

# Azure, Kubernetes, and AKS

**Ian Choi**

February 26, 2020

# 목차

0. Preface

1. Change & Challenge

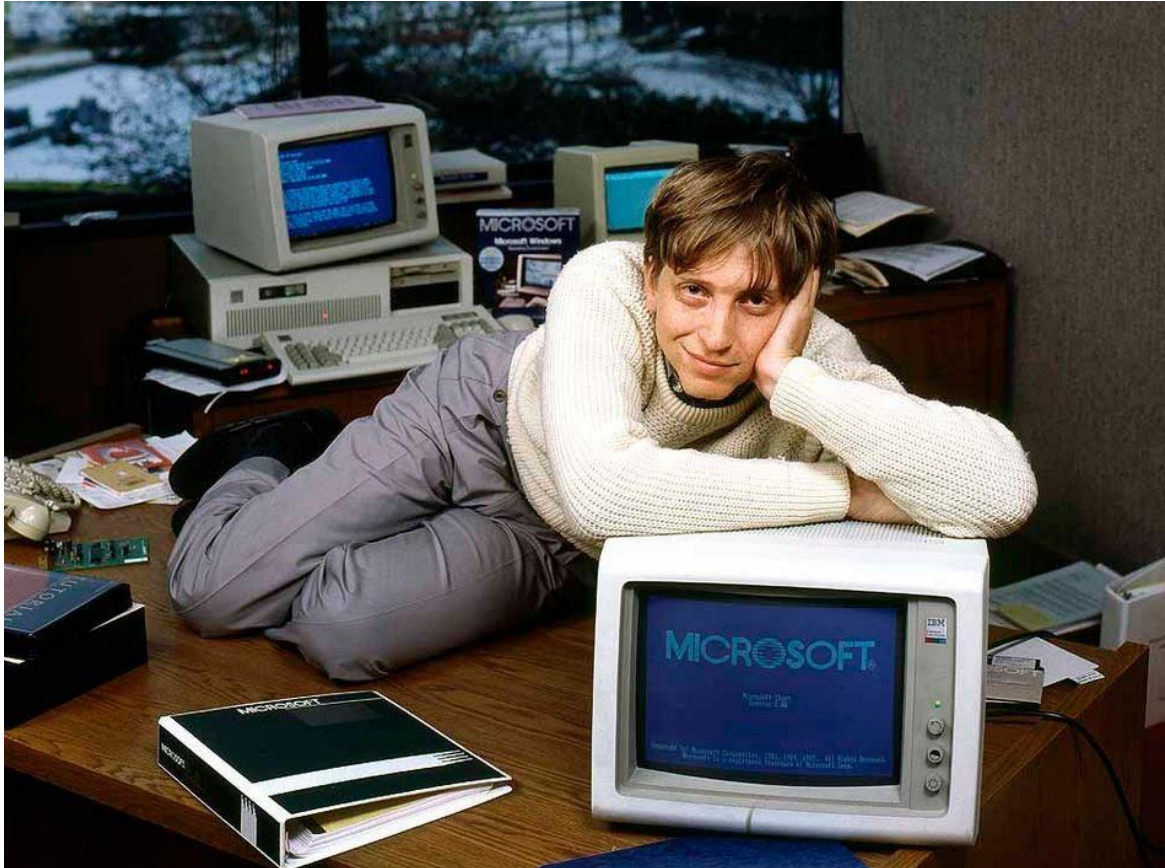
2. Kubernetes

3. Technical Differentiation

4. Hands-on-Lab: AKS with Java (simple Spring app)

# 0. Preface

# 참고: Microsoft와 개발자



```
### COMMODORE BASIC ###  
7167 BYTES FREE  
READY.  
█
```

```
C:\ E:\GWBASI-1.5\GWBASIC.EXE  
Ok  
10 CLS  
20 FOR B=2 TO 40 STEP 2  
30 PRINT B  
40 NEXT B  
50 END  
Ok  
RUN
```

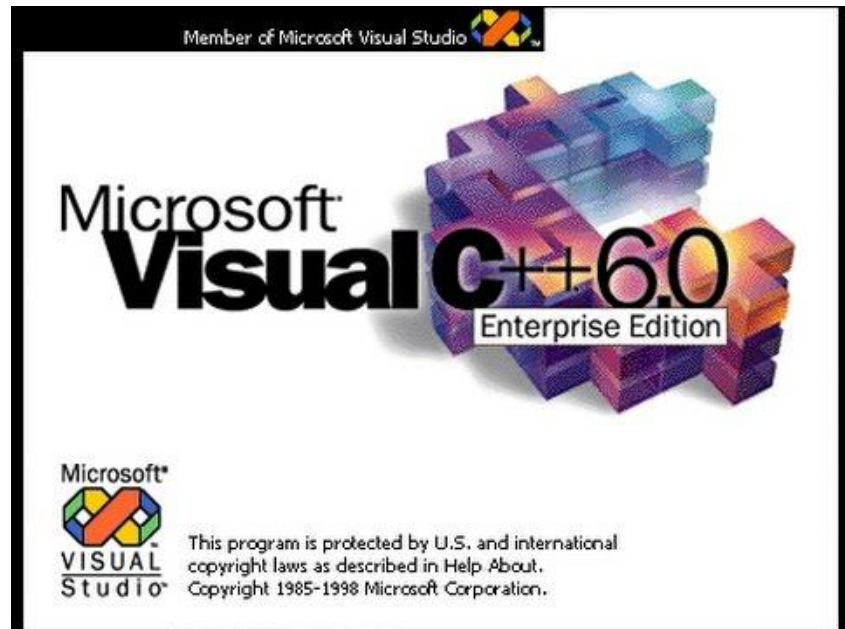
1995

1LIST 2RUN← 3LOAD" 4SAV

```
import json  
data_store = 'todos.json'  
  
class Todo:  
    def __init__(self, text):  
        self.text = text  
        self.isComplete = False  
    def complete(self):  
        self.isComplete = True  
  
class Todos:  
    def __init__(self, data_store):  
        self.data_store = data_store  
        self.todos = []  
        with open(self.data_store) as f:  
            self.todos = json.load(f)  
    def add(self, text):  
        self.todos.append(Todo(text))  
        self.save()
```

2019

# Microsoft와 개발자: only MS?





# Microsoft와 개발자: 변해가는 MS...



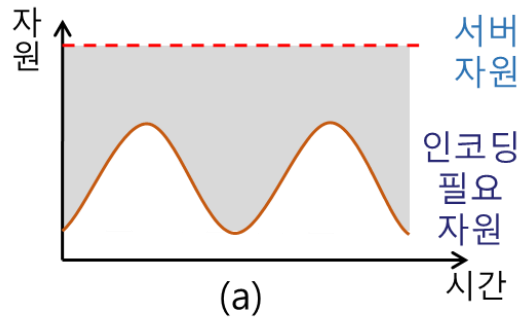
오늘 공유드리고자 하는 이야기는...



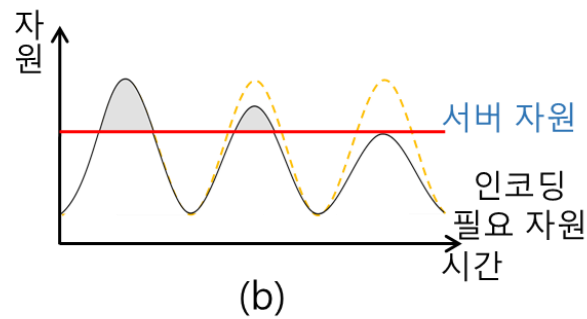
# 1. Change & Challenge



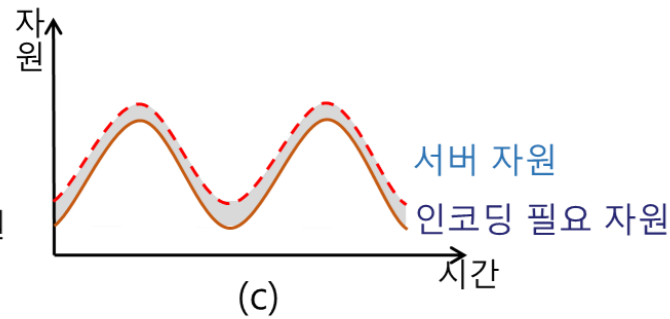
# 클라우드: VM & 가상화에서 컨테이너로



오버-프로비저닝  
인코딩

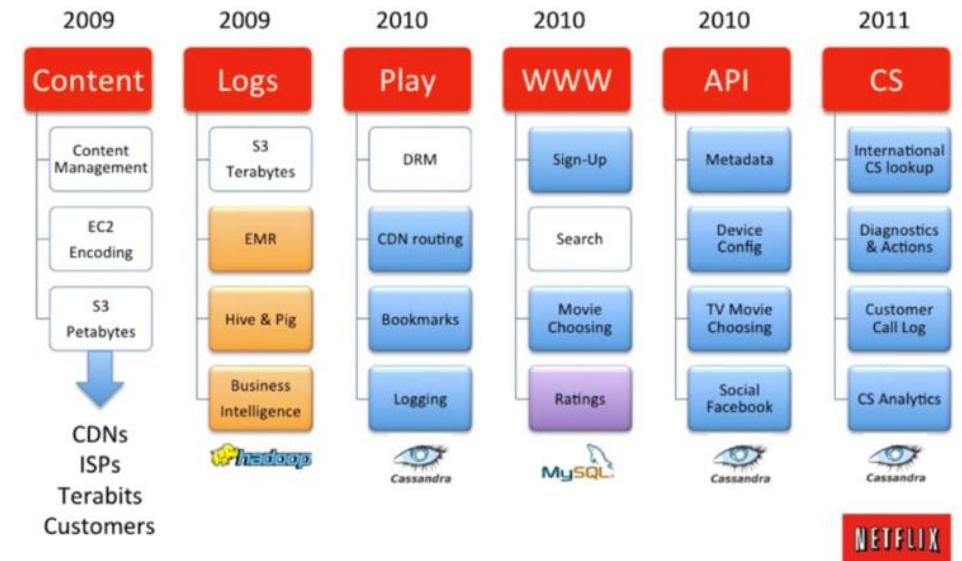
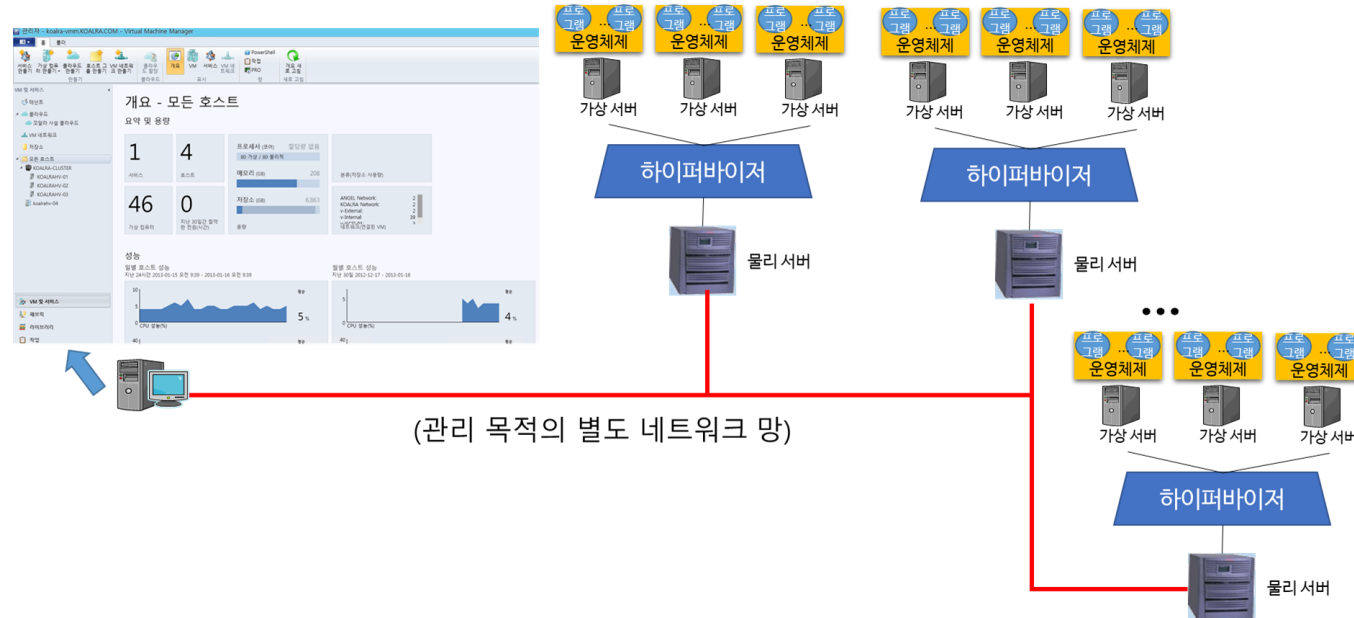


언더-프로비저닝  
인코딩



온-디맨드, 신축성있는  
인코딩 자원 환경

## Netflix's case



➔ " 컨테이너 " 관점에서 다시 보시다

# Containers, Why Change



## Aging infrastructure

- 데이터 센터의 Hardware, Operating systems, Business applications 의 노후화에 따른 영향
- 운영 비용, 효율성 및 안정성
- 자본 지출, 요구 사항
- 보안감사 및 규정 준수



