

※ 로컬 볼륨
hostPath : host의 local directory를 공유
emptyDir : pod가 실행되는 동안에만 임시로 사용할 수 있는 공간

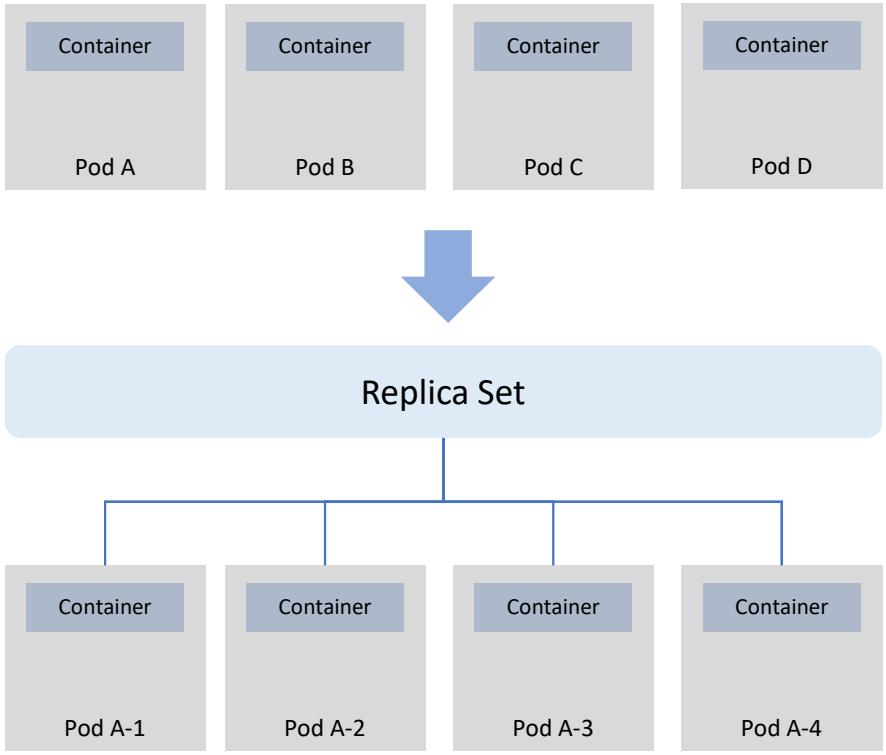
※ 네트워크 볼륨
NFS, AWS EBS, GCP Persistent

```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 containers:
 - name: nginx
   image: nginx:1.7.9
   ports:
   - containerPort: 8090
   volumeMounts:
   - name : my-volume
     mountPath : /etc
 volumes:
  - name : my-volume
    hostPath:
      path: /tmp
```

