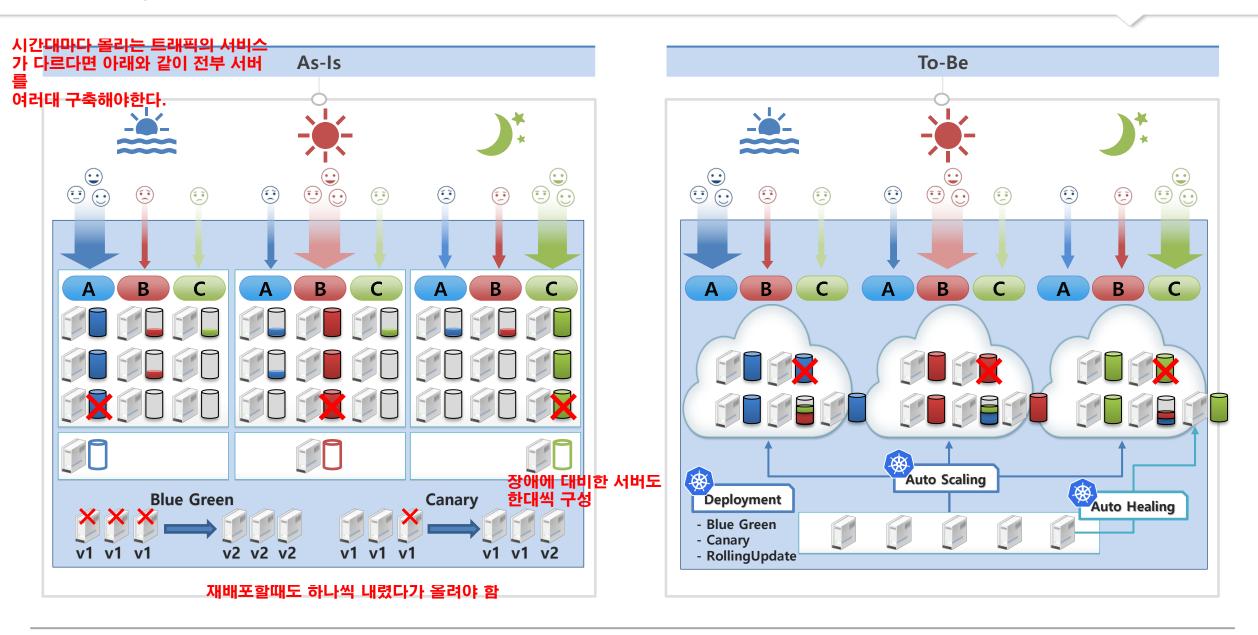
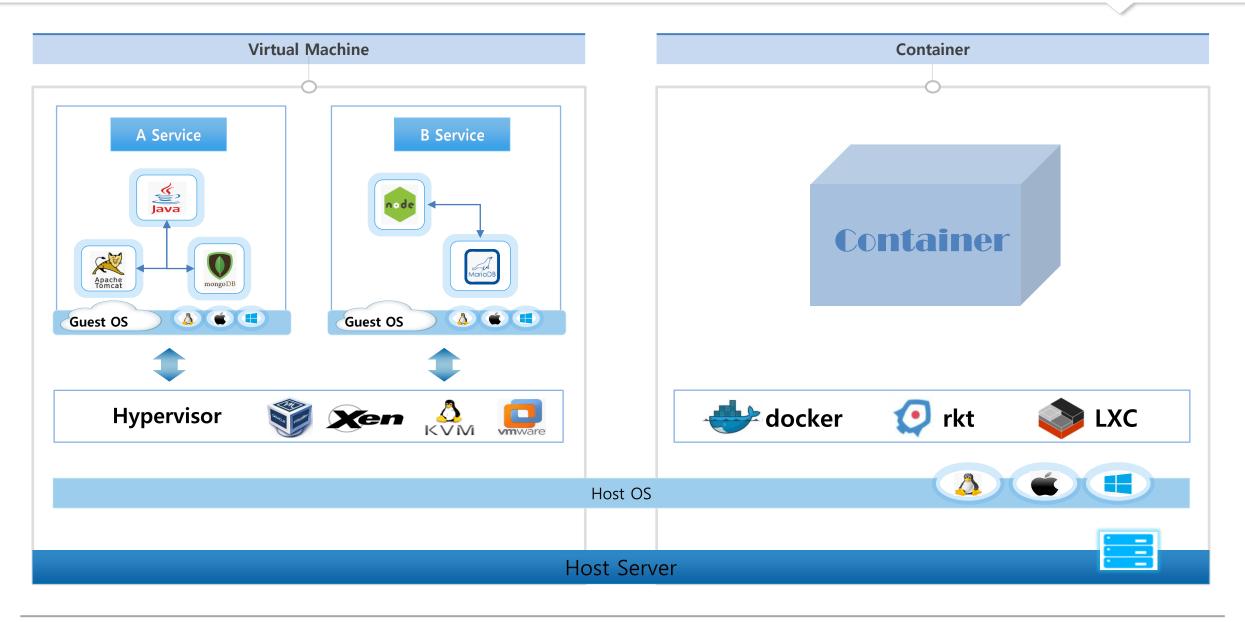
# 1. Why Kubernetes?