## Lack of agility

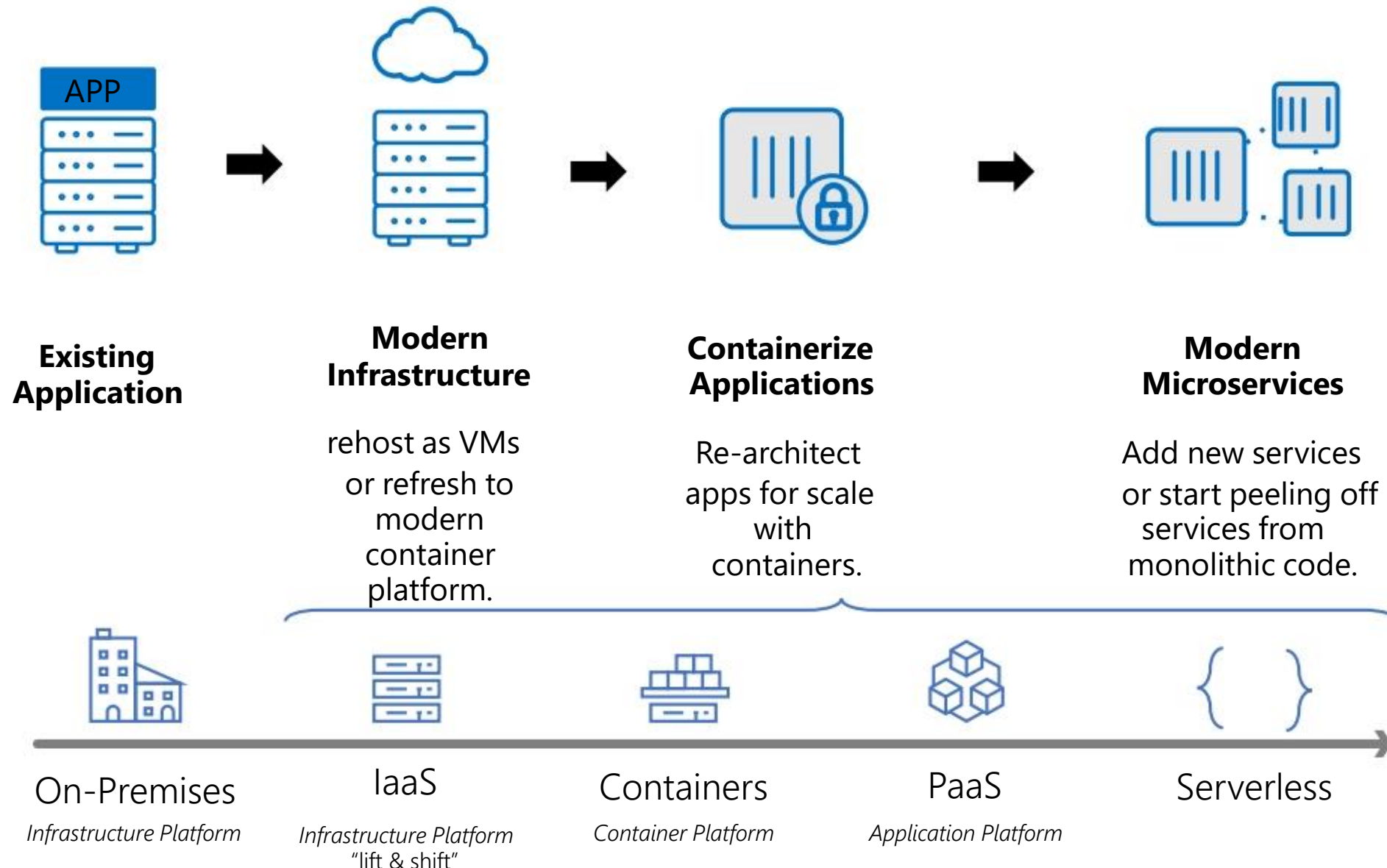
- 새로운 서비스에 대한 개발 시간
- 운영시간이 Budget과 연결
- Innovation is happening outside IT inside business areas



## High Cost

- Longer release cycles, monolithic and highly coupled architecture
- Highly IT dependent
- Low application performance and time-to-market compromise business agility

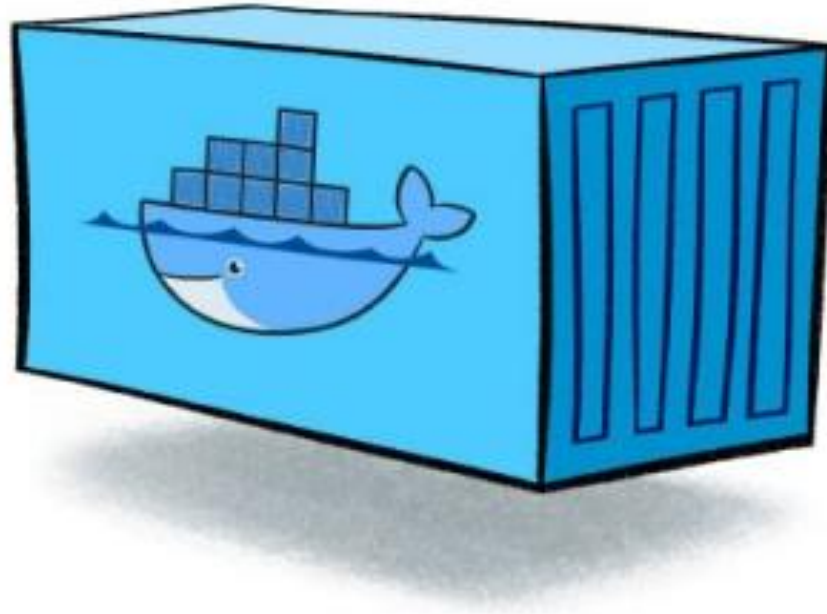
# From traditional app to modern app



# Microservice Architecture

- 시스템을 여러개의 독립된 서비스로 나눠서 , 이 서비스를 조합함으로써 기능을 제공하는 아키텍처
- SOA의 경량화 버전(실패한다면 실패하는 이유도 같음 )
- 서비스란?
  - ✓ 단일화된 기능의 묶음으로 개발된 서비스 컴포넌트
  - ✓ REST API 등을 통하여 기능을 제공
  - ✓ 데이터를 공유하지 않고 독립적으로 가공 저장
  - ✓ Kubernetes의 Service 개념과 Mapping 시킬 수 있음

# What is Container





# What is Container

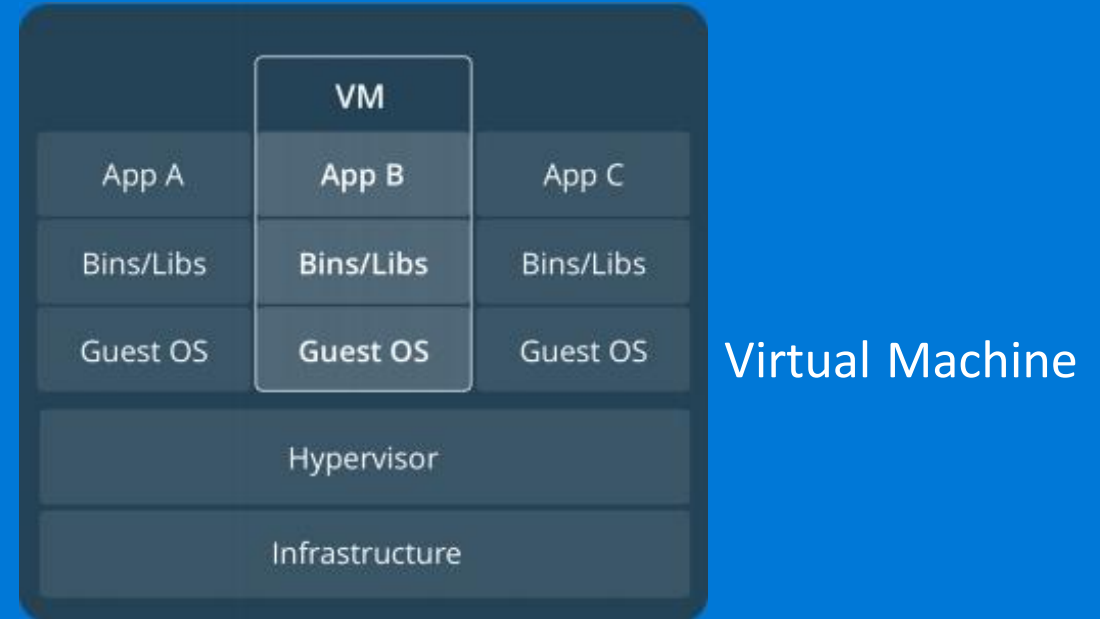


- VM보다 가벼운 형식으로 애플리케이션을 pack, ship, run
- 컨테이너는 프로세스 격리를 기반으로 하는 애플리케이션 전달 메커니즘
- Linux Kernel 기술 사용 : cgroups, namespaces + overlay fs + tooling
- 컨테이너 이미지를 사용하면 응용 프로그램 코드, 런타임 및 모든 Dependency들을 Pre-Defined Format으로 제한 가능
- Container 는 새로운 것이 아닙니다 - 리눅스 LXC, Solaris Zones, BSD Jails 처럼 기존에 있던 개념!!!
  - ✓ Docker 는 기존의 있는 것을 사용하여 빌드하고 관리할 오픈소스 소프트웨어를 만들었음

# Containers vs. VM's

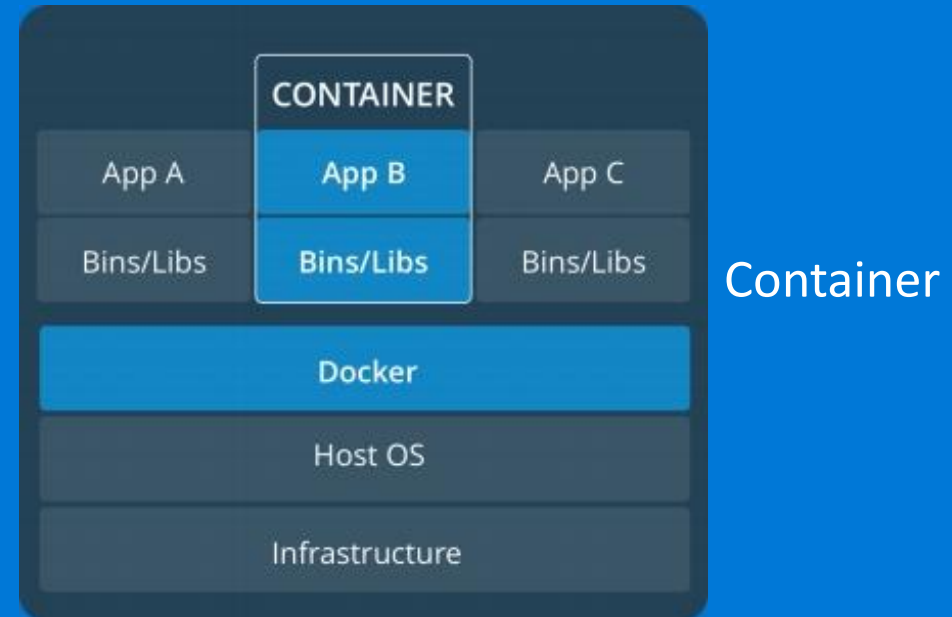
## Virtual Machines

- Each VM has independent, full OS
- Full isolation
- Separate app frameworks
- Support features such as live migration
- Slow to boot



## Containers

- Shared Host OS
- Near instant start-up
- Processes in containers are isolated
- Dependent app services and libraries are tied to container (layers)
- Every container has an isolated view and gets it's own file system, it's own PID0 and eth0 network interface



