

# 쿠버네티스

Kubernetes Administration

기초부터 실무까지  
한번에 ( 성장하는 )  
쿠버네티스



COMING SOON



kakaoenterprise

※ 강의의 내용은 쿠버네티스 공식 홈페이지(Kubernetes.io)의 문서를 기반으로 합니다.

※ 수강생의 교육상 이해를 돋기 위한 자료로, 외부 배포를 금합니다.



# 강사소개

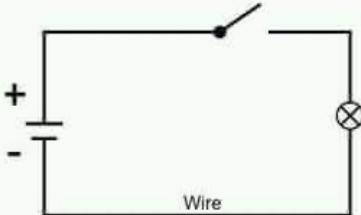


- 오성근
- 현 SK텔레콤 CoreEng팀 근무 및 사내 강사
  - (7년간) 5G/LTE Gateway, MEC - 구축, 운용, 상용망 기술적 문제 해결, 성능 개선 등
  - (2년차) 5G/LTE Gateway, MEC - 신기술 도입/기획, 설계, Capex 투자, 용량 관리 등
- 컴퓨터공학 학/석사 졸업, 과학기술관리학 박사과정
- IT Infra 관련 도서 5권 번역 출간/1권 감수 (매니징 쿠버네티스, 클라우드 팁옵스, OpenStack, AWS Lambda, VMware), 쿠버네티스를 활용한 클라우드 네이티브 데브옵스 감수
- CKA(Kubernetes Admin), RHCE in OpenStack, RHCVA, RHCE, CISSP, CISA, Oracle OCP, AWS SAA/MS Admin 등 취득

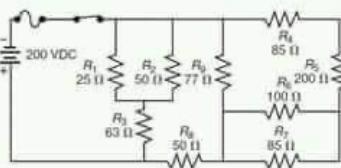


## 들어가기 전에...

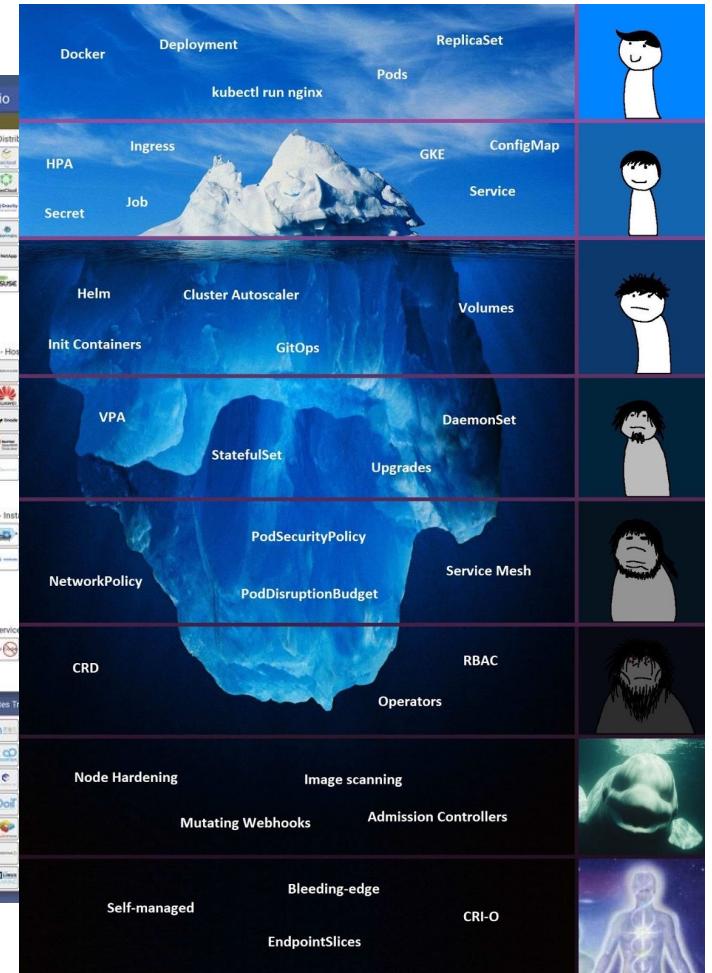
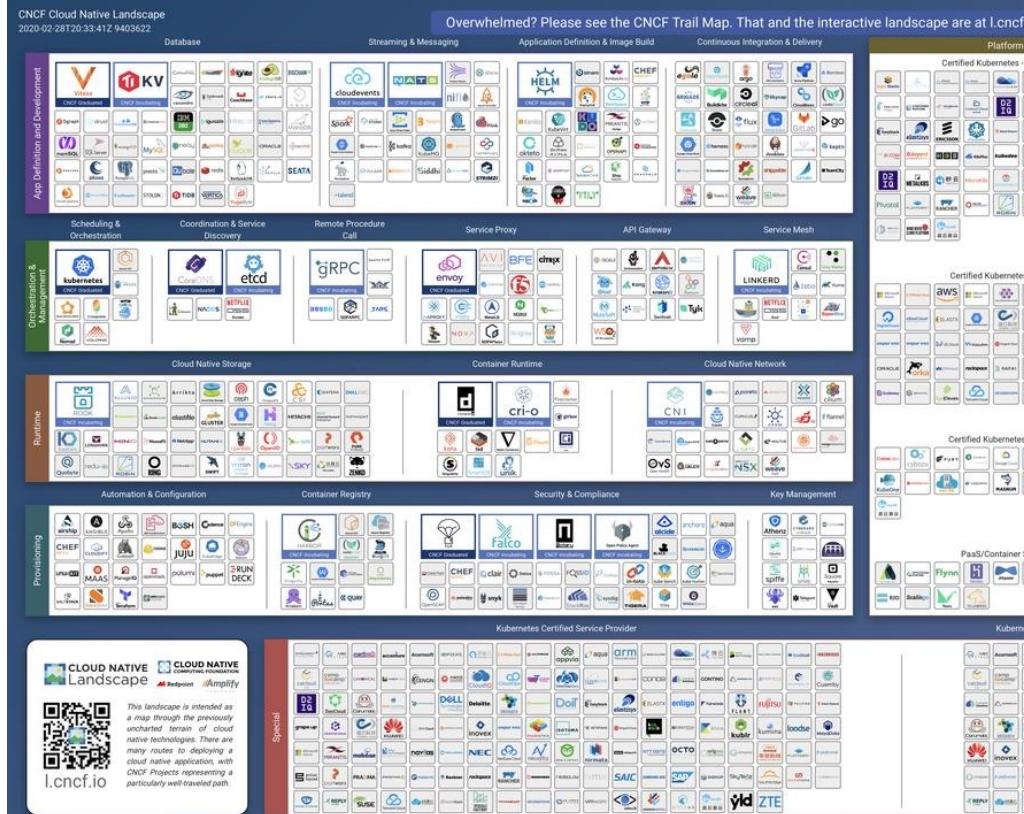
## **Class**



## Exam

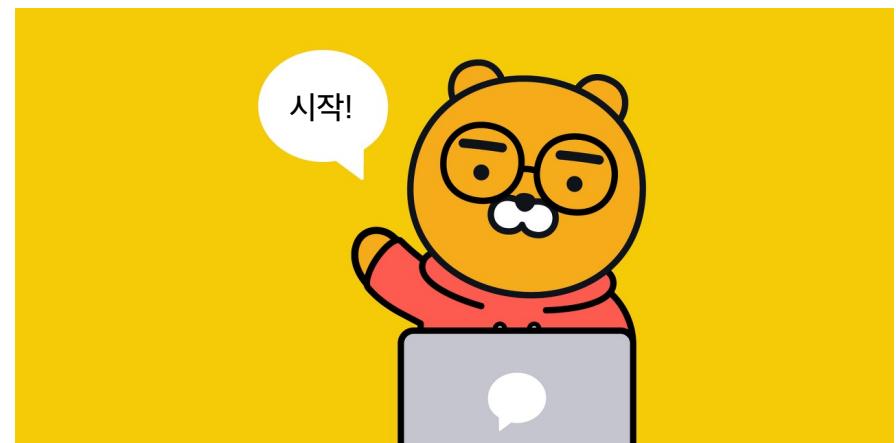


## Reality



# 교육 순서

- OT & PC세팅
    - Kakao i Cloud 활용한 실습준비
  - [배경지식] IT Infra & Kubernetes
1. [구조/개념] Kubernetes Architecture, [실습] 쿠버네티스 설치
  2. [접근/권한] APIs and Access
  3. [Compute] Managing State With Deployments, Scheduling
  4. [Storage] Volumes and data
  5. [Network] Services & Ingress
  6. [Operation] Logging & Troubleshooting, Backup, Upgrade



# OT&PC 세팅 (카카오 클라우드 활용)

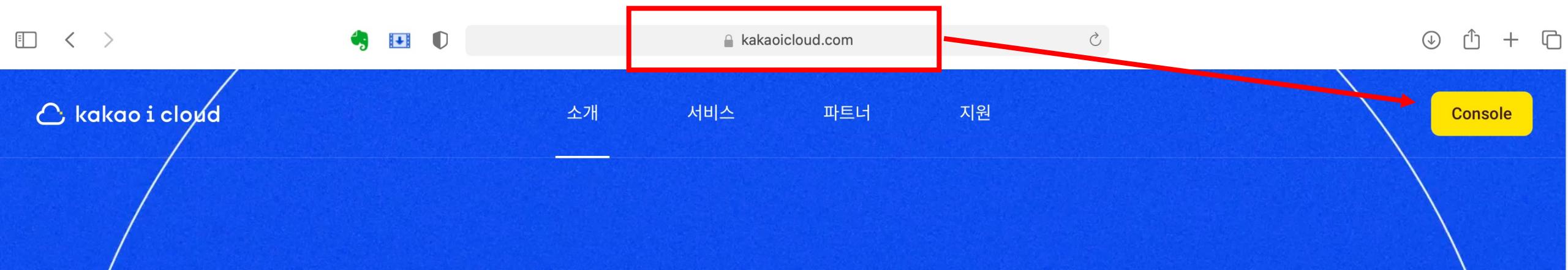
# OT - 실습 환경 안내

- 카카오 엔터프라이즈 계정 사용
  - 카카오 콘솔 접속, 카카오 내 쿠버네티스 클러스터 배포
  - 카카오 접속을 위한 PC 환경설정, 카카오 API 키/ 인증 설정
- 강의 진행방식
  - 이론
  - 실습 : 카카오 i 클라우드 활용
- 자료 안내
  - 일관된 설명을 위해 PPT 로만 가급적 단일화



# OT - 카카오 콘솔 접속 방법

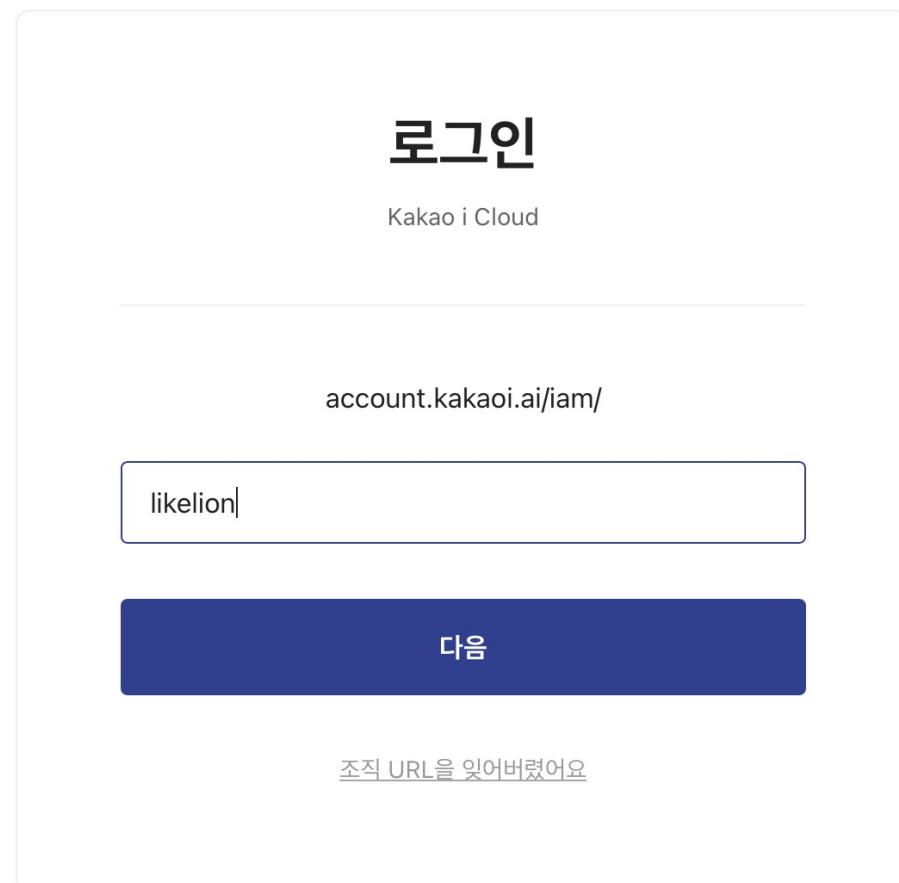
- Kakao i Cloud 접속 ([kakaoicloud.com](https://kakaoicloud.com)) 후 Console 선택



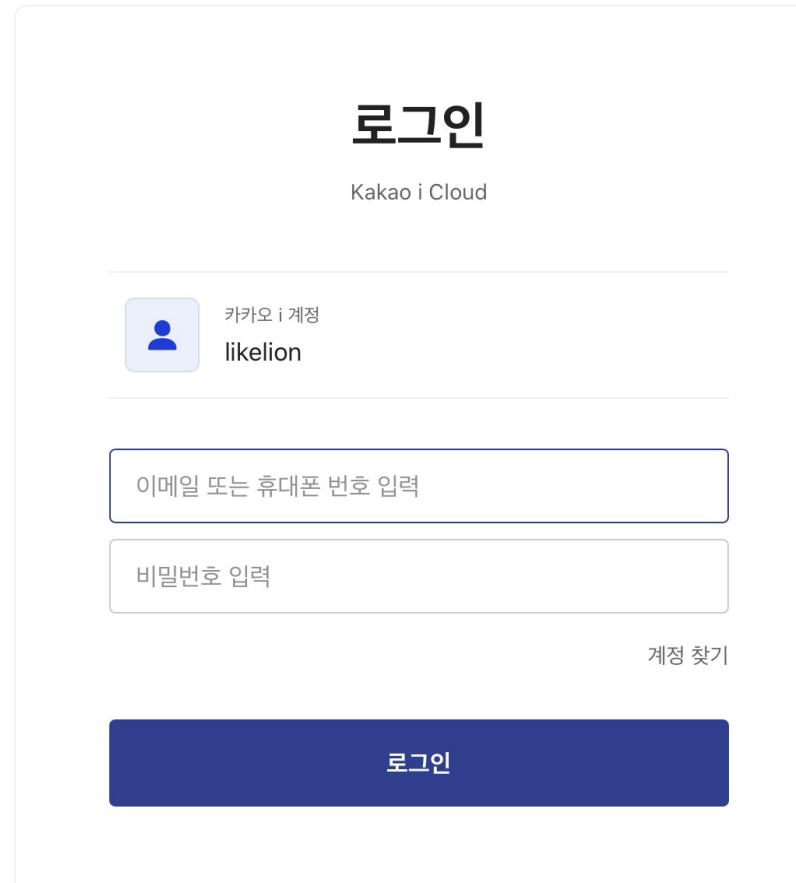
# OT - 카카오 콘솔 접속 방법

- 로그인에 필요한 조직URL은 likelion 입력
- 메일로 초대받아서 가입 한 ID/PW 입력

kakao i 계정



kakao i 계정



# OT - 카카오 콘솔 접속 방법

- 접속 완료

The screenshot shows the Kakao i cloud management console interface. At the top, there's a navigation bar with the 'kakao i cloud' logo and a dropdown menu set to 'likelion-icloud'. The left sidebar contains several sections: 'Openstandard IaaS' (Virtual Machine, VPC), 'Container Pack' (Kubernetes Engine), 'Storage' (Object Storage), and 'Management' (IAM). Below the sidebar, the main content area displays a banner with 'Kakao's Expertise' and '스마트한 운영 경험을 공유합니다.' (Sharing smart operating experience). The main dashboard is titled 'likelion-icloud 4' and shows four service statistics:

- Virtual Machine** (OpenStandard IaaS):
  - 인스턴스: 7 개
  - vCPU: 16 개
  - 메모리: 40 GB
- VPC** (OpenStandard IaaS):
  - 네트워크: 2 개
  - 시큐리티 그룹: 3 개
  - 공인 IP: —
- Kubernetes Engine** (Container Pack):
  - 클러스터: 2 개
  - 노드풀: 2 개
  - 노드: 4 개

At the bottom of the main content area, there are two additional boxes: 'Block Storage' (Storage) with 7 volumes (350 GiB total) and 'Object Storage' (Storage) which is described as having high integrity and durability for object storage.

사용자 가이드

# OT - 카카오 내 인스턴스 배포

- 인스턴스 만들기 클릭
- (1단계) 이미지 설정
  - CentOS7.9-cloudimg 선택
- (2단계) 인스턴스 설정
  - 인스턴스 개수 3개 선택
  - 인스턴스 이름은 실명 영어약자로 (ex:osk-master-01)
  - 인스턴스 설명에 한국어로 이름 추가 표현 (과제 통과 여부 시 확인용)
  - 인스턴스 타입 : a1.2c4m 선택
  - 키가 없는 경우 생성 후 개인 PC에 다운로드

2단계: 인스턴스 설정

인스턴스 개수  - +

인스턴스 이름  
osk-master-01  
osk-master-02  
osk-master-03

인스턴스 설명 (선택)  
오설근

인스턴스 타입

<input type="radio"/> a1.2c4m	vCPU: 2개	Memory: 4 GB
<input type="radio"/> a1.4c8m	vCPU: 4개	Memory: 8 GB
<input type="radio"/> a1.4c16m	vCPU: 4개	Memory: 16 GB
<input type="radio"/> a1.8c32m	vCPU: 8개	Memory: 32 GB
<input checked="" type="radio"/> a1.8c16m	vCPU: 8개	Memory: 16 GB
<input type="radio"/> a1.14c32m	vCPU: 14개	Memory: 32 GB

볼륨 타입 / 크기  ▼

# OT - 카카오 내 인스턴스 배포

- (3단계) 네트워크 보안 설정
  - 네트워크 및 서브넷은 likelion private으로 선택
  - 시큐리티 그룹도 default로 all inbound 허용으로 선택

3단계: 네트워크/보안 설정

네트워크 구성

관리 페이지 ↗

네트워크

likelion-private-01

서브넷

likelion-private-01 (172.30.4.0/22)

(사용 중인 IP : 5개 / 사용 가능한 IP : 1008개)

시큐리티 그룹

관리 페이지 ↗

적용 가능한 시큐리티 그룹



적용된 시큐리티 그룹

default



적용된 정책

Inbound

Outbound

시큐리티 그룹

프로토콜

패킷 출발지(Source)

포트 번호

default

all

@default

all

default

tcp

0.0.0.0/0

all

default

tcp

0.0.0.0/0

22

# OT - 카카오 내 인스턴스 배포

- (4단계) 확인 후 생성하기
- (1~4단계) 한번 더 반복해서,  
**2개짜리 worker node 추가 생성**  
ex) osk-worker-01

- 1 이미지 설정  
CentOS7.9-cloudimg
- 2 인스턴스 설정  
인스턴스 개수 : 3  
인스턴스 이름 : osk-master-01  
인스턴스 설명 (선택) : 오성근  
인스턴스 타입 : a1.8c1  
6m  
볼륨 타입 : SSD  
볼륨 크기 : 100  
키페이지 : kakao-osk
- 3 네트워크/보안 설정  
네트워크 : likelion-priv  
-ate-01  
서브넷 : likelion-private-01  
적용된 시큐리티 그룹 : 1개
- 4 검토

4단계: 검토

## 1단계: 이미지 설정

이미지 이름 CentOS7.9-cloudimg  
이미지 설명 release 21.05.25 (for Cloudimg)

## 2단계: 인스턴스 설정

인스턴스 개수 3  
인스턴스 이름 osk-master-01  
인스턴스 설명 오성근  
인스턴스 타입 a1.8c16m  
볼륨 타입 SSD  
볼륨 크기 100  
키페이지 kakao-osk

## 3단계: 네트워크/보안 설정

네트워크 likelion-private-01  
서브넷 likelion-private-01  
적용된 시큐리티 그룹 default

# OT - 카카오 내 인스턴스 배포 완료 확인

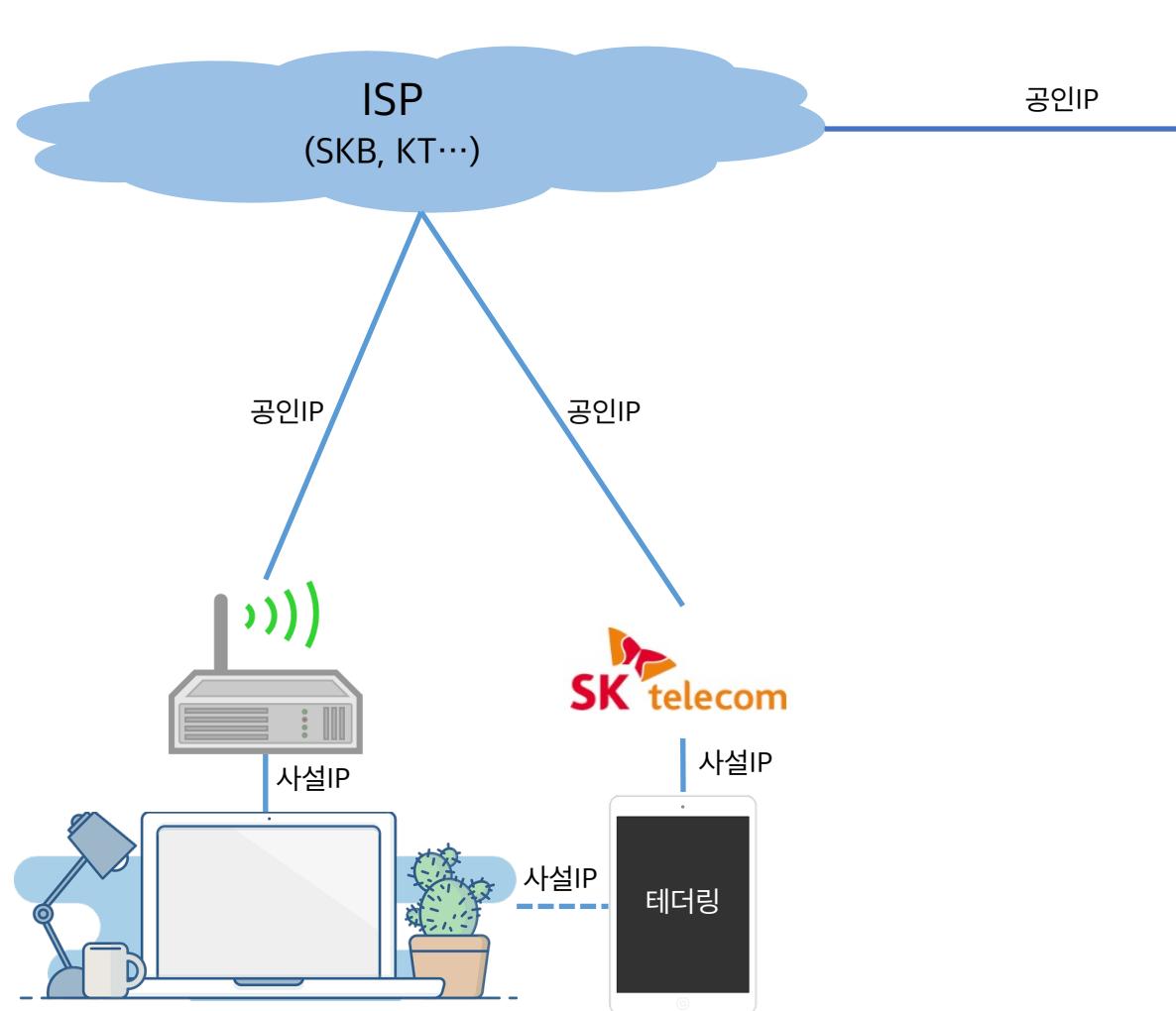
- 172 사설 IP로 배포됨을 확인

## 인스턴스 목록

인스턴스 필터		인스턴스 ID		상태	타입	가동 시간	사설 DNS	사설 IP	공인 IP	항목 새로고침
<input type="checkbox"/>	인스턴스 이름									
<input type="checkbox"/>	osk-worker-02	b3216624-ebc0-43f0-b65c-ed923d2f...	<span>● Active</span>	a1.8c16m	0분	-	-	172.30.6.73	-	<span>⋮</span>
<input type="checkbox"/>	osk-worker-01	73ae8d52-0f64-4bc9-867b-960903e...	<span>● Active</span>	a1.8c16m	0분	-	-	172.30.6.82	-	<span>⋮</span>
<input type="checkbox"/>	osk-master-02	101202c0-1da5-48b4-8c65-2cb90dd...	<span>● Active</span>	a1.8c16m	5분	-	-	172.30.7.170	-	<span>⋮</span>
<input type="checkbox"/>	osk-master-03	feb8f674-69cd-4a24-8b93-e5b4271c...	<span>● Active</span>	a1.8c16m	5분	-	-	172.30.6.161	-	<span>⋮</span>
<input type="checkbox"/>	osk-master-01	458f0e7f-66c3-4a73-a2d9-669ed2b6...	<span>● Active</span>	a1.8c16m	5분	-	-	172.30.4.190	-	<span>⋮</span>

# OT - 실습 환경 구성도

kakaoenterprise

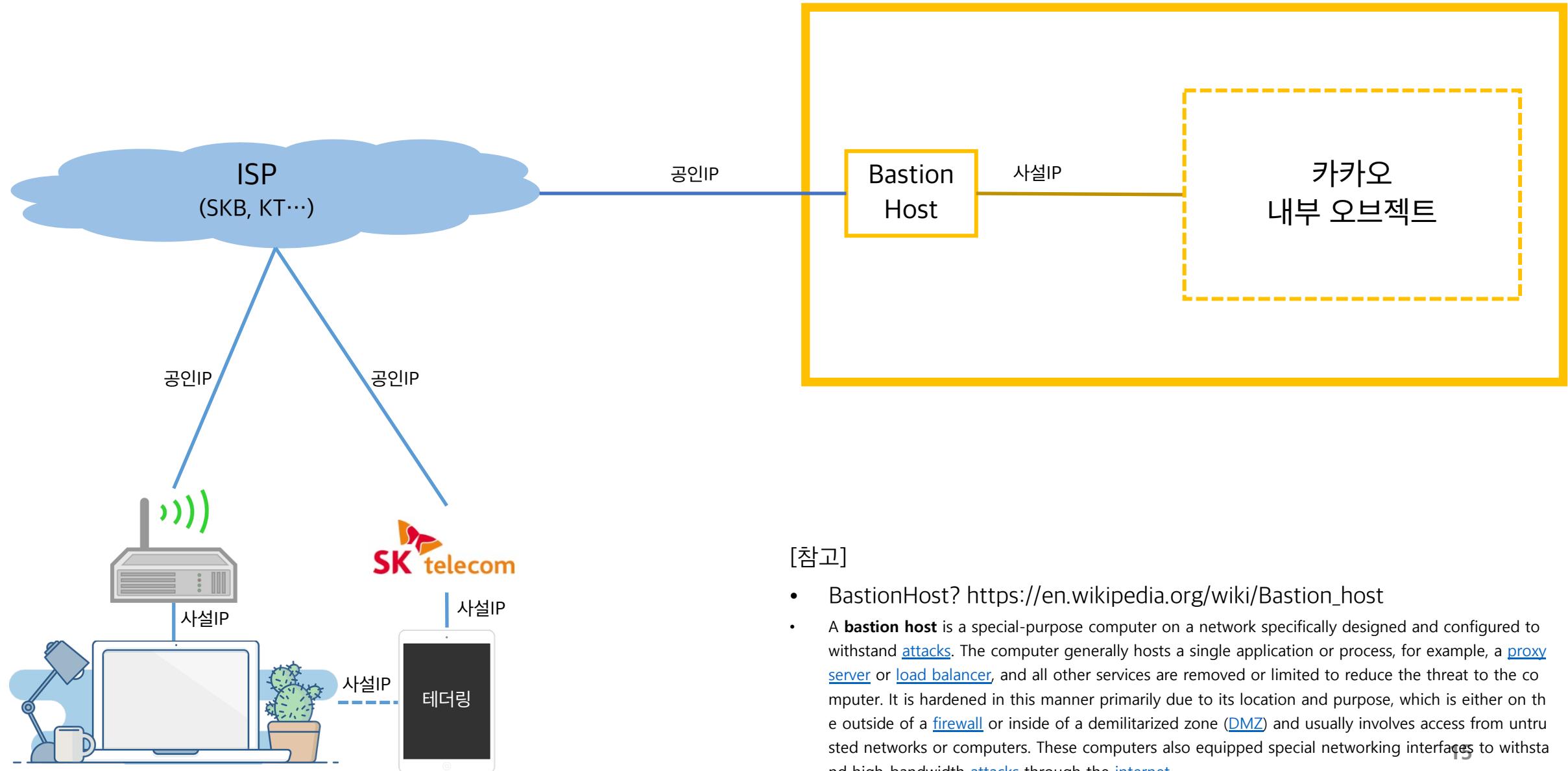


[참고]

- 사설IP?
  - 클래스 A 규모 : 10.0.0.0 ~ 10.255.255.255 (10.0.0.0/8)
  - 클래스 B 규모 : 172.16.0.0 ~ 172.31.255.255 (172.16.0.0/12)
  - 클래스 C 규모 : 192.168.0.0 ~ 192.168.255.255 (192.168.0.0/16)

# OT - 실습 환경 구성도

kakaoenterprise



# OT - 실습 환경 구성도

kakaoenterprise



- 교육생 PC가 카카오의 사설 대역을 받을 수 있게 설정 필요
    - 1) PC에 VPN 클라이언트 다운받아서 설치
    - 2) 전달받은 .ovpn 파일을 활용하여 연결

# OT - PC 네트워크 설정

- 1) PC에 VPN 클라이언트 다운받아서 설치
  - Mac OS 용: Tunnelblick (<https://tunnelblick.net/downloads.html>에서 stable버전 다운로드)
  - Window OS 용: OpenVPN 클라이언트 (<https://openvpn.net/community-downloads/>에서 다운로드)
- 2) 전달받은 .ovpn, ta.key를 같은 경로에 놓고, 파일을 활용하여 연결

[맥북]

We need translators

Tunnelblick free software for OpenVPN on macOS

Home Downloads Support Documents Issues Source Contribute Contact

On This Page Release Downloads Verifying Downloads User Contributions Download Integrity Downloading and Installing on macOS Mojave and Higher

### Release Downloads

To be notified of new releases, use Tunnelblick's built-in update mechanism or subscribe to the [Tunnelblick Announce Mailing List](#).

Beta versions are suitable for many users. See [Stable vs. Beta](#) for details.

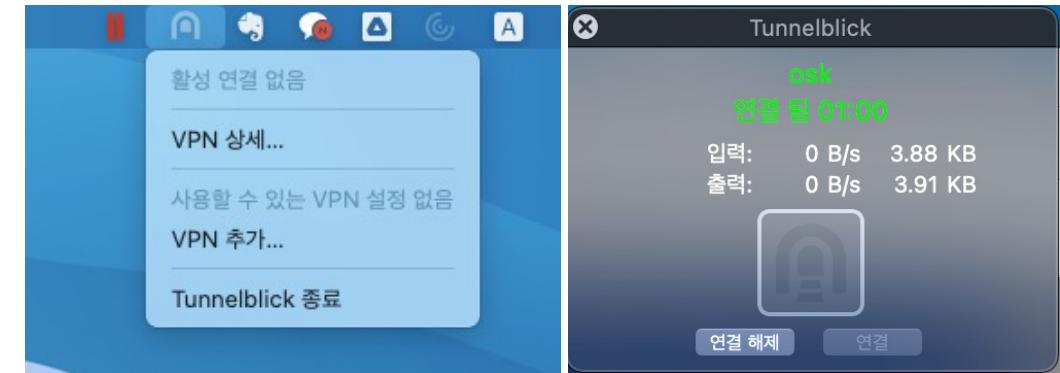
As a [Free Software](#) project, Tunnelblick puts its users first. There are no ads, no affiliate marketers, no tracking — we don't even address or other information. We just supply open technology for fast, easy, private, and secure control of VPNs.