Kubernetes

▣ Deployment

- 동일한 pod에 대하여 명시한 개수에 대한 가용성을 보증하는데 사용

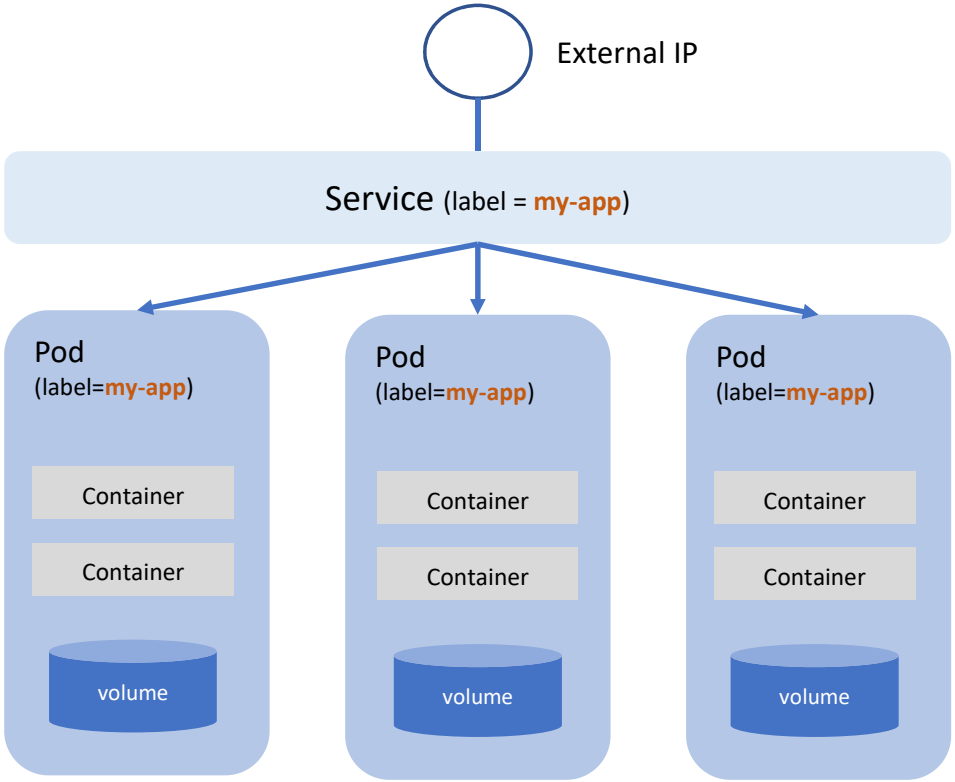



```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: my-nginx-deployment
spec:
 replicas: 4
 selector:
  matchLabels:
   app: my-nginx
 template:
  metadata:
   name: my-nginx-pod
   labels:
    app: my-nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.10
    ports:
    - containerPort: 80
```

Kubernetes

▣ Service

- 여러 개의 pod를 서비스 할때 이를 하나의 IP와 포트로 묶어서 서비스를 제공하는 오브젝트

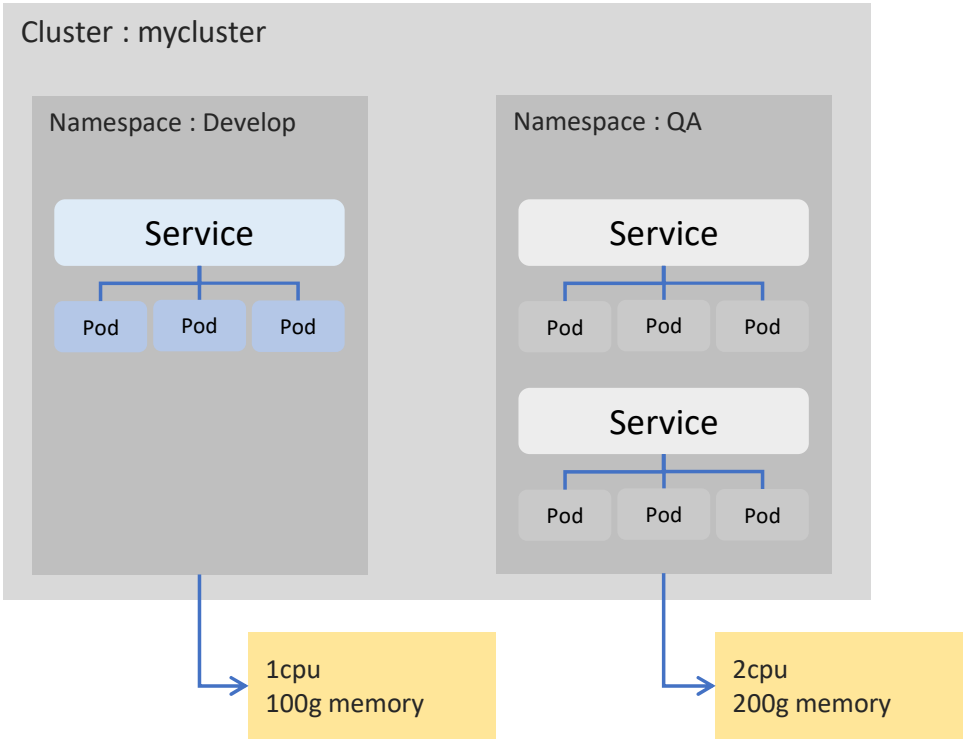



```
kind: Service
apiVersion: v1
metadata:
 name: my-service
spec:
 selector:
   app: my-app
 ports:
 - protocol: TCP
   port: 80
   targetPort: 9376
```