# What are Containers?

## 컨테이너에 의한 서버 환경 - 개발자들에게 인기 있는 이유

어플리케이션은 어디에서든, 동일하게 동작함

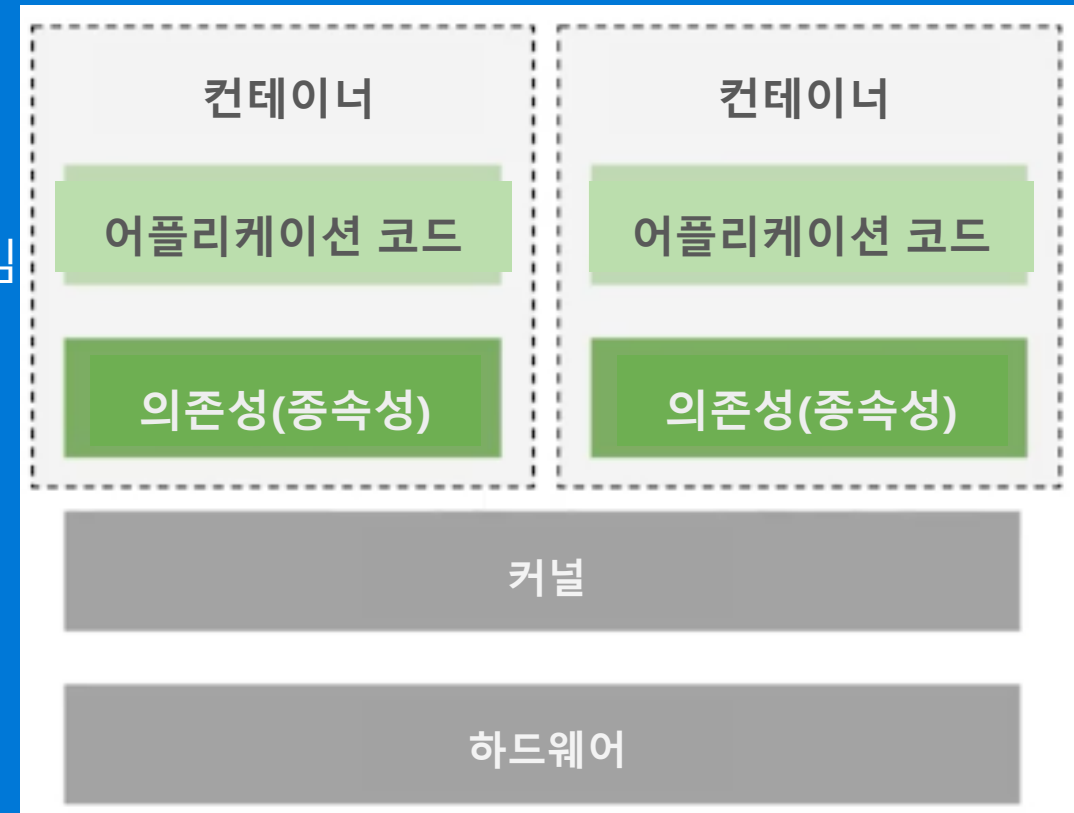
- 개발, 테스트, 프로덕션 어떤 환경에서든
- 베어메탈, 가상머신, 클라우드 어떤 환경에서든

패키지화된 어플리케이션은 개발 사이클의 속도를 빠르게 회전시킴

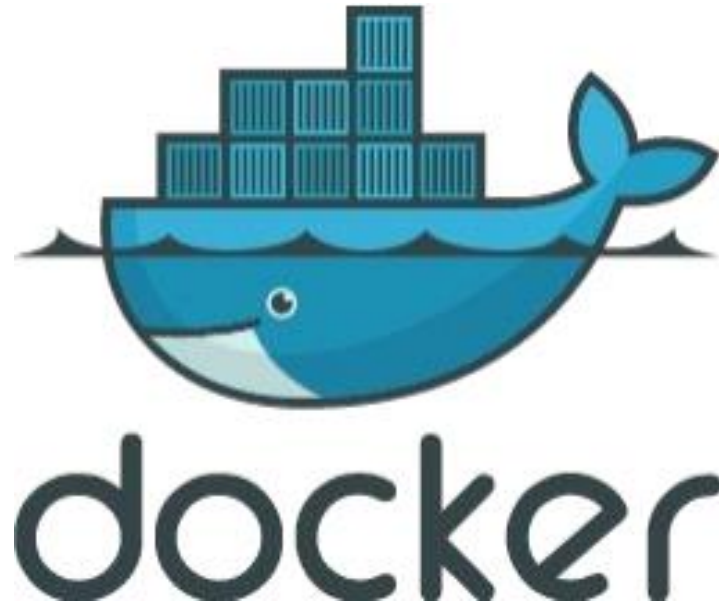
- 애자일한 생성과 배포가 가능
- 지속적인 통합/배포가 가능
- 단일 파일의 복사만으로 이를 가능하게 해줌

마이크로서비스를 가능하게 해 주기 위한 방법을 제공:

- 분석 가능성(introspectable),  
격리성(isolated), 탄력성(elastic)



# Docker



2013년 3월 PyCon Conference – Docker (Solomon Hykes)

컨테이너 기반의 오픈소스 가상화 플랫폼

**Docker - Build, Ship, and Run Any App, Anywhere**

# Docker



docker

---

## Linux Kernel

### Storage

Device Mapper

Btrfs

Aufs

### Namespaces

PID

MNT

IPC

UTS

NET

### Networking

veth

bridge

iptables

### Cgroups

cpu

cpuset

memory

device

### Security

Capability

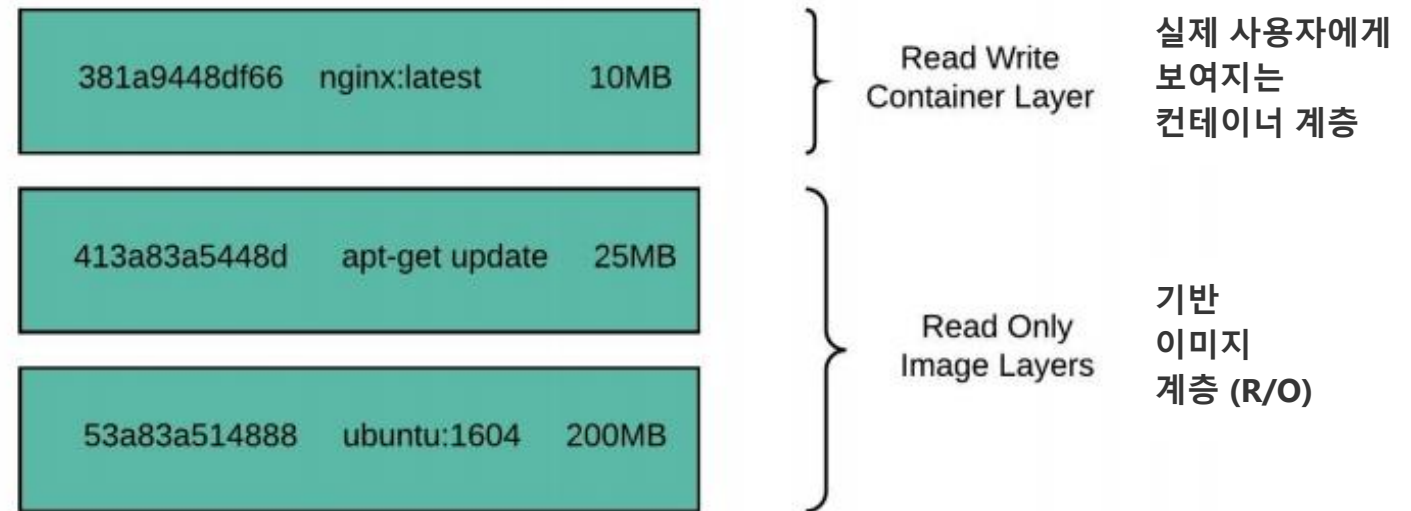
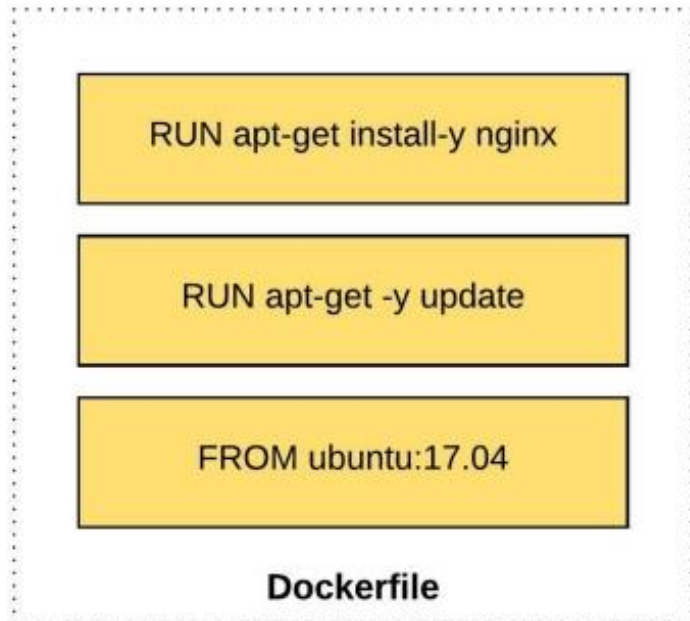
SELinux

seccomp

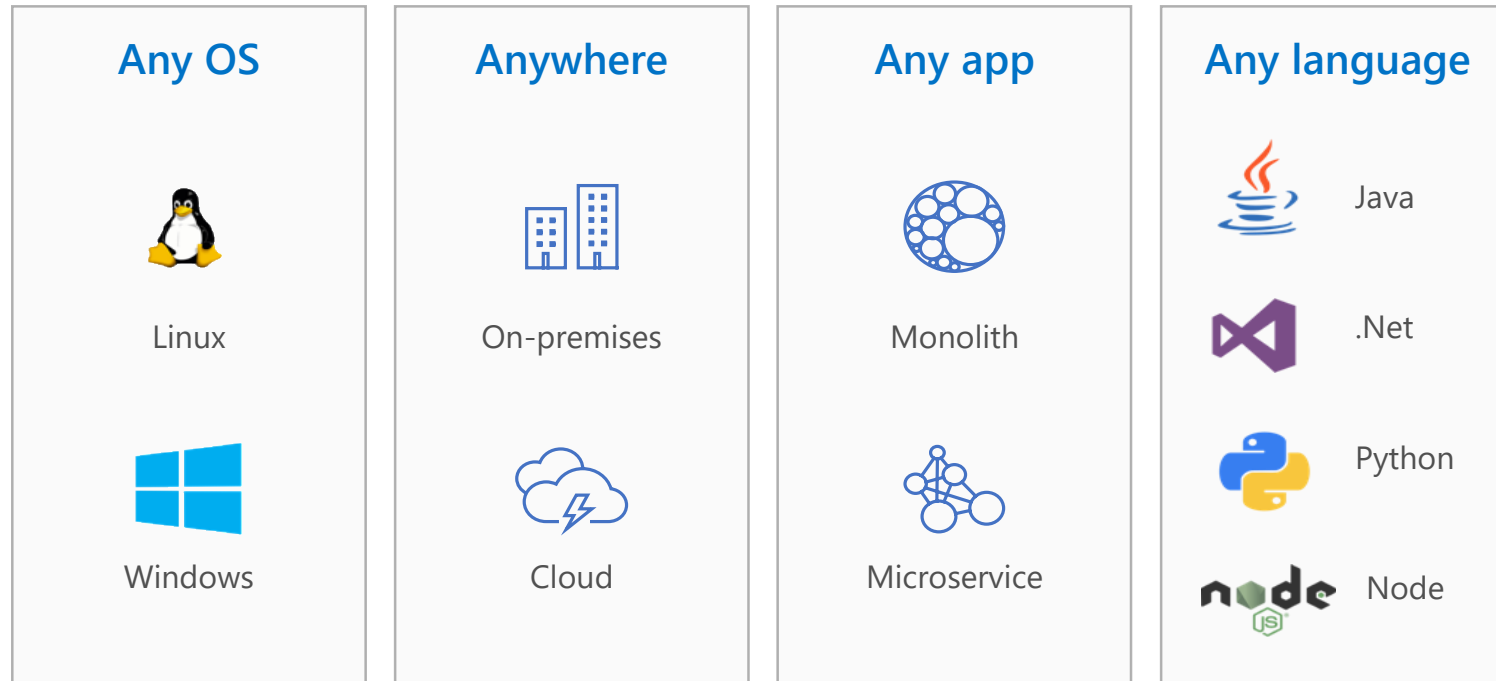


# Layers within a Docker Image

- 도커파일은 텍스트 기반 문서로, 도커 이미지를 자동으로 빌드하기 위한 명령들을 기술함
- Container 는 Multiple Layer로 구성되어 있음 - Top layer는 read/write, 나머지는 read-only
- This is made possible by an overlay filesystem
- The lowest layer is referred to as the base image (ubuntu:1704)



# The **benefits** of using containers



## 2. Kubernetes

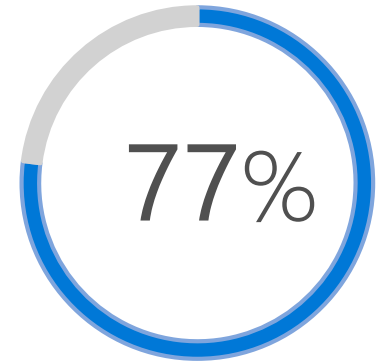
# 컨테이너 & Kubernetes 가속화는 진행중!

"By 2020, more than **50%** of enterprises will run **mission-critical, containerized cloud-native applications** in production."

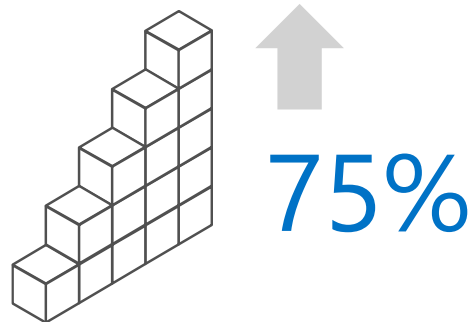
**Gartner**

Half of container environment is orchestrated.<sup>1</sup>

**77%** of companies<sup>2</sup> who use container orchestrators choose Kubernetes.

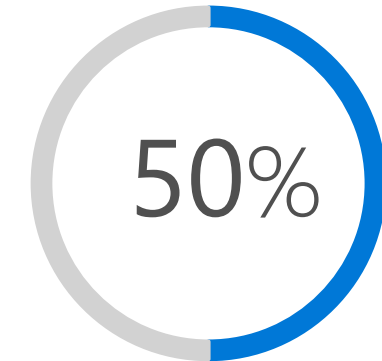


The average size of a container deployment has grown 75% in one year. <sup>1</sup>



Larger companies are leading the adoption.<sup>1</sup>

Nearly **50%** of organizations<sup>1</sup> running 1000 or more hosts have adopted containers.