**Beta** **PREVIEW: For testing only. Runs natively on M1 and Intel Macs.**

**Tunnelblick 3.8.6beta05** (build 5706, macOS 10.10+, (mixed Intel-64, M1), notarized) released 2021-06-26 [Release Notes](#)  
SHA1: ddf3db886a95de512588beb0fd42c77b087056d8 MD5: 985e8051353d1b2f95eb6d77b2c482b  
SHA256: dcd8b47cbee91846f09c02842cd85c52c12a556722be090bdc67c13cd5accde1  
GnuPG v2 signature

**Beta** **Tunnelblick 3.8.6beta03** (build 5700, macOS 10.10+, (mixed Intel-64, M1), notarized) released 2021-04-22 [Release Notes](#)  
SHA1: 24787eb0a1b3f0692cba8f4d27ed2b00a64867a6 MD5: c77747cceedd8c14794a893eed15c2c  
SHA256: de0f6d24cbc45650c1789f6963f7b454099e6cd9a00ec067178977614415d256  
GnuPG v2 signature

**Stable** **Tunnelblick 3.8.5a** (build 5671, macOS 10.10+, (mixed Intel-64, M1), notarized) released 2021-04-21 [Release Notes](#)  
SHA1: 9e6bb2717f0924fdf2fed306fe40837170b2d1ba MD5: ecaad1485e2691343ecb2c6ade161cbd  
SHA256: 88f8cd776bf237a8b1c72531cf44bf7440e3bb4f94b16a77747351014c2de3a7  
GnuPG v2 signature



▲ 상단 새로 생긴 터널에 ovpn  
파일을 Drag&Drop

▲ 연결 완료

# OT - PC 네트워크 설정

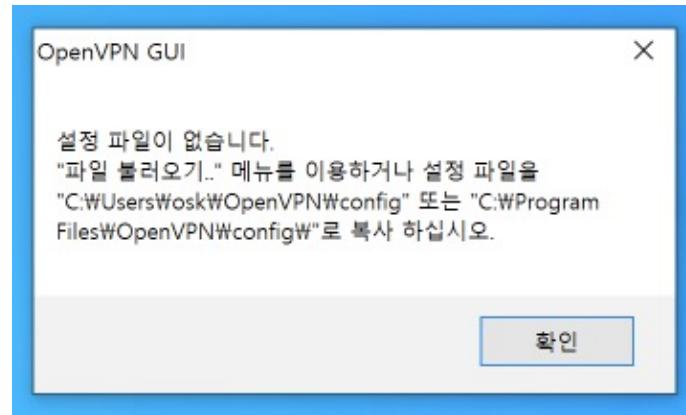
- 1) PC에 VPN 클라이언트 다운받아서 설치
  - Mac OS 용: Tunnelblick (<https://tunnelblick.net/downloads.html> 에서 stable버전 다운로드)
  - Window OS 용: OpenVPN 클라이언트 (<https://openvpn.net/community-downloads/> 에서 다운로드)
- 2) 전달받은 .ovpn, ta.key를 같은 경로에 놓고, 파일을 활용하여 연결

[윈도우]

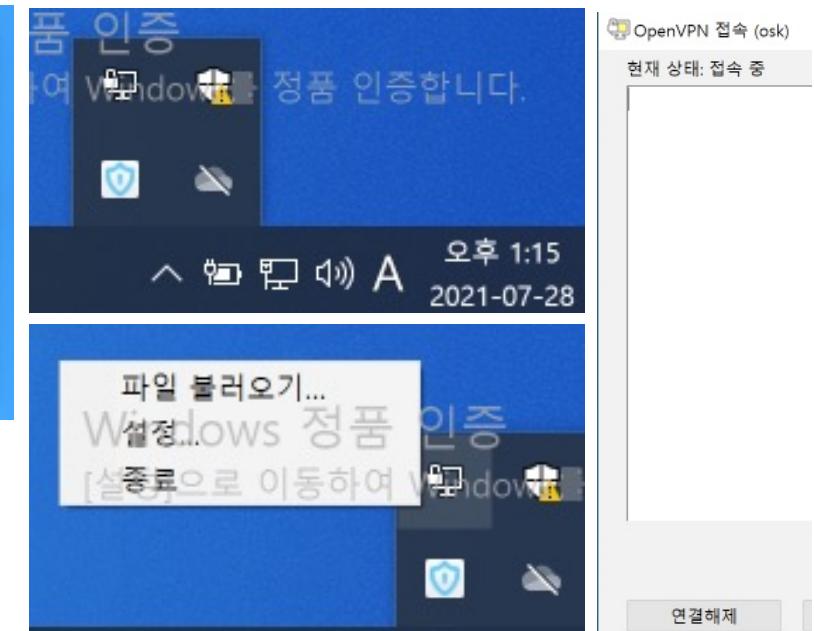
The screenshot shows the 'Community Downloads' section of the OpenVPN website. It lists several download links for different file formats:

- Source tarball (gzip): openvpn-2.5.3.tar.gz
- Source tarball (xz): openvpn-2.5.3.tar.xz
- Source zip: openvpn-2.5.3.zip
- Windows 32-bit MSI installer: OpenVPN-2.5.3-I601-x86.msi
- Windows 64-bit MSI installer: OpenVPN-2.5.3-I601-amd64.msi

The 'Windows 64-bit MSI installer' link is highlighted with a yellow box.

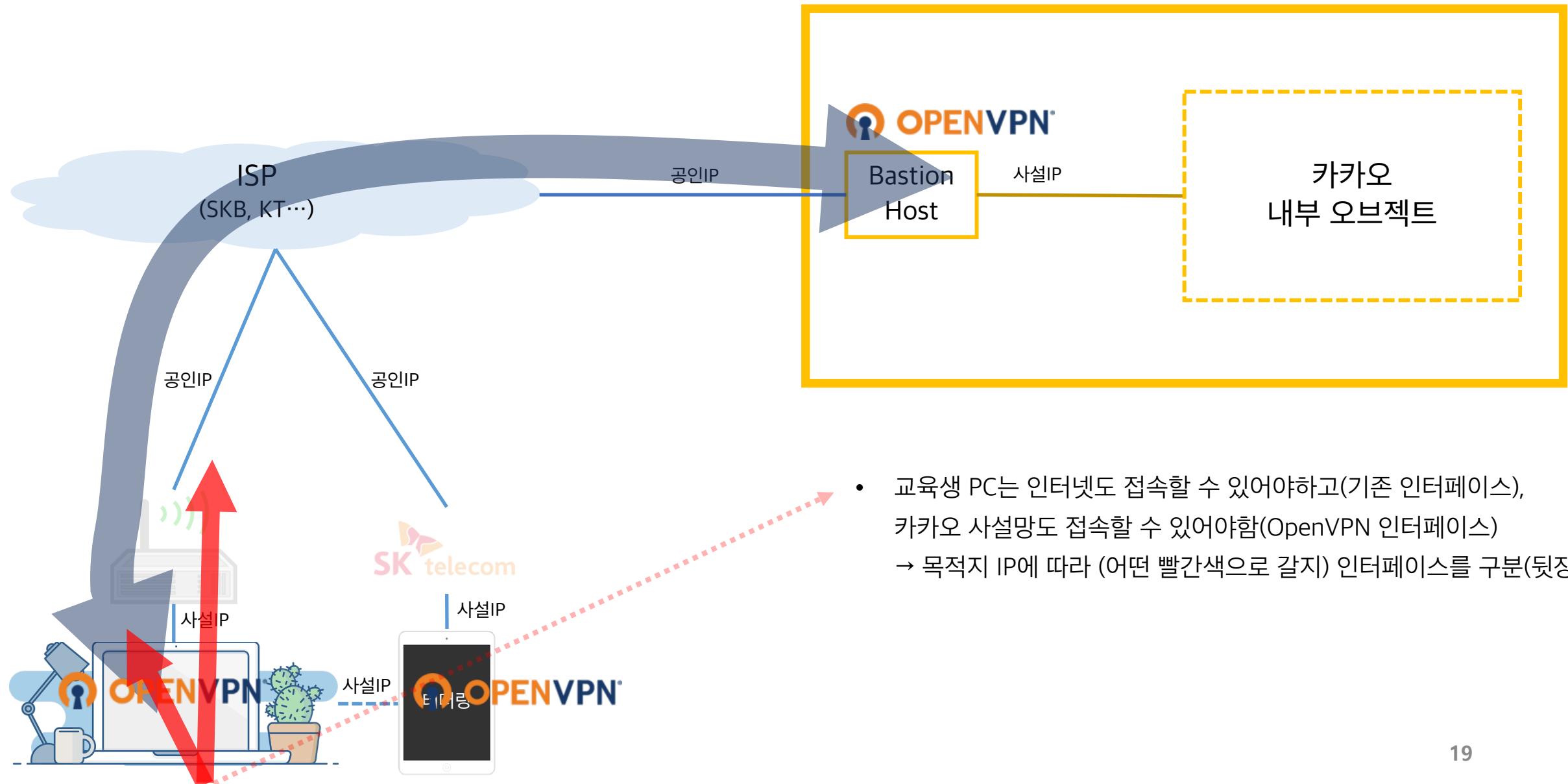


- ▲ 설치 직후 나오는 팝업
- ▶ OpenVPN 실행 후 파일 불러오기
- ▶▶ 연결완료



# OT - PC 네트워크 설정

kakaoenterprise



# OT - PC 네트워크 설정

- Mac OS

- Public Subnet에 대한 라우팅 룰 적용

```
$ sudo route add -net 172.30.0.0 -netmask 255.255.252.0 10.8.0.1
```

- Private Subnet에 대한 라우팅 룰 적용

```
$ sudo route add -net 172.30.4.0 -netmask 255.255.252.0 10.8.0.1
```

- Kubernetes API 서버 엔드포인트에 대한 라우팅 룰 적용

```
$ sudo route add -net <API 서버 엔드포인트 주소> -netmask 255.255.255.255 10.8.0.1
```

- 적용된 룰 확인

```
$ netstat -nr | grep 10.8.
```

- 윈도우(-p옵션)과는 다르게 맥북은 재부팅마다 반복해야하며, 귀찮다면 별도 스크립트 만들어야 함.

```
osk@osk-MacBook-Pro ~ % sudo route add -net 172.30.0.0 -netmask 255.255.252.0 10.8.0.1
```

```
Password:
```

```
add net 172.30.0.0: gateway 10.8.0.1
```

```
osk@osk-MacBook-Pro ~ % sudo route add -net 172.30.4.0 -netmask 255.255.252.0 10.8.0.1
```

```
add net 172.30.4.0: gateway 10.8.0.1
```

```
osk@osk-MacBook-Pro ~ % sudo route add -net 10.183.67.34 -netmask 255.255.255.255 10.8.0.1
```

```
add net 10.183.67.34: gateway 10.8.0.1
```

# OT - PC 네트워크 설정

- Windows OS

- Public Subnet에 대한 라우팅 룰 적용

```
$ route -p ADD 172.30.0.0 MASK 255.255.252.0 10.8.0.1
```

- Private Subnet에 대한 라우팅 룰 적용

```
$ route -p ADD 172.30.4.0 MASK 255.255.252.0 10.8.0.1
```

- Kubernetes API 서버 엔드포인트에 대한 라우팅 룰 적용

```
$ route -p ADD <API 서버 엔드포인트 주소> MASK 255.255.255.255 10.8.0.1
```

- 적용된 룰 확인

```
$ route print 10.8.
```

# OT - 설정 완료.

- 정상적으로 접속 가능한지 확인

```
osk@osk-MacBook-Pro lkln-kakao % ssh -i kakao-osk.pem centos@172.30.0.176
The authenticity of host '172.30.0.176 (172.30.0.176)' can't be established.
ECDSA key fingerprint is SHA256:TVUqUxQvFR4Cte3C9abeXecnDHic9qnZWfZJHdCc3Vo.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.30.0.176' (ECDSA) to the list of known hosts.
Last login: Tue May 25 06:18:33 2021 from 172.25.59.92
[centos@osk-master-01 ~]$
[centos@osk-master-01 ~]$
```

- 강사들이 개별 평가할 수 있도록 강사의 Public Key를 각자 VM에 세팅 (접속 불가능하면 과제 평가 불가)

```
# sudo yum install -y wget
# wget http://172.30.5.154/instructor.pub
# cat instructor.pub >> ~/.ssh/authorized_keys
```

# 쉬어가는 페이지



## [배경지식] IT Infra & Kubernetes

# 기본 용어

**제품 광고아님**

상세정보 리뷰 1 문의 6

**세부사항**

운영체계	Windows 10 Pro
프로세서 / 칩셋	Intel® Celeron® Processor 6305 (1.8 GHz, 4 MB L3 Cache)
메모리	4 GB DDR4 Memory (4 GB x 1) 2 SODIMM
저장장치	128 GB NVMe SSD 총 HDD 슬롯 : 1 총 SSD 슬롯 : 1
ODD	없음
디스플레이	39.6 cm (15.6 inch) FHD LED Display (1920 x 1080), Anti-Glare
그래픽	Intel® UHD Graphics
멀티미디어	스테레오 스피커 (1.5 W x 2) SoundAlive™ 내장 디지털마이크 웹 카메라
네트워크	Bluetooth v5.1 Wi-Fi 6 (Gig+), 802.11 ax 2x2 Gigabit Ethernet [10/100/1000]
포트	1 HDMI 1 USB-C® 1 USB3.0, 1 USB2.0 MicroSD Multi-media Card Reader 1 헤드폰 출력/마이크 입력 콤보 1 RJ45 (랜) 1 DC-in
입력장치	래티스 키보드 / 숫자키 클릭패드
보안기능	시큐리티 슬롯 TPM
색상	퓨어 화이트
전원	43 Wh 40 W 어댑터
* 어댑터 용량은 사양에 따라 상이합니다.	
크기 (가로 x 세로 x 높이)	359.2 x 241.3 x 18.8 mm
무게	1.81 kg

**제품 광고아님**

상세정보 리뷰 문의

**H/W 제품사양**

주요사항	600Mbps급 11n 4LAN 포트 유무선공유기 / 4dbi 4ANT 모바일 설치도우미 & IUX 제공
CPU	Realtek RTL8196D (620MHz)
WAN Interface	1 x 10/100Mbps WAN - 케이블 자동 감지
LAN Interface	4 x 10/100Mbps PC Port - 케이블 자동 감지
Wireless Interface	802.11 b/g/n
Status LED	CPU, Wireless, WAN, LAN X 4
DRAM	32 Mbytes
FLASH	4 Mbytes
본체 크기 / 무게	175 x 110 x 30 mm (안테나 크기 제외) / 236g
파키지 크기 / 무게	227 x 146 x 88 mm / 514g (제품 포함, 본체 및 구성품 전체의 무게 포함)
최대소비전력	5.4W
동작온도	설비 0 ~ 50도
전원	외장형 DC 어댑터 (최저소비효율기준 만족제품)
색상	White

**S/W 제품사양**

NAT	SNAT, DNAT, IP Masquerade
Protocols	HTTP, DHCP, PPPoE
Application Protocol	H233, MSN, BattleNet ...
QoS	Rate limiting, Rate Guarantee



**무선사항**

무선	IEEE 802.11b, IEEE 802.11g, IEEE 802.11n
주파수	2.4GHz
수신감도	-55dBm at 600Mbps -65dBm at 54M -97dBm at 11M
변조방식	OFDM, CCK, BPSK, QPSK
보안설정	WPA-PSK WPA2-PSK WPA2PSK/WPA-PSK WPS (외부버튼지원) 802.1X



**기타제품사양**

KCC 인증번호	R-CMM-EFM-IPTIMEN804R
출시년월	2018.06
제조국	중국
제조/수입/AS책임자	(주)이에프엠네트웍스
품질보증기준	무상 A3 2년
AS전화번호	1544-8695

# 기본 용어

- 데이터센터 / 랙 / 서버 / 네트워크 / 스토리지



# 기본 용어

- <https://www.google.com/about/datacenters/gallery/>



# 기본 용어

- 재미 삼아 보는,  
컴퓨터구조/OS 관련 용어 비유

익명 09/15 10:31  
컴잘알있니  
나노트북 새로사려는데 진짜 외계어가 따로없다  
ㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋㅋ i5..? 숫자높을수록  
좋은거같은디 이건 뭐야..? 글고 렌이 클수록 좋다던디  
맞니.. 진짜 1도모르겠음;  
유튜브좀보고 포토샵이랑 메이플 자주할건디 어느정도  
성능으로사야해...?? 게이밍노트북은 과한거같은뎁..  
44 38 142

익명1 09/15 10:33  
코어의 수 = 니가 부릴 노예의 수  
48

익명1 09/15 10:33  
쓰레드 = 니가부릴 노예의 손 갯수(

익명1 09/15 10:33  
오버클럭 = 노예가 발까지 써서 옥수수 수확

익명1 09/15 10:34  
캐쉬 메모리 = 노예가 짊어지고 있는 백팩

익명1 09/15 10:34  
램 = 노예가 준비한 리어카

익명1 09/15 10:35  
하드 디스크 = 수확한 옥수수의 저장창고

익명1 09/15 10:35  
i3 = 노예가 3마리( 실제로 3코어 아님 강 이해를  
돕기 위해서 숫자 통일하는거 )  
09/15 10:35

익명1 09/15 10:35  
i5 = 노예가 5마리  
09/15 10:36

익명1 09/15 10:37  
i7 = 노예가 7마리  
09/15 10:44

익명1 09/15 10:46  
파워 = 노예에게 주는 월급 = 일많이하는놈한텐  
많이주고, 적게하는놈한텐 적게줌 = 딱 맞춰주면  
가성비 좋긴하겠지만 조금 여유있게 주면 서로 좋음=  
부족하게 주면 건강 악해져서 일개못함

익명1 09/15 10:47  
ssd = 존나 최신식 좋은 저장창고

익명1 09/15 10:47  
NVme ssd = 존나 좋은 창고 존나 더 업그레이드 함  
, 옥수수 보관할때도 존나 빠르게 가능하고, 꺼낼때도  
ㅈㄴ 빠르게 끼내옴, 안에 거중기 같은게 노예들  
도와줌  
09/15 10:47

익명18 09/15 14:03  
배워갑니다 교수님

익명19 09/15 14:26  
개지린다 와

익명20 09/15 15:03  
와우 :: 비유 쓰스트ㅊ

익명1 09/15 10:39  
?????????? 교수님 왜여기계세요  
09/15 10:39

익명(글쓴이) 09/15 10:39  
와우 그스그스  
09/15 10:39

익명1 09/15 10:39  
그래픽카드 = gpu = 옥수수 말고 다른거 시킬 노예 =  
옥수수 딸수 있긴한데 일 험같이 못함 근데 감자  
캐는건 존내 잘함  
09/15 10:41 2

# 기본 용어

- 재미 삼아 보는,  
컴퓨터구조/OS 관련 용어 비유



VISUAL DIVE

# 컴알못을 위한 노예 비유법

**코어의 수**  
네가 부릴 노예의 수

**쓰레드**  
네가 부릴 노예의 손 것수

**오버클럭**  
노예가 밥까지 써서  
온수수 수확

---

**캐시 메모리**  
노예가 짊어지고 있는 백팩

**램**  
노예가 준비한 리어카

**하드 디스크**  
수확한 온수수 저장창고

---

**i3, i5, i7**  
노예가 3마리, 5마리, 7마리  
(실제 코어 수와 다름.  
이해를 돋기 위해 숫자 통일한 것)

**i7 10900**  
7마리 10년도 생 노예의 손이 900개  
(7마리의 10년에 태어난 노예들이  
손이 900개라 수확도 빠름)

**램 다다익선 이유**  
온수수를 많이 따도  
리어카가 작으면 한 번에  
창고로 못 가져감

---

**그래픽 카드  
= gpu**  
온수수 말고 감자  
캐는 거 시킬 노예

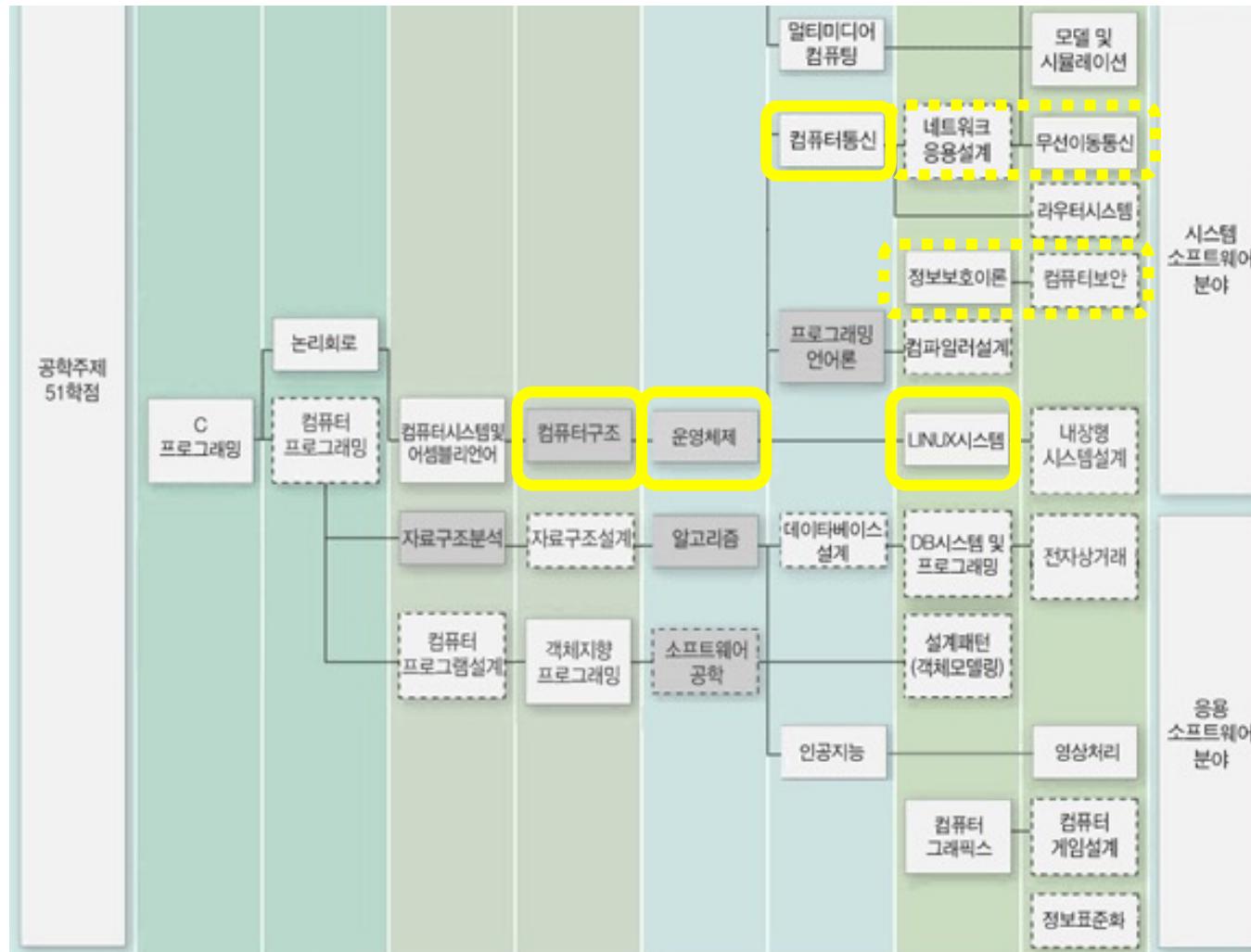
**파워**  
노예에게 주는 월급  
부족하면 의해져서  
일 못함

**ssd**  
최신식으로 좋은  
저장창고

**NVme ssd**  
더 업그레이드해서  
정말 빠르고  
좋은 저장창고

# 선수 지식

- 인프라 분야 관련 과목

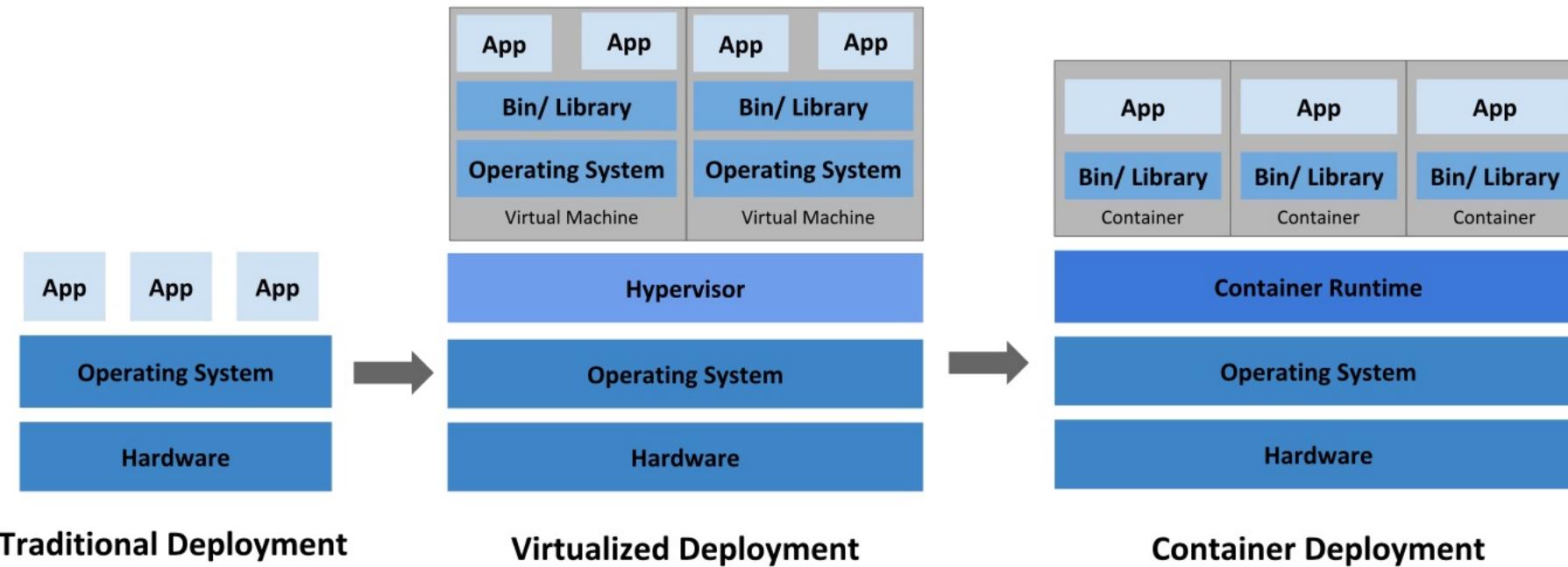


# 쿠버네티스란?

- 쿠버네티스는 컨테이너화된 애플리케이션을 배포, 관리하기 위한 오픈소스 오케스트레이터
- 쿠버네티스란 명칭은 키잡이(helmsman)나 파일럿을 뜻하는 그리스어에서 유래 (그리스어 : 퀴베르네테스 kubernētēs)
- K8s라는 표기는 "K"와 "s"와 그 사이에 있는 8글자를 나타내는 약식 표기
- 구글이 2014년에 쿠버네티스 프로젝트를 오픈소스화 (기존 이름 : Borg)
- 쿠버네티스는 프로덕션 워크로드를 대규모로 운영하는 15년 이상의 구글 경험과 커뮤니티의 아이디어와 적용 사례가 결합되어 있음
- 쿠버네티스에서 관리할 수 있는 가장 작은 단위는 파드(Pod)이며, 하나의 애플리케이션을 나타내고, 스토리지 리소스 및 IP 주소를 공유하는 1개 이상의 컨테이너로 구성
- 쿠버네티스 공식 사이트 <https://kubernetes.io/ko/>
- 쿠버네티스 커뮤니티 <https://www.facebook.com/groups/k8skr/>