Kubernetes

▣ Namespace

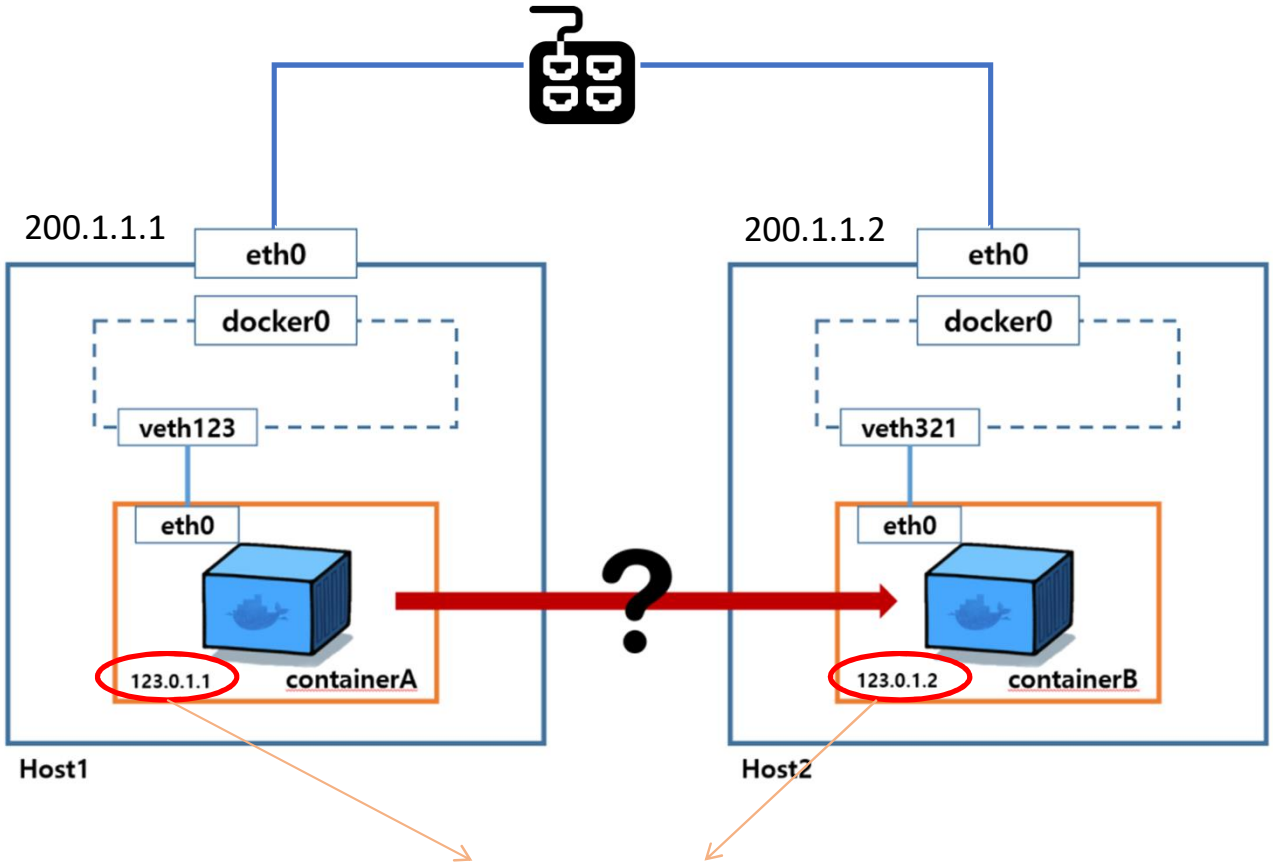
- 한 쿠버네티스 클러스터 내에 논리적인 분리 단위