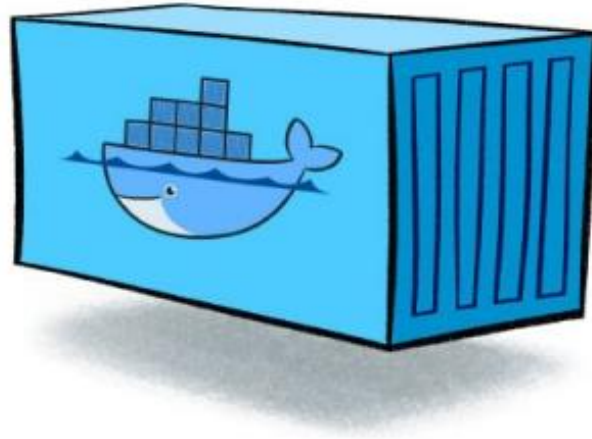
1: Datadog [report](#): 8 Surprising Facts About Real Docker Adoption

2: CNCF [survey](#): cloud-native-technologies-scaling-production-applications

# Why Container Orchestration

컨테이너 역시 하나부터 시작

=> 하나일 경우에는 문제없이 동작!!





# Why Container Orchestration

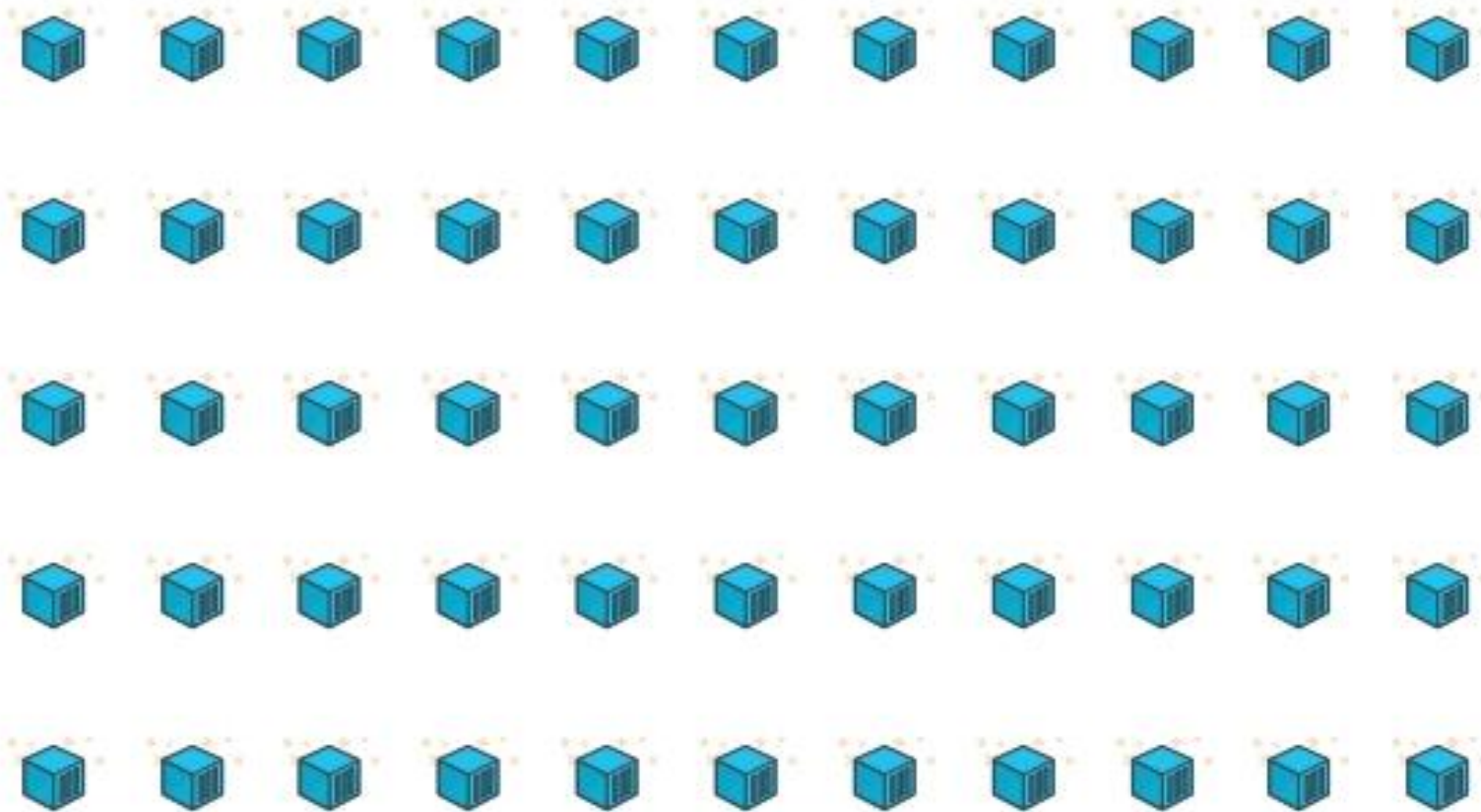
몇 개가 늘어난다 하더라도 간단한 스크립트의 도움을 받으면  
=> 문제없이 동작!!



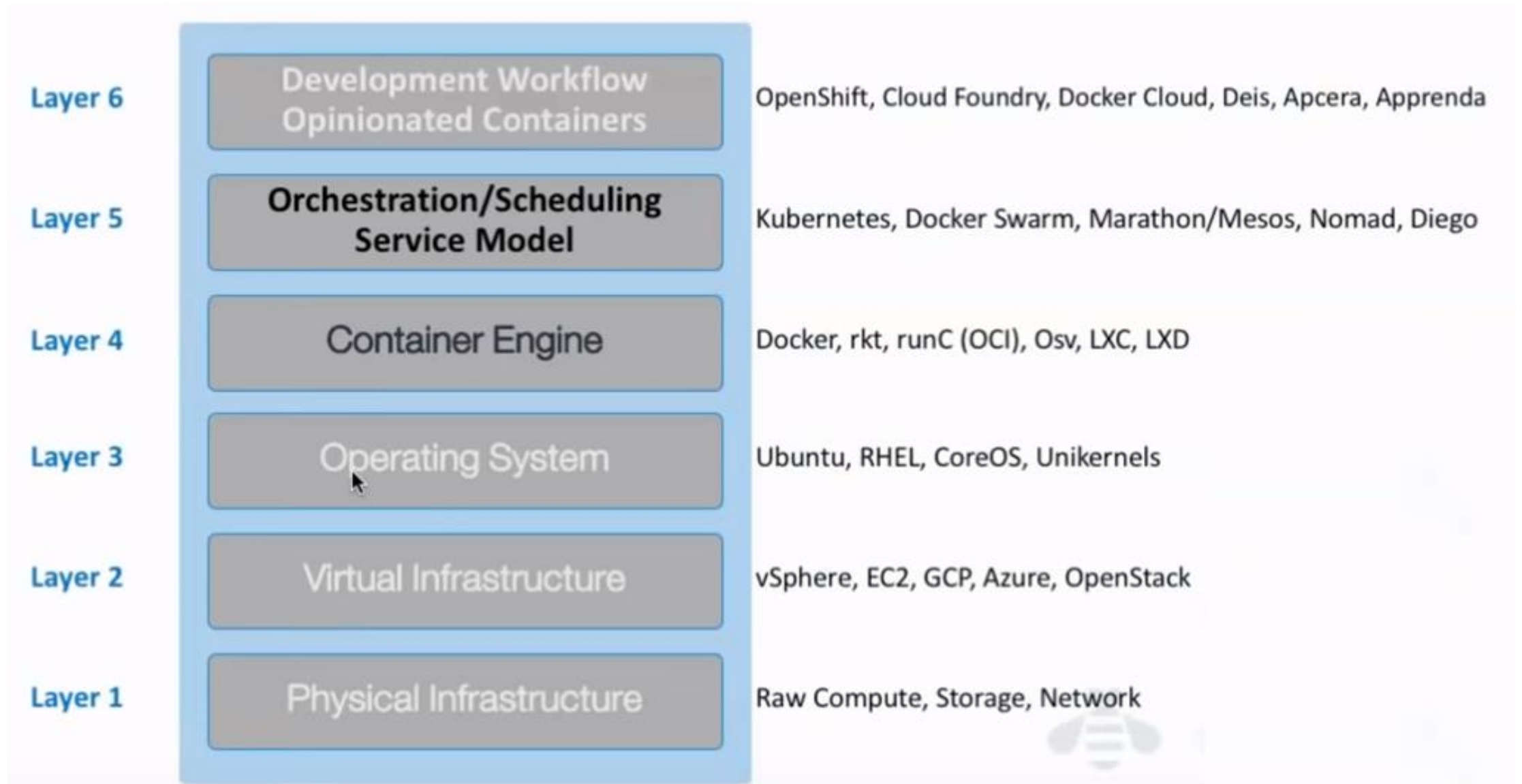
# Why Container Orchestration

Container가 수십 개일 경우엔

=> 점점 늘어만 가는 Container 감당 할 수 있을까요

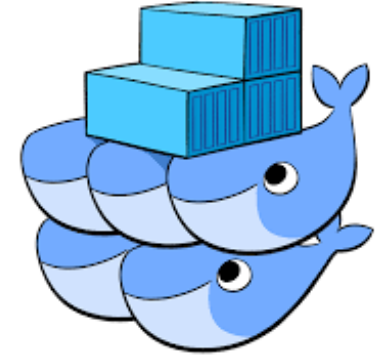


# Container Orchestration Players



# Available Orchestrators

Docker Swarm



Apache Mesos



MESOS

Nomad (from HashiCorp)



HashiCorp  
**Nomad**

Rancher



**RANCHER**

Service Fabric



Microsoft Azure  
Service Fabric



...

Kubernetes

# What is Kubernetes(k8s)?

**Kubernetes** is "an open-source software for automating deployment, scaling, and management of containerized applications".

**Kubernetes**, in Greek κυβερνήτης, means the Helmsman, or pilot of the ship.

Keeping with the maritime theme of **Docker** containers, **Kubernetes** is the pilot of a ship of containers.

## History

Google open sourced Borg. Google still actively involved

Kubernetes v1.0 was released on July 21, 2015 by Joe Beda, Brendan Burns and Craig McLuckie

Most discussed repo in Github last year. Over 1,700 authors; releases every three month

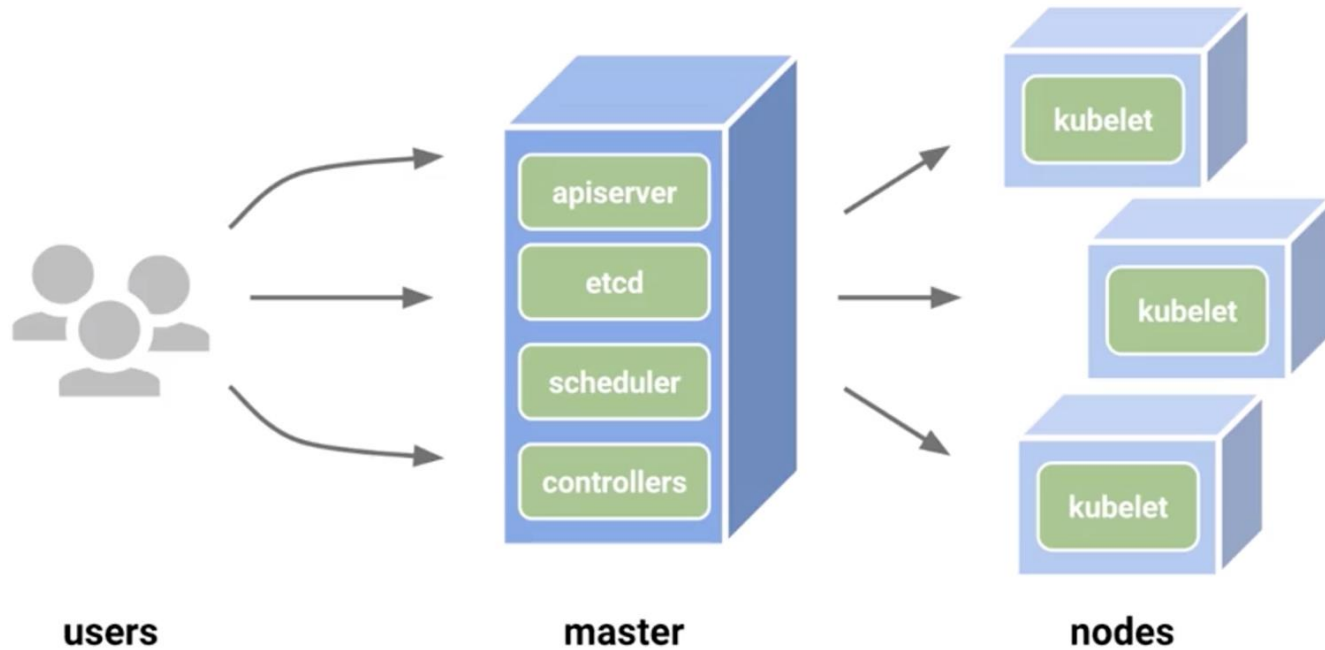
To learn more about the ideas behind Kubernetes: read the [Large-scale cluster management at Google with Borg](#) paper