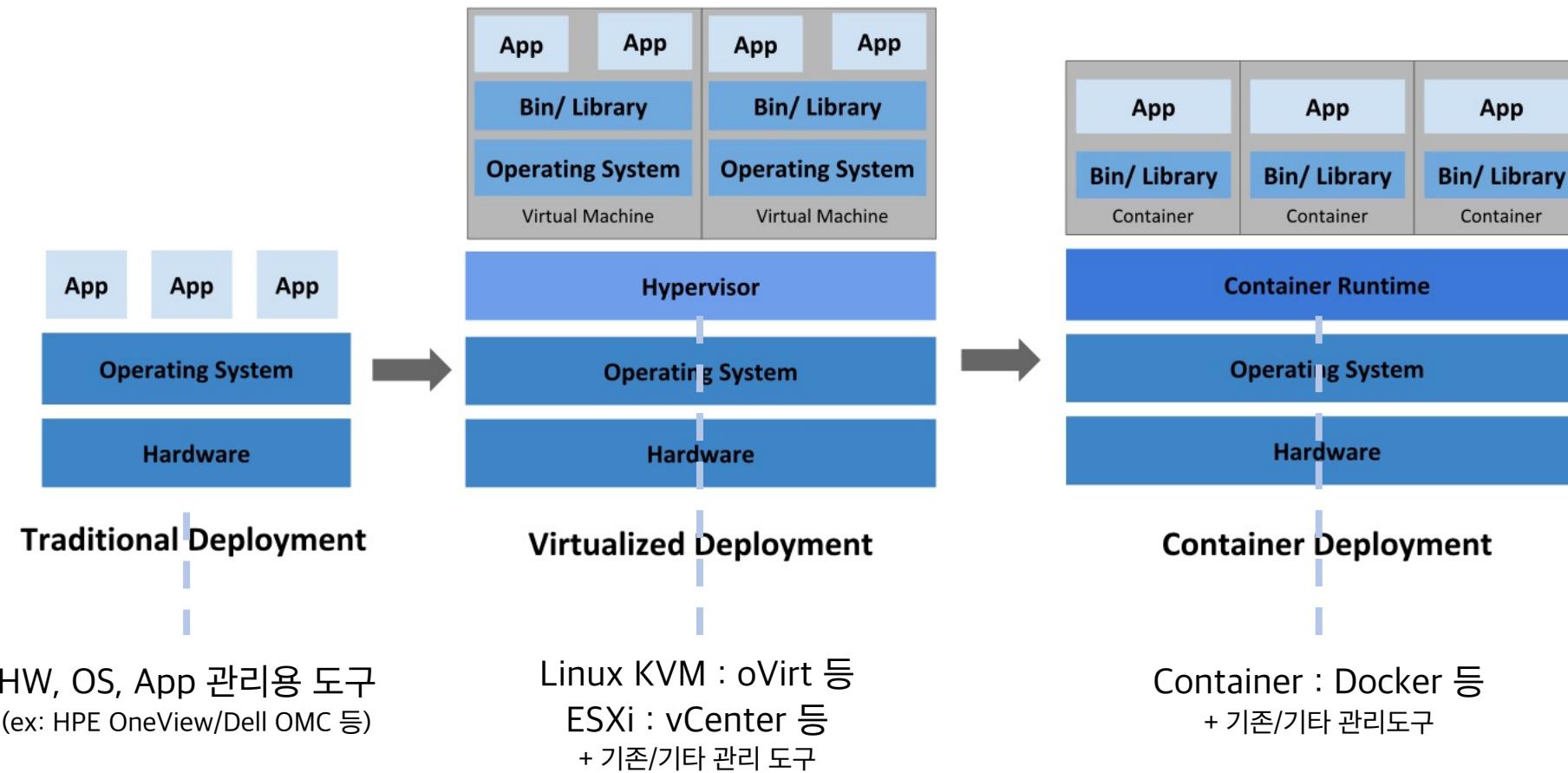


# 인프라 발전 히스토리

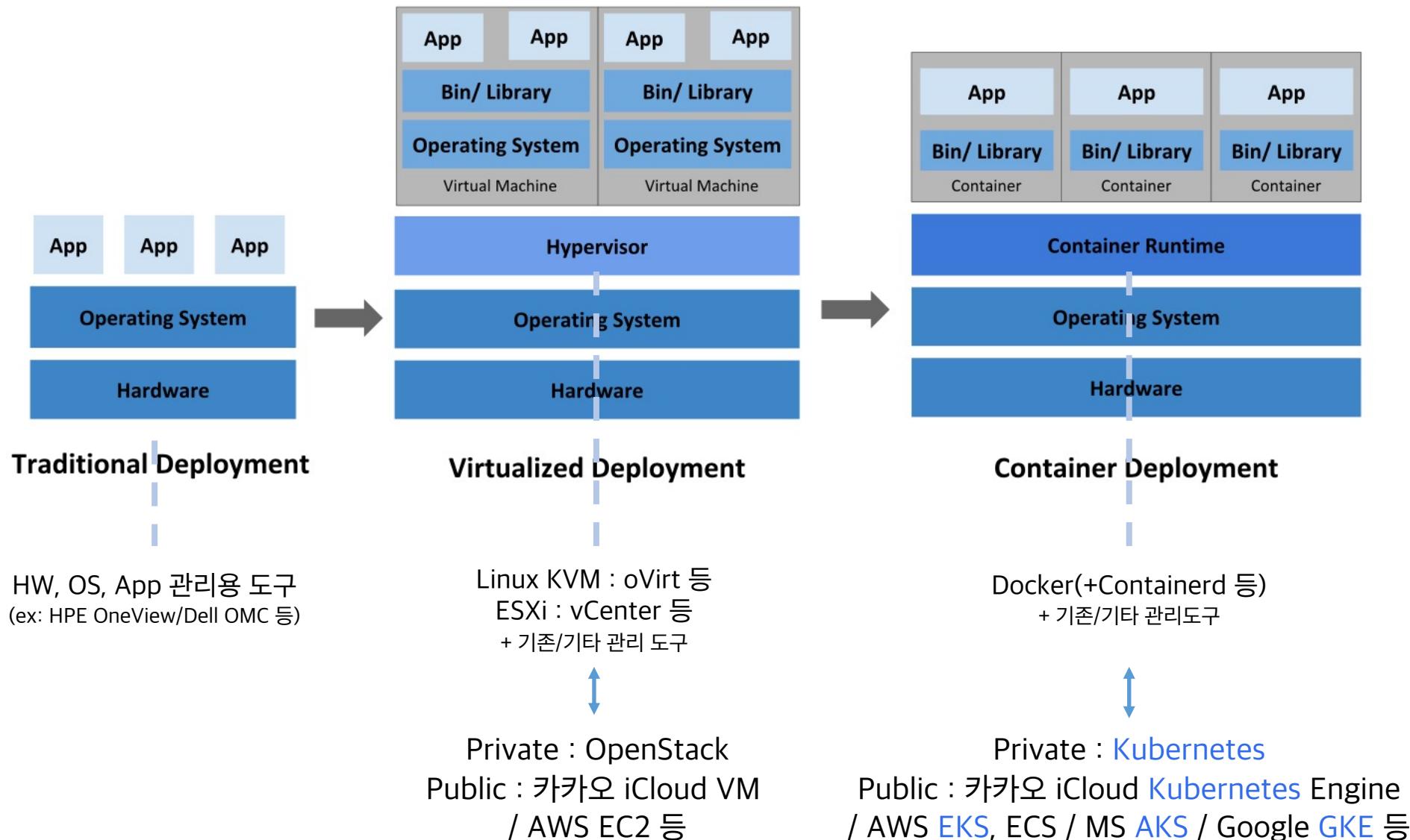


- 컨테이너에서는 OS를 공유하므로 가볍고 빠름
- 작고 독립적인 단위로 쪼개져서 구동
- 고효율/고집적

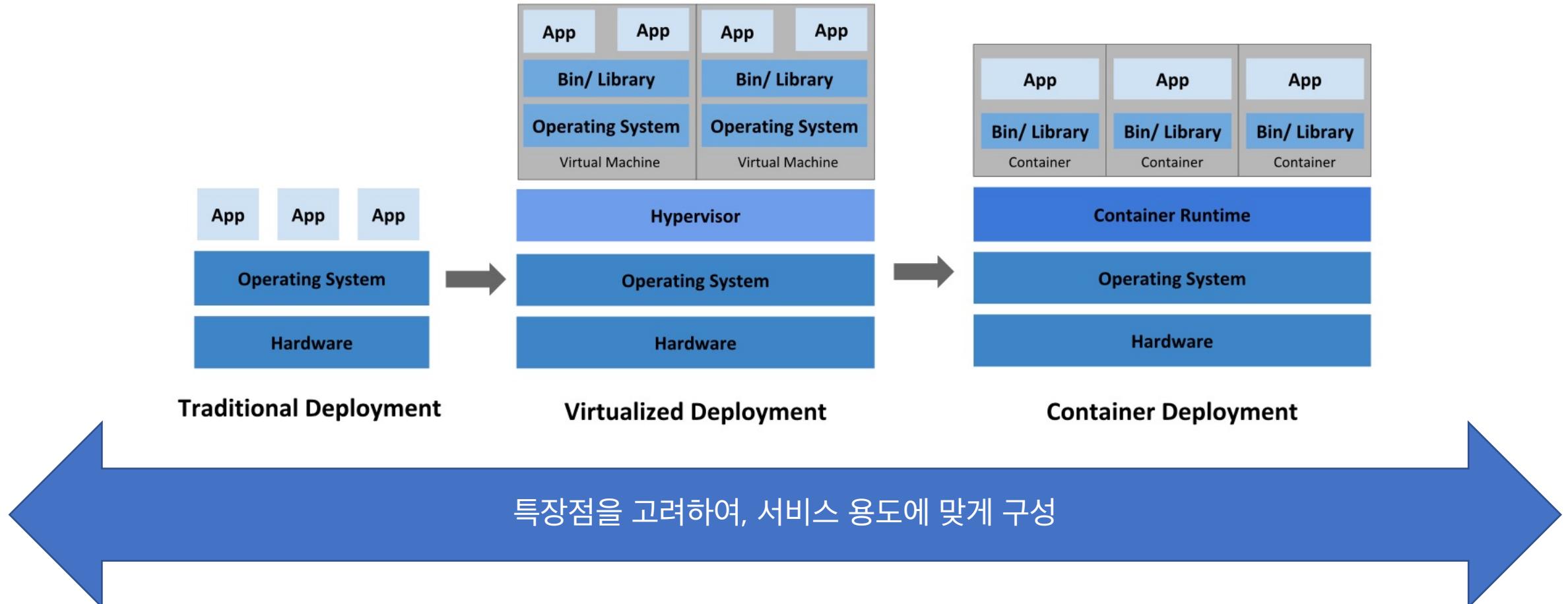
# 인프라 발전 히스토리 - 관리도구



# 인프라 발전 히스토리 - 클라우드로의 발전, 대규모



# 인프라 발전 히스토리

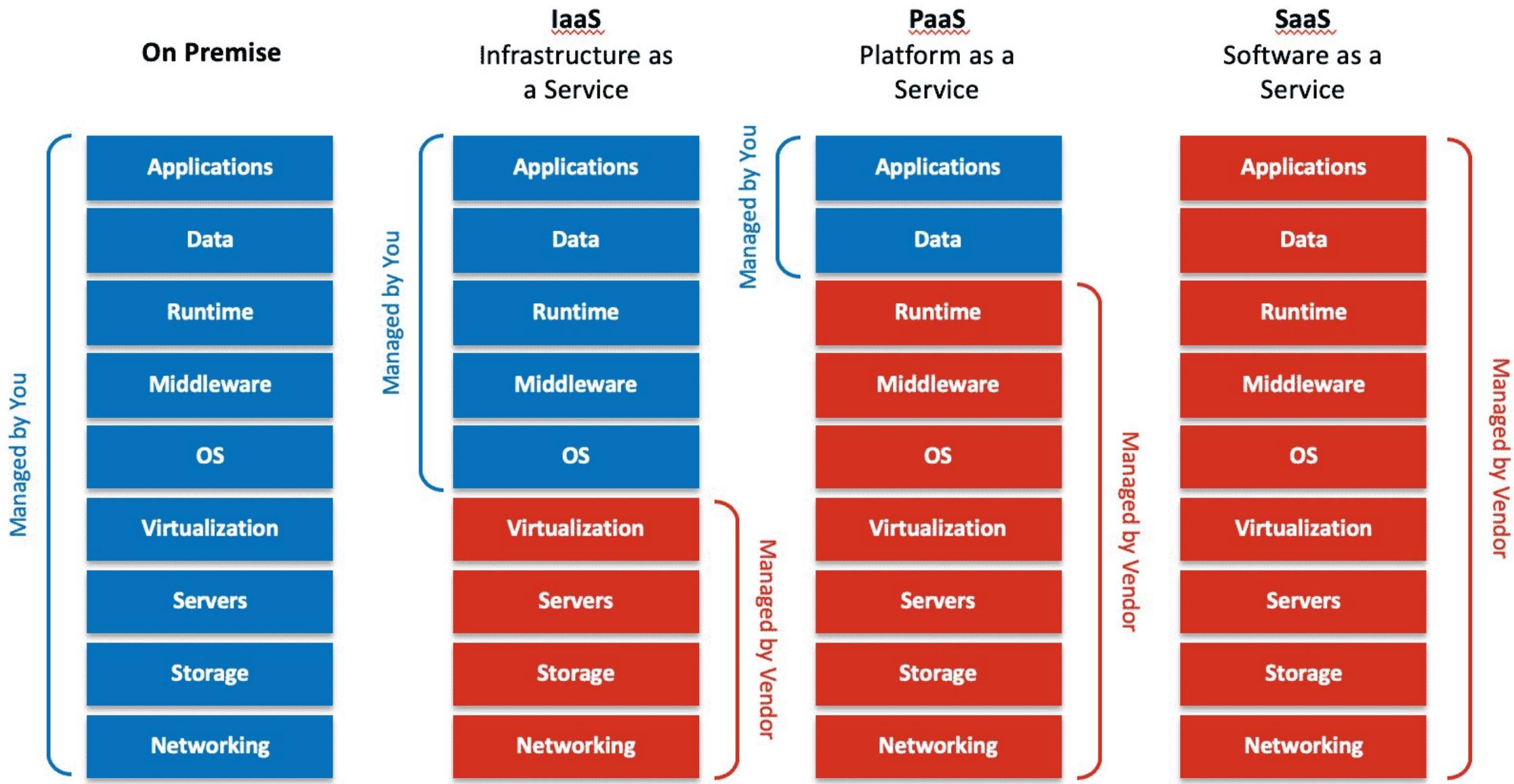


- 장애 대응
- 성능 유지 용이
- 전체 관점으로 운용
- 고효율/고집적
- 빠른 시작 / 코드화 용이
- 애플리케이션에만 집중

# 인프라 발전 히스토리



# 클라우드에서 제공하는 리소스



그림출처 : Microsoft

# 쿠버네티스 리소스 : 쿠버네티스가 제공하는 것

PaaS  
Platform as a  
Service



# 쿠버네티스가 필요한 이유

컨테이너 관리용 오케스트레이션으로  
가장 빠르게 발전하는 사실상 표준(de facto)

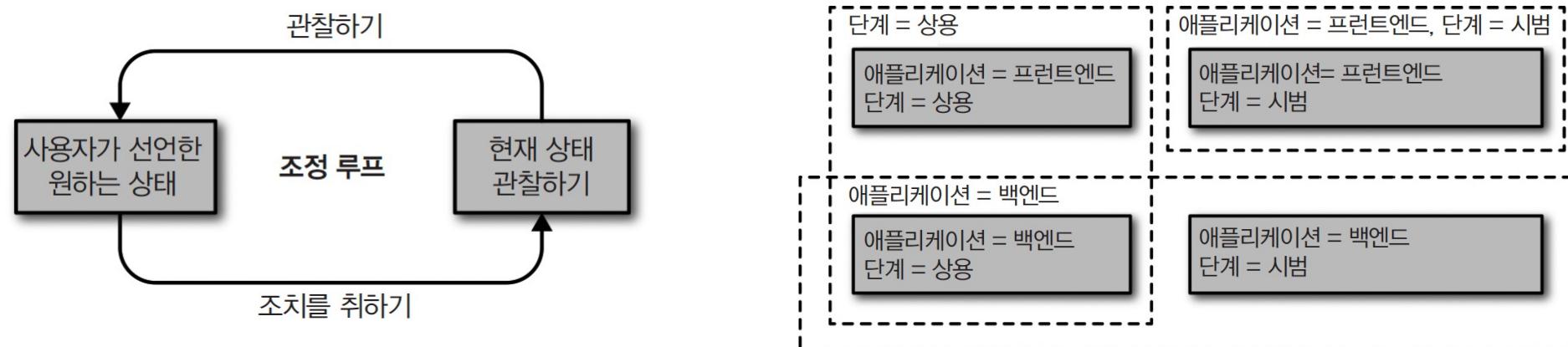
- 서비스 디스커버리와 로드 밸런싱
  - 쿠버네티스는 DNS 이름을 사용하거나 자체 IP 주소를 사용하여 컨테이너를 노출 가능
  - 컨테이너에 대한 트래픽이 많으면, 쿠버네티스는 네트워크 트래픽을 로드밸런싱하고 배포하여 안정적인 배포 가능
- 스토리지 오케스트레이션 : 쿠버네티스를 사용하면 로컬 저장소, 공용 클라우드 공급자 등과 같은 저장소를 자동 탑재
- 자동화된 롤아웃과 롤백
  - 쿠버네티스를 사용하여 배포된 컨테이너의 원하는 상태를 서술 가능
  - 쿠버네티스를 자동화해서 배포용 새 컨테이너 생성/기존 컨테이너 제거/모든 리소스를 새 컨테이너에 적용 가능
- 자동화된 배치 - 빈 패킹(bin packing)
  - 컨테이너화된 작업을 실행하는데 사용할 수 있는 쿠버네티스 클러스터 노드를 제공
  - 쿠버네티스는 컨테이너를 노드에 맞추어서 리소스를 가장 잘 사용할 수 있도록 선택
- 자동화된 복구(self-healing)  
쿠버네티스는 실패한 컨테이너를 다시 시작/교체 및 '사용자 정의 상태 검사'에 응답하지 않는 컨테이너를 제거
- 시크릿과 구성 관리
  - 쿠버네티스를 사용하면 암호, OAuth 토큰 및 SSH 키와 같은 중요한 정보를 저장하고 관리 가능



## [구조/개념] Kubernetes Architecture

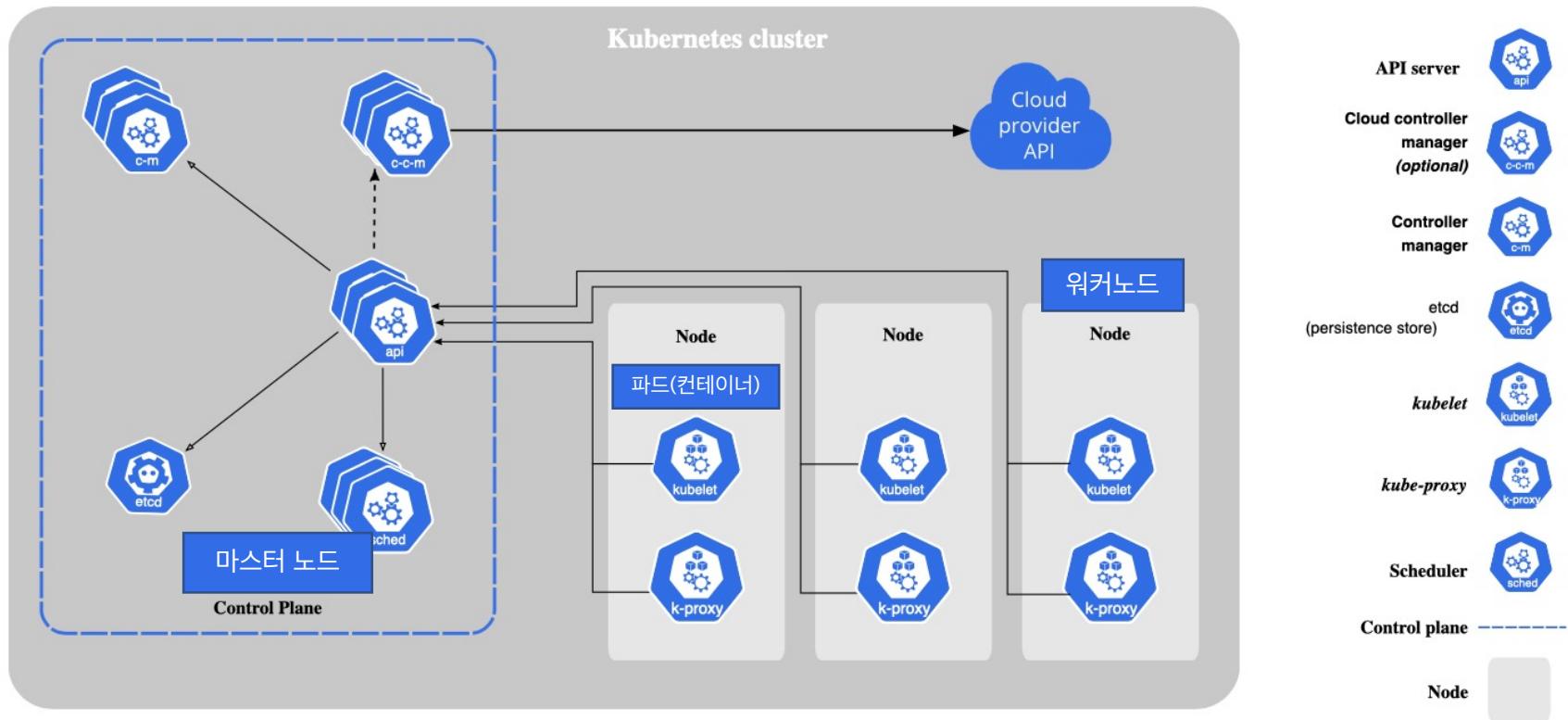
# 쿠버네티스 설계 사상

- 선언적 구성 : “내 웹서버의 레플리카를 항상 5개씩 실행하고 싶습니다” cf) 명령어 구성 : ”레플리카를 5개 만들어”
  - 제어(조정)루프 : 현재 상태를 관찰하여 사용자가 원하는 상태로 유지 ex) 에어컨
- 컨트롤러 구성 : 각 기능별로 독립적으로 분산되게 설계 ► 유연하고 안정적이지만 복잡함
- 동적 그룹화 : “우리 팀원은 오렌지색 옷을 입은 사람들입니다.” cf) 정적 그룹화 : 우리 팀원은 철수와 영희입니다.
  - 레이블(쿼리 가능) 및 어노테이션(메타데이터용)으로 구성
- API 기반 상호작용 : 쿠버네티스 요소들이 서로 직접 접근 불가, 구성 요소를 대체 구현하기 용이



# 쿠버네티스 구조 : 클러스터

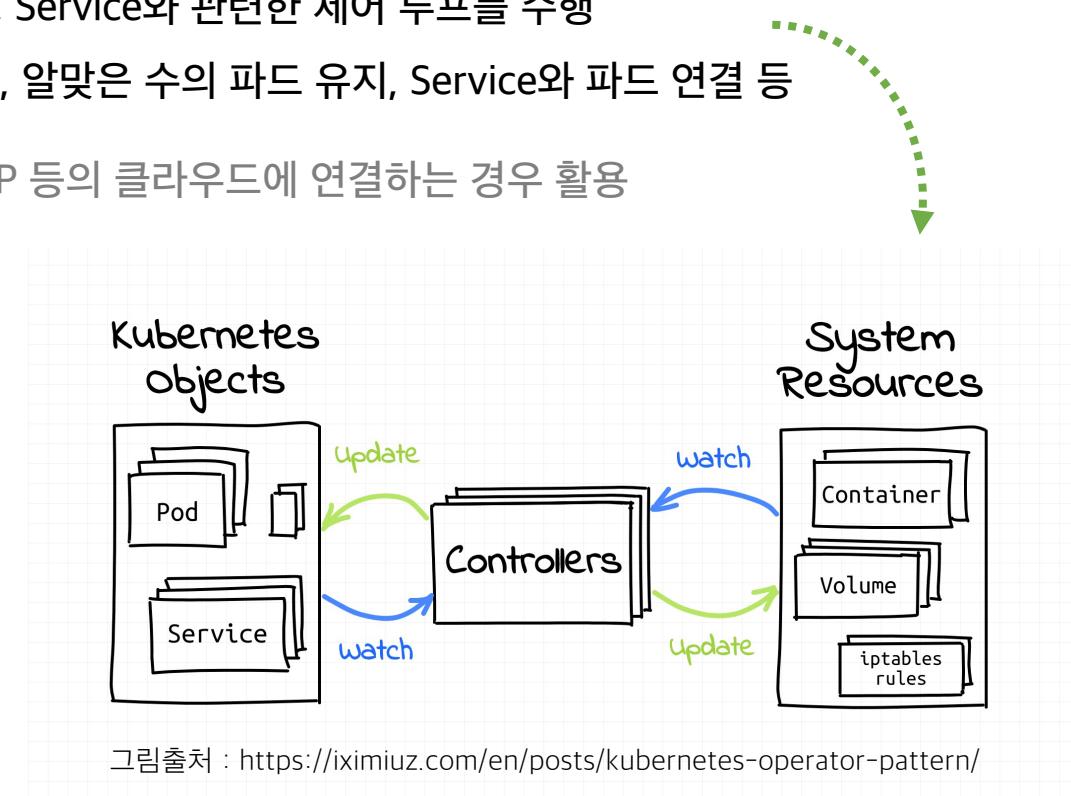
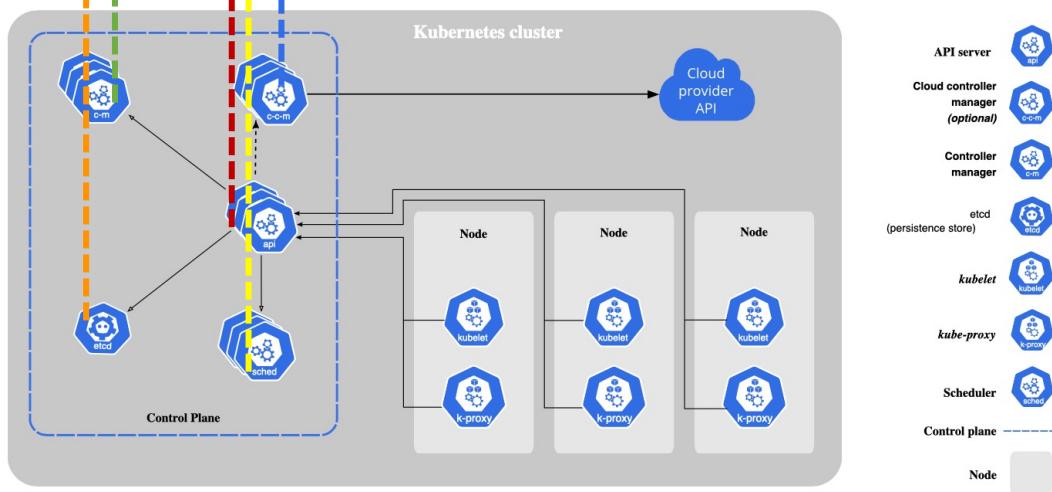
- 클러스터란? 노드라고 하는 워커 머신의 집합
- 노드란? 컨테이너화된 애플리케이션을 실행하는 서버(가상 또는 물리 둘다 가능)
- 워커 노드는 애플리케이션의 구성요소인 파드(↪ 컨테이너)를 호스트
- 컨트롤 플레인은 워커 노드와 클러스터 내 파드를 관리



# 쿠버네티스 구조 : 클러스터 구성요소

- 컨트롤 플레인(마스터 노드)

- API서버 : 쿠버네티스 API를 노출, 쿠버네티스 컨트롤 플레인의 프론트 엔드, 수평확장 가능
- etcd : 모든 클러스터 데이터를 담는 쿠버네티스 뒷단의 저장소로 사용되는 일관성·고가용성 키-값 저장소
- 스케줄러 : 노드가 배정되지 않은 새로 생성된 파드를 감지하고, 실행할 노드를 선택
- 컨트롤러 매니저 : ReplicaSets, Deployment, Service와 관련한 제어 루프를 수행  
노드가 다운되면 통지/대응, 알맞은 수의 파드 유지, Service와 파드 연결 등
- 클라우드 컨트롤러 매니저 : AWS, Azure, GCP 등의 클라우드에 연결하는 경우 활용

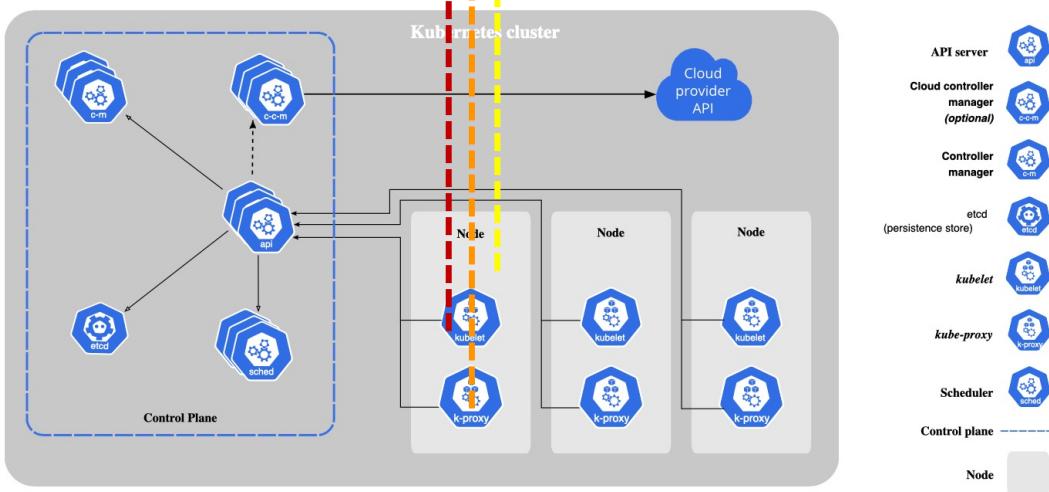


그림출처 : <https://iximiuz.com/en/posts/kubernetes-operator-pattern/>

# 쿠버네티스 구조 : 클러스터 구성요소

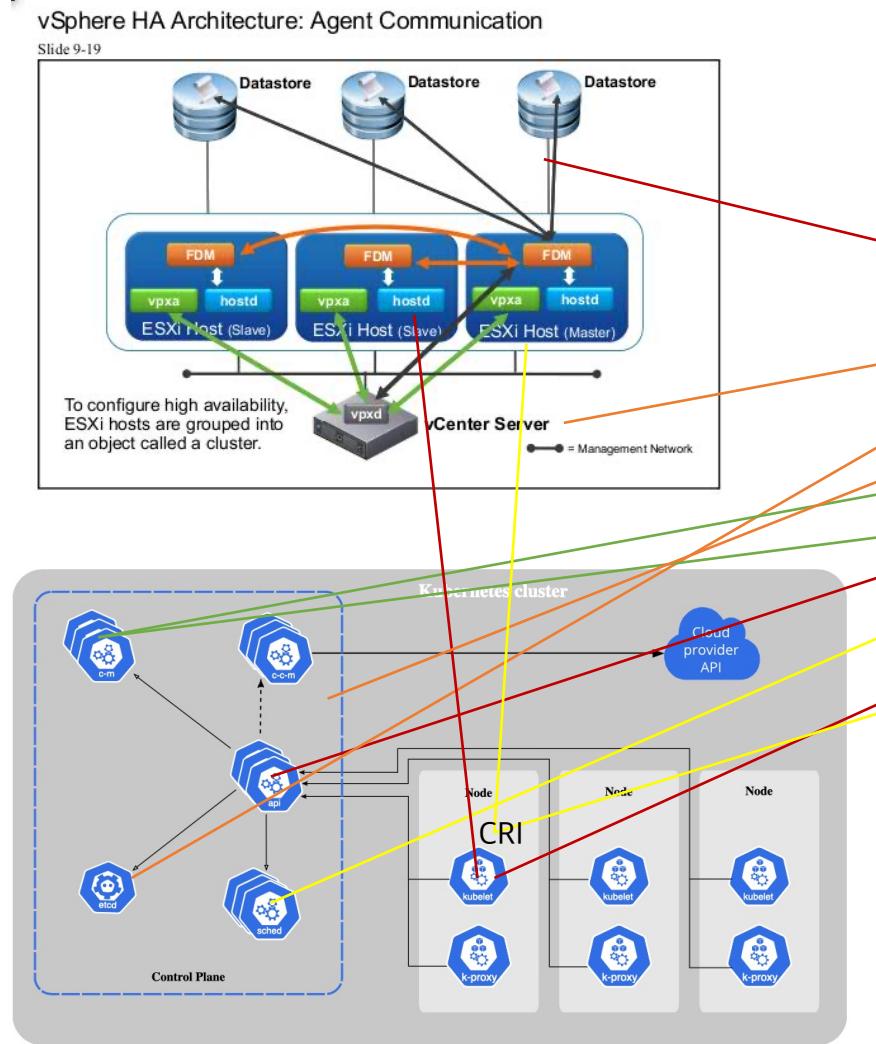
- 워커 노드

- Kubelet(쿠블렛) : 컨테이너가 동작하도록 관리, 쿠버네티스 클러스터와 워커노드의 CPU/Mem/Disk 간을 연결
- Kube-proxy : 쿠버네티스 Service(로드밸런서 리소스)에 맞게 커널의 netfilter(iptables) 등을 관리하는 역할
- 컨테이너 런타임 : 컨테이너 실행을 담당하는 소프트웨어(도커, containerd, CRI-O 등)

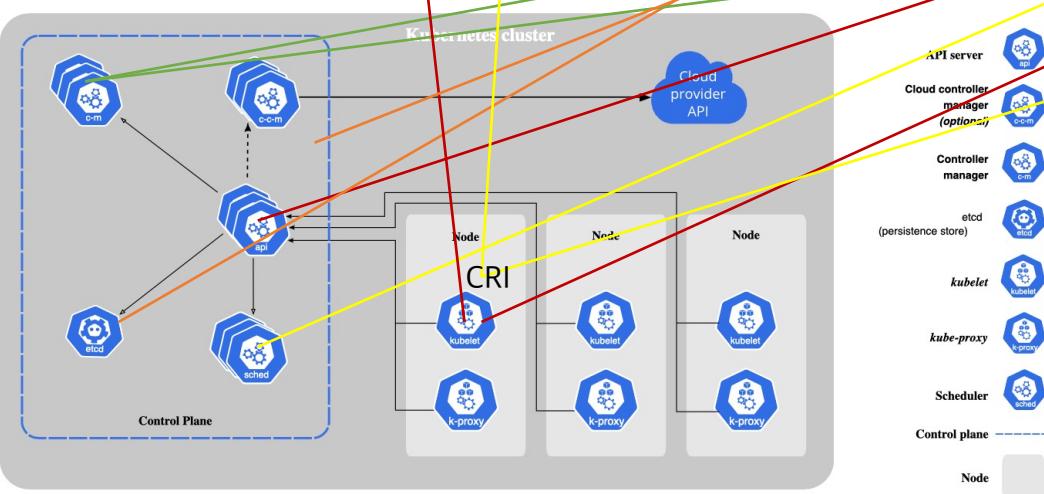
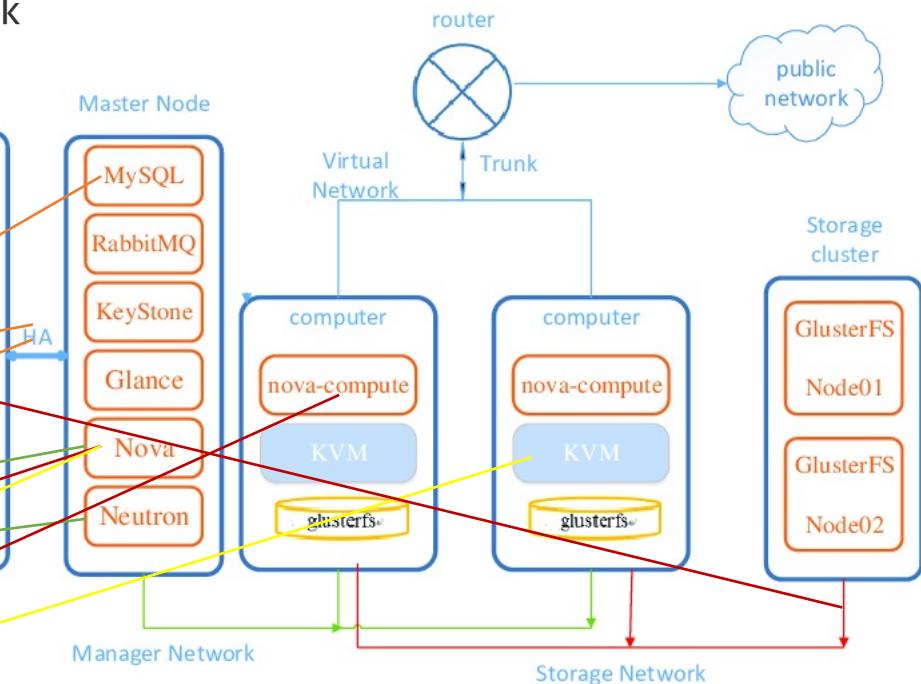


# 쿠버네티스 구조(번외) : 타 솔루션과 비교

- VMware



- OpenStack



# 쿠버네티스 설치

- 설치방법(배포도구 활용)

- Kubeadm, Kops, Kubespray

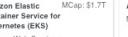
차이? 공식docs 링크 문서 / <https://github.com/kubernetes-sigs/kubespray/blob/master/docs/comparisons.md>

- 설치방법(턴키솔루션 활용)

- Redhat Openshift, AWS EKS, MS AKS 등

## 턴키 클라우드 솔루션

이 페이지는 인증된 쿠버네티스 솔루션 제공자 목록을 제공한다. 각 제공자 페이지를 통해서, 프로덕션에 준비된 클러스터를 설치 및 설정하는 방법을 학습할 수 있다.