Chapter. 5

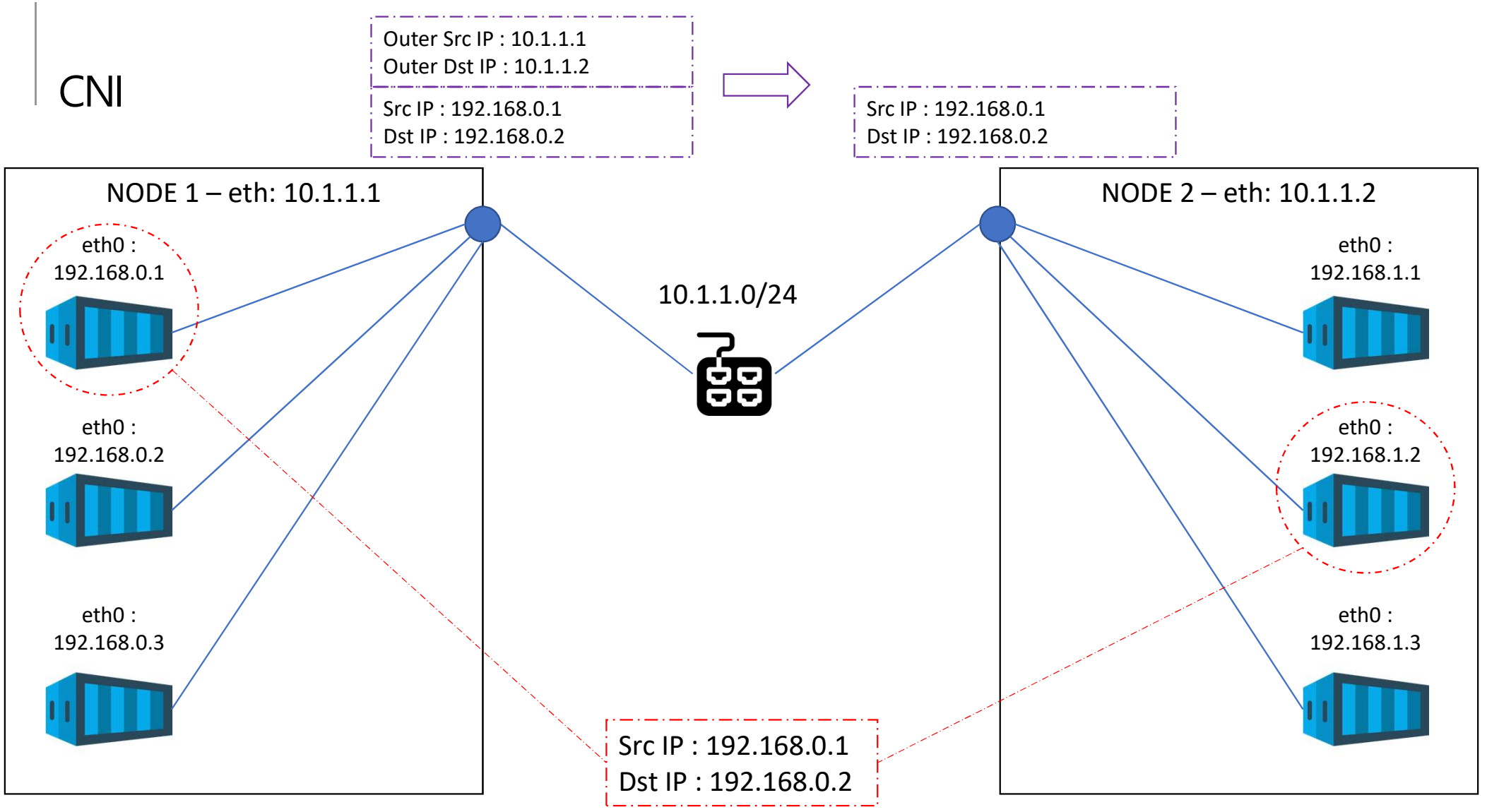
CNI(Container Network Interface)