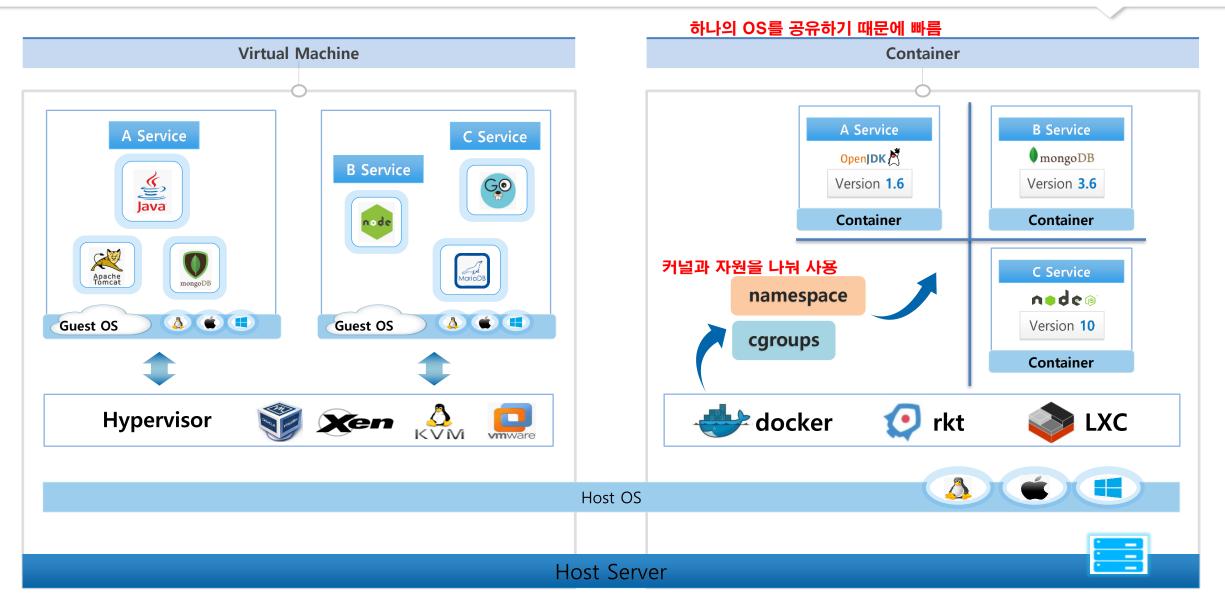
## 2. VM vs Container



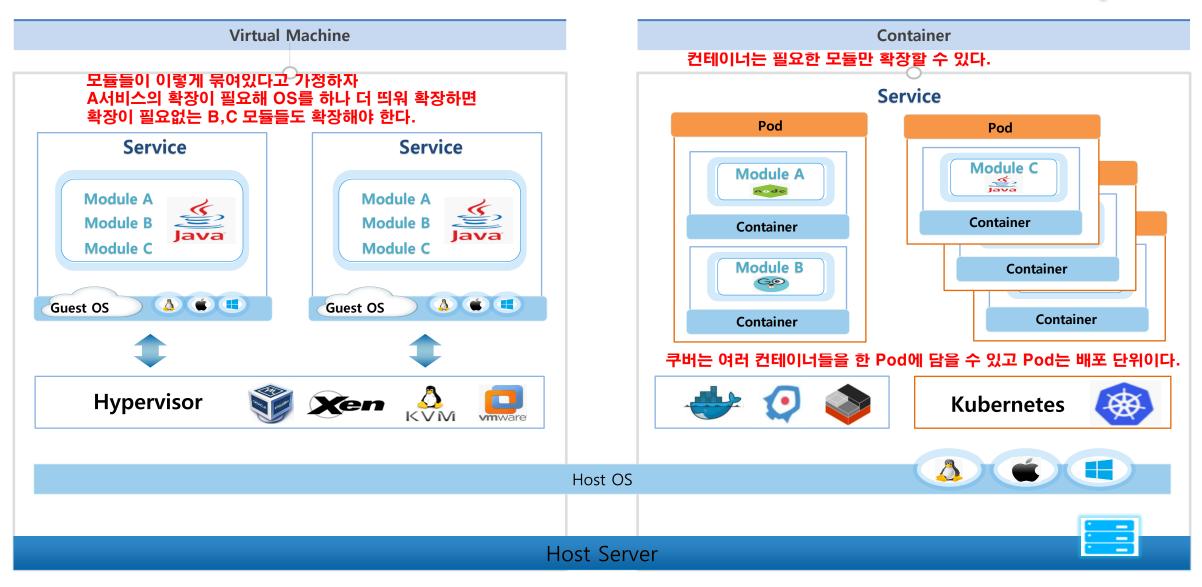
2. VM vs Container Container Virtual Mach A Service OpenJ**D**K Version 1.6 A Service **B** Service **Container** Open**JDK** mongoDB Version 1.6 Version 3.6 Container Container A Service C Service mnt, pid, net, ipc, uts, user OpenJ**DK** namespace node® memory, CPU, I/O, network cgroups A 6 H Version 1.6 Guest OS Version 10 **docker** docker **Container** Linux 6 Linux 7 Linux 7 Hypervisor  $|V \setminus V|$ Host OS Host Server

### 2. VM vs Container

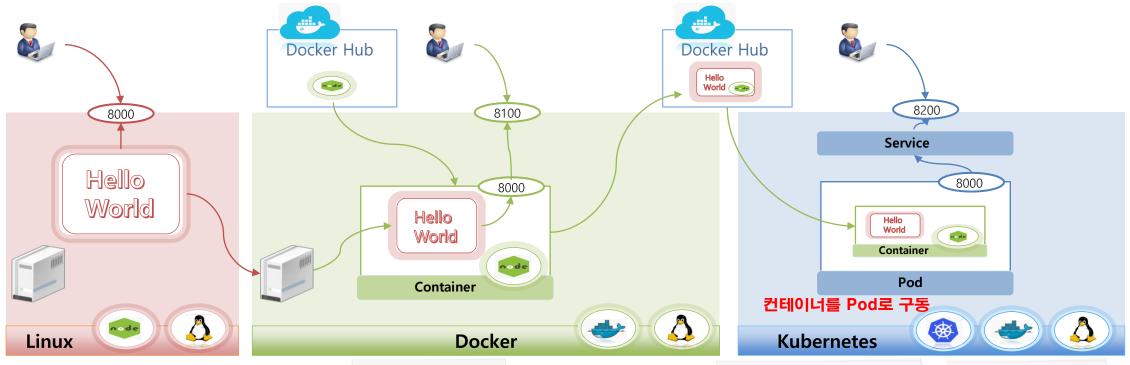
#### OS에서 제공하는 자원 격리 기술을 이용해서 컨테이너란 단위로 서비스를 분리 가능



### 2. VM vs Container



## 3. Getting started - Kubernetes



노드가 깔려있는 환경

#### **Dockerfile**

FROM node:slim EXPOSE 8000 COPY hello.js . CMD node hello.js

#### 노드가 깔려있지 않은 환경 도커로 띄움

pod 명

pod안의 컨테이너 컨테이너명: hello-컨

#### Pod

apiVersion: v1
kind: Pod
metadata:
name: hello-pod
labels:
app: hello
spec:
containers:
- name: hello-container
image: tmkube/hello
ports:

- containerPort: 8000

### Service

apiVersion: v1
kind: Service
metadata:
name: hello-svc
spec:
selector:
app: hello
ports:
- port: 8200
targetPort: 8000
externallPs:
- 192.168.0.30

컨테이너 포트: 8천 / 외부 오픈 8200 ip를 통해 접근

### 4. Kubernetes Overview