Platform - Certified Kubernetes - Hosted (45)												
 Microsoft Azure	AKS Engine for Azure Stack Microsoft	MCap: \$2.2T	 Alibaba Cloud Container Service for Kubernetes Alibaba Cloud	MCap: \$510.2B	 Amazon Elastic Container Service for Kubernetes (EKS) Amazon Web Services	MCap: \$1.7T	 Microsoft Azure	Azure (AKS) Engine Microsoft	MCap: \$8B	 Azure Kubernetes Service (AKS) Microsoft	MCap: \$2.2T	
 BIZMICRO	Bizmicro KBSYS		 BoCloud BeyondContainer BoCloud	Funding: \$20.8M	 catalyst cloud	Catalyst Kubernetes Service Catalyst Cloud	 CIVO	Civo Kubernetes Civo	Funding: \$840K	 DigitalOcean Kubernetes DigitalOcean	MCap: \$5.9B	
 eBaoCloud	eBaoTech		 ELASTX Private Kubernetes ELASTX	ELASTX	 Exoscale	Exoscale	 GERMAN EDGE CLOUD Kubernetes Services (GKS) German Edge Cloud	German Edge Cloud Kubernetes Services (GKS) German Edge Cloud		 Google Kubernetes Engine Google	MCap: \$1.8T	
 HUAWEI	Huawei Cloud Container Engine (CCE) Huawei Technologies		 HUAWEI	Huawei NFV FusionStage Huawei Technologies	 IBM Cloud Kubernetes Service IBM	IBM	 inspur ICKS Inspur Group	inspur ICKS Inspur Group	MCap: \$7.1B	 InsureMO eBaoTech	MCap: \$1.8T	
 KUBESPHERE	KubeSphere		 linode	linode	 MAIL.RU CLOUD SOLUTIONS	MAIL.RU CLOUD SOLUTIONS	 Nexastack	Nexastack		 UCloud Kubernetes Service (UKS) UCloud Information Technology	MCap: \$2.4B	
							 VENTUS	VENTUS		 VEXXHOST	VEXXHOST	
							 WoCloud	WoCloud		 ZTE	ZTE	
							 OpenPalette	OpenPalette				

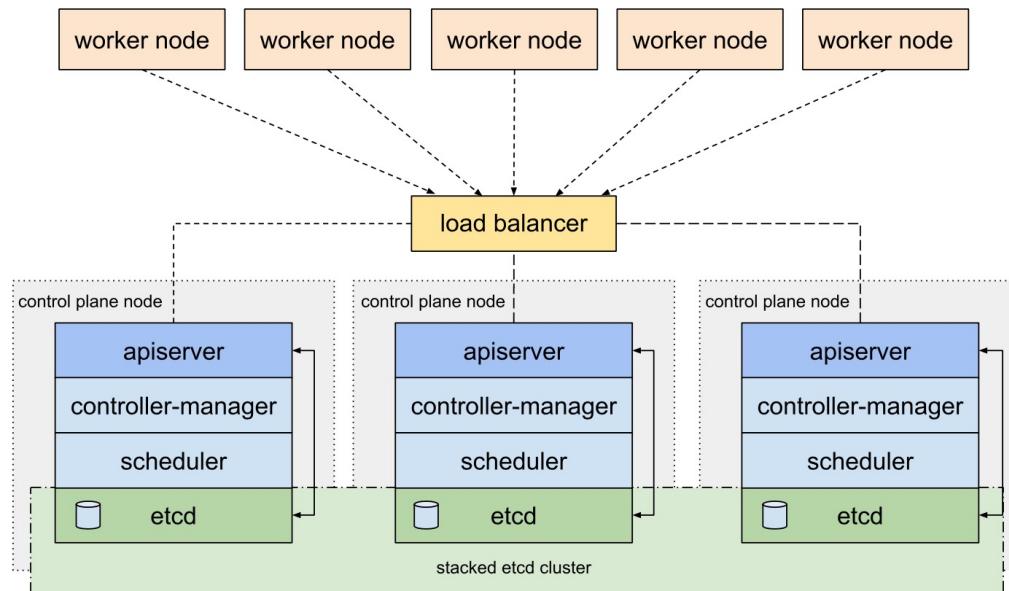
# 쿠버네티스 설치

- 고가용성 토플로지

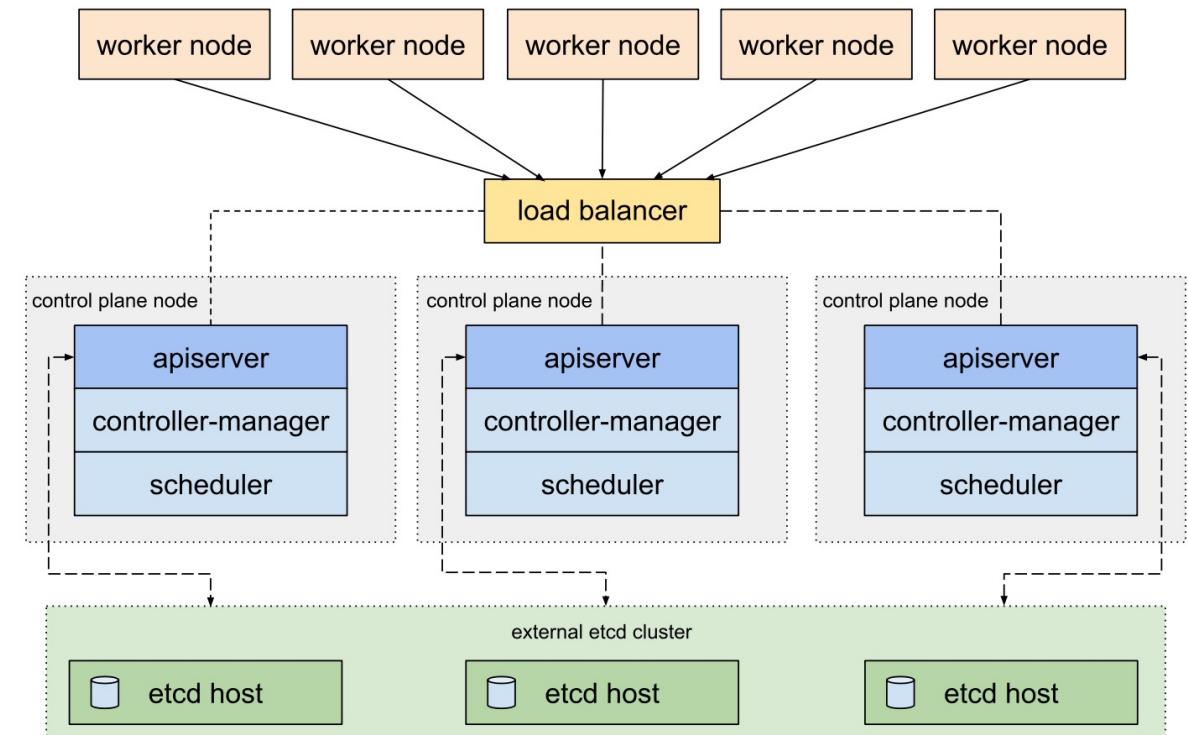
<https://kubernetes.io/ko/docs/setup/production-environment/tools/kubeadm/ha-topology/>

- 중첩된 etcd 토플로지 vs 외부 etcd 토플로지 : etcd 배포 위치 차이

kubeadm HA topology - stacked etcd

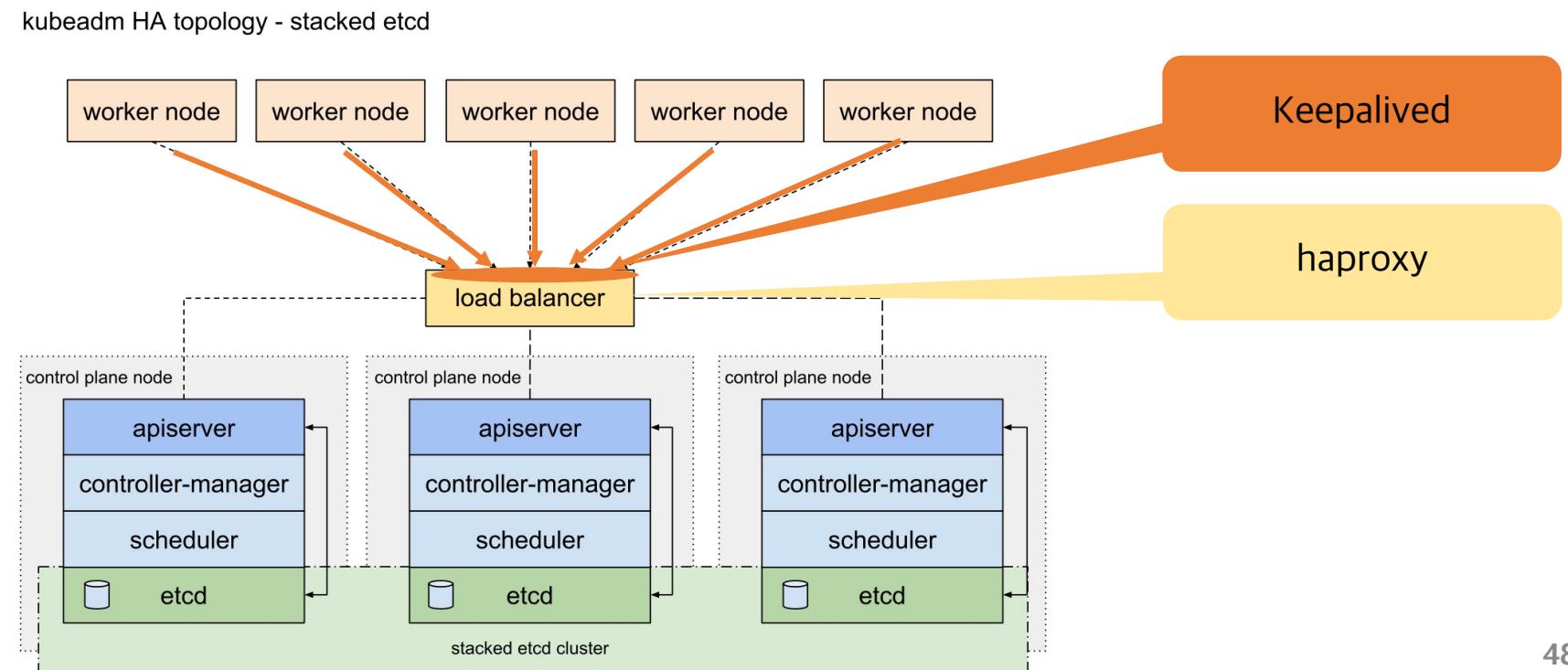


kubeadm HA topology - external etcd



# 쿠버네티스 설치

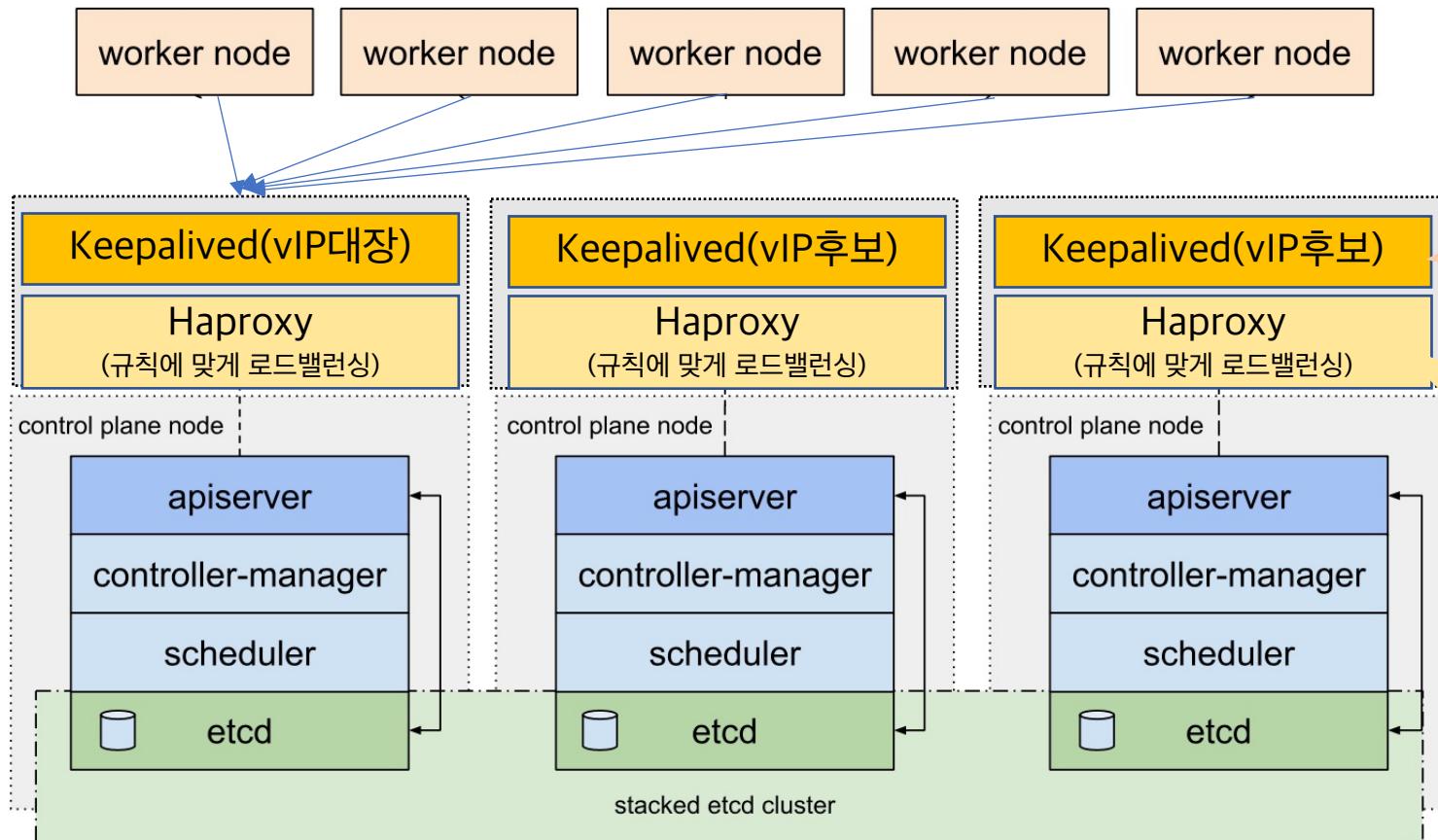
- Kubeadm으로 High Availability 클러스터 설치  
<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/>
- 3개의 Master에 연결하기 위한 물리적인 L4스위치 / ha-proxy / nginx / CSP에서 제공하는 Load Balancer 등 다양한 Load Balancer 구성 방식이 있으며, 본 교육에서는 ha-proxy(로드밸런서) 사용



# 쿠버네티스 설치

- 물리적 구성도

kubeadm HA topology - stacked etcd



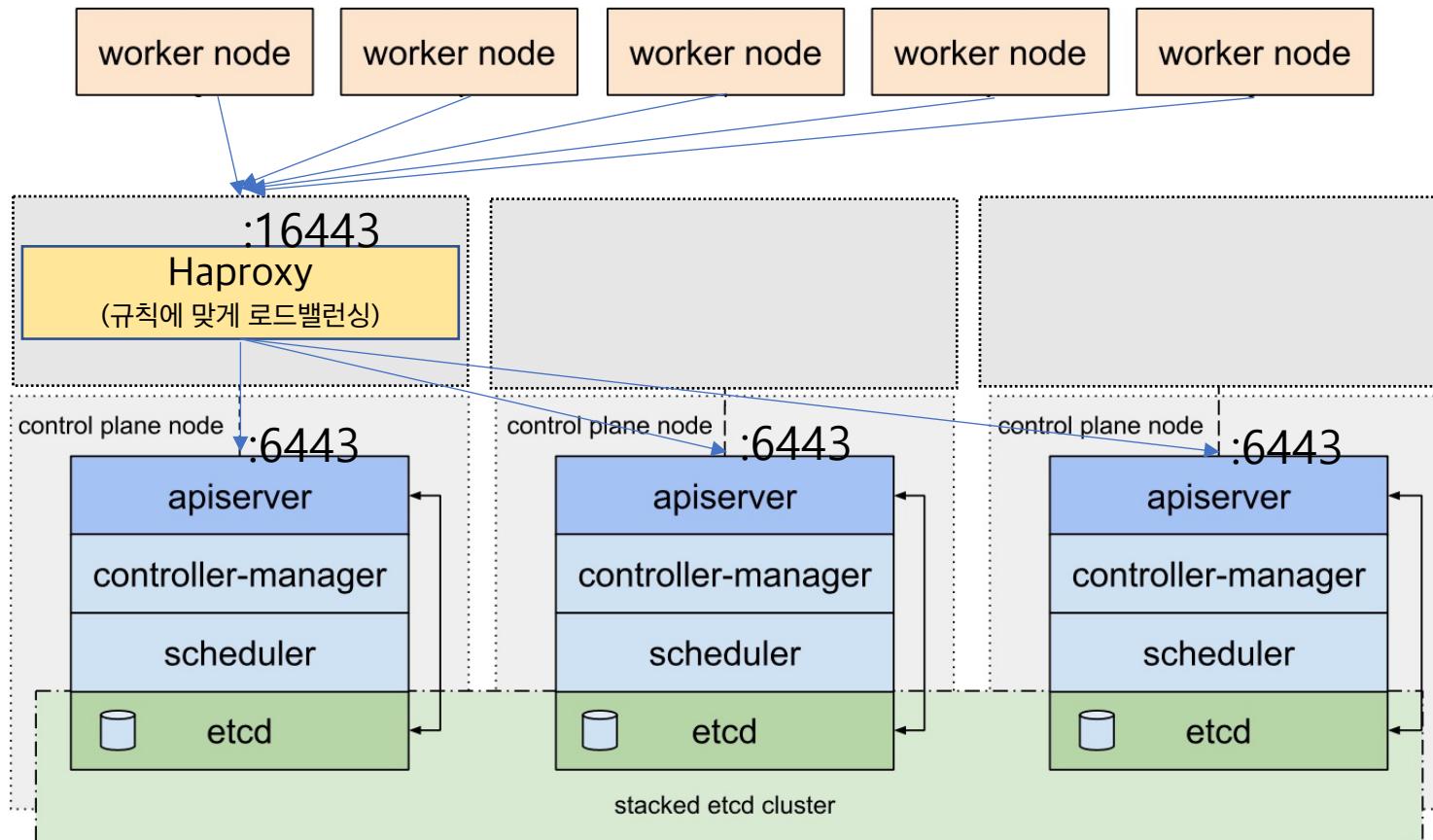
Keepalived 끼리 이야기해서 대장 정하고, 대장은 스위치에 내가 대장(vIP)라고 알림.  
Worker node가 vIP 목적지로 패킷을 보내면 스위치는 대장에게 보냄

HA Proxy의 설정을 동일하게 맞추게 되면 어떤 노드가 대장이 되던지 해당 haproxy가 골고루 controller node 3개로 분배할 것임  
(controller가 죽으면 안 보냄)

# 쿠버네티스 설치

- 물리적 구성도

kubeadm HA topology - stacked etcd



Keepalived는  
현 실습 환경에서 불가

yum install haproxy

# 쿠버네티스 설치

- [Master 1번 노드만] Load balancer인 haproxy 설치, 설정

```
# sudo setenforce 0
# sudo yum install haproxy -y
# sudo vi /etc/haproxy/haproxy.cfg

frontend kubernetes-master-lb
    bind 0.0.0.0:16443
    option tcplog
    mode tcp
    default_backend kubernetes-master-nodes

backend kubernetes-master-nodes
    mode tcp
    balance roundrobin
    option tcp-check
    option tcplog
    server master1 172.30.4.88:6443 check
    server master2 172.30.7.75:6443 check
    server master3 172.30.6.170:6443 check
```

- 16443 포트가 Listen 중인지 확인

```
# netstat -nltp
```

# 쿠버네티스 설치

- Kubeadm 설치

<https://kubernetes.io/ko/docs/setup/production-environment/tools/kubeadm/install-kubeadm/>



- 설치 전,
  - 리눅스 기반. 2GB Ram, 2 CPU 이상의 성능
  - SWAP 미사용 (카카오 기본 이미지에 SWAP 미설정됨)
  - 고유한 hostname, MAC, UUID, 포트 개방, 네트워크 연결
- [전체노드] # hostname 으로 hostname 확인 후 /etc/hosts 에 반영 (worker도)
- MAC주소 및 UUID가 고유한지 확인
- 네트워크 어댑터 확인
  - 카카오 내 VM은 1개만 이므로 관계없음

# 쿠버네티스 설치

- [전체 노드] br\_netfilter 모듈 로딩 : 리눅스 노드의 iptables가 브리지된 트래픽을 올바르게 보기 위한 요구 사항

```
# 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
```

# 쿠버네티스 설치

- 주요 포트 확인
- 외부에서 쿠버네티스 API를 접근 시 6443 포트 사용

## 필수 포트 확인

### 컨트롤 플레인 노드

프로토콜	방향	포트 범위	목적	사용자
TCP	인바운드	6443*	쿠버네티스 API 서버	모두
TCP	인바운드	2379-2380	etcd 서버 클라이언트 API	kube-apiserver, etcd
TCP	인바운드	10250	kubelet API	자체, 컨트롤 플레인
TCP	인바운드	10251	kube-scheduler	자체
TCP	인바운드	10252	kube-controller-manager	자체

### 워커 노드

프로토콜	방향	포트 범위	목적	사용자
TCP	인바운드	10250	kubelet API	자체, 컨트롤 플레인
TCP	인바운드	30000-32767	NodePort 서비스†	모두

† [NodePort 서비스](#)의 기본 포트 범위.

\*로 표시된 모든 포트 번호는 재정의할 수 있으므로, 사용자 지정 포트도 열려 있는지 확인해야 한다.

etcd 포트가 컨트롤 플레인 노드에 포함되어 있지만, 외부 또는 사용자 지정 포트에서 자체 etcd 클러스터를 호스팅할 수도 있다.

사용자가 사용하는 파드 네트워크 플러그인(아래 참조)은 특정 포트를 열어야 할 수도 있다. 이것은 각 파드 네트워크 플러그인마다 다르므로, 필요한 포트에 대한 플러그인 문서를 참고한다.

# 쿠버네티스 설치

- [전체 노드] CRI (Container runtime interface) 설치 필요 : Docker 설치/시작
- ```
# sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```
- ```
# sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```
- ```
# sudo yum install docker-ce -y
```
- ```
# sudo vi /usr/lib/systemd/system/docker.service
```

```
ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --exec-opt native.cgroupdriver=systemd
```
- ```
# sudo systemctl daemon-reload
```
- ```
# sudo systemctl start docker && sudo systemctl enable docker
```
- ```
# sudo docker info | grep -i cgroup
```

## Configuring the container runtime cgroup driver

The [Container runtimes](#) page explains that the `systemd` driver is recommended for kubeadm based setups instead of the `cgroupfs` driver, because kubeadm manages the kubelet as a systemd service.

## Configuring the kubelet cgroup driver

kubeadm allows you to pass a `KubeletConfiguration` structure during `kubeadm init`. This `KubeletConfiguration` can include the `cgroupDriver` field which controls the cgroup driver of the kubelet.

**Note:** In v1.22, if the user is not setting the `cgroupDriver` field under `KubeletConfiguration`, `kubeadm` will default it to `systemd`.

# 쿠버네티스 설치

- [전체 노드] Redhat 배포판 버전으로 따라하기

<https://kubernetes.io/ko/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#kubeadm-kubelet-및-kubectl-설치>

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
EOF
```

- # sudo setenforce 0
- # sudo sed -i 's/^SELINUX=enforcing\$/SELINUX=permissive/' /etc/selinux/config

# 쿠버네티스 설치

- [전체 노드] 단, kubelet / kubeadm / kubectl 은 최신보다 한단계 낮은 버전으로 설치 (향후 교육과정 고려)

```
# sudo yum info kubelet --disableexcludes=Kubernetes -y  
# sudo yum install kubelet-1.21.0 --disableexcludes=kubernetes -y  
# sudo yum install kubectl-1.21.0 --disableexcludes=kubernetes -y  
# sudo yum install kubeadm-1.21.0 --disableexcludes=kubernetes -y  
# sudo systemctl enable --now kubelet
```

- 앞에서 도커의 cgroup driver를 systemd로 설정했기에, 쿠버도 마찬가지로 설정

```
# sudo vi /usr/lib/systemd/system/kubelet.service.d/10-kubeadm.conf 내에
```

[Service]

```
Environment="KUBELET_CGROUP_ARGS=--cgroup-driver=systemd --runtime-cgroups=/systemd/system.slice --kubelet-cgroups=/systemd/system.slice"
```

- daemon reload, kubelet restart 수행. 이제 필요한 패키지는 모두 설치 완료. 클러스터를 만들어야 함

# 쿠버네티스 설치

- [컨트롤 첫번째만] (etcd가 함께 포함된) Stack Control node 설치

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/high-availability/#stacked-control-plane-and-etcd-nodes>

Stacked control plane and etcd nodes

```
sudo kubeadm init --control-plane-endpoint "LOAD_BALANCER_DNS:LOAD_BALANCER_PORT" --upload-certs
```

- # sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --control-plane-endpoint "172.30.~.~:16443" --upload-certs

---

Calico 활용 예정

HA Proxy가 설치된 master1 및 해당 포트

인증서 전달하여

손쉽게 구성



Your Kubernetes control-plane has initialized successfully!

# 쿠버네티스 설치

- [결과 메세지에 따라 이어서 진행]
- 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  
  
Alternatively, if you are the root user, you can run:  
export KUBECONFIG=/etc/kubernetes/admin.conf
```

Kubeconfig  
인증에 사용되는 쿠베컨피그

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/>

→ <https://docs.projectcalico.org/getting-started/kubernetes/self-managed-onprem/onpremises>  
→ # curl https://docs.projectcalico.org/manifests/calico.yaml -o  
→ # kubectl apply -f calico.yaml

# 쿠버네티스 설치

- [결과 메세지에 따라 진행]

(다른 마스터노드 2,3에서 진행)

You can now join any number of the control-plane node running the following command on each **as root**:

```
kubeadm join 172.30.5.193:16443 --token 1ecqfl.3grbrnswq5ggwakz \
--discovery-token-ca-cert-hash sha256:6daa1758e52cf26c5ba797700c7fb33fdc89294a3c2fc02a185fdb309907b3ea \
--control-plane --certificate-key 6e924d9d18d75d979eda3599e9d62da107e3b72a9f8c64b71bea4b00d78c01ed
```

Please note that the certificate-key gives access to cluster sensitive data, keep it secret!

As a safeguard, uploaded-certs will be deleted in two hours; If necessary, you can use

"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.