CNI



host 내부의 각 docker0 에서 할당한 IP
→ 통신 불가능

CNI


Outer Src IP : 10.1.1.1
Outer Dst IP : 10.1.1.2
Src IP : 192.168.0.1
Dst IP : 192.168.0.2
Src IP : 192.168.0.1
Dst IP : 192.168.0.2
NODE 1 – eth: 10.1.1.1
eth0 :
192.168.0.1
eth0 :
192.168.0.2
eth0 :
192.168.0.3
10.1.1.0/24
NODE 2 – eth: 10.1.1.2
eth0 :
192.168.1.1
eth0 :
192.168.1.2
eth0 :
192.168.1.3
Src IP : 192.168.0.1
Dst IP : 192.168.0.2

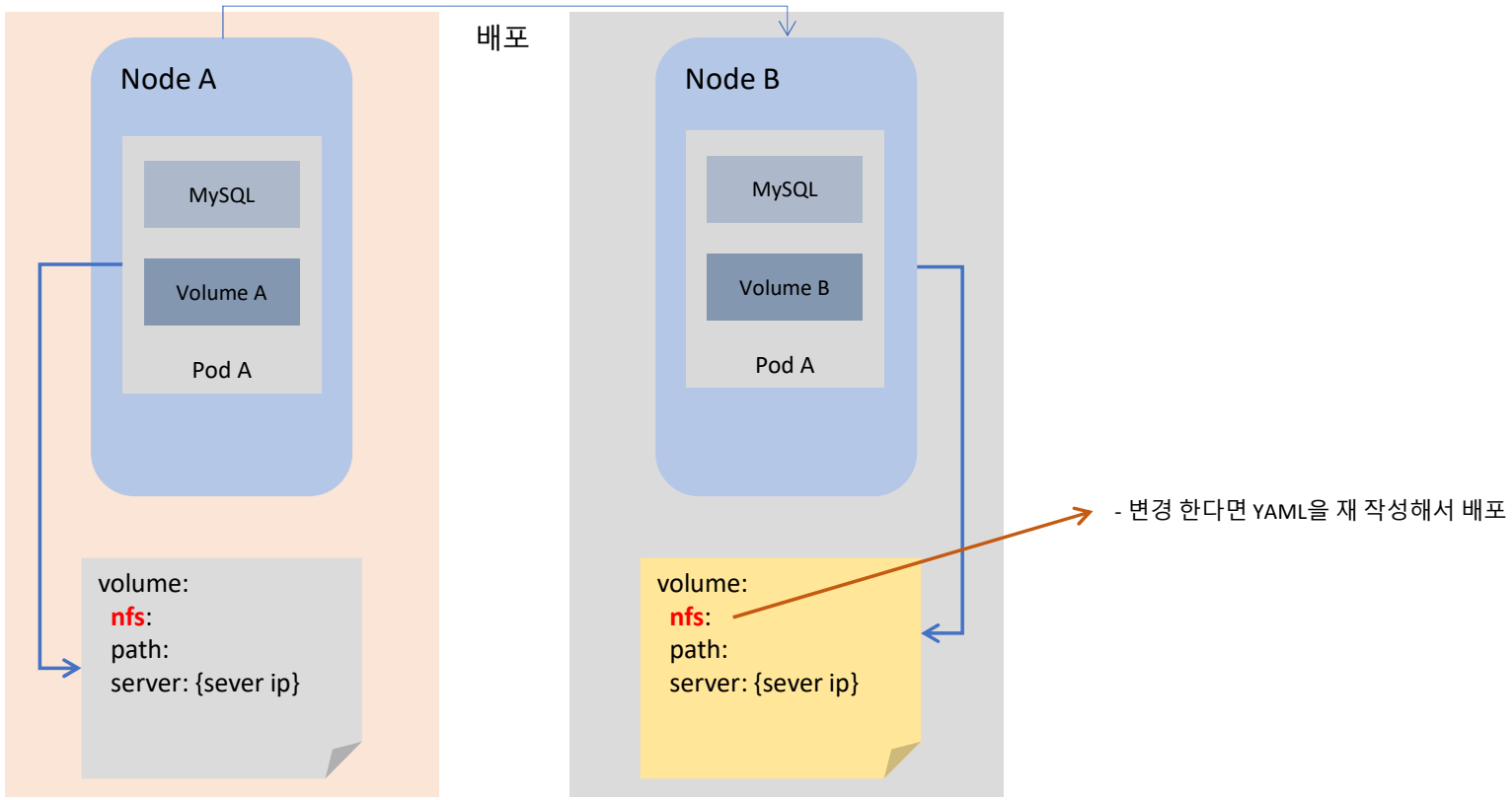
Chapter. 6

Extra

Kubernetes

▣ Volume – PersistentVolume / PersistentVolumeClaim

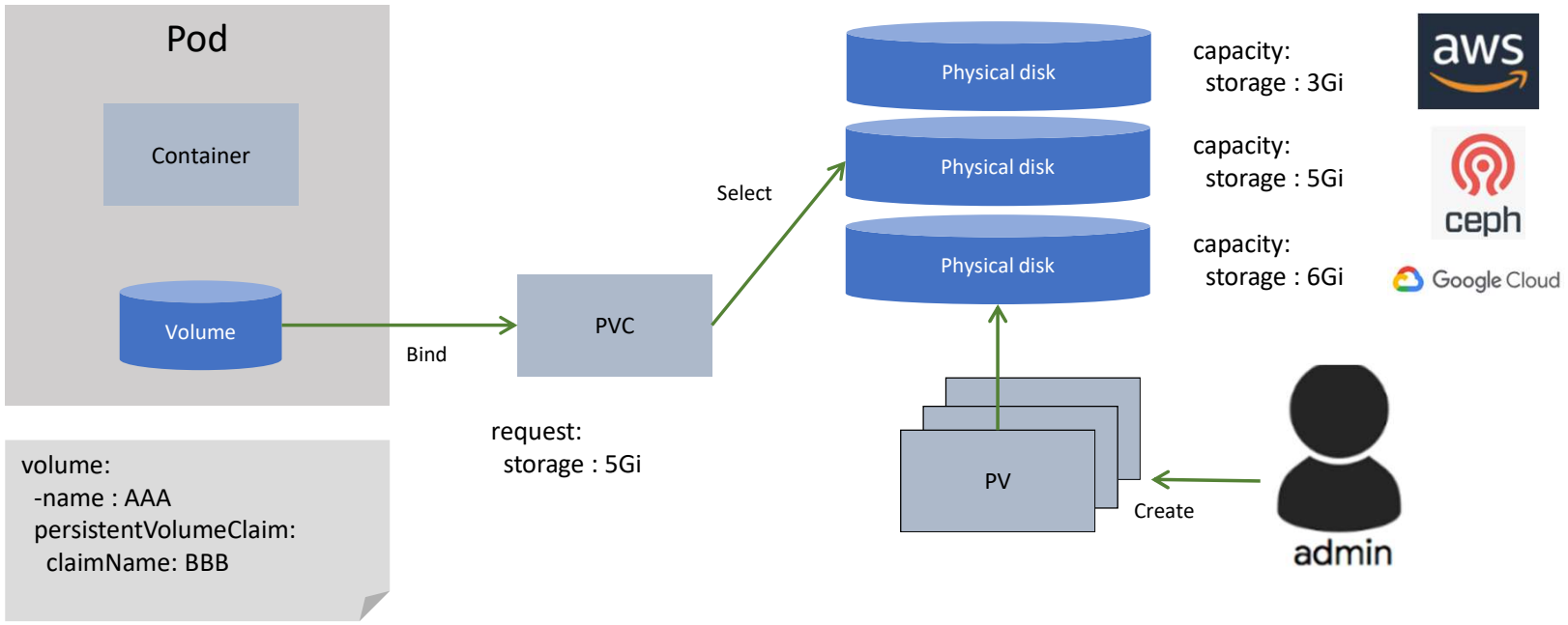
- Pod에 일반 Volume을 선언하여 배포하는 방식은 가용성이 떨어짐
- 다른 네트워크 볼륨을 사용한다면 변경할 때 마다 yaml 재 작성 필요