# Clusters, Nodes, and Pods

## 쿠버네티스의 일반적인 설명



\* 쿠버네티스는 오픈소스기반,

컨테이너 환경의 오케스트레이터,  
통합관리격의 역할을 한다.

users

사용자

master

마스터 클러스터

- 모든 작업을 제어함
- 작업들의 스케줄링도 제어함
- 분산 키-밸류의 안정성 있는 제어/관리
- API 서버 제어/관리

nodes

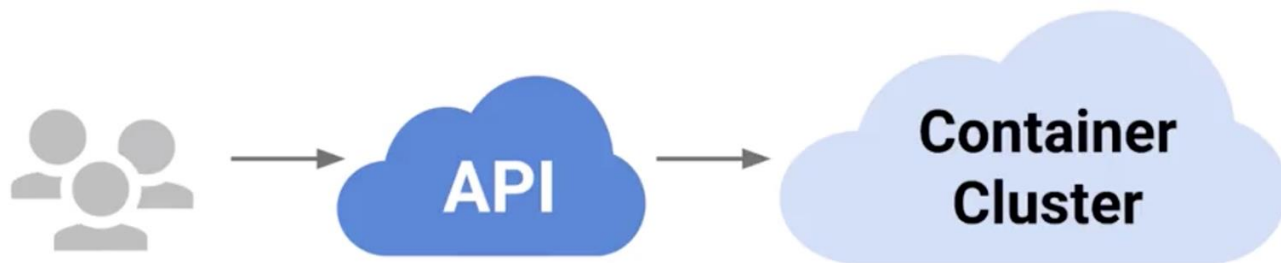
kubelet (쿠블릿)

- 쿠버네티스 에이전트 줌으로  
생각해볼 수 있음

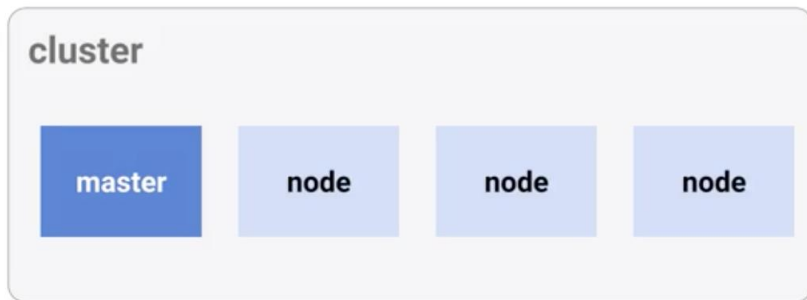
**etcd:** distributed reliable key-value store for the most critical data of a distributed system

# Clusters, Nodes, and Pods

## 쿠버네티스의 일반적인 설명

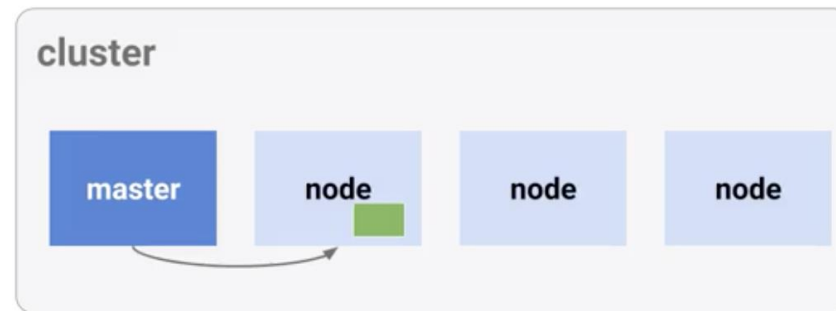


\* 쿠버네티스는 개발자로 하여금 컨테이너 클러스터에 접근할 수 있는 Open API 접근 방법을 제공함



\* 클러스터는 포함된 여러 노드들을 관리하는 컴퓨터 인스턴스

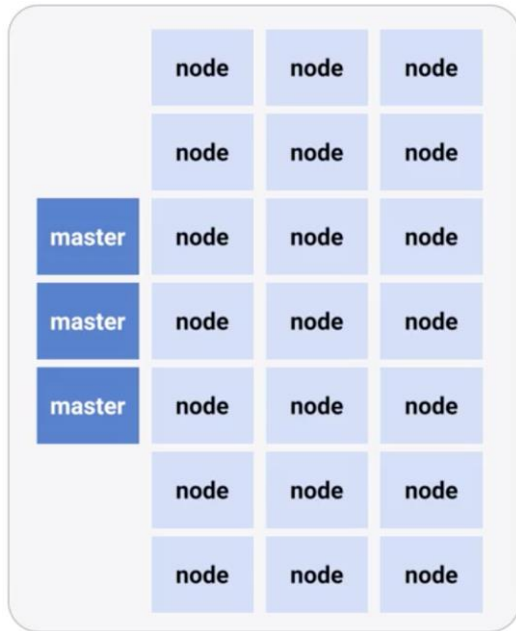
\* 모든 전체 그림은 쿠버네티스에 의해서 관리됨



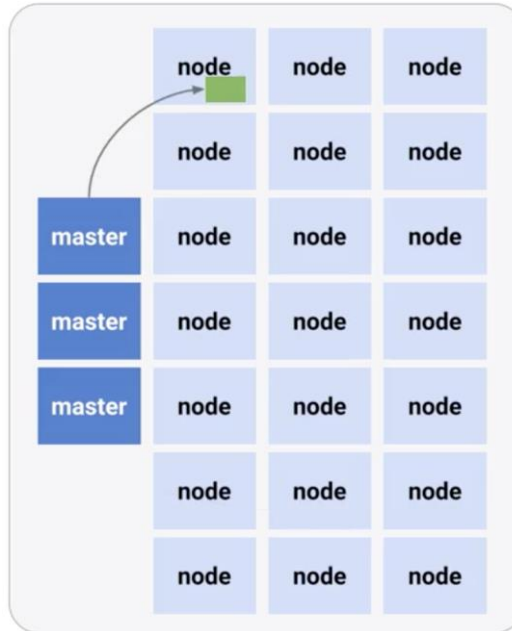
\* 어떤 작업이 발생시, 이를 어떻게 어디로 배치할지를 쿠버네티스가 결정하고 관리함

# Clusters, Nodes, and Pods

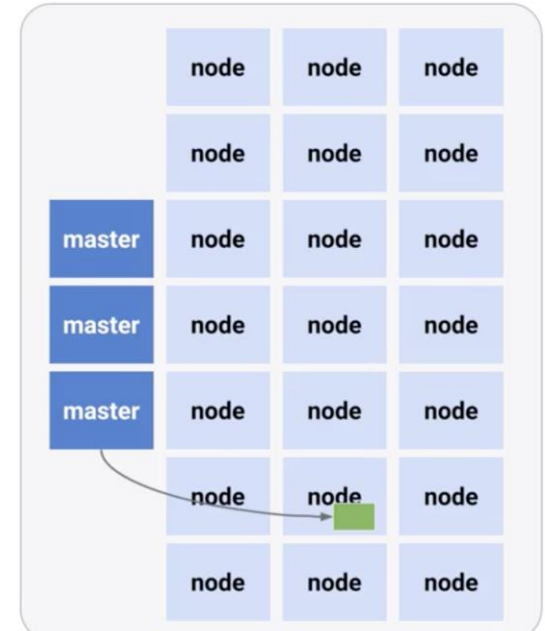
## 쿠버네티스의 일반적인 설명



\* 현실 세계에서, 일반적으로 하나의 클러스터는 ~ 수천개의 노드, 여러개의 마스터로 구성됨



\* 리소스가 널널한 노드로 작업을 할당



\* 리소스가 꽉찬 노드에서 특정 작업을 제거

# Kubernetes: the industry leading orchestrator



## Portable

Public, private, hybrid,  
multi-cloud

## Extensible

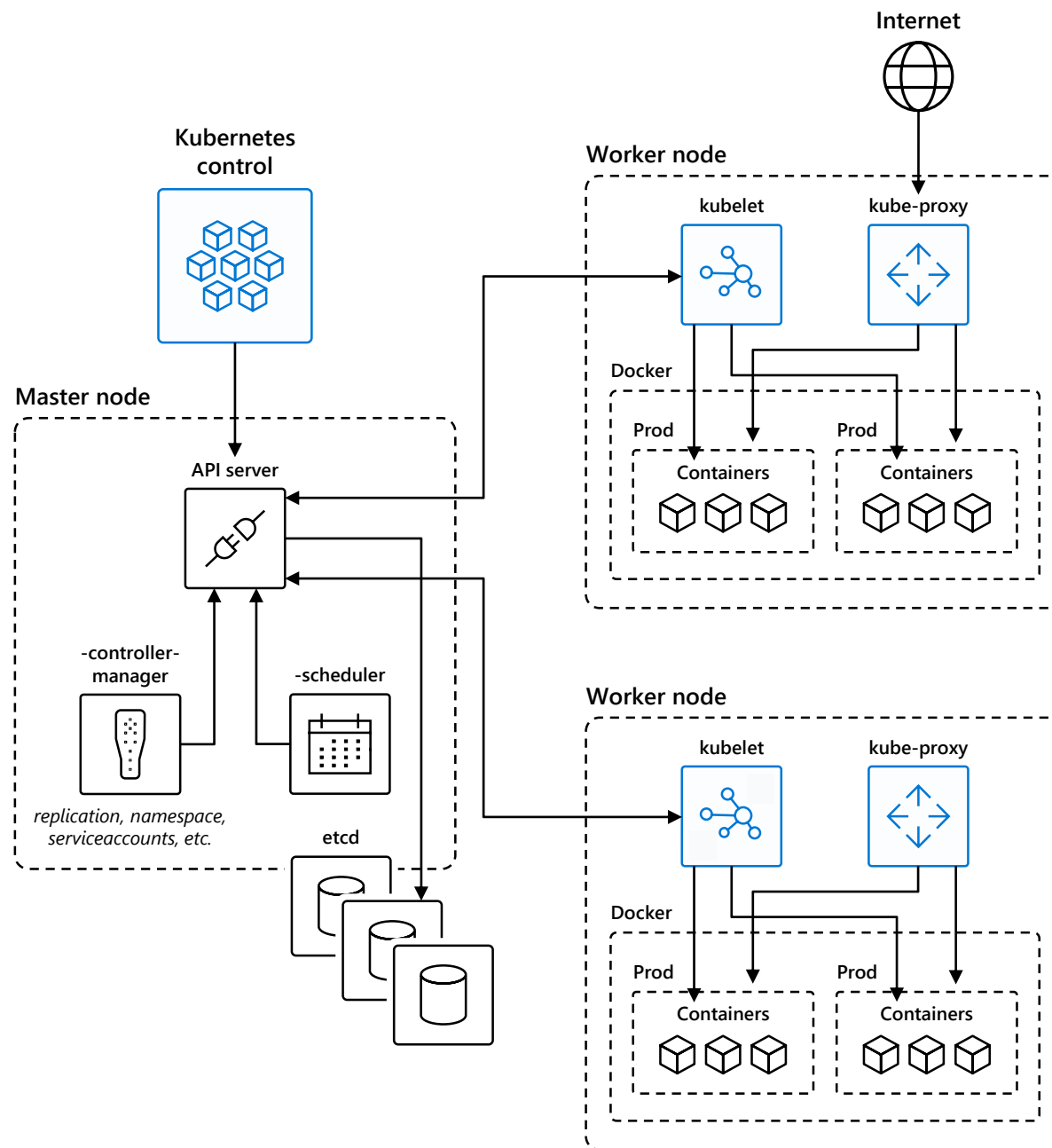
Modular, pluggable,  
hookable, composable

## Self-healing

Auto-placement, auto-restart,  
auto-replication, auto-scaling

# Kubernetes 101

















1. Kubernetes 사용자: API 서버와 통신을 하여 "state" 를 가짐
2. Master node: worker node들이 해당 "state"를 가지도록 설정 & 보장
3. Worker node: 컨테이너를 직접적으로 관리
4. 또한 인터넷 연결에 대한 관리가 Worker node에서 필요