Then you can join any number of worker nodes by running the following on each **as root**:

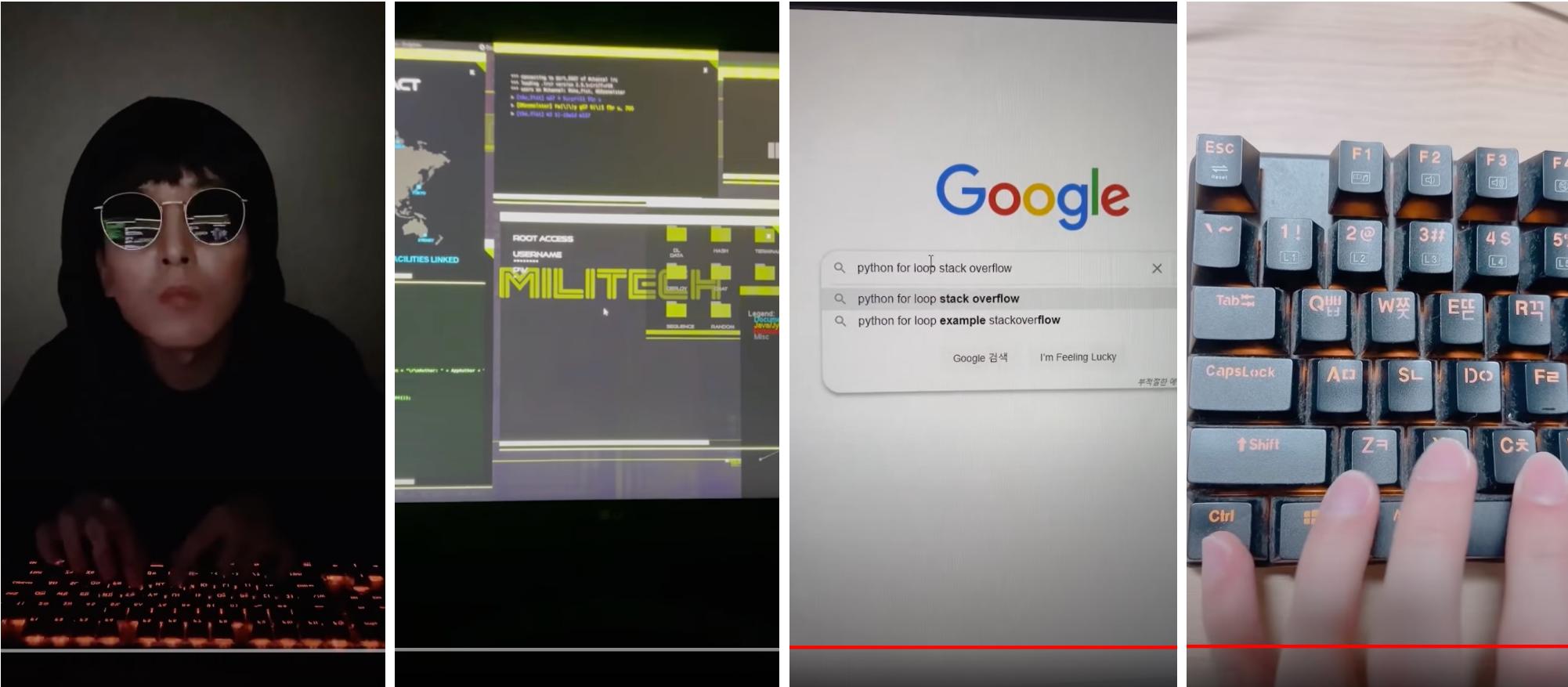
(워커노드1,2에서 진행)

```
kubeadm join 172.30.5.193:16443 --token 1ecqfl.3grbrnswq5ggwakz \
--discovery-token-ca-cert-hash sha256:6daa1758e52cf26c5ba797700c7fb33fdc89294a3c2fc02a185fdb309907b3ea
```

# 쿠버네티스 설치

- 쉬엄쉬엄, 차근차근

<https://www.youtube.com/watch?v=cXB8rCW7lto>



# 쿠버네티스 설치

- 클러스터 전체 구성 완료!

```
[root@osk-master-01 ~]# kubectl get nodes
```

| NAME                                  | STATUS | ROLES                | AGE   | VERSION |
|---------------------------------------|--------|----------------------|-------|---------|
| osk-master-01.kr-central-1.c.internal | Ready  | control-plane,master | 22m   | v1.21.0 |
| osk-master-02.kr-central-1.c.internal | Ready  | control-plane,master | 4m36s | v1.21.0 |
| osk-master-03.kr-central-1.c.internal | Ready  | control-plane,master | 2m29s | v1.21.0 |
| osk-worker-01.kr-central-1.c.internal | Ready  | <none>               | 32s   | v1.21.0 |
| osk-worker-02.kr-central-1.c.internal | Ready  | <none>               | 58s   | v1.21.0 |

```
[root@osk-master-01 ~]#
```



KAKAO FRIENDS

# 설치 완료 후, Kubectl?

- Kubectl 은 쿠버네티스 클러스터를 제어하기 위한 커맨드 라인 도구
  - ✓ kubectl [command] [TYPE] [NAME] [flags]
    - command: 명령을 하려는 ‘동사’. 예: create, get, describe, delete
    - TYPE: 리소스 타입
    - NAME: 리소스 이름
    - flags: 선택적 옵션
  - ✓ kubectl help를 사용하여 command를 확인 가능 (`-- help`)
  - ✓ kubectl에서 자주사용하는 output flag는 -o wide, -o yaml, -o json, --sort-by=<jsonpath\_exp> 등  
`--dry-run=client -o yaml > filename.yaml`

```
osk@osk-MacBook-Pro ~ % kubectl get nodes -o wide
osk@osk-MacBook-Pro ~ %
osk@osk-MacBook-Pro ~ % kubectl run test --image=nginx --dry-run=client -o yaml > 1.yaml
```

# 설치 완료 후, 인증서

- Preflight-check가 완료되면 kubeadm은 CA(자체 인증) 파일과 키를 생성

```
[root@osk-master-01 pki]# ll /etc/kubernetes/pki
total 56
-rw-r--r--. 1 root root 1326 Aug 14 11:29 apiserver.crt
-rw-r--r--. 1 root root 1155 Aug 14 11:29 apiserver-etcd-client.crt
-rw-----. 1 root root 1675 Aug 14 11:29 apiserver-etcd-client.key
-rw-----. 1 root root 1679 Aug 14 11:29 apiserver.key
-rw-r--r--. 1 root root 1164 Aug 14 11:29 apiserver-kubelet-client.crt
-rw-----. 1 root root 1679 Aug 14 11:29 apiserver-kubelet-client.key
-rw-r--r--. 1 root root 1066 Aug 14 11:29 ca.crt
-rw-----. 1 root root 1675 Aug 14 11:29 ca.key
drwxr-xr-x. 2 root root 162 Aug 14 11:29 etcd
-rw-r--r--. 1 root root 1078 Aug 14 11:29 front-proxy-ca.crt
-rw-----. 1 root root 1679 Aug 14 11:29 front-proxy-ca.key
-rw-r--r--. 1 root root 1119 Aug 14 11:29 front-proxy-client.crt
-rw-----. 1 root root 1679 Aug 14 11:29 front-proxy-client.key
-rw-----. 1 root root 1679 Aug 14 11:29 sa.key
-rw-----. 1 root root 451 Aug 14 11:29 sa.pub
```

→ public key infrastructure

→ .crt : 서버 인증서

→ .key : 서버 개인키

# 설치 완료 후, 팁

- 자동 완성 설정 : <https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/#install-bash-completion>

## ✓ Bash

```
sudo yum install -y bash-completion
source /usr/share/bash-completion/bash_completion # bash-completion 패키지를 먼저 설치 후, bash의 자동 완성 셸에 설정
echo "source <(kubectl completion bash)" >> ~/.bashrc    # 자동 완성을 bash 셸에 영구적으로 추가
kubectl completion bash >/etc/bash_completion.d/kubectl # root권한으로 실행
```

## ✓ zsh(맥북)

```
source <(kubectl completion zsh)                                # 현재 셸에 zsh의 자동 완성 설정
echo "[[ $commands[kubectl] ]] && source <(kubectl completion zsh)" >> ~/.zshrc # 자동 완성을 zsh 셸에 영구적으로 추가한다.
```

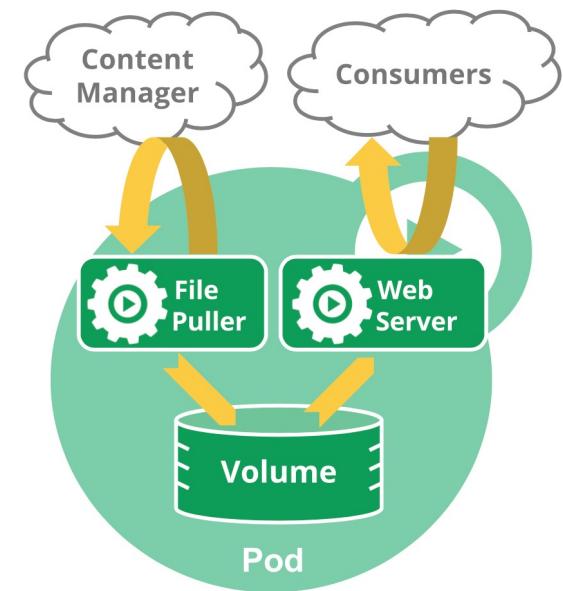
\* 맥에서 complete:13 에러 발생시, .zshrc 내부에 아래 추가

```
autoload -Uz compinit
```

```
compinit -i
```

# 쿠버네티스 리소스 : 워크로드 오브젝트

- POD : (고래 떼를 일컫음. 도커의 고래에서 유래) 하나 이상의 컨테이너로 구성
  - 스케일링의 단위, 어플리케이션에 친숙(환경변수 / 정상 여부 상태검사 정의 등이 용이)
- 1개 파드에 2개 이상의 각각 다른 이미지 가진 컨테이너가 가능함
- 파드 리소스는 노드 IP와 별개로 파드 만의 고유한 IP를 할당받으며 파드 내의 컨테이너들은 IP를 공유함
- 파드 내의 컨테이너들은 동일한 볼륨과 연결이 가능
- 파드는 배포의 최소단위이며 특정 네임스페이스에 실행됨



```
osk@osk-MacBook-Pro lkln-kakao % kubectl run test --image=nginx --dry-run=client -o yaml > 1.yaml
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl get pod
```

```
No resources found in default namespace.
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl run test --image=nginx
```

```
pod/test created
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl get pod
```

| NAME | READY | STATUS            | RESTARTS | AGE |
|------|-------|-------------------|----------|-----|
| test | 0/1   | ContainerCreating | 0        | 4s  |

```
osk@osk-MacBook-Pro lkln-kakao % kubectl delete pod test
```

```
pod "test" deleted
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl apply -f 1.yaml
```

```
pod/test created
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl get pod
```

| NAME | READY | STATUS  | RESTARTS | AGE |
|------|-------|---------|----------|-----|
| test | 1/1   | Running | 0        | 20s |

1) kubectl 명령어 활용  
2) yaml 파일 생성 후 apply로 적용

# 쿠버네티스 리소스 : 워크로드 오브젝트

- POD 명령어

```
osk@osk-MacBook-Pro lkln-kakao % kubectl get pod -o wide
```

| NAME | READY | STATUS  | RESTARTS | AGE | IP           | NODE                        | NOMINATED NODE | READINESS GATES |
|------|-------|---------|----------|-----|--------------|-----------------------------|----------------|-----------------|
| test | 1/1   | Running | 0        | 73m | 10.240.32.66 | osk-test-lkln-5c439d5-gm5hf | <none>         | <none>          |

```
osk@osk-MacBook-Pro lkln-kakao % kubectl describe pod test
```

```
Name:           test
Namespace:      default
Priority:       0
Node:          osk-test-lkln-5c439d5-gm5hf/172.30.6.194
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl logs test
```

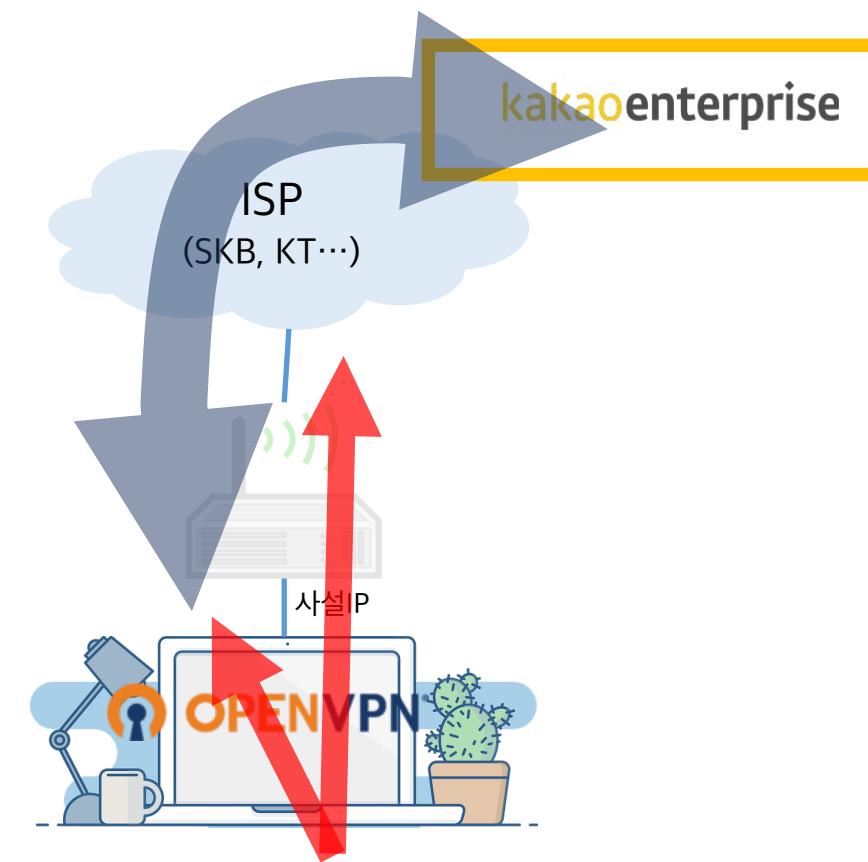
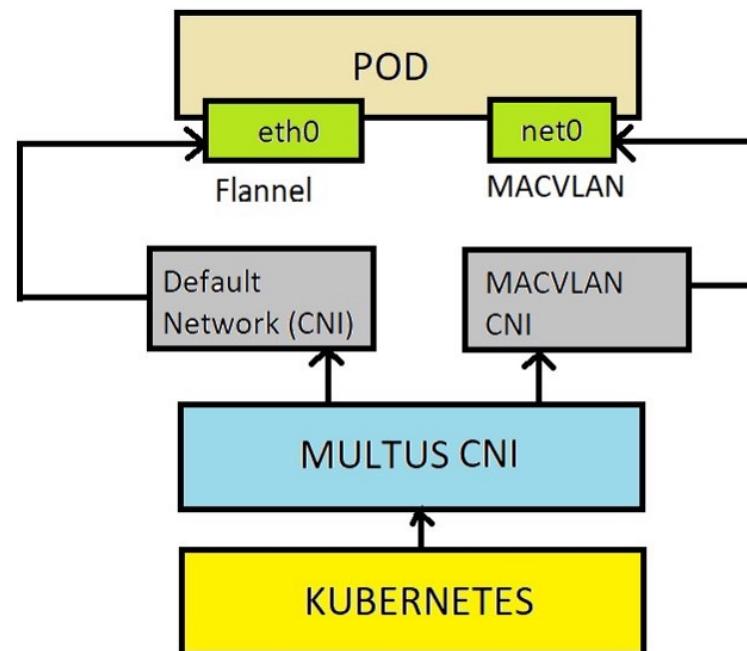
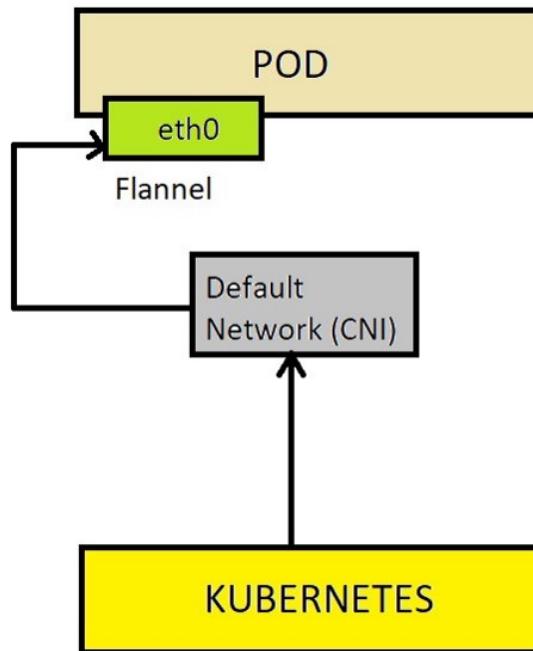
```
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl exec -it test -- /bin/sh
```

```
# hostname
test
```

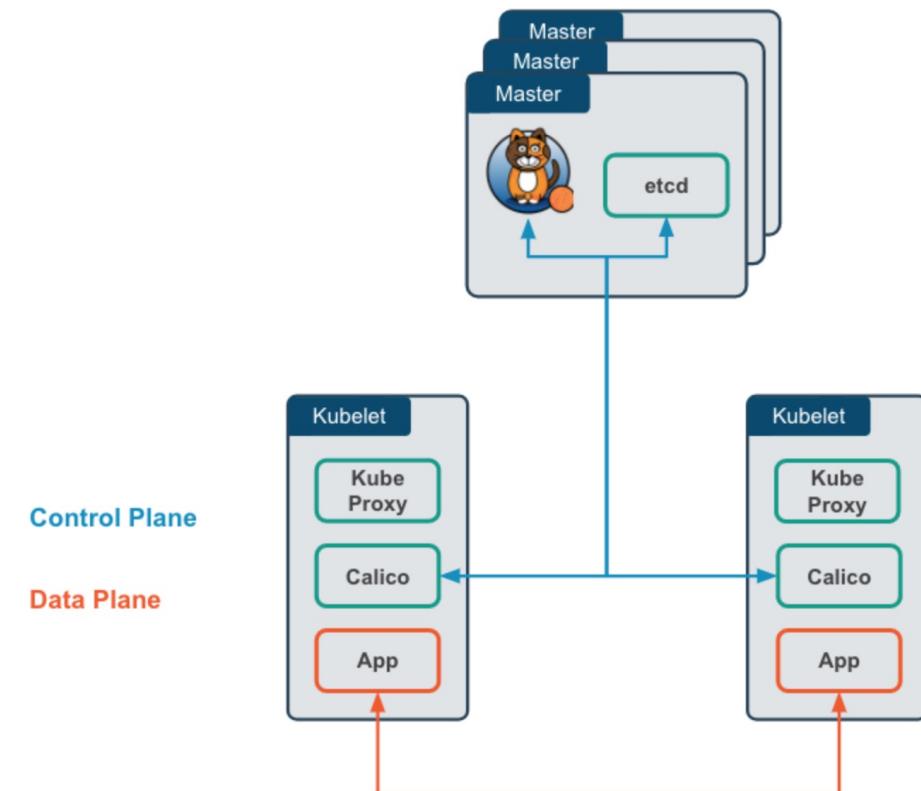
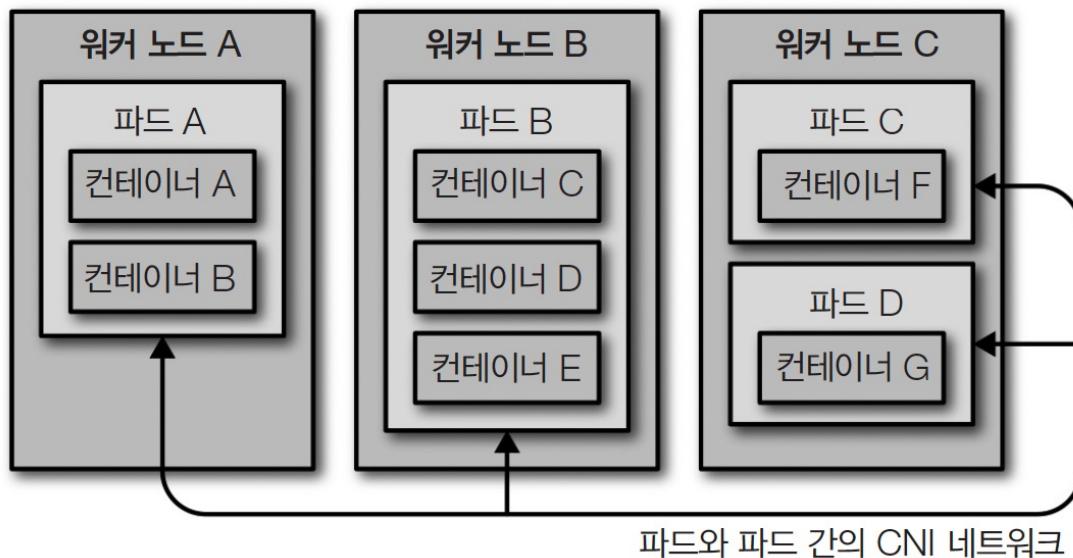
# 쿠버네티스 리소스 : 워크로드 오브젝트

- POD는 1개의 IP만 가짐 (CNI 플러그인이 할당)  
(참고) Multus 를 활용시 POD에 2개 네트워크도 연결 가능
- 2개 인터페이스 활용 사례



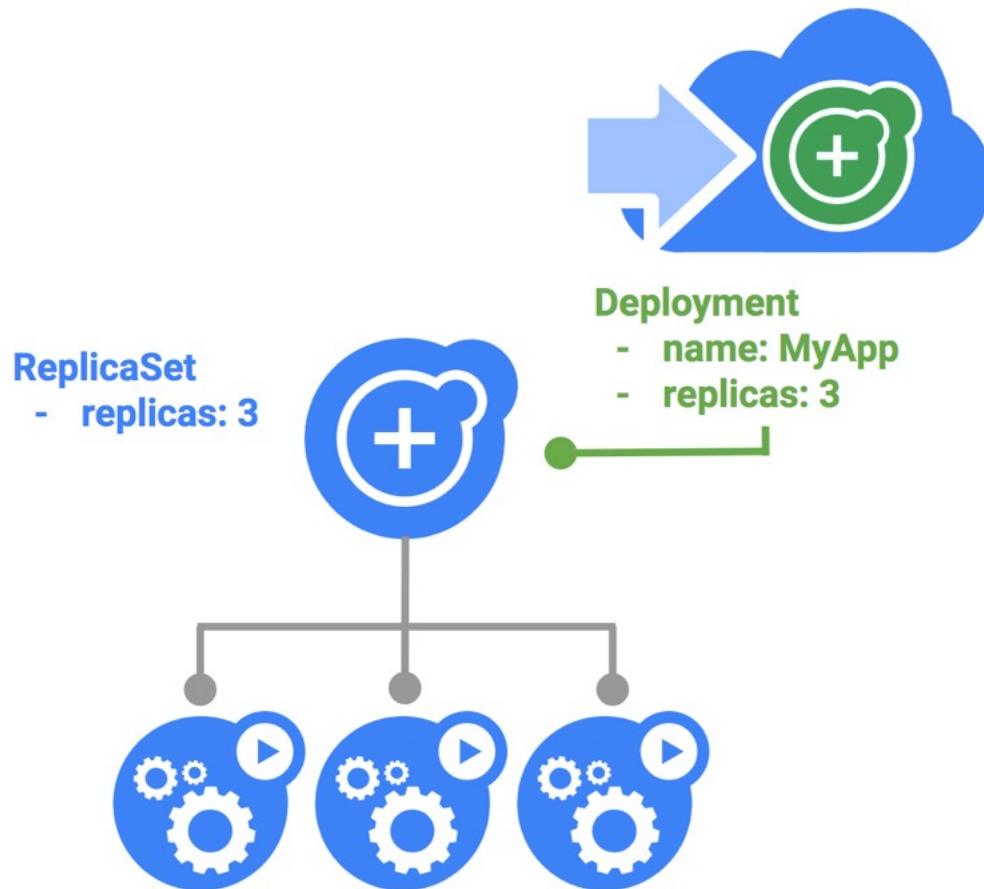
# 쿠버네티스 리소스 : 워크로드 오브젝트

- 컨테이너 네트워크 인터페이스(CNI)  
네트워크로 연결될 파드는 동일 노드에, 다른 노드에 있을 수도 있음. CNI의 역할은 단순히 파드간 연결을 용이하게 만드는 것
- 컨테이너 런타임(예:도커)은 CNI 플러그인 실행파일(예:칼리코)을 호출하여 컨테이너의 네트워킹 네임스페이스에 인터페이스를 추가/제거



# 쿠버네티스 리소스 : 컨트롤러 오브젝트

- ReplicaSet : (Node 고장 / Pod 삭제 등 발생 시) 파드 복제하여 개수 유지
- Deployment : 구 버전에서 신 버전으로 복제, 레플리카셋 관리(보통 pod를 일일이 관리하기보다는 deployment 를 서비스 단위로 관리)



# 쿠버네티스 리소스 : 컨트롤러 오브젝트

- Deployment 생성시 Replica 숫자 지정. 파드를 지워도 ReplicaSet에 의해 다시 재생성

```
osk@osk-MacBook-Pro lkln-kakao % kubectl create deployment deploy-test --image nginx --replicas=3
deployment.apps/deploy-test created
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl get deployments.apps deploy-test -o wide
```

| NAME        | READY | UP-TO-DATE | AVAILABLE | AGE | CONTAINERS | IMAGES | SELECTOR        |
|-------------|-------|------------|-----------|-----|------------|--------|-----------------|
| deploy-test | 1/3   | 3          | 1         | 12s | nginx      | nginx  | app=deploy-test |

```
osk@osk-MacBook-Pro lkln-kakao %
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl get pod
```

| NAME                         | READY | STATUS  | RESTARTS | AGE |
|------------------------------|-------|---------|----------|-----|
| deploy-test-76bc8c5457-5xjsm | 1/1   | Running | 0        | 47s |
| deploy-test-76bc8c5457-145sb | 1/1   | Running | 0        | 47s |
| deploy-test-76bc8c5457-lppr7 | 1/1   | Running | 0        | 47s |
| test                         | 1/1   | Running | 0        | 12m |

```
osk@osk-MacBook-Pro lkln-kakao %
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl get replicasets.apps
```

| NAME                   | DESIRED | CURRENT | READY | AGE |
|------------------------|---------|---------|-------|-----|
| deploy-test-76bc8c5457 | 3       | 3       | 3     | 54s |

```
osk@osk-MacBook-Pro lkln-kakao %
```

```
osk@osk-MacBook-Pro lkln-kakao % kubectl delete pod deploy-test-76bc8c5457-5xjsm
```

```
pod "deploy-test-76bc8c5457-5xjsm" deleted
```

```
osk@osk-MacBook-Pro lkln-kakao %
```

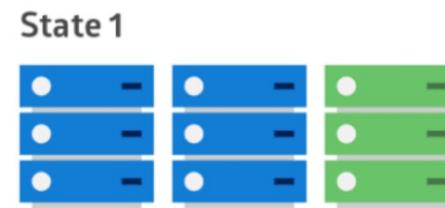
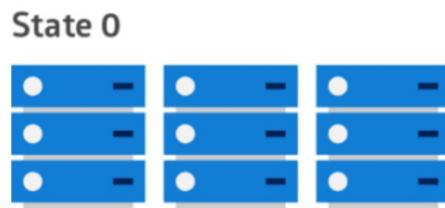
```
osk@osk-MacBook-Pro lkln-kakao % kubectl get pod
```

| NAME                         | READY | STATUS  | RESTARTS | AGE   |
|------------------------------|-------|---------|----------|-------|
| deploy-test-76bc8c5457-145sb | 1/1   | Running | 0        | 7m45s |
| deploy-test-76bc8c5457-lppr7 | 1/1   | Running | 0        | 7m45s |
| deploy-test-76bc8c5457-wgpn6 | 1/1   | Running | 0        | 17s   |
| test                         | 1/1   | Running | 0        | 19m   |

# 쿠버네티스 리소스 : (참고) App 업데이트 방법

- 롤링 업데이트

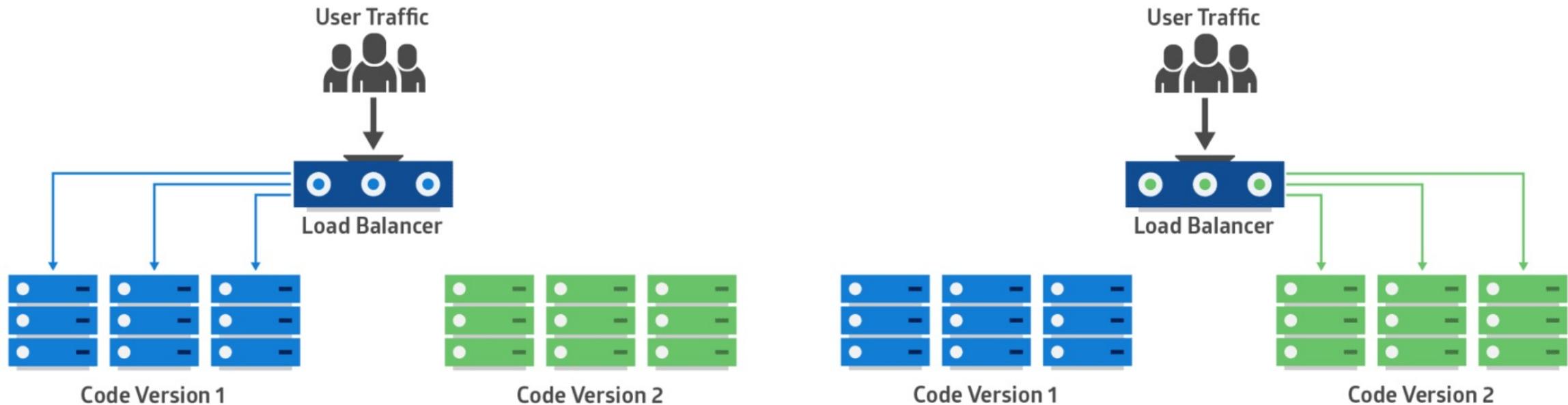
- ✓ 장점 : Down Time 없음. 추가 인프라 투자비 별로 없음.
- ✓ 단점 : 신/구 버전 공존에 따른 개발 및 검증 사항 고려 필요. 신/구 버전이 같이 활용하는 인프라에서 문제 발생 가능



```
osk@osk-MacBook-Pro ~ % kubectl describe deployments.apps  
Name:          deploy-test  
Namespace:     default  
Labels:        app=deploy-test  
Annotations:   deployment.kubernetes.io/revision: 1  
Selector:      app=deploy-test  
StrategyType:  RollingUpdate  
MinReadySeconds: 0  
RollingUpdateStrategy: 25% max unavailable, 25% max surge
```

# 쿠버네티스 리소스 : (참고) App 업데이트 방법

- 블루그린 업데이트 방식 ( 레드블랙 업데이트 / AB 배포 )
  - ✓ 장점 : Down Time 없음. 복구가 빠름
  - ✓ 단점 : 인프라 투자가 2배로 필요. 로드 밸런서 추가 필요

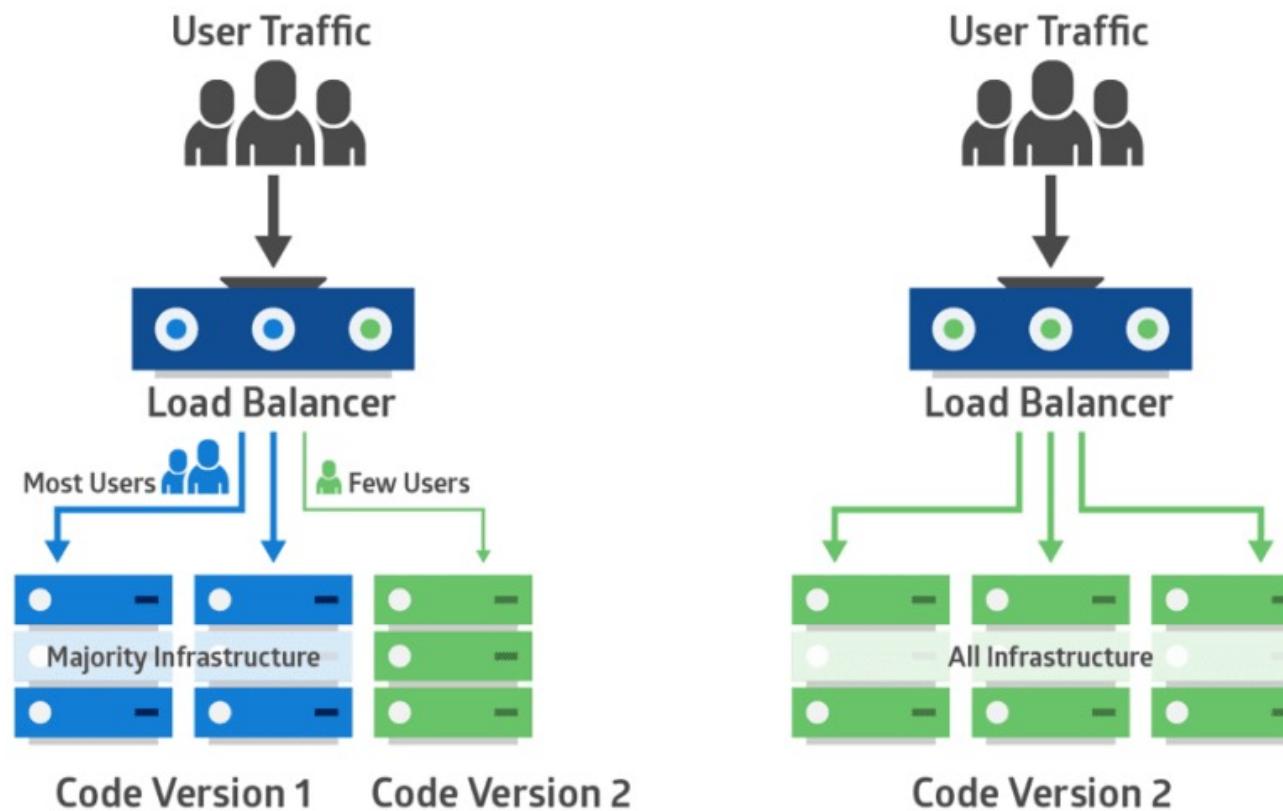


# 쿠버네티스 리소스 : (참고) App 업데이트 방법

- 카나리 업데이트 방식

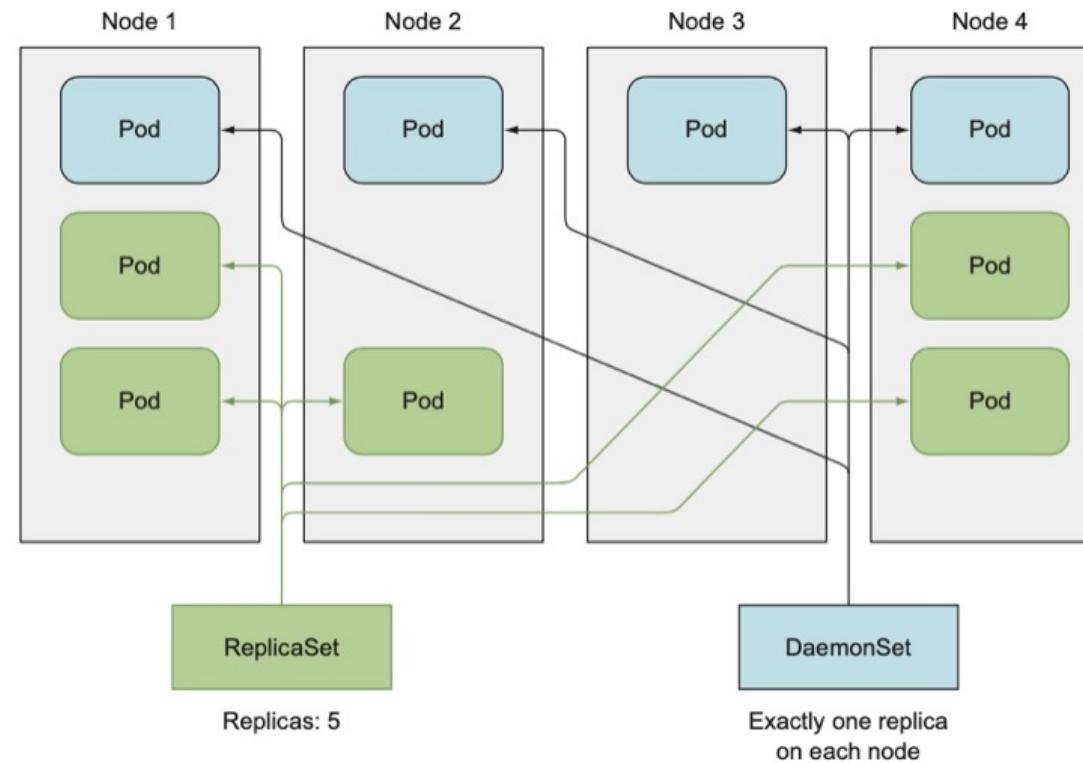
- ✓ 장점 : Downtime 없음. 아주 적은 사용자만 새 버전에 유입시키므로 영향도 최소화, 추가 인프라 투자 별로 없음