Kubernetes

▣ Volume – PersistentVolume / PersistentVolumeClaim

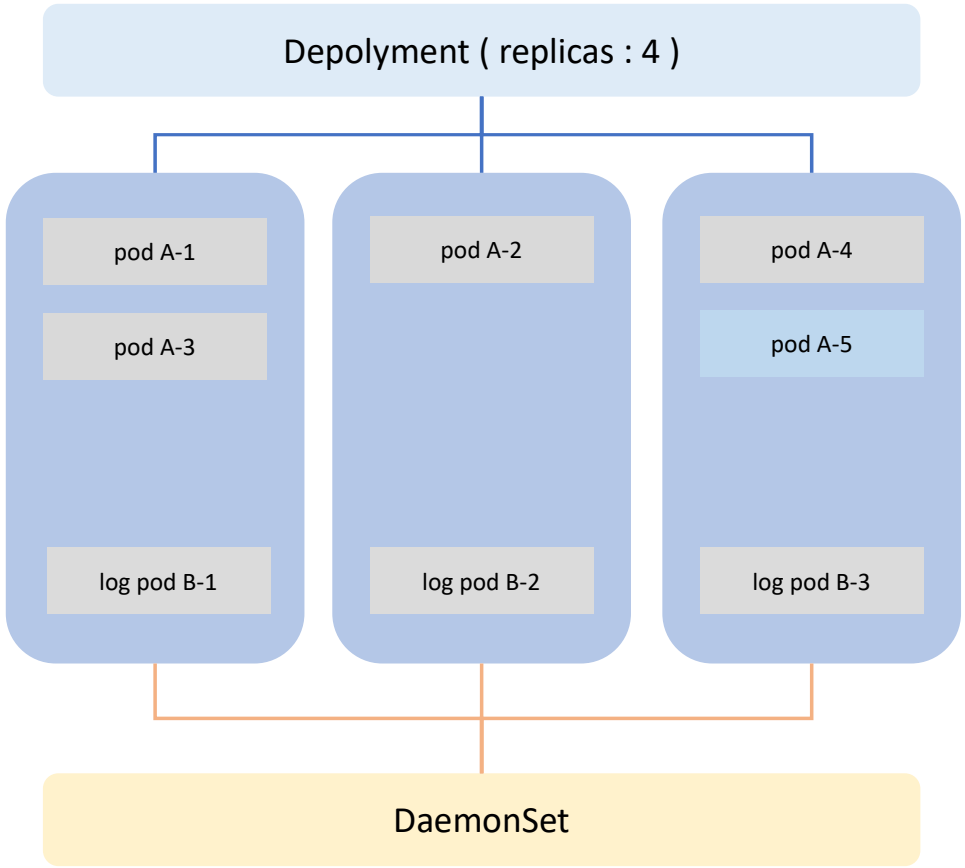
- PersistentVolume(PV) : 관리자가 개발자가 미리 필요할 것 같은 볼륨을 생성
- PersistentVolumeClaim (PVC): 개발자가 필요한 볼륨의 크기와 속성만 명시, 만들어진 PV중 조건에 맞는 PV를 bound



Kubernetes

▣ DaemonSet

- 모든 노드에 동일한 pod 사본을 실행하도록 하는 오브젝트

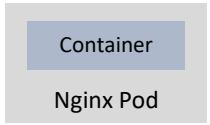



```
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: daemonset-example
spec:
 selector:
  matchLabels:
   name: my-daemonset
 template:
  metadata:
   labels:
    name: my-daemonset
  spec:
   tolerations:
   - key: node-role.kubernetes.io/master
    effect: NoSchedule
   containers:
   - name: daemonset
    image: busybox
    args: ["tail", "-f", "/dev/null"]
    resources:
     limits:
      cpu: 100m
      memory: 200Mi
```

Kubernetes

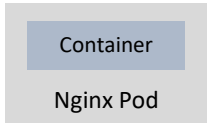
▣ StatefulSet

- Database와 같은 volume을 사용하는 application을 관리하는데 적합한 오브젝트



Deployment

Name	Ready	Status	Restarts	AGE
nginx-afscf	1/1	Running	0	13m
nginx-fa2sf	1/1	Running	0	12m
nginx-elela	1/1	Running	0	12m