# 인프라에서 혁신으로

## Managed Kubernetes empowers you to do more

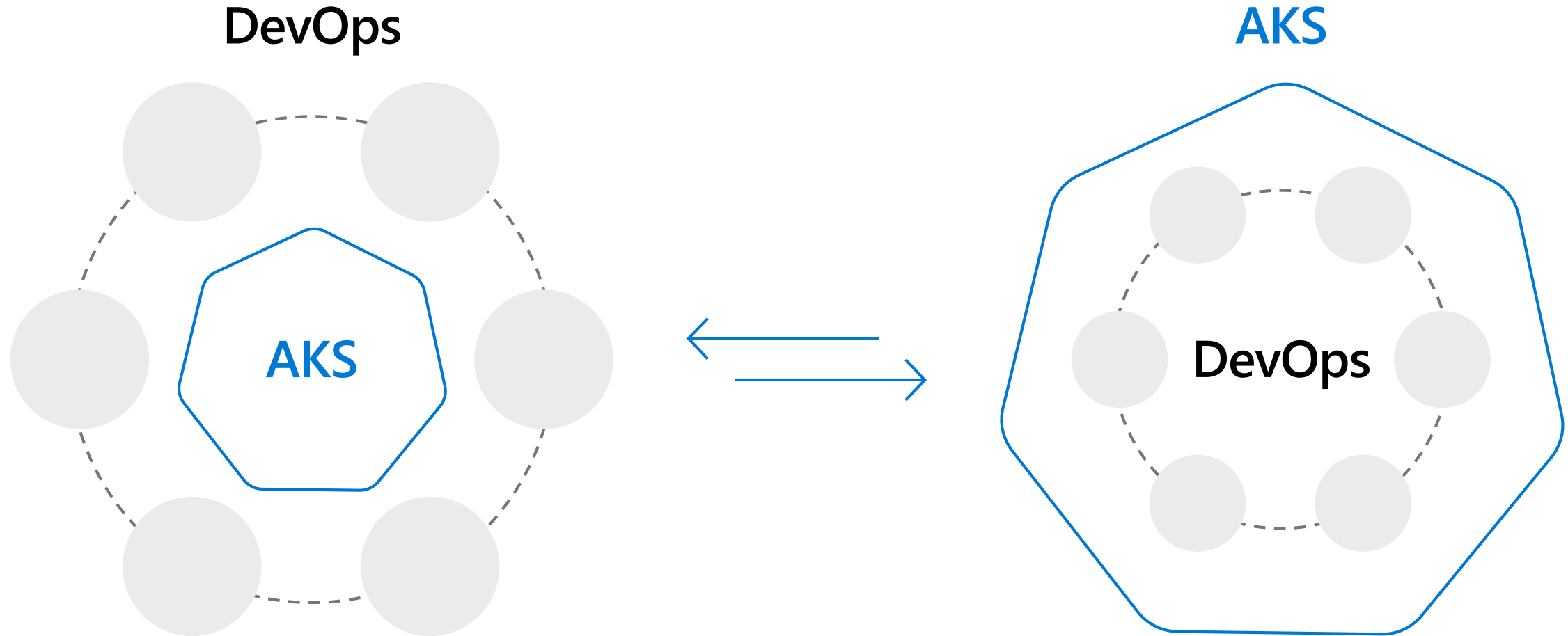
Focus on your containers and code, not the plumbing of them.

Responsibilities	DIY with Kubernetes	Managed Kubernetes on Azure
Containerization		
Application iteration, debugging		
CI/CD		
Cluster hosting		
Cluster upgrade		
Patching		
Scaling		
Monitoring and logging		

 Customer

 Microsoft

# Kubernetes 와 DevOps: better together



# 3. Technical differentiation



# What are the features of Kubernetes ?

가상 또는 물리적 인프라 환경 모두에서 클러스터 구축 가능(Cloud & Bare metal)

고가용성(High availability) 구성

자동배포(rollout & rollback)

원할한 수평(horizontal) scaling

키 관리(Secret Management)

Service discovery and load balancing

간단한 로그 수집 기능

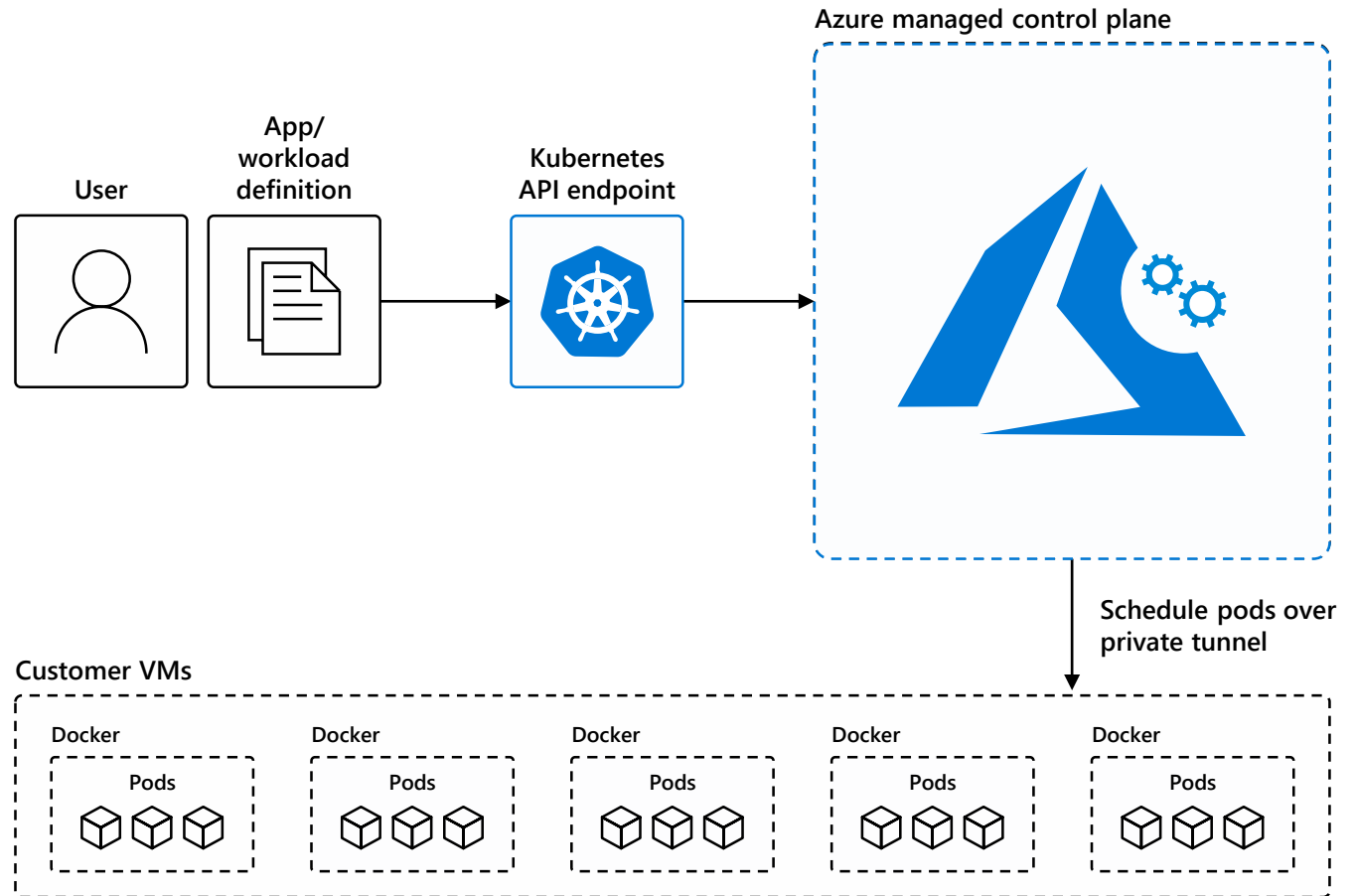
Stateful application support

Persistent volume 관리

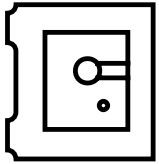
CPU/memory quotas

# How managed Azure Kubernetes Service

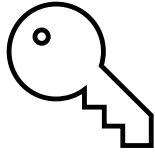
- 자동 업그레이드, patches
- 높은 신뢰성, 가용성(availability)
- 쉽고 안전한 클러스터 scaling
- 자기 치유(Self-healing)
- API server monitoring
- Control Plane 비용 추가가 없음(Master Node 무료!)
- GPU Support
- 100% upstream Kubernetes



# Secure your Kubernetes environment with layers of isolation



AAD 및 RBAC 을 통한  
액세스 제어



Key Vault와 함께 Key와  
Secret 보호



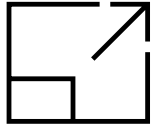
VNET 및 policy와 안전한  
네트워크 통신



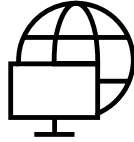
SOC, HIPAA 및 PCI 인증을  
획득 한 Kubernetes 서비스



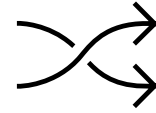
# Scale applications on the fly



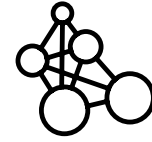
Built-in 자동  
스케일링



글로벌 데이터 센터로  
성능 및 범위 확대



ACI를 함께 사용하여  
AKS cluster를  
Elastically burst



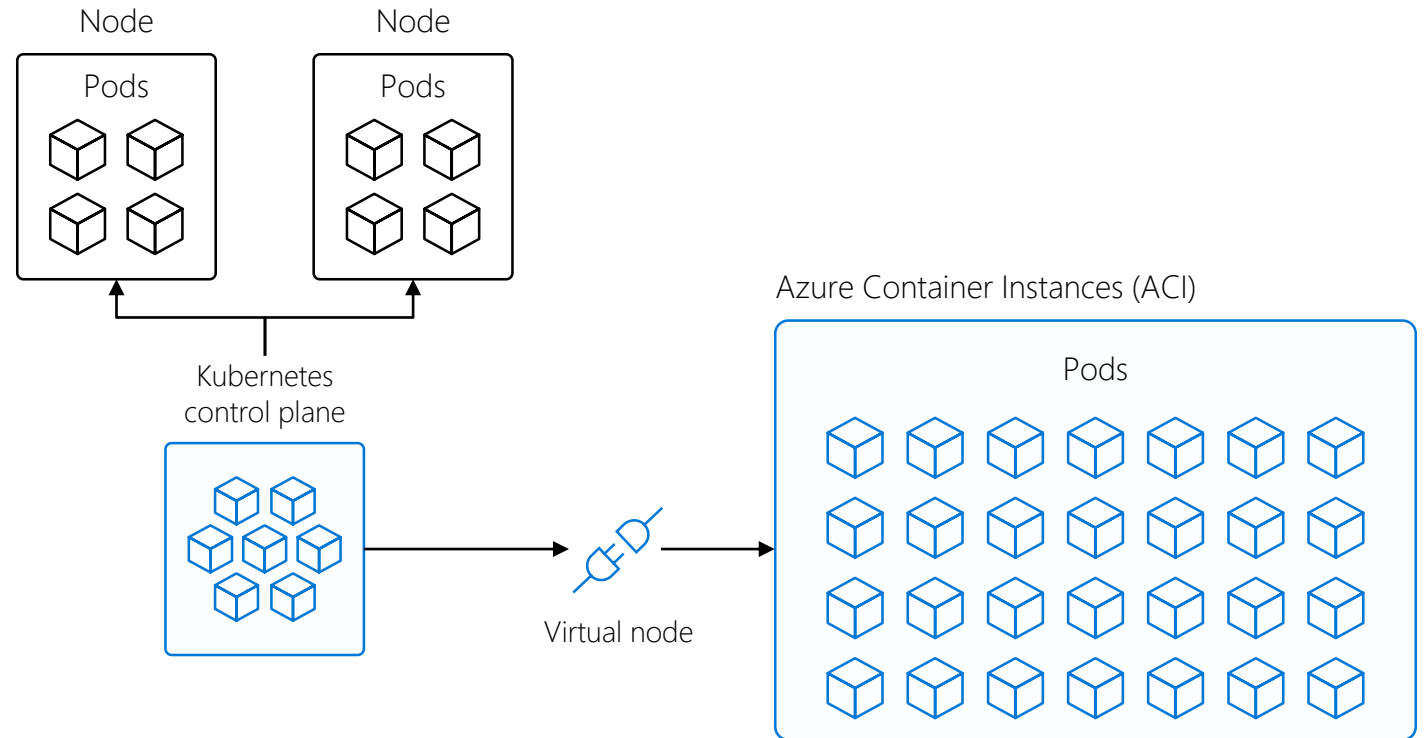
이미지 레이턴시를 줄이기  
위해 Geo-replicated  
container registry