- ✓ 단점 : 신/구 버전 공존에 따른 개발 및 검증 사항 고려 필요. 업데이트 시간이 롤링 업데이트보다 더 걸림. 로드 밸런서 추가 필요



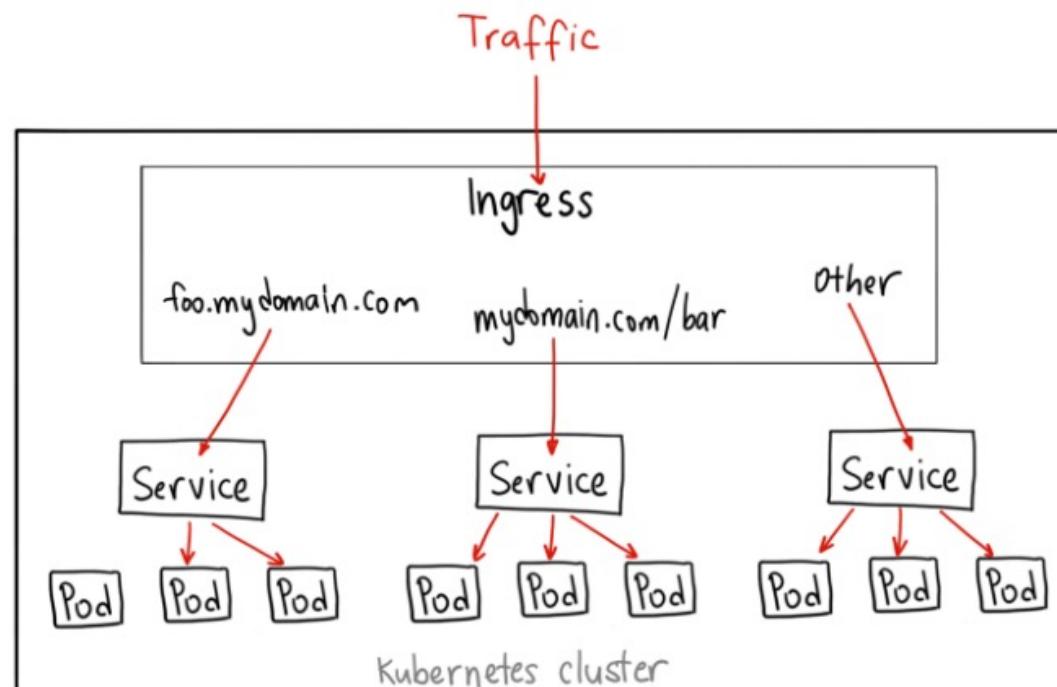
# 쿠버네티스 리소스 : 컨트롤러 오브젝트

- DaemonSet
  - ✓ 모든 노드에 동일한 파드를 실행시키고 싶은 경우 활용
  - ✓ 리소스 모니터링, 로그 수집기 등에 유용. 클러스터에 노드가 추가/삭제 되면 자동으로 파드도 생성/삭제됨



# 쿠버네티스 리소스 : 로드밸런서 오브젝트

- Service : POD는 임시적인(ephemeral) 오브젝트. TCP/UDP 로드밸런싱을 담당하는 추상적 개념
  - 파드는 생성시마다 IP가 변하기 때문에, 파드 외부와 통신시 고정 IP를 갖는 'Service'를 통하여 서비스
  - kubectl create service 명령어도 서비스 생성이 가능하지만, kubectl expose 명령어를 활용하면 생성 ~ 연결까지 한번에 가능
  - 종류 : Cluster IP(기본값), Node Port, LoadBalancer, ExternalName으로 구분
- Ingress : HTTP 로드밸런서. Service로 라우팅 가능

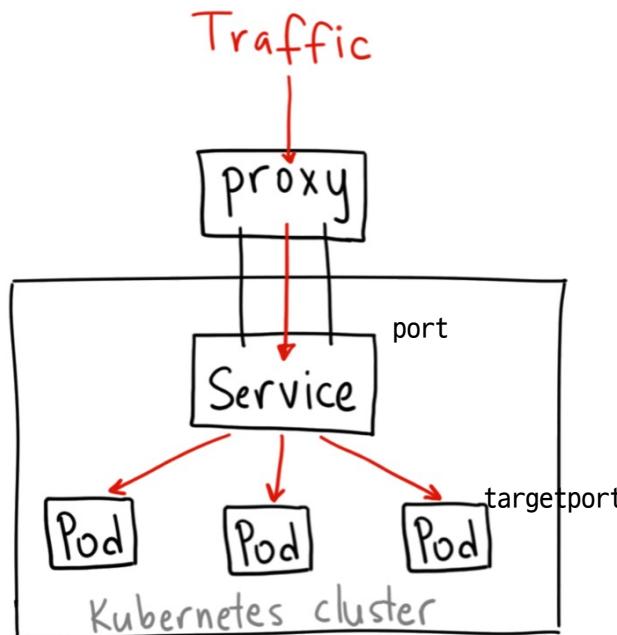


# 쿠버네티스 리소스 : 로드밸런서 오브젝트

- Service 종류 : Cluster IP(기본값), Node Port, LoadBalancer, ExternalName으로 구분

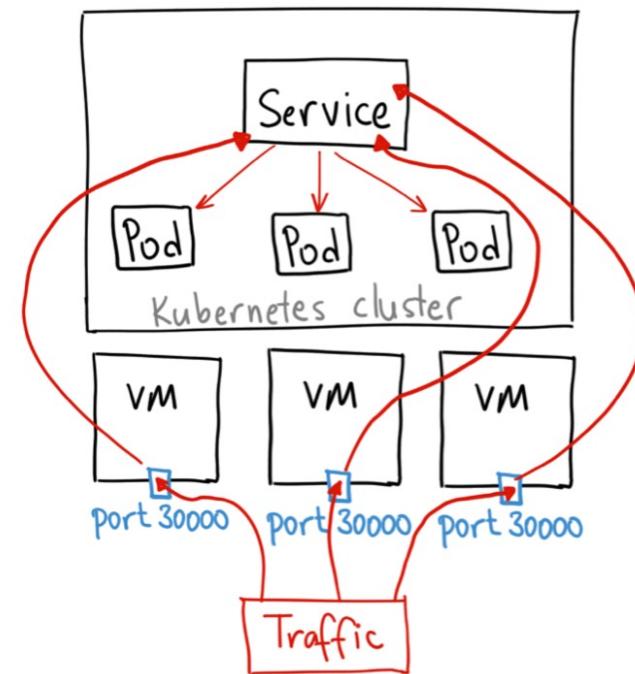
## [Cluster IP]

- 쿠버네티스 클러스터 내에서만 통신 가능
- 외부와 통신이 필요하면 별도 프록시를 두어야함



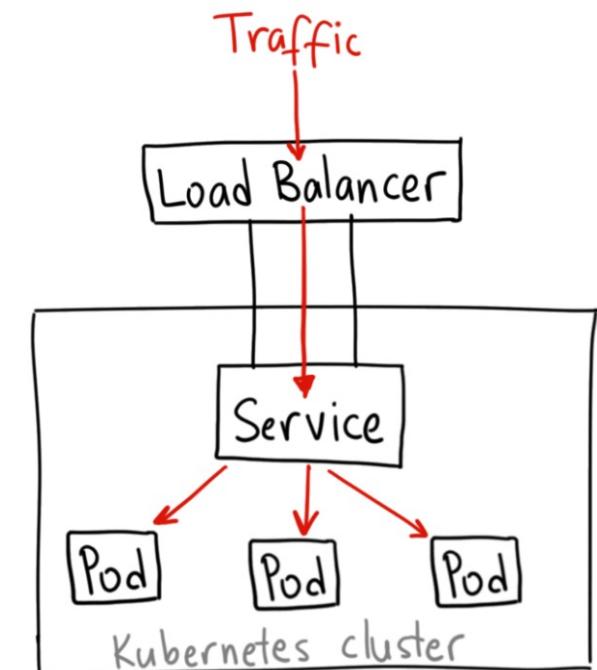
## [Node Port]

- 어떤 노드더라도, '특정 포트'로 들어오게 된다면 서비스가 알맞은 파드로 로드밸런싱 해줌
- 노드에 해당 Pod가 있든 없든 모든 노드 전체에 해당



## [LoadBalancer]

- 각각의 노드에 트래픽 분산처리 가능
- 단일 IP로 여러 노드에 로드밸런싱 가능  
(현재 카카오 클라우드 내에서는 사설IP로만 가능)

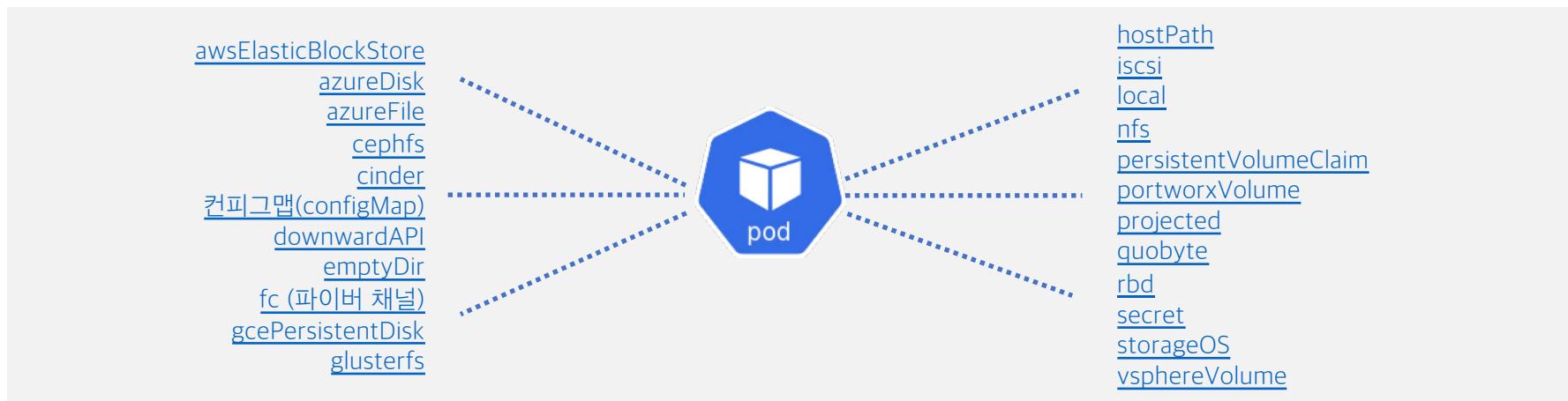


[ExternalName] 외부 서비스를 쿠버네티스 내부에서 호출 시 활용

그림 출처 : Medium, Sandeep Dinesh

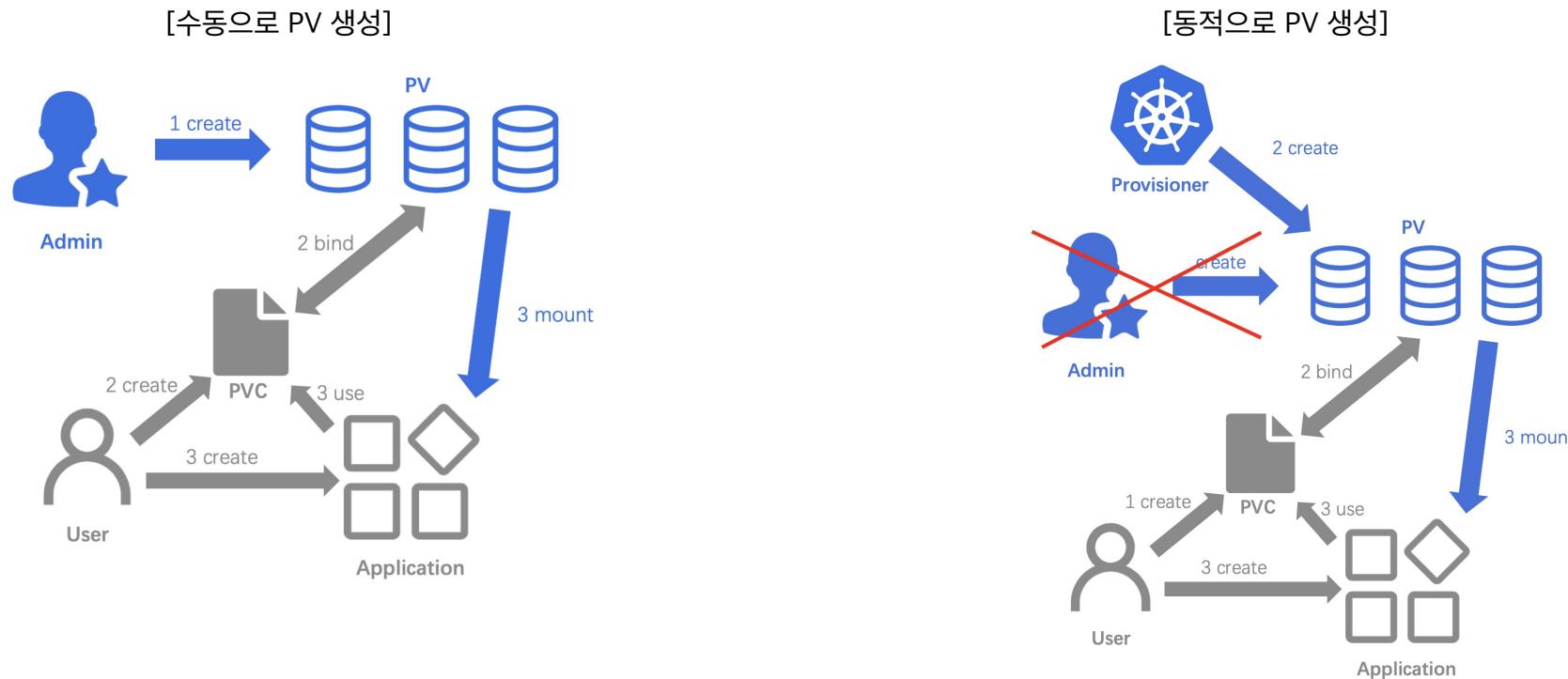
# 쿠버네티스 리소스 : 스토리지 오브젝트

- 컨테이너 내의 디스크에 있는 파일은 임시적. 퍼시스턴트 볼륨은 지속 존재.



# 쿠버네티스 리소스 : 스토리지 오브젝트(PV, PVC)

- Persistent Volume(PV) : 관리자가 프로비저닝하거나 스토리지 클래스를 사용하여 동적으로 프로비저닝한 클러스터의 스토리지
- Persistent Volume Claim(PVC) : 사용자의 스토리지에 대한 요청. 파드와 비슷.
  - ✓ 파드 : 노드 리소스를 사용 / POD 명세서 내 특정 수준의 리소스(CPU 및 메모리)를 요청
  - ✓ PVC : PV 리소스를 사용 / 클레임 명세서 내 특정 크기 및 접근 모드를 요청(예: ReadWriteOnce, ReadOnlyMany, ReadWriteMany)



## [접근/권한] APIs and Access

# Label과 Annotation

- 레이블 : 오브젝트를 식별하는데 도움이 되는 문자열 키/쌍 (쿼리 가능)
- 어노테이션 : 단순 주석. 모든 API오브젝트는 주석 포함 가능(쿼리 불가)
  - 쿠버네티스의 실험적인 기능
  - 제작사별 특이한 기능. 메타데이터로 가능하므로 그래픽 아이콘도 가능.

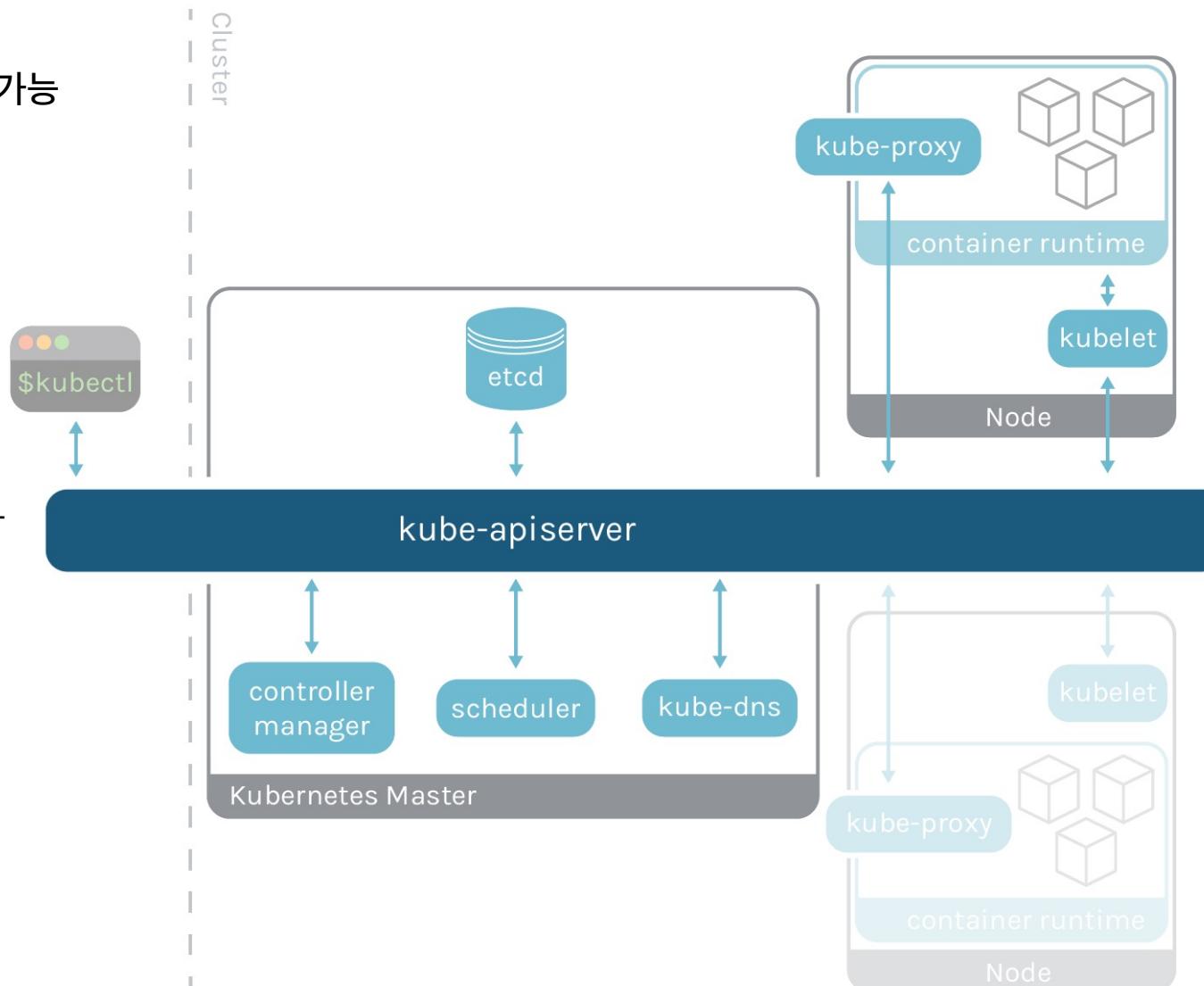
```
[centos@osk-master-01 .kube]$ kubectl describe pod -n kube-system calico-kube-con
Name:           calico-kube-controllers-58497c65d5-wnfnv
Namespace:      kube-system
Priority:      2000000000
Priority Class Name: system-cluster-critical
Node:          osk-master-01.kr-central-1.c.internal/172.30.5.193
Start Time:    Sat, 14 Aug 2021 11:43:38 +0000
Labels:        k8s-app=calico-kube-controllers
               pod-template-hash=58497c65d5
Annotations:   cni.projectcalico.org/containerID: d315adb5bb72f128f74dac3663e481356be6ae4ca58efc9820a2f47a14de0583
               cni.projectcalico.org/podIP: 192.168.205.193/32
               cni.projectcalico.org/podIPs: 192.168.205.193/32
Status:        Running
```

# API 접근

- API 서버 : 중앙 접근 포인트. Stateless이며(etcd 활용), 복제 가능
- 주요 기능

- 1) API 관리 : 서버에서 API를 노출하고 관리하는 프로세스
- 2) 요청 처리 : 클라이언트의 개별 API 요청을 처리하는 기능
  - 대부분 HTTP 형태로 요청, 콘텐츠는 JSON 기반이 많음
  - 요청의 유형 예 : GET / LIST / POST / DELETE

<https://kubernetes.io/ko/docs/reference/access-authn-authz/authorization/>
- 3) 내부 제어 루프 : API 작동에 필요한 백그라운드 작업을 담당  
(대부분은 컨트롤러 매니저에서 수행)



# V1 Group API / API 리소스

- API 그룹은 쿠버네티스 API를 더 쉽게 확장하게 설계됨  
API 그룹은 REST 경로와 오브젝트의 apiVersion 필드에 명시
- 쿠버네티스에는 두 종류의 API 그룹 존재

|                       | REST 경로                      | apiVersion                         |
|-----------------------|------------------------------|------------------------------------|
| V1 group(Core/Legacy) | /api/v1                      | apiVersion: v1                     |
| 이름 있는 그룹(후속)          | /apis/\$GROUP_NAME/\$VERSION | apiVersion: \$GROUP_NAME/\$VERSION |

- API 리소스 탐방  
<https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.22>
- (참고) API 버전 : alpha=불안정하고 상용환경에 부적합,  
beta=안정적이나 최종 개선 예정,  
GA(General Availability)=안정적

# 인증서

- 쿠버네티스에서는 다양한 인증방법이 있으나, 대부분 PKI(공개 키 인프라) 기반으로 상호 인증
- PKI? 그 전에 대칭키/비대칭키 이론 학습
  - 대칭 키 : 암호화와 복호화에 같은 암호 키(비밀 키만 있음)를 쓰는 알고리즘  
(군대 암구호. 서로 동일하게 알고 있는 키)
  - 비대칭 키 : 공개 키와 비밀 키가 존재하며, 공개 키는 누구나 알 수 있지만  
그에 대응하는 비밀 키는 키의 소유자만이 가지고 있음  
(공개 키 : 촛농, 비밀 키 : 물감)  
→ 역으로 생각하면 물감이 있으면 비밀 키 소유자가 맞겠구나
- PKI : 공개된 키를 개인이나 집단을 대표하는 믿을 수 있는 주체와 엮는 것이며  
이는 인증 기관(Certificate authority, 이하 CA)의 등록/인증 발행을 통해 성립  
(성근이가 지은이에게 편지를 보내는데.. 빨간 물감으로, 성근이는 지은이 좋아해)  
(지은이를 사모하던 나쁜남자가 편지 슬쩍하더니, 노란 물감으로, 성근이는 지은이 싫어)  
(→ 선생님 등판, '성근이는 빨간 물감이야' 선생님 인증서 + 빨간 물감으로, 나 너 좋아해)

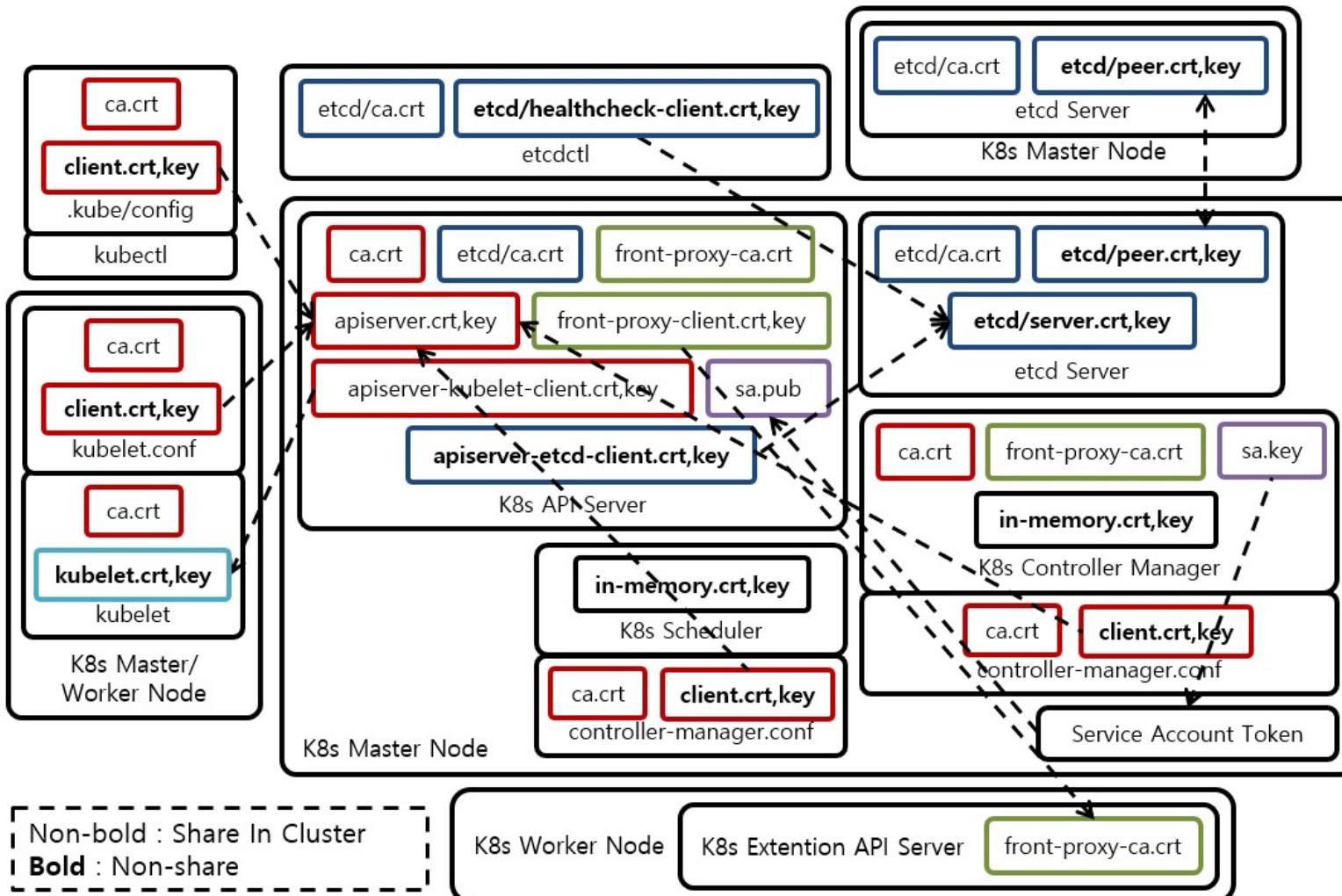
.crt 파일

.key 파일



# 인증서

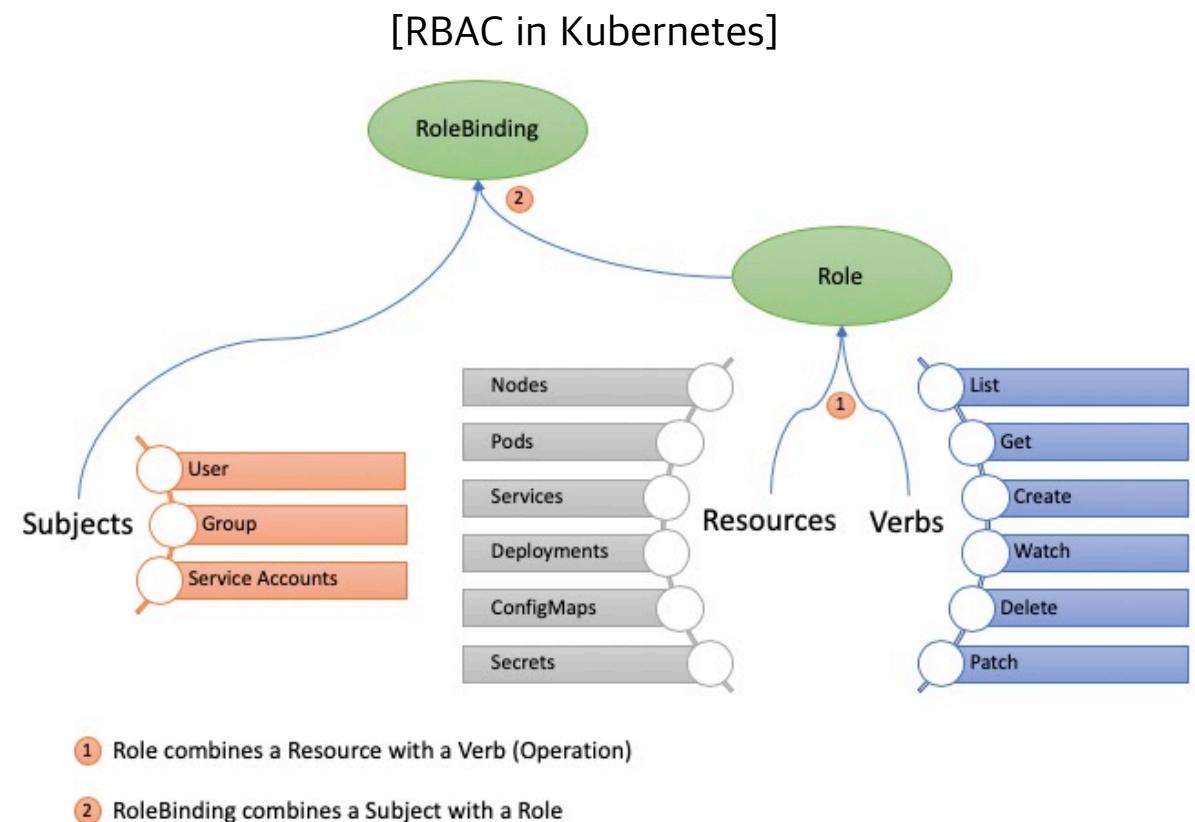
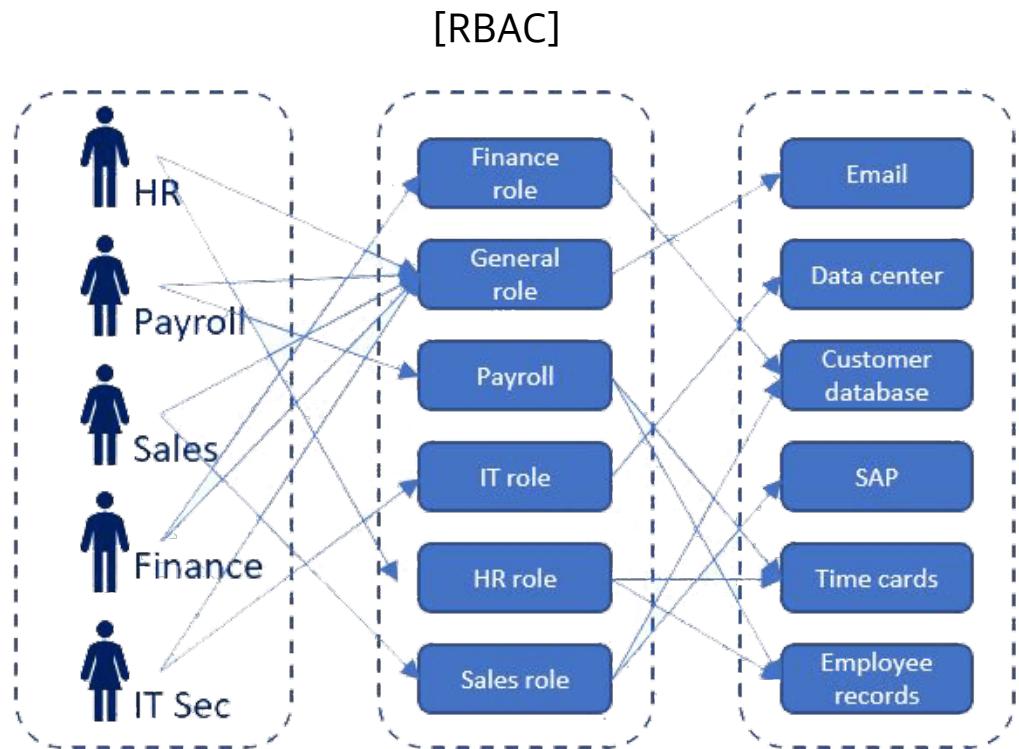
- 쿠버네티스 컨트롤 플레인에서 자체 CA 역할 수행