StatefulSet

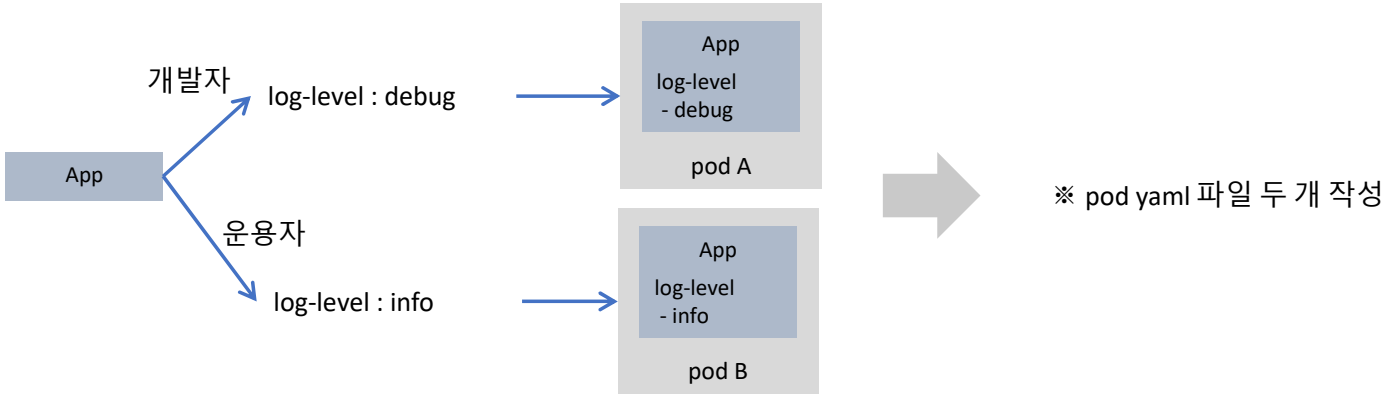
Name	Ready	Status	Restarts	AGE
nginx-0	1/1	Running	0	13m
nginx-1	1/1	Running	0	12m
nginx-2	1/1	Running	0	12m

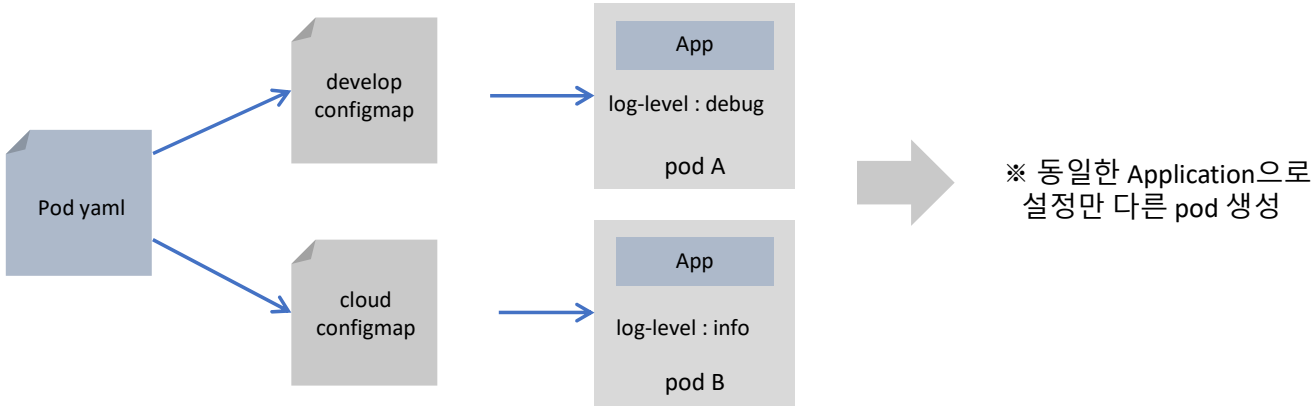
⦿ 0 ~ 오름차순으로 실행, 내림차순 으로 종료

Kubernetes

▣ ConfigMap, Secret

- 운용에 필요한 API Key 값이나, 환경설정 등을 저장 하여 pod에게 전달하는 오브젝트






원하는 이미지로 변경하여 사용하세요