# Serverless for AKS with Virtual Node

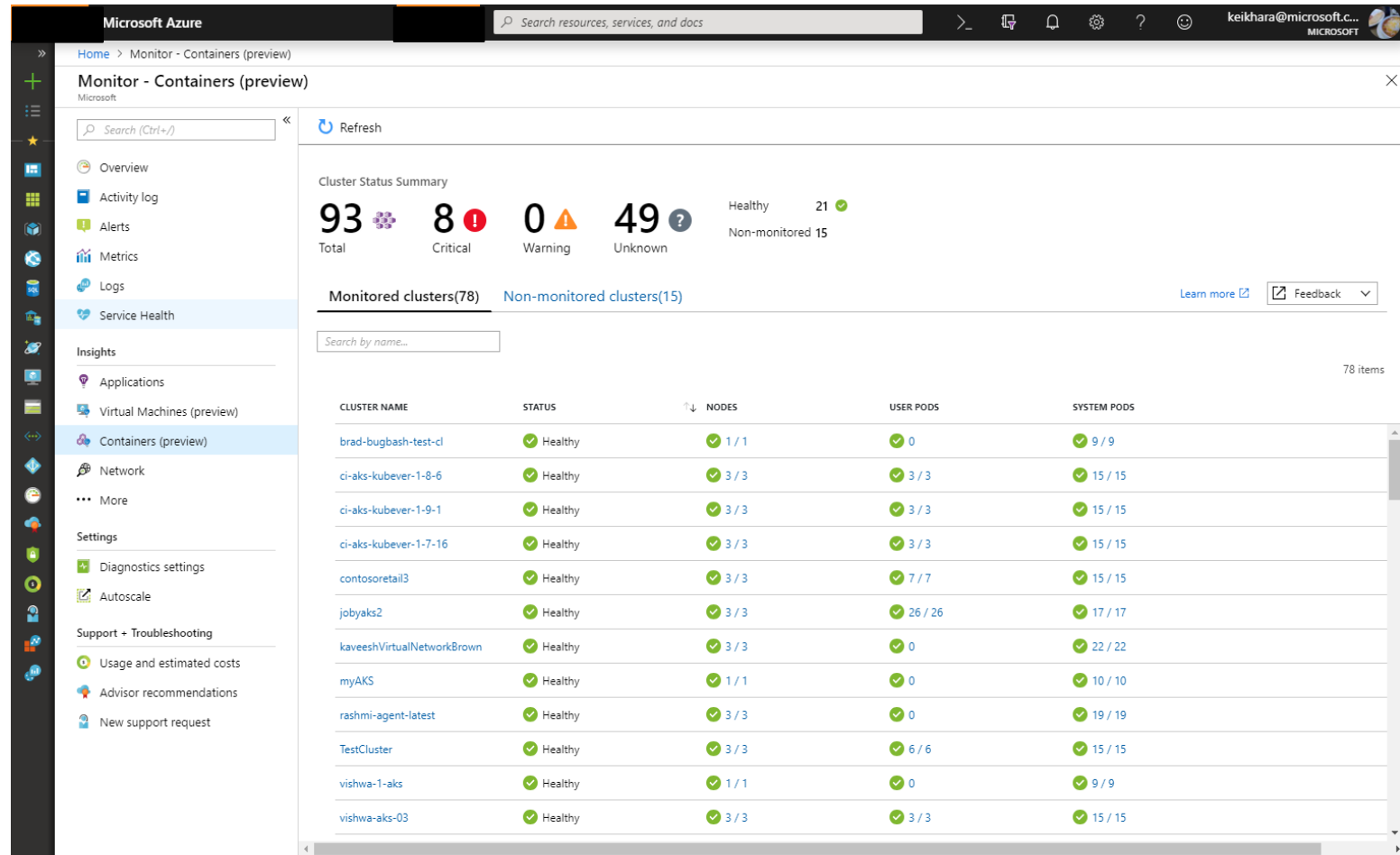
Computing 용량을 몇 초만에 증설할 수  
있음

관리 할 인프라가 없음(Serverless  
Architecture)



# Azure Monitor for Containers

- Cross subscription multi-cluster 모니터링 뷰 제공
- 노드, 컨트롤러 및 컨테이너에서 전반적인 상태 및 성능보기
- namespace, service, and node filter에 대한 다양한 모니터링 항목 제공
- 문제 해결을 위한 Kubernetes 이벤트 및 컨테이너 로그 분석
- 가상 노드가있는 AKS에 대한 요청시 컨테이너 모니터링
- Azure DevOps Project와 연계



# Azure Kubernetes momentum

10x

Azure Kubernetes Service 는 2018년  
6월에 GA 된 이후로 10배가 넘게  
성장해 왔습니다.

Trusted by thousands of customers





# Kubernetes Leadership @ Microsoft

Kubernetes 운영위원회 두 명

CNCF Technical Oversight Committee 두 명

CNCF Governing Board 두 명

Member on the Linux Foundation Governing Board

Kubernetes release team (1.13 and 1.14) 에 수많은  
Contributors 및 리딩

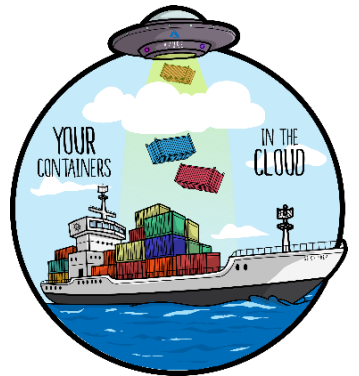


# Azure Kubernetes Service(1/4)

Azure 는 전 세계적으로 55 리전에서 사용 가능함

Azure는 모든 클라우드 서비스 제공 업체 중 가장 포괄적인 컴플라이언스 제품군을 제공함(70 이상)

Azure Kubernetes는 다른 클라우드 제공 업체보다 더 많은 지역에서 사용할 수 있음(17이상)



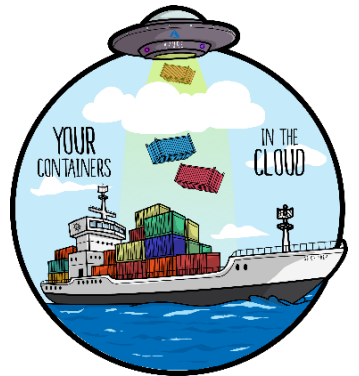
# Azure Kubernetes Service(2/4)

Azure 는 Control Plane 이 무료 (마스터 노드가 무료로 제공)

[Helm](#), [Draft](#), [Brigade](#), [CNAB](#), [Virtual Kubelet](#) 와 같은 프로젝트를 통해 Azure 는 커뮤니티  
Contribute 위주의 Kubernetes의 미래를 구체화

EKS는 오직 Control Plane만 관리함 (Worker 노드는 관리하지 않음)

AKS는 Kubernetes Control Plane과 더불어 Worker 노드의 통합된 프로비저닝 및 관리 기능 제공



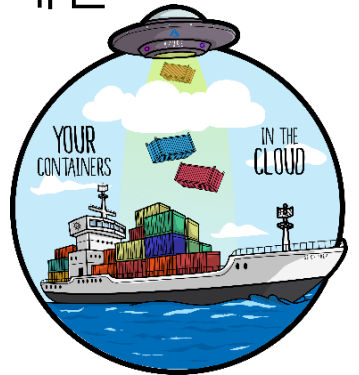
# Azure Kubernetes Service(3/4)

Azure DevOps Project를 사용하면 완벽한 기능의 CI / CD 파이프 라인을 생성하고 사용가능.

다양한 개발 언어를 지원하고 몇 번의 클릭만으로 완벽한 앱 분석 기능을 사용하여 Kubernetes에 응용 프로그램을 배포 할 수 있음

Azure Container Registry는 Docker 및 OCI (Open Container Initiative) 이미지 외에도 Helm 차트의 저장을 지원함

AKS는 Azure Active Directory와 통합되어 클러스터에서 컨테이너로 Kubernetes 리소스에 대한 세밀한 액세스 제어 기능을 제공함

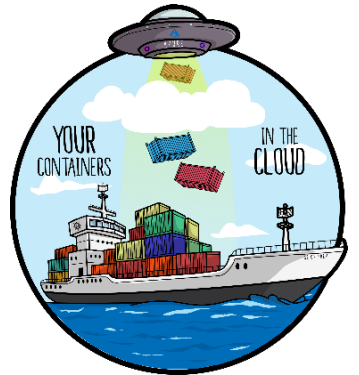


# Azure Kubernetes Service(4/4)

Azure는 Virtual Node를 통해 최초의 서버리스 Kubernetes를 제공. 인프라를 사용하지 않고도 컴퓨팅 용량을 수초 내에 탄력적으로 관리 할 수 있음

Microsoft는 Kubernetes 상에서 Azure Kubernetes Service를 운영하고 있음. 이러한 대규모 서비스를 실행함으로써 수집 된 정보들은 Kubernetes 커뮤니티에 피드백됨.(선순환)

Brendan Burns - Kubernetes의 공동 창립자 3 명 중 한 명이 현재 Microsoft에서 일하고 AKS 팀을 이끌고 있음!



# 4. Hands-on-Lab: AKS with Java (simple Spring App)

<http://aka.ms/20200226>

# 실습 결과

```
<properties>
  <docker.image.prefix>ianaksspringwinregistry.azurecr.io</docker.image.prefix>
  <jib-maven-plugin.version>2.0.0</jib-maven-plugin.version>
  <java.version>1.8</java.version>
</properties>
```

```
ian@~:$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
gs-spring-boot-docker-7c6c4cddc5-jvbm	1/1	Running	0	21m

```
ian@~:$ kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
gs-spring-boot-docker	LoadBalancer	10.0.193.196	104.43.18.218	80:30299/TCP	20m
kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP	30m

```
ian@~:$
```

104.43.18.218



Not secure | 104.43.18.218

Hello Docker World

# Clusters, Nodes, and Pods

## Node와 Pod의 개념

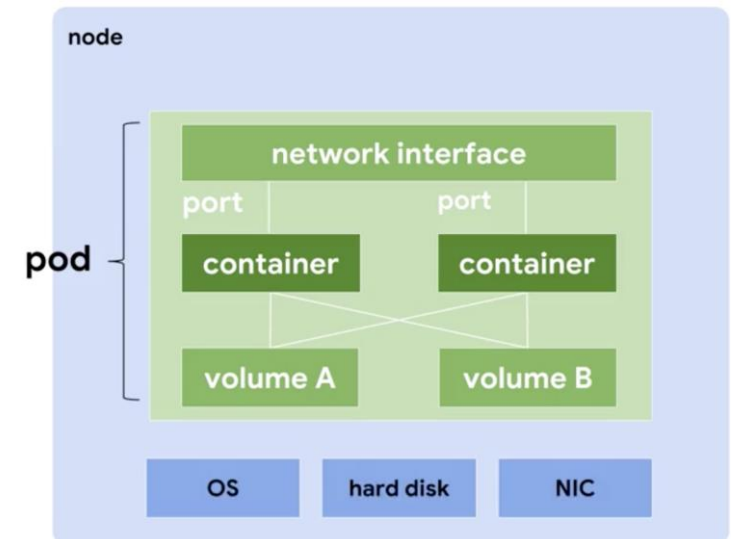
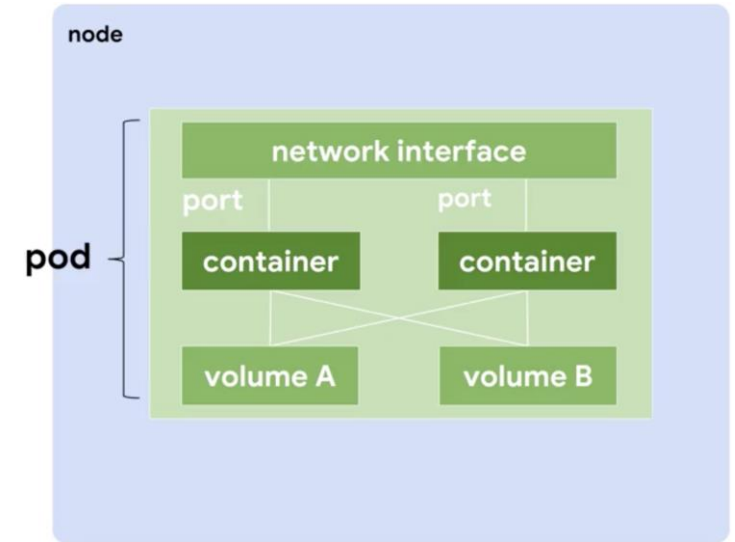
pod은 쿠버네티스에서의 하나의 작업의 단위

- pod는 일종의 가상머신과 유사하다고 볼 수 있음
- 동일한 하나의 네트워크와 여러 스토리지들을 공유하는 컨테이너들의 그룹
- pod의 네트워크, 컨테이너, 스토리지는 노드와는 분리되어 있음

pod 과는 분리되어 있는 node에는

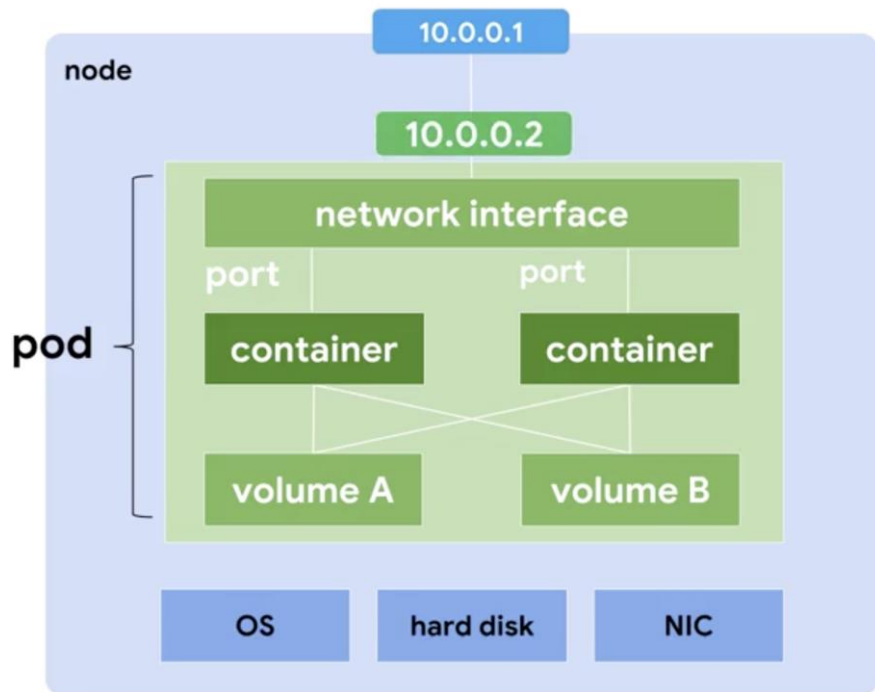
- OS 계층
- 하드웨어 하드디스크
- 하드웨어 NIC (네트워크 인터페이스 카드)