# RBAC(Role Based Access Control)

- 인증 모듈은 4개가 존재하며, 이중 RBAC이 가장 대표적이고 효과적인 방식

<https://kubernetes.io/docs/reference/access-authn-authz/rbac/>



# Role, Cluster Role

- 작업 수행에 대한 권한. ‘어떤 resource에 어떤 verb 권한을?’
- 차이 : kind 에 들어가는 종류명, namespace 존재 여부 (범위만 다름)
- Cluster Role 사용 예시 : 클러스터 관리자, 쿠버네티스 컨트롤러

[Role]

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
- apiGroups: ["extensions", "apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "watch", "create", "update",
  "patch", "delete"]
```

[Cluster Role]

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata: # 클러스터 룰이므로 "네임스페이스" 는 생략됨
  name: secret-reader
rules:
- apiGroups: [""]
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]
- apiGroups: ["extensions", "apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "watch", "create", "update",
  "patch", "delete"]
```

# RoleBinding, ClusterRoleBinding

- Role을 사용자/그룹/Service Account에 연결. ‘어떤 resource에 어떤 verb 권한을,’ + ‘누구에게 줄 것인가?’
- 차이 : kind에 들어가는 종류명, namespace 존재 여부 (범위만 다름)
- Cluster Role 사용 예시 : 클러스터 관리자, 쿠버네티스 컨트롤러

[Role]

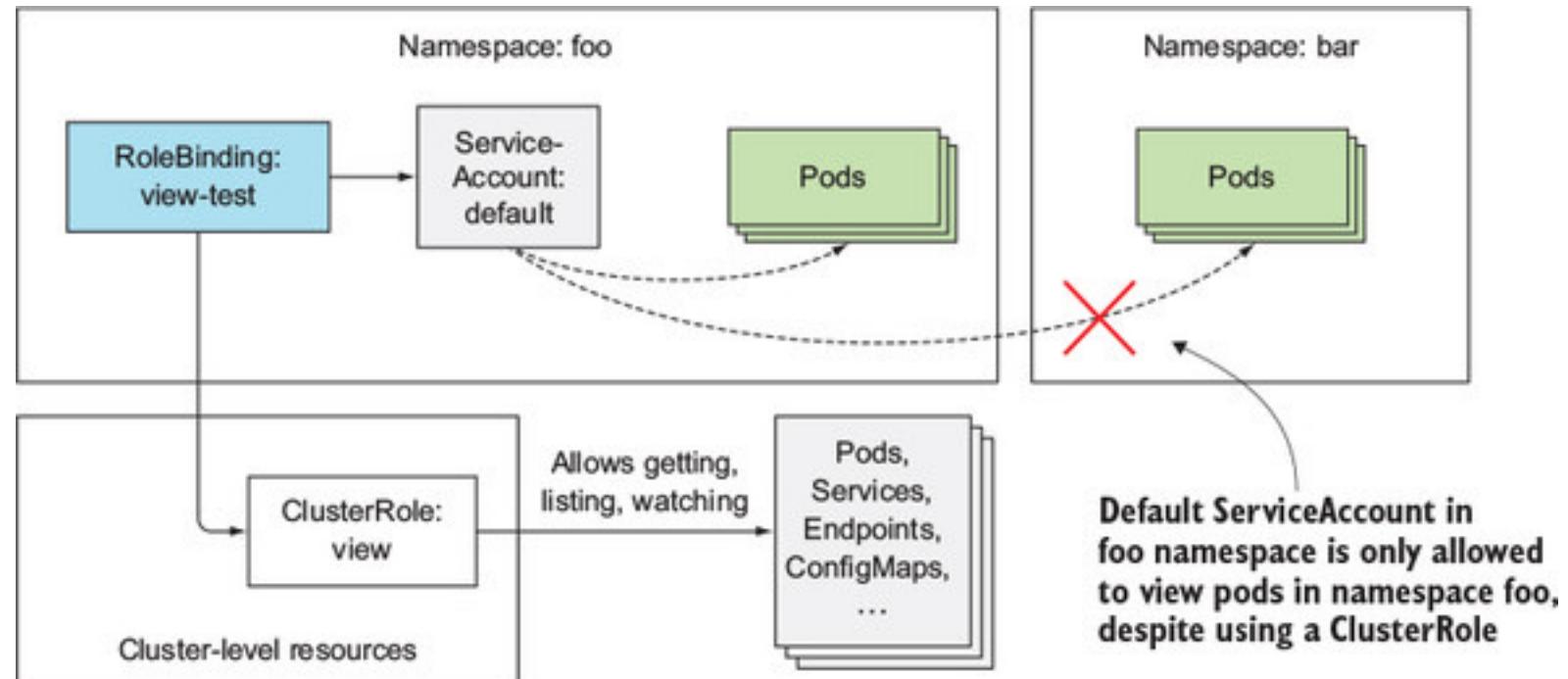
```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: jane
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

[Cluster Role]

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

# Service Account

- 파드 내부에서 실행하는 프로세스가 쿠버네티스 API를 호출하고 싶다면? Service Account 활용  
# kubectl create serviceaccount test



# [Compute] Managing State With Deployments, Scheduling

# Pod / Replicaset 기본 실습

- Simple Pod 생성

```
# kubectl run test --image=nginx
```

- 생성한 pod 확인

```
# kubectl get pod test
```

```
# kubectl describe pod test
```

- Replicaset 확인/생성

```
# kubectl get replicsets
```

```
# kubectl create replicaset test ~
```

- Pod/Replicaset 삭제

```
# kubectl delete pod / replicaset ~
```

- Replicaset을 edit 하여 pod 숫자 확인

```
# kubectl edit replicaset ~
```



꾸준한 연습만이 합격의 지름길  
익숙해져야 합니다

# Deployment 기본 실습

- Deployment 생성 및 수정

```
[centos@osk-master-01 ~]$ kubectl create deployment nginx_deployment --image=nginx --dry-run=client -o yaml > deployment.yaml
[centos@osk-master-01 ~]$ vi deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx_deployment
    name: nginx_deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx_deployment
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx_deployment
    spec:
      containers:
        - image: nginx
          name: nginx
          resources: {}
status: {}
```

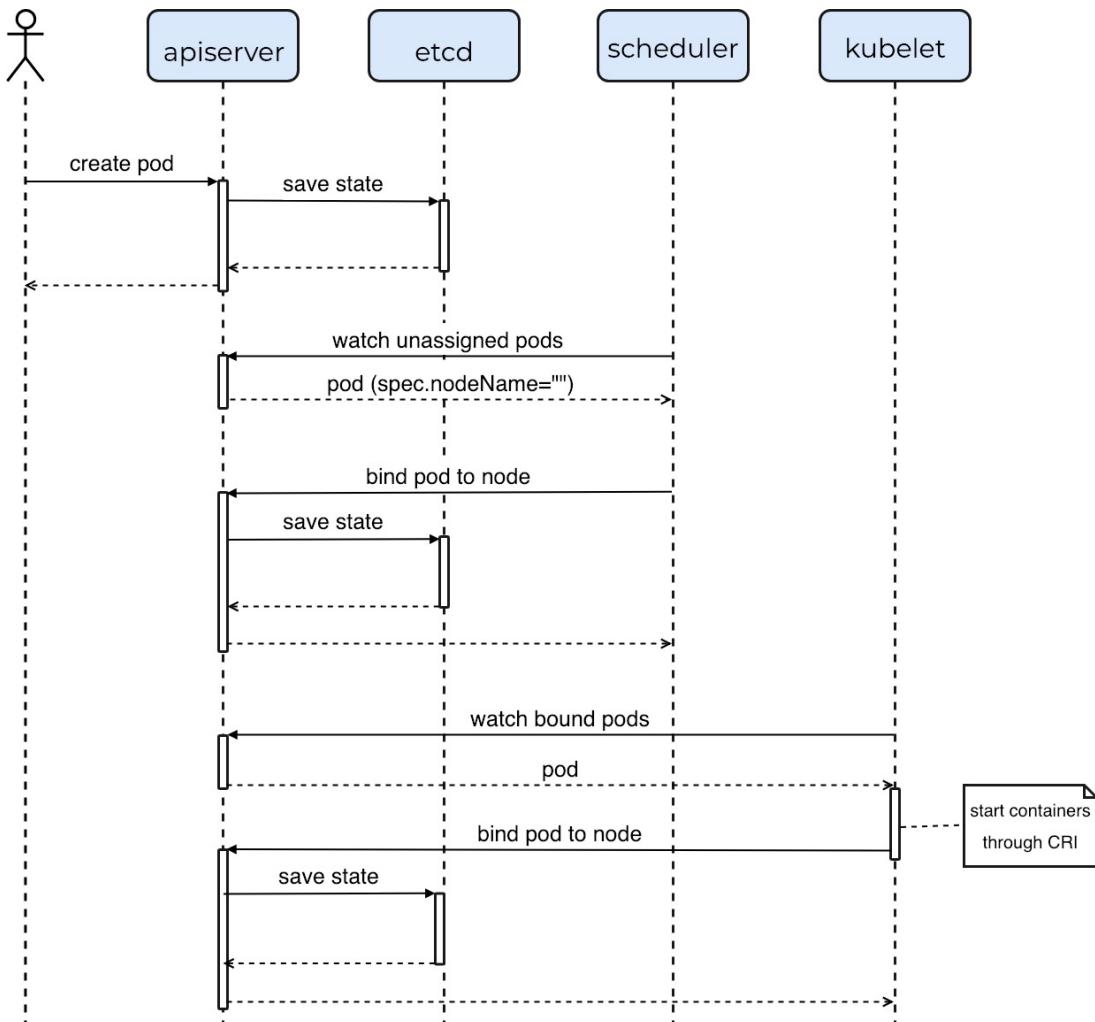


꾸준한 연습만이 합격의 지름길

익숙해져야 합니다

# Scheduler

- 컨테이너를 적절한 노드에 스케줄링(배치) 하는 역할(전용 바이너리)



- 스케줄러는 nodeName이 없는 파드가 있는지 API 서버를 지속 감시
- 해당 파드를 적합한 노드로 선택하고, 파드의 nodeName 업데이트
- 해당 노드의 쿠블렛이 계속 API서버를 보다가 자기꺼면 실행

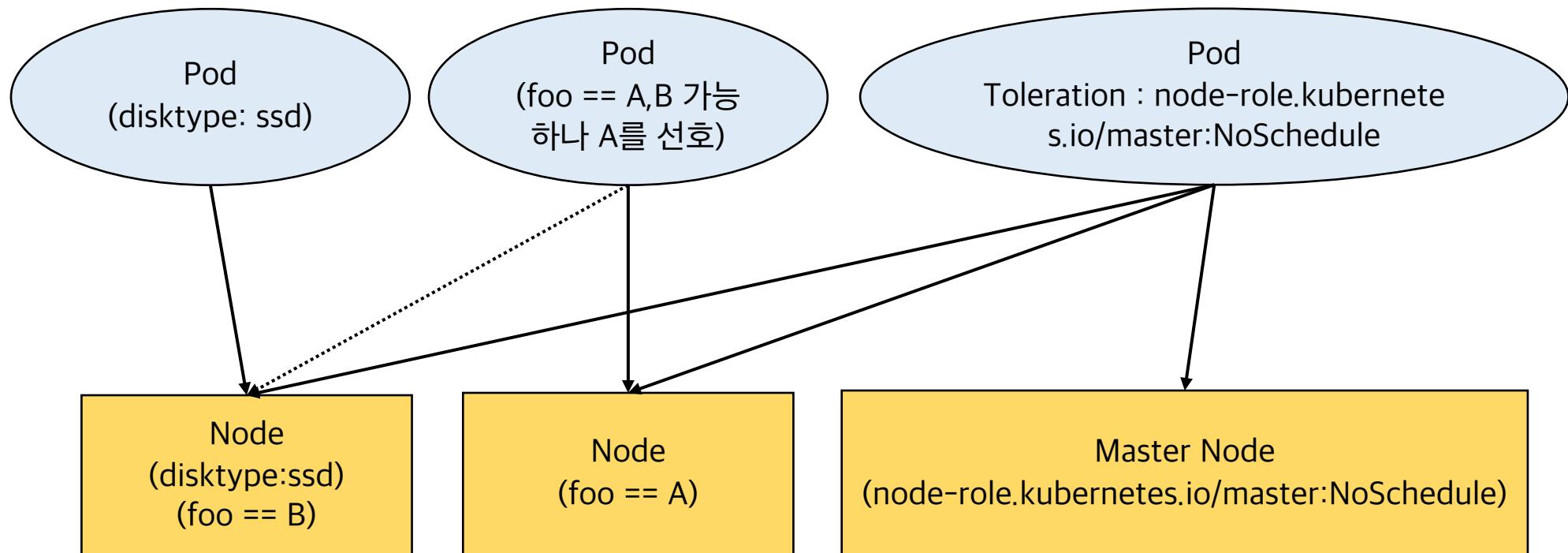
# Scheduling 프로세스

- 스케줄러가 파드를 배치하는 주요 기준
  - 1) 사전 조건
    - 파드가 특정 노드에 적합한지 확인.
    - ex) 파드에서 요청한 메모리 양을 노드가 감당할 수 있을지, 유저가 node selector로 지정한 경우 등
  - 2) 우선순위
    - 파드가 노드에서 실행할 수 있다고 가정 후, 상대적 가치를 판단
    - ex) 이미지가 이미 존재하는 노드에 가중치를 부여 : 빠른 시작 가능, 동일한 Service 내 있는 파드라면 spreading 시켜놓음

※ 스케줄링은 한 순간에만 최적이므로, 이후 충돌 등 여러 이유로 실행중인 파드를 이동할 수 도 있음. 이 경우 해당 pod를 삭제하고 재 생성

# Policy 정책

- 쿠버네티스 스케줄링과 다르게 운용자가 직접 스케줄 하고 싶다면 주로 아래 3가지 방안을 활용
  - 1) Node Selector
  - 2) Node Affinity
  - 3) Taint/Toleration



# Node Selector

- Node Selector
  - 노드에도 파드에도 레이블을 부여하여 동일한 조건에 맞게 스케줄 되도록 설정  
<https://kubernetes.io/ko/docs/concepts/scheduling-eviction/assign-pod-node/>  
(1) 노드에 레이블 붙이기. (예) #kubectl label nodes kubernetes-foo-node-1.c.a-robinson.internal disktype=ssd  
(2) 파드에 node selector 추가하기

```
spec:  
  containers:  
    nodeSelector:  
      disktype: ssd
```

# Node Affinity

- Node Selector는 선택에 대한 요구사항을 이야기하는 사전 조건
- 복잡한 선택에는 Affinity가 유연. (ex) 레이블 foo이면 A 또는 B 노드를 선택해주세요.

```
kind: Pod
...
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              # foo == A or B
              - key: foo
                operator: In
              values:
                - A
                - B
...
...
```

# Node Affinity

- Node Affinity
  - (ex) A, B 둘다 가능하지만, A로 라벨링 된 노드를 더 선호합니다.

```
kind: Pod
...
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              # foo == A or B
              - key: foo
                operator: In
                values:
                  - A
                  - B
            [operator]
            In : 해당 Value가 있는지?(키+쌍 모두)
            Exists : Key만 있으면 됨(키만 확인)
      preferredDuringSchedulingIgnoredDuringExecution:
        preference:
          - weight: 1
            matchExpressions:
              # foo == A
              - key: foo
                operator: In
                values:
                  - A
...
...
```

# Taint & Toleration

- Taint : 스케줄러가 pod를 배치할 때 Taint (오염)되어 있는 node들은 배치하지 않도록 함
- Toleration : Taint가 묻어 있어도 배치할 수 있도록 함
- Kubeadm은 설치 시, 자동으로 Controller Plane에 node-role.kubernetes.io/master 테인트로 제어하여 일반 파드와 분리

```
[root@osk-master-01 pki]# kubectl describe node osk-master-01
Name:           osk-master-01.kr-central-1.c.internal
Roles:          control-plane,master
Labels:         beta.kubernetes.io/arch=amd64
                beta.kubernetes.io/os=linux
                kubernetes.io/arch=amd64
                kubernetes.io/hostname=osk-master-01.kr-central-1.c.internal
                kubernetes.io/os=linux
                node-role.kubernetes.io/control-plane=
                node-role.kubernetes.io/master=
                node.kubernetes.io/exclude-from-external-load-balancers=
Annotations:   kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
                node.alpha.kubernetes.io/ttl: 0
                projectcalico.org/IPv4Address: 172.30.5.193/22
                projectcalico.org/IPv4IPIP TunnelAddr: 192.168.205.192
                volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp: Sat, 14 Aug 2021 11:30:14 +0000
Taints:         node-role.kubernetes.io/master:NoSchedule
Unschedulable:  false
```

# 멀티 컨테이너 Pod

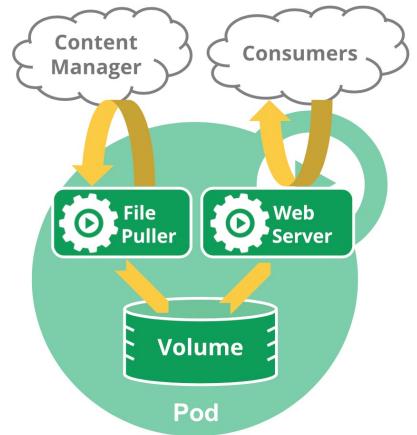
- Deployment 를 파일로 수정
- (실수 주의) Container가 2번 들어가는 것이 아님

spec:

```
containers:  
- image: busybox  
  name: lemon  
containers:  
- image: redis  
  name: gold
```

- 두개의 컨테이너가 한개의 emptydir 볼륨을 바라보게 하기

<https://kubernetes.io/docs/concepts/storage/volumes/#emptydir-configuration-example>



```
apiVersion: v1  
kind: Pod  
metadata:  
  name: test-pd  
spec:  
  containers:  
    - image: k8s.gcr.io/test-webserver  
      name: test-container  
    volumeMounts:  
      - mountPath: /cache  
        name: cache-volume.  
    - image: k8s.gcr.io/test-webserver  
      name: side-car  
    volumeMounts:  
      - mountPath: /cache  
        name: cache-volume.  
  volumes:  
    - name: cache-volume  
      emptyDir: {}
```

# Namespace

- 쿠버네티스는 동일한 물리 클러스터를 기반으로 하는 여러 가상 클러스터를 지원, 이런 가상 클러스터가 네임스페이스
- 네임스페이스 확인

```
[centos@osk-master-01 ~]$ kubectl get pods -A
```

| NAMESPACE   | NAME                                     | READY | STATUS  | RESTARTS | AGE |
|-------------|------------------------------------------|-------|---------|----------|-----|
| kube-system | calico-kube-controllers-58497c65d5-wnfnv | 1/1   | Running | 0        | 22h |

- 네임스페이스 생성, 해당 네임스페이스에 파드 생성

```
[centos@osk-master-01 ~]$ kubectl create namespace kakao
```

```
namespace/kakao created
```

```
[centos@osk-master-01 ~]$ kubectl run likelion --image=nginx --namespace=kakao
```

```
pod/likelion created
```

- 생성한 파드 확인

```
[centos@osk-master-01 ~]$ kubectl get pods
```

```
No resources found in default namespace.
```

```
[centos@osk-master-01 ~]$ kubectl get pods -n kakao
```

| NAME     | READY | STATUS  | RESTARTS | AGE |
|----------|-------|---------|----------|-----|
| likelion | 1/1   | Running | 0        | 20s |

# Label과 Selector

- [centos@osk-master-01 ~]\$ kubectl get pod --help  
Display one or many resources  
Prints a table of the most important information about the specified resources. You can filter the list using a label selector and the --selector flag. (생략)  
[centos@osk-master-01 ~]\$ kubectl run pod --image=nginx -l teacher=osk --dry-run=client -o yaml > label.yaml  
[centos@osk-master-01 ~]\$ cat label.yaml  
apiVersion: v1  
kind: Pod  
metadata:  
 creationTimestamp: null  
 labels:  
 teacher: osk  
 name: pod  
spec:  
 containers:  
 - image: nginx  
 name: pod  
 resources: {}  
 dnsPolicy: ClusterFirst  
 restartPolicy: Always  
status: {}  
[centos@osk-master-01 ~]\$ kubectl apply -f label.yaml  
pod/pod created  
[centos@osk-master-01 ~]\$ kubectl get pod --selector teacher=osk  
NAME READY STATUS RESTARTS AGE  
pod 1/1 Running 0 23s  
[centos@osk-master-01 ~]\$  
[centos@osk-master-01 ~]\$ kubectl get all --selector teacher=osk  
NAME READY STATUS RESTARTS AGE  
pod/pod 1/1 Running 0 39s  
[centos@osk-master-01 ~]\$

# Staticpod

- Static pod(Manifest)는 kubectl 및 control plane에서 관리하지 않는 pod
- pod spec을 디스크에 직접 쓸수 있으며, kubelet은 시작과 함께 바로 지정된 컨테이너를 시작.
- Kubelet은 지속적으로 변경사항이 있는지 Manifest 파일을 지속적으로 모니터링

```
[root@osk-master-01 manifests]# pwd
/etc/kubernetes/manifests
[root@osk-master-01 manifests]#
[root@osk-master-01 manifests]# ls
etcd.yaml  kube-apiserver.yaml  kube-controller-manager.yaml  kube-scheduler.yaml
[root@osk-master-01 manifests]#
[root@osk-master-01 manifests]# kubectl get pod -A -o wide
NAMESPACE     NAME                               READY   STATUS    RESTARTS   AGE      IP           NODE          NOMINATED NODE  READINESS GATES
kube-system   calico-kube-controllers-58497c65d5-wnfnv   1/1    Running   0          14h     192.168.205.193  osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   calico-node-9nc4s      1/1    Running   0          14h     172.30.7.246    osk-master-03.kr-central-1.c.internal  <none>        <none>
kube-system   calico-node-b4ghk      1/1    Running   0          14h     172.30.5.179    osk-worker-02.kr-central-1.c.internal  <none>        <none>
kube-system   calico-node-dqx6d      1/1    Running   0          14h     172.30.5.3       osk-worker-01.kr-central-1.c.internal  <none>        <none>
kube-system   calico-node-gg6vc      1/1    Running   0          14h     172.30.5.193    osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   calico-node-mvbn5      1/1    Running   0          14h     172.30.4.105    osk-master-02.kr-central-1.c.internal  <none>        <none>
kube-system   coredns-558bd4d5db-ctf7n   1/1    Running   0          14h     192.168.205.194  osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   coredns-558bd4d5db-q8zph   1/1    Running   0          14h     192.168.205.195  osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   etcd-osk-master-01.kr-central-1.c.internal  1/1    Running   0          14h     172.30.5.193    osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   etcd-osk-master-02.kr-central-1.c.internal  1/1    Running   0          14h     172.30.4.105    osk-master-02.kr-central-1.c.internal  <none>        <none>
kube-system   etcd-osk-master-03.kr-central-1.c.internal  1/1    Running   0          14h     172.30.7.246    osk-master-03.kr-central-1.c.internal  <none>        <none>
kube-system   kube-apiserver-osk-master-01.kr-central-1.c.internal  1/1    Running   0          14h     172.30.5.193    osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   kube-apiserver-osk-master-02.kr-central-1.c.internal  1/1    Running   0          14h     172.30.4.105    osk-master-02.kr-central-1.c.internal  <none>        <none>
kube-system   kube-apiserver-osk-master-03.kr-central-1.c.internal  1/1    Running   0          14h     172.30.7.246    osk-master-03.kr-central-1.c.internal  <none>        <none>
kube-system   kube-controller-manager-osk-master-01.kr-central-1.c.internal  1/1    Running   1          14h     172.30.5.193    osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   kube-controller-manager-osk-master-02.kr-central-1.c.internal  1/1    Running   0          14h     172.30.4.105    osk-master-02.kr-central-1.c.internal  <none>        <none>
kube-system   kube-controller-manager-osk-master-03.kr-central-1.c.internal  1/1    Running   0          14h     172.30.7.246    osk-master-03.kr-central-1.c.internal  <none>        <none>
kube-system   kube-proxy-h4hqp      1/1    Running   0          14h     172.30.5.3       osk-worker-01.kr-central-1.c.internal  <none>        <none>
kube-system   kube-proxy-jvnsv      1/1    Running   0          14h     172.30.5.193    osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   kube-proxy-qb95g      1/1    Running   0          14h     172.30.7.246    osk-master-03.kr-central-1.c.internal  <none>        <none>
kube-system   kube-proxy-s7kl6      1/1    Running   0          14h     172.30.5.179    osk-worker-02.kr-central-1.c.internal  <none>        <none>
kube-system   kube-proxy-tkzjh      1/1    Running   0          14h     172.30.4.105    osk-master-02.kr-central-1.c.internal  <none>        <none>
kube-system   kube-scheduler-osk-master-01.kr-central-1.c.internal  1/1    Running   1          14h     172.30.5.193    osk-master-01.kr-central-1.c.internal  <none>        <none>
kube-system   kube-scheduler-osk-master-02.kr-central-1.c.internal  1/1    Running   0          14h     172.30.4.105    osk-master-02.kr-central-1.c.internal  <none>        <none>
kube-system   kube-scheduler-osk-master-03.kr-central-1.c.internal  1/1    Running   0          14h     172.30.7.246    osk-master-03.kr-central-1.c.internal  <none>        <none>
[root@osk-master-01 manifests]#
```

# Staticpod

- 기본 값 외우지 않고 static pod manifest 저장된 위치 찾기

```
[centos@osk-master-01 ~]$ ps -ef | grep kubelet | grep conf
root      13731      1  3  8월14 ?    00:46:03 /usr/bin/kubelet --bootstrap-kubeconfig=/etc/kubernetes/bootstrap
-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf --config=/var/lib/kubelet/config.yaml --network-plugin=c
ni --pod-infra-container-image=k8s.gcr.io/pause:3.4.1
```

```
[centos@osk-master-01 ~]$ sudo cat /var/lib/kubelet/config.yaml
(생략)
```

```
shutdownGracePeriod: 0s
shutdownGracePeriodCriticalPods: 0s
staticPodPath: /etc/kubernetes/manifests
streamingConnectionIdleTimeout: 0s
syncFrequency: 0s
volumeStatsAggPeriod: 0s
```

# Staticpod

- Static pod 만들기

```
[centos@osk-master-01 ~]$ kubectl run --restart=Never --image=busybox static-busybox --dry-run=client -o yaml --command -- sleep 1000 > ./static-busybox.yaml
```

```
[centos@osk-master-01 ~]$ cat static-busybox.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: static-busybox
    name: static-busybox
spec:
  containers:
    - command:
        - sleep
        - "1000"
      image: busybox
      name: static-busybox
      resources: {}
    dnsPolicy: ClusterFirst
    restartPolicy: Never
  status: {}
```

# 클러스터 노드 작업

- Cordon/Drain/Taint 비교



## cor·don

1. [군사] 초병선(哨兵線); 비상[경계]선 2. 방역선, 교통 차단선 3. 수장으로 장식하다

- 노드에 더이상 어떤 파드도 스케줄링 되지 않도록 하는 기능
- ex) 오늘 야간에 정기점검이 잡혀있는 노드



## drain

1. 물을 빼내다 2. (액체를) 따라 내다 3. 배수관

- 노드에 있는 파드를 지금 바로 다른 노드로 이동시키는 기능
- Drain 명령어 입력 시 Cordon을 수행 후, Drain하게 됨
- ex) 해당 노드에 고장이 발생하여 파드를 다른 곳에서 새로 시작해야 하는 경우



## taint

1. (평판 등을) 더럽히다, 오염시키다, 오점을 남기다 2. 오점, 오명

- 파드가 부적절한 노드에 실행되지 않도록 설정하는 기능
- ex) Toleration과 결합하여 특정 파드만 실행시키고 싶을 때 (Control Plane으로 쓰는 Master Node 등)

# 클러스터 업그레이드

- 참고 : <https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/>
- Kubelet과 kubeadm 자체는 kubeadm으로 관리되지 않기에, OS 패키지 관리자인 apt/yum으로 설치 필요
- 업그레이드 순서 : [컨트롤 플레인 1번 노드 먼저 업그레이드](#) ► 2/3번 마스터 업그레이드 ► 워커 노드 업그레이드 수행
  - (참고1) 업그레이드 시, kubectl cordon/drain을 활용하여 사용자 워크로드를 조정도록 함
  - (참고2) 업그레이드가 완료 된 후에는 uncordon 하여 다시 파드가 스케줄링 되도록 설정

[1번 노드] #sudo yum list --showduplicates kubeadm --disableexcludes=Kubernetes

[1번 노드] #sudo yum install -y kubeadm-1.22.1-0 --disableexcludes=kubernetes

[1번 노드] #kubeadm upgrade plan

→ preflight 수행하여 클러스터가 정상인지 확인, kube-system kubeadm-config 컨피그맵 검사

→ 앞으로 업그레이드하는게 좋을지 알려줌 (root 계정으로 실행)

[1번 노드] #kubeadm upgrade apply v1.22.1

[1번 노드] #kubectl uncordon master01

[upgrade/successful] SUCCESS! Your cluster was upgraded to "v1.22.1". Enjoy!



# 클러스터 업그레이드

- 참고 : <https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/>
- Kubelet과 kubeadm 자체는 kubeadm으로 관리되지 않기에, OS 패키지 관리자인 apt/yum으로 설치 필요
- 업그레이드 순서 : 컨트롤 플레인 1번 노드 먼저 업그레이드 ► [2/3번 마스터 업그레이드](#) ► 워커 노드 업그레이드 수행

[2,3번 노드] #sudo yum install -y kubeadm-1.22.1-0 --disableexcludes=kubernetes

[2,3번 노드] #kubeadm upgrade node

[2,3번 노드] #kubectl uncordon master02