같은 자원들이 존재함



# Clusters, Nodes, and Pods

## Node와 Pod의 연결성



node 에는 외부 네트워크와 교류하기 위한 IP가 존재함

pod의 네트워크 인터페이스에는 node의 IP와 연결되기 위한 IP가 존재함

- 실질적으로 사용되는 컨테이너는 pod안에 있기 때문에 접근하기 위한 방법이 필요



# Clusters, Nodes, and Pods

## Pod을 정의 방법 및 생성되는 과정

YAML 포맷의 파일로 Pod을 정의할 수 있음

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
  - name: my-app
    image: my-app
  - name: nginx-ssl
    image: nginx
    ports:
    - containerPort: 80
    - containerPort: 443
```

버전 정

보

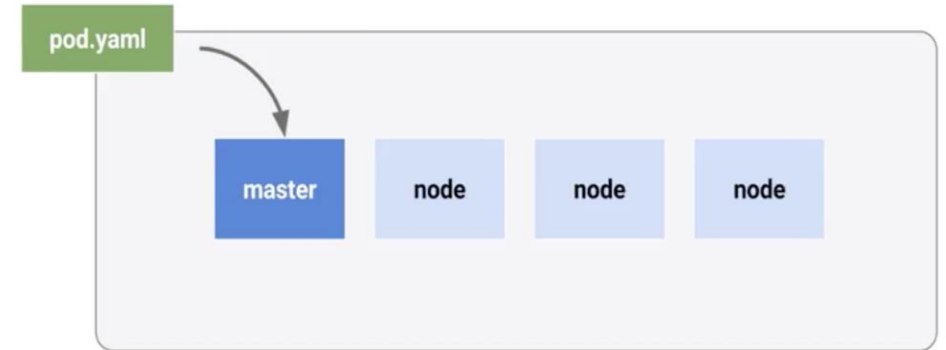
메타데이터

pod에 포함될

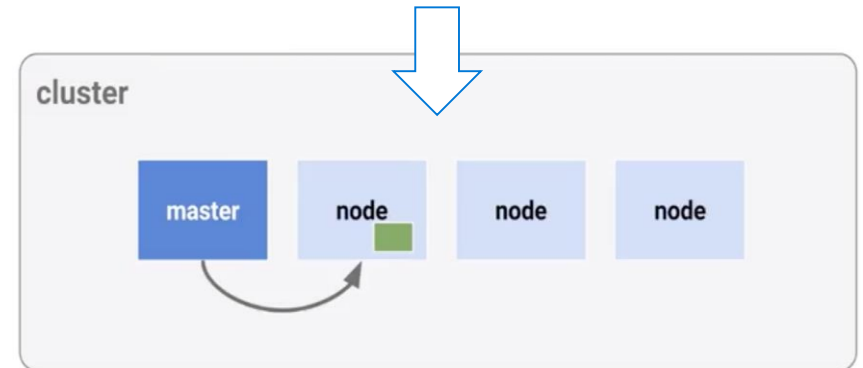
컨테이너 목록 명시

컨테이너에 접근할때

사용될 포트 명시



pod 이 정의된 YAML 파일을 마스터에 업로드



정의된 내용에 기반하여, pod을 생성함

# Clusters, Nodes, and Pods

## Node 또는 Pod의 장애 발생



쿠버네티스는 N개의 pod이 클러스트 내부에서 실행될 수 있음을 보장함

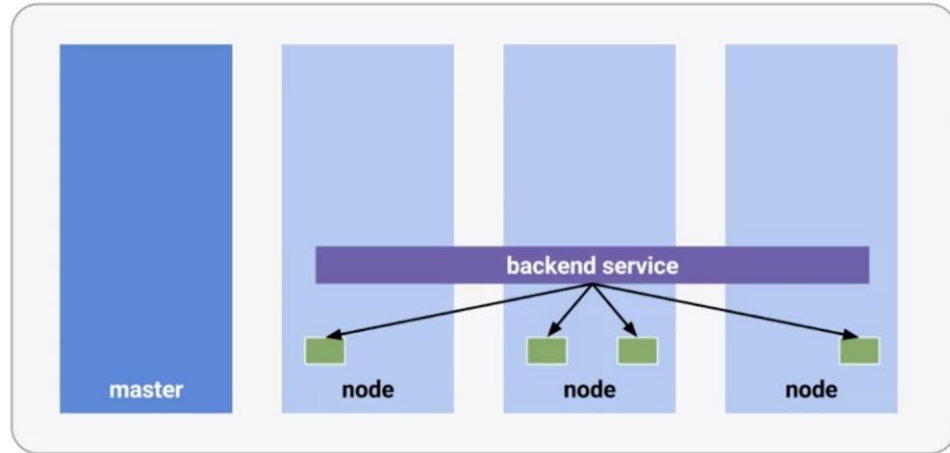
- node 개수 만큼의 N

특정 node 또는 pod에 장애가 발생시

- 다른 node에서 이를 대체하는 동일한 pod을 구동하여, 장애가 발생한 pod을 대체함

# Services, Labels, and Selectors

## Service의 개념



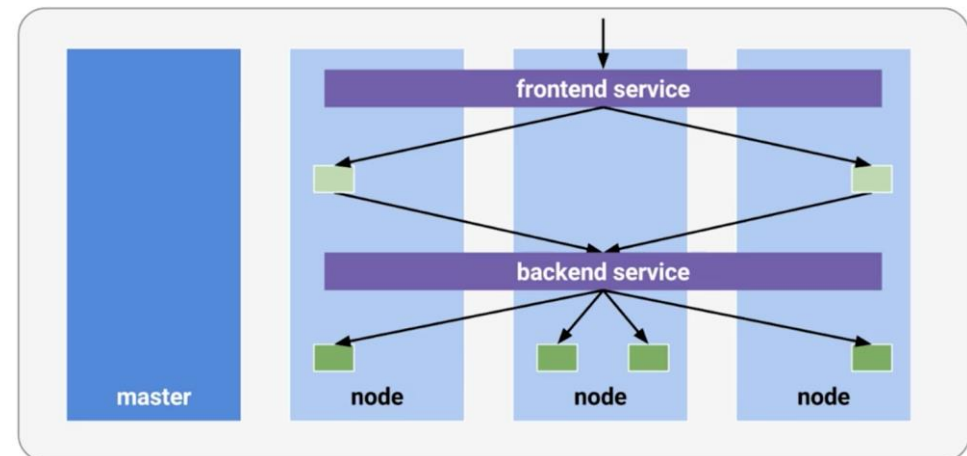
Service는 하나의 고정 IP를 복제된 Pod들 에게 할당해줌

Service는 여러 Pod들 사이에서,  
상호 소통을 가능하게 해 주는 방법을 제공함

서로다른 특징, 설정을 가진 여러개의 Service가 혼재 가능

일반적으로  
frontend service는 웹서버, 앱서버등 서버종류  
backend service는 데이터베이스 서버와 같은 것들임

각 Service들은 구글 클라우드 로드밸런서와 함께 구동됨  
⇒ 어떤 GCP API를 사용해야 하는지를 상세히 알려줌



# Services, Labels, and Selectors

## Service의 정의

마찬가지로, YAML 포맷의 파일로 정의

```
kind: Service
apiVersion: v1
metadata:
  name: web-frontend
spec:
  ports:
    - name: http
      port: 80
      targetPort: 80
      protocol: TCP
  selector:
    role: web
  type: LoadBalancer
```

Service를 위한  
자원임을 명시

Service 접근  
Port 스펙을 명시

로드밸런서의  
종류를 명시

# Services, Labels, and Selectors

## Label의 개념

Label은 메타데이터의 하나로, 모든 API 객체에 할당 가능함

객체마다의 정체를 표현하기 위한 것으로, 일종의 식별자라고 볼 수 있음

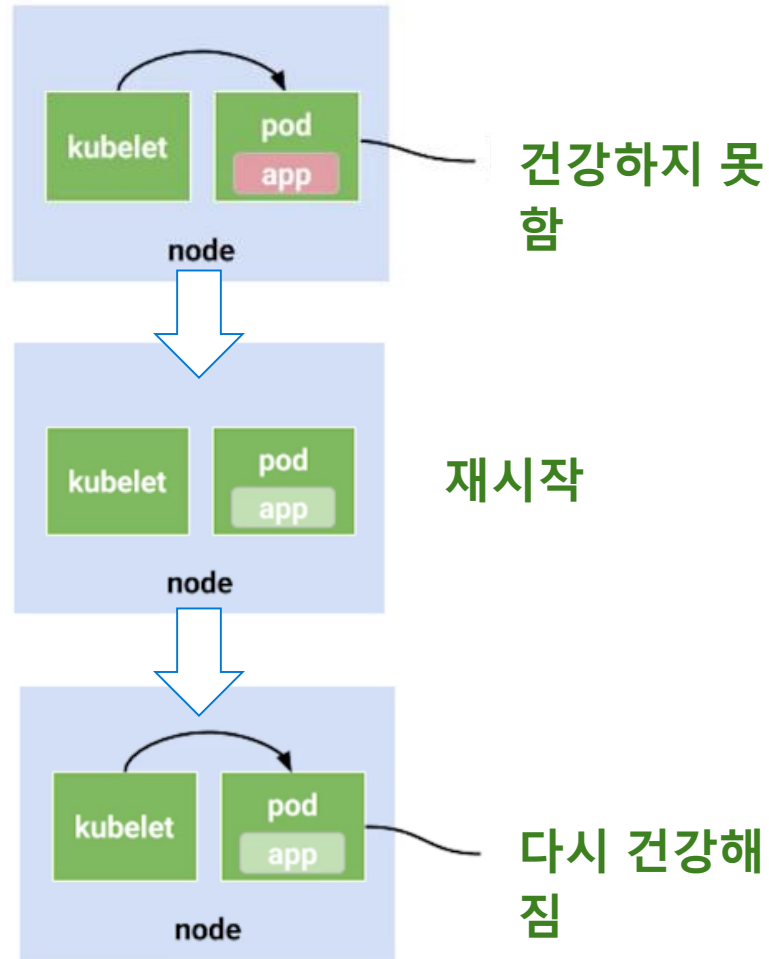
⇒ 확장해서 생각해 보면, pod들을 그룹화 하기 위한 메커니즘도 됨

⇒ 수 많은 pod들이 존재할 때, 어떤 레이블링 정보를 통한 검색이 용이함



# Services, Labels, and Selectors

## Kublet의 Pod 상태 관리



\* **Kublet (쿠블릿)**은  
**Node**마다 배치되는 에이전트 프로그램

Kublet은 Pod이 살아 있는지, 건강한 상태인지를 점검함

⇒ 점검했는데, 응답이 없으면, 건강하지 못한 상태라고 판단

⇒ 건강하지 못한 환경이라는 상태가 있음

⇒ a.k.a. Doctor in a can

⇒ 만약, 재실행 해야한다고 판단되면,

해당 Pod을 자동으로 재실행함

⇒ 재 실행 후, 괜찮다고 판단되면 건강한 상태로 돌아감

쿠버네티스에는 Pod의 상태를 나타내고,  
관리하는 방법이 있음

## Resource: Kubernetes & AKS 관련 정보

### Kubernetes 101 Docs

[aka.ms/ko-kr/LearnAKS](https://aka.ms/ko-kr/LearnAKS) (한글 문서)



### Best practices

[aka.ms/ko-kr/aks/bestpractices](https://aka.ms/ko-kr/aks/bestpractices)  
(한글 문서)



### Hear from experts

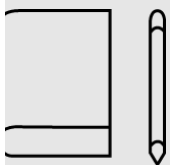
[aka.ms/k8s/lightboard](https://aka.ms/k8s/lightboard)

(Brendan Burns, Kubernetes  
Founder)



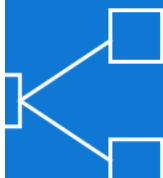
### Case studies

[aka.ms/aks/casestudy](https://aka.ms/aks/casestudy)



### Microservices architecture

[aka.ms/ko-kr/aks/microservices](https://aka.ms/ko-kr/aks/microservices)  
(한글 문서)



### Try for free

[aka.ms/ko-kr/aks/trial](https://aka.ms/ko-kr/aks/trial)

(12개월 체험 서비스)



Feedback on the roadmap? Tell us at <https://aka.ms/aks/feedback>