```
[upgrade] The control plane instance for this node was successfully updated!
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[upgrade] The configuration for this node was successfully updated!
[upgrade] Now you should go ahead and upgrade the kubelet package using your package manager.
```

[1~3번 노드] #sudo yum install -y kubelet-1.22.1-0 kubectl-1.22.1-0 --disableexcludes=kubernetes

[1~3번 노드] #sudo systemctl daemon-reload

[1~3번 노드] #sudo systemctl restart kubelet

[1~3번 노드] kubectl uncordon osk-master-01.kr-central-1.c.internal osk-master-02.kr-central-1.c.internal osk-master-03.kr-central-1.c.internal

# 클러스터 업그레이드

- 참고 : <https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/>
- Kubelet과 kubeadm 자체는 kubeadm으로 관리되지 않기에, OS 패키지 관리자인 apt/yum으로 설치 필요
- 업그레이드 순서 : 컨트롤 플레인 1번 노드 먼저 업그레이드 ► 2/3번 마스터 업그레이드 ► 워커 노드 업그레이드 수행

```
[워커노드1,2] #sudo yum install -y kubeadm-1.22.1-0 --disableexcludes=Kubernetes
```

```
[워커노드1,2] #sudo kubeadm upgrade node
```

```
[centos@osk-master-01 ~]$ kubectl drain osk-worker-01.kr-central-1.c.internal --ignore-daemonsets
```

```
[워커노드1] #sudo yum install -y kubelet-1.22.1-0 kubectl-1.22.1-0 --disableexcludes=kubernetes
```

```
[워커노드1] #sudo systemctl daemon-reload
```

```
[워커노드1] #sudo systemctl restart kubelet
```

```
[centos@osk-master-01 ~]$ kubectl uncordon osk-worker-01.kr-central-1.c.internal
```

```
[centos@osk-master-01 ~]$ kubectl get nodes
```

| NAME                                  | STATUS | ROLES                | AGE  | VERSION |
|---------------------------------------|--------|----------------------|------|---------|
| osk-master-01.kr-central-1.c.internal | Ready  | control-plane,master | 127m | v1.22.1 |
| osk-master-02.kr-central-1.c.internal | Ready  | control-plane,master | 121m | v1.22.1 |
| osk-master-03.kr-central-1.c.internal | Ready  | control-plane,master | 116m | v1.22.1 |
| osk-worker-01.kr-central-1.c.internal | Ready  | <none>               | 105m | v1.22.1 |
| osk-worker-02.kr-central-1.c.internal | Ready  | <none>               | 105m | v1.21.0 |

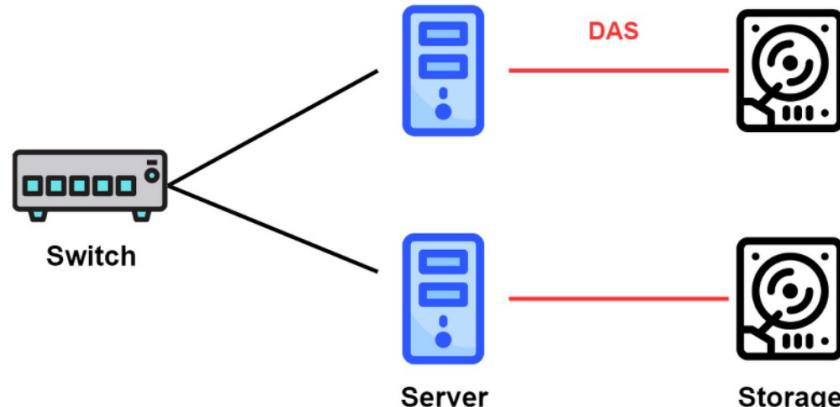
```
[워커노드2도 drain부터 반복]
```

## [Storage] Volumes and data

# Storage 개요

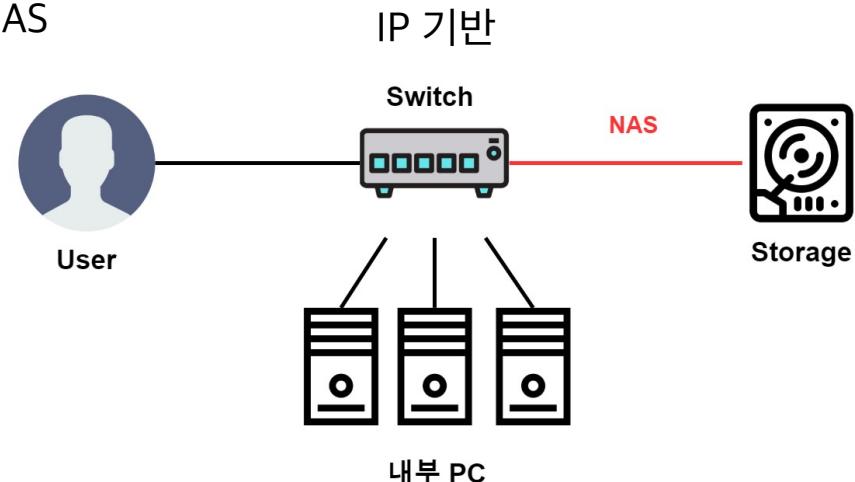
- 스토리지 종류

- DAS (직접 연결)

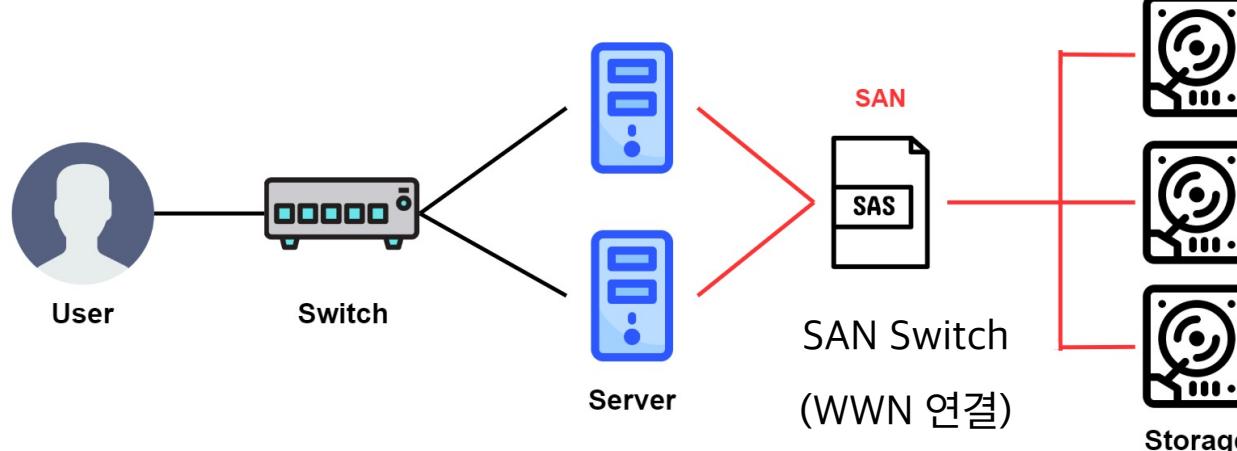


저렴 : Ceph, GlusterFS로 발전 꾀하는 중

- NAS



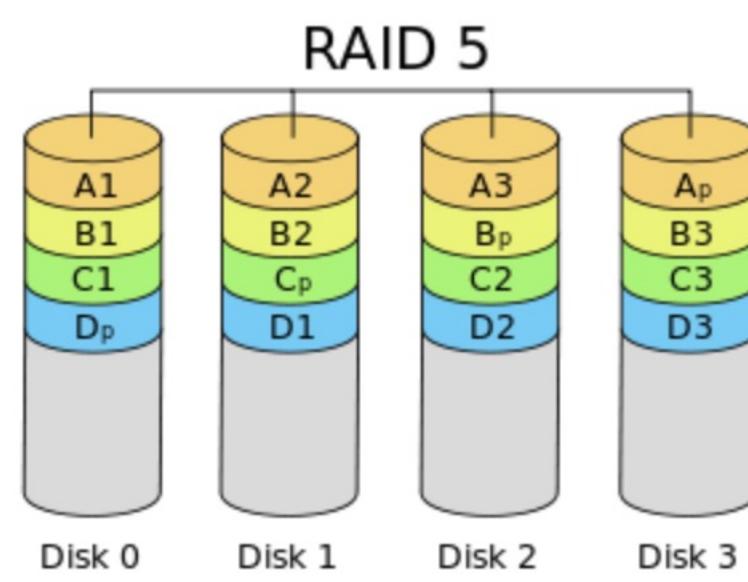
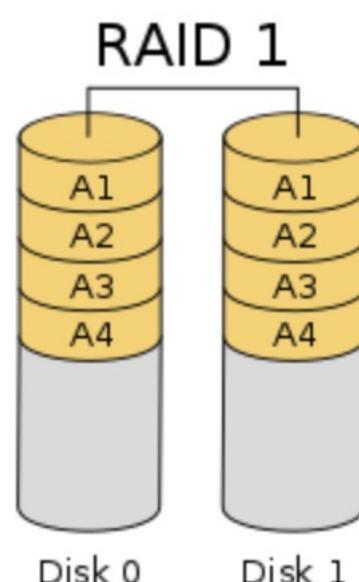
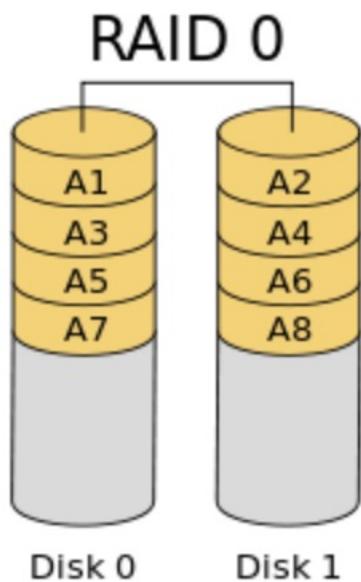
- SAN



- 가격 : DAS < NAS < SAN (비쌈)
- 성능 : NAS < DAS=SAN (좋음)
- 확장성 : DAS < SAN(보통 광) < NAS

# Storage 개요

- RAID 0 / 1 / 5 ( 그 외 RAID 2~4, 6도 있음 )
  - RAID 0 : 구매한 그대로 사용
  - RAID 1 : 구매한 절반 사용
  - RAID 5 : 패리티 비트 이용하여 데이터 복구 / 2개 이상 고장 시 복구 불가



# Volumes 소개

- 볼륨(volume)은 하나의 파일 시스템을 갖춘 하나의 접근 가능한 스토리지 영역

| 물리 디스크   | 파티션   | 파일 시스템 | 드라이브 문자 |
|----------|-------|--------|---------|
| 하드 디스크 1 | 파티션 1 | NTFS   | C:      |
|          | 파티션 2 | FAT32  | D:      |
| 하드 디스크 2 | 파티션 1 | FAT32  | E:      |



이 예에서

- "C:", "D:", "E:"는 볼륨이다.
- 하드 디스크 1과 하드 디스크 2는 물리 디스크이다.
- 이들 중 어떠한 것도 "드라이브"라 부를 수 있다.

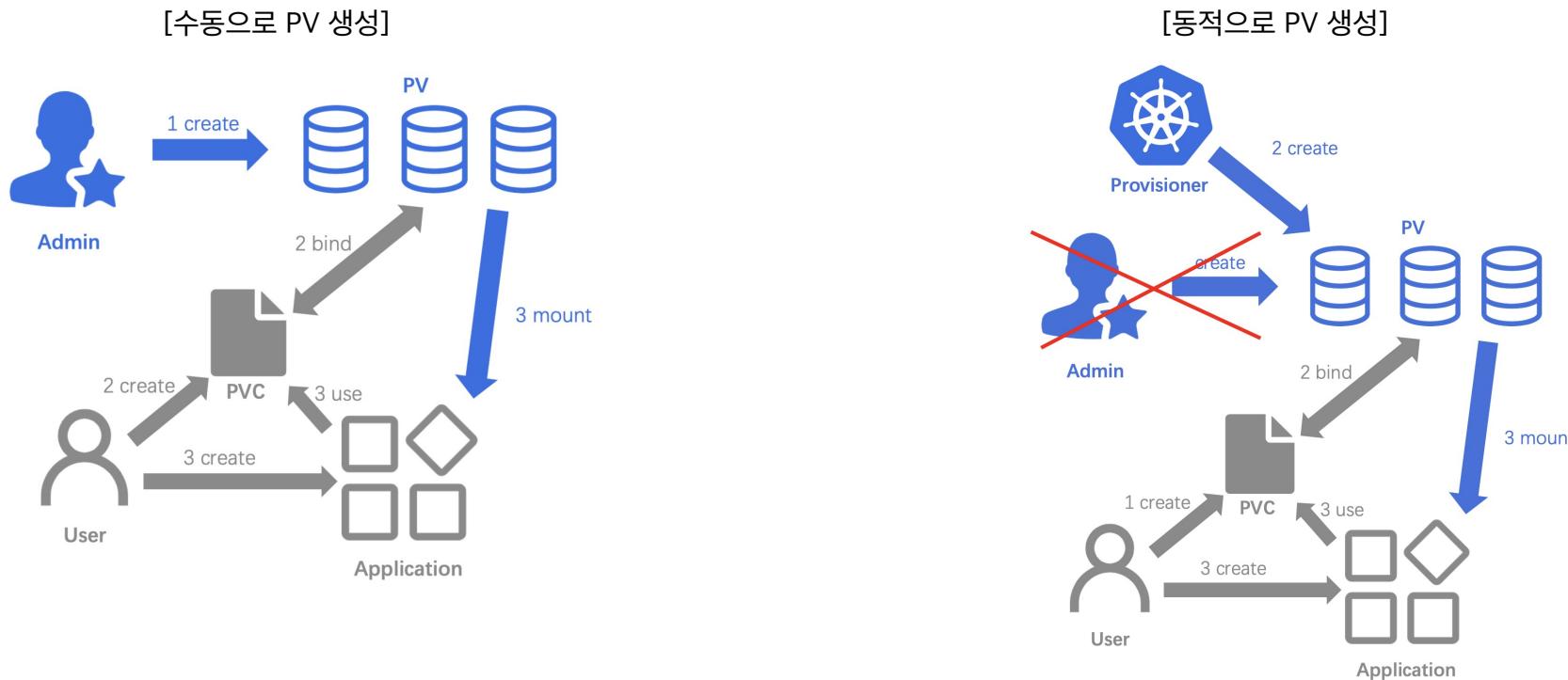
```
[centos@osk-master-01 ~]$ sudo fdisk -l  
[centos@osk-master-01 ~]$ sudo cat /etc/fstab
```

파일 시스템은 컴퓨터에서 [파일](#)이나 [자료](#)를 쉽게 발견 및 접근할 수 있도록 보관 또는 조직하는 체계

파일 시스템은 통상 [하드 디스크](#)나 [CD-ROM](#) 같은 실제 자료 보관 장치를 사용하여 파일의 물리적 소재를 관리하는 것을 가리키나 네트워크 프로토콜([NFS](#), [SMB](#), [9P](#) 등)을 수행하는 클라이언트를 통하여 파일 서버 상의 자료로의 접근을 제공하는 방식과 가상의 형태로서 접근 수단만이 존재하는 방식([procfs](#) 등)도 파일 시스템의 범위에 포함

# Persistent Volume 소개(앞 설명 동일)

- Persistent Volume(PV) : 관리자가 프로비저닝하거나 스토리지 클래스를 사용하여 동적으로 프로비저닝한 클러스터의 스토리지
- Persistent Volume Claim(PVC) : 사용자의 스토리지에 대한 요청. 파드와 비슷.
  - ✓ 파드 : 노드 리소스를 사용 / POD 명세서 내 특정 수준의 리소스(CPU 및 메모리)를 요청
  - ✓ PVC : PV 리소스를 사용 / 클레임 명세서 내 특정 크기 및 접근 모드를 요청(예: ReadWriteOnce, ReadOnlyMany, ReadWriteMany)



# Persistent Volume 소개

- PV 생성 : <https://kubernetes.io/ko/docs/concepts/storage/volumes/#hostpath>

```
[centos@osk-master-01 ~]$ cat hostpath.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
    - image: k8s.gcr.io/test-webserver
      name: test-container
      volumeMounts:
        - mountPath: /test-pd
          name: test-volume
```

```
volumes:
  - name: test-volume
    hostPath:
```

```
      # 호스트의 디렉터리 위치
      path: /data
      # 이 필드는 선택 사항이다
      type: Directory
```

```
[centos@osk-master-01 ~]$ kubectl apply -f hostpath.yaml
```

```
pod/test-pd created
```

```
[centos@osk-master-01 ~]$ kubectl describe pod test-pd
```

```
Name:           test-pd (이후 생략)
```

```
Volumes:
```

```
  test-volume:
    Type:            HostPath (bare host directory volume)
    Path:             /data
    HostPathType:    Directory
```

# Pod으로 Data(Volumes) 연결

```
[centos@osk-master-01 ~]$ kubectl get pod -o wide
```

| NAME    | READY | STATUS  | RESTARTS | AGE | IP              | NODE                                  | NOMINATED NODE | READINESS GATES |
|---------|-------|---------|----------|-----|-----------------|---------------------------------------|----------------|-----------------|
| pod     | 1/1   | Running | 0        | 64m | 192.168.224.3   | osk-worker-01.kr-central-1.c.internal | <none>         | <none>          |
| test-pd | 1/1   | Running | 0        | 6s  | 192.168.183.131 | osk-worker-02.kr-central-1.c.internal | <none>         | <none>          |

```
[centos@osk-worker-02 ~]$ cd /data
```

```
[centos@osk-worker-02 data]$ sudo touch likelion
```

```
[centos@osk-worker-02 data]$ sudo ls
```

```
likelion
```

```
[centos@osk-master-01 ~]$ kubectl delete pod test-pd  
pod "test-pd" delete
```

```
[centos@osk-worker-02 data]$ sudo ls  
likelion
```

```
[centos@osk-master-01 ~]$ kubectl apply -f hostpath.yaml  
pod/test-pd created
```

(결과 확인시 원활히 접속토록 공식홈페이지 yaml에서 이미지 nginx 등 수정바람)

```
[centos@osk-master-01 ~]$ kubectl get pod -o wide
```

| NAME    | READY | STATUS            | RESTARTS | AGE | IP            | NODE                                  | NOMINATED NODE | READINESS GATES |
|---------|-------|-------------------|----------|-----|---------------|---------------------------------------|----------------|-----------------|
| pod     | 1/1   | Running           | 0        | 17h | 192.168.224.3 | osk-worker-01.kr-central-1.c.internal | <none>         | <none>          |
| test-pd | 0/1   | ContainerCreating | 0        | 9s  | <none>        | osk-worker-02.kr-central-1.c.internal | <none>         | <none>          |

```
[centos@osk-master-01 ~]$
```

```
[centos@osk-master-01 ~]$ kubectl exec -it test-pd -- /bin/sh
```

```
# cd /test-pd
```

```
# ls
```

```
likelion
```

```
#
```

# ConfigMap

- 컨피그맵은 키-값 쌍으로 기밀이 아닌 데이터를 저장하는 데 사용하는 API 오브젝트
- 컨테이너에 필요한 환경 설정 내용을 컨테이너 내부가 아닌 외부에 분리하는데 용이
- 클러스터가 구성된 Config 방식을 이해하는 데 사용할 수도 있고, 클러스터를 업그레이드 할 때 활용할 수도 있음
- <https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/#configure-all-key-value-pairs-in-a-configmap-as-container-environment-variables>

```
# kubectl create -f https://kubernetes.io/examples/configmap/configmap-multikeys.yaml  
# kubectl create -f https://kubernetes.io/examples/pods/pod-configmap-envFrom.yaml
```

```
[centos@osk-master-01 ~]$ kubectl create -f https://kubernetes.io/examples/configmap/configmap-multikeys.yaml
```

```
configmap/special-config created
```

```
[centos@osk-master-01 ~]$ kubectl create -f https://kubernetes.io/examples/pods/pod-configmap-envFrom.yaml
```

```
pod/dapi-test-pod created
```

```
[centos@osk-master-01 ~]$ kubectl logs dapi-test-pod
```

생략

SPECIAL\_LEVEL=very

생략

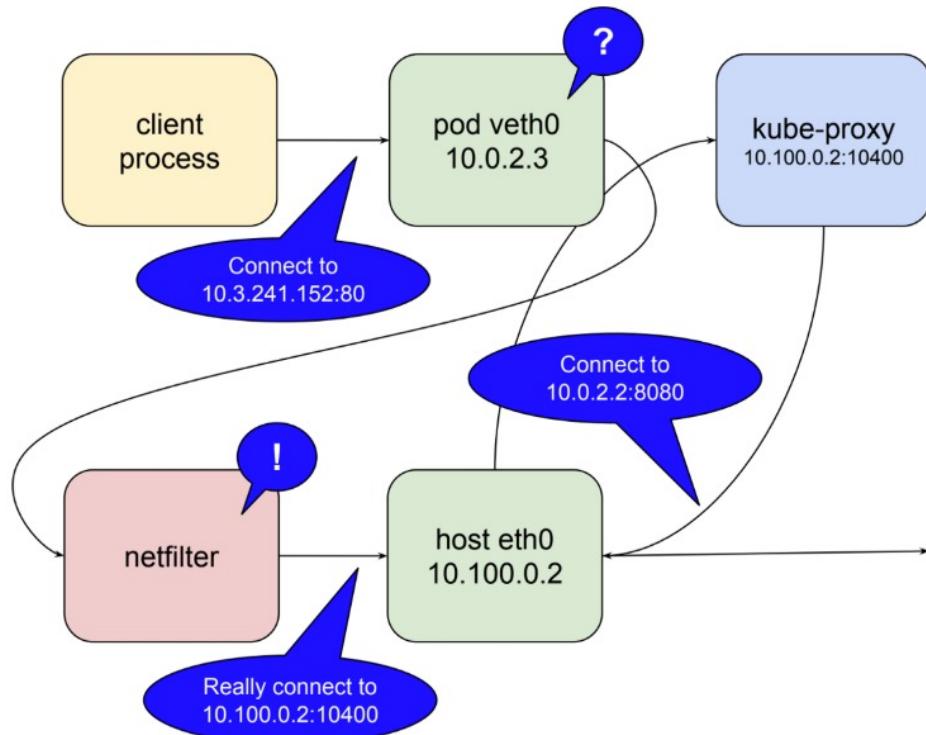
SPECIAL\_TYPE=charm

# [Network] Services & Ingress

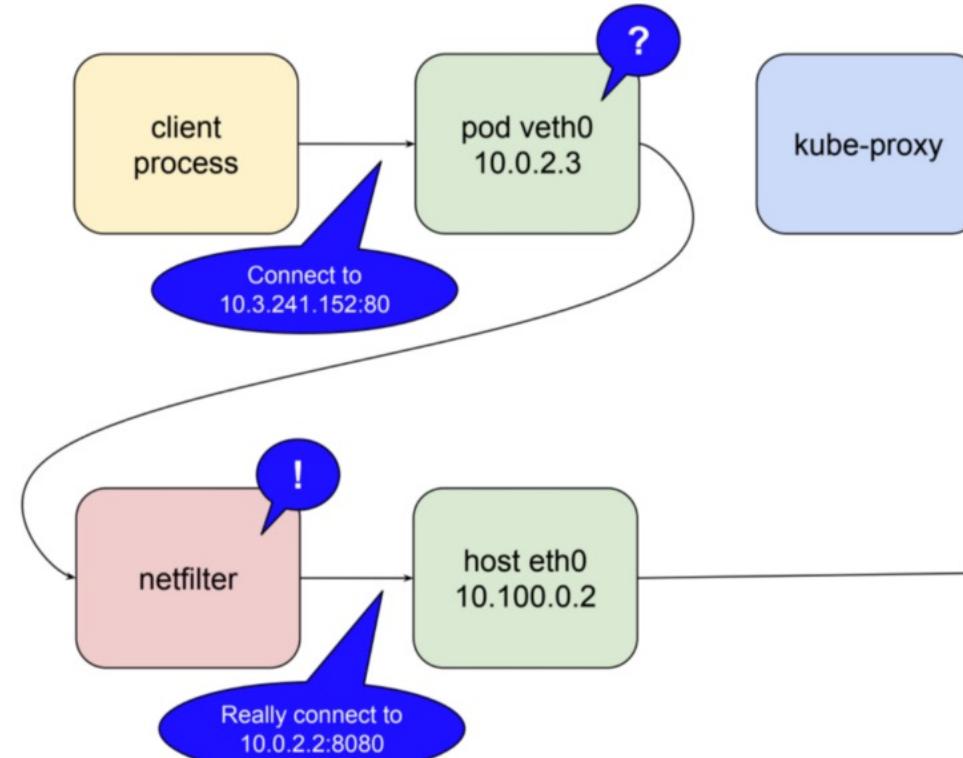
# Services 개요

- 여러 레플리카에 트래픽을 분산시키는 로드밸런서 (TCP, UDP 모두 가능)
- 노드의 kube-proxy 를 활용하여 엔드포인트로 라우팅 되도록 처리

[ Kube-proxy가 직접 UserSpace Proxy역할 시 ]



[ Kube-proxy가 Iptables를 통해 netfilter조작하는 역할 시 ]



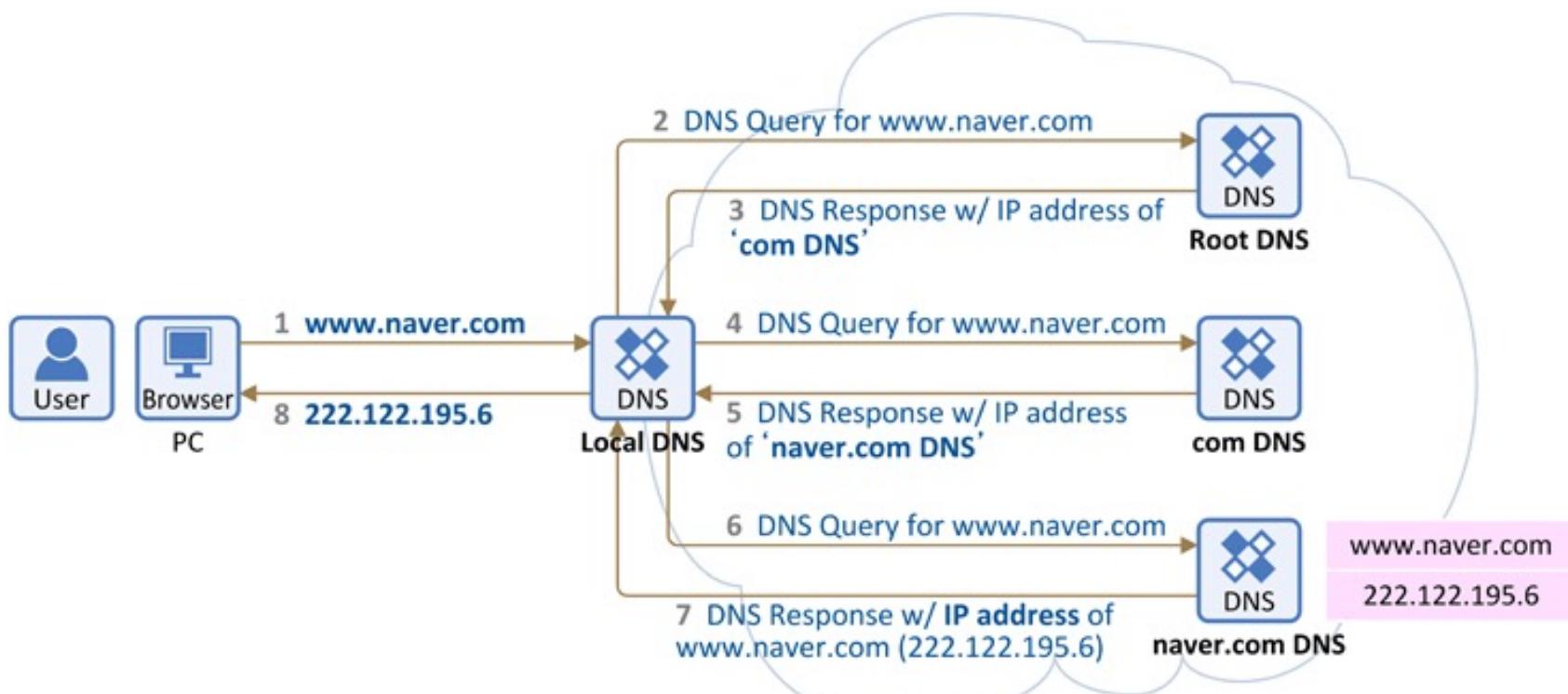
# Services 실습

- <https://kubernetes.io/ko/docs/concepts/services-networking/connect-applications-service> 참고

```
[centos@osk-master-01 ~]$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/main/content/ko/examples/service/networking/run-my-nginx.yaml
deployment.apps/my-nginx created
[centos@osk-master-01 ~]$ kubectl expose deployment/my-nginx
service/my-nginx exposed
[centos@osk-master-01 ~]$ kubectl describe svc my-nginx
Name:           my-nginx
Namespace:      default
Labels:          <none>
Annotations:    <none>
Selector:        run=my-nginx
Type:            ClusterIP
IP Family Policy: SingleStack
IP Families:    IPv4
IP:              10.104.164.89
IPs:             10.104.164.89
Port:            <unset>  80/TCP
TargetPort:      80/TCP
Endpoints:       192.168.183.140:80,192.168.224.7:80
Session Affinity: None
Events:          <none>
[centos@osk-master-01 ~]$
```

# DNS

- (위키백과) 도메인 네임 시스템(Domain Name System, DNS)은 호스트의 도메인 이름을 호스트의 네트워크 주소로 바꾸거나 그 반대의 변환을 수행할 수 있도록 하기 위해 개발되었다. 특정 컴퓨터(또는 네트워크로 연결된 임의의 장치)의 주소를 찾기 위해, 사람이 이해하기 쉬운 도메인 이름을 숫자로 된 식별 번호(IP 주소)로 변환해 준다. 도메인 네임 시스템은 흔히 "전화번호부"에 비유된다. 인터넷 도메인 주소 체계로서 TCP/IP의 응용에서, www.example.com과 같은 주 컴퓨터의 도메인 이름을 192.168.1.0과 같은 IP 주소로 변환하고 라우팅 정보를 제공하는 분산형 데이터베이스 시스템이다.



# DNS

- 공유기에서는 단말에 사설 IP를 할당하면서 DNS 주소도 배포 가능



- 공유기, DNS 를 모두 나쁜 목적으로 활용 시 :



# DNS

- 설치 과정 중, 기본 값인 CoreDNS를 애드온으로 설치 완료

```
[centos@osk-master-01 ~]$ kubectl get svc -A
NAMESPACE      NAME        TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)          AGE
default        kubernetes  ClusterIP   10.96.0.1      <none>        443/TCP         44h
default        my-nginx    ClusterIP   10.104.164.89  <none>        80/TCP          20m
kube-system    kube-dns    ClusterIP   10.96.0.10     <none>        53/UDP,53/TCP,9153/TCP 44h
```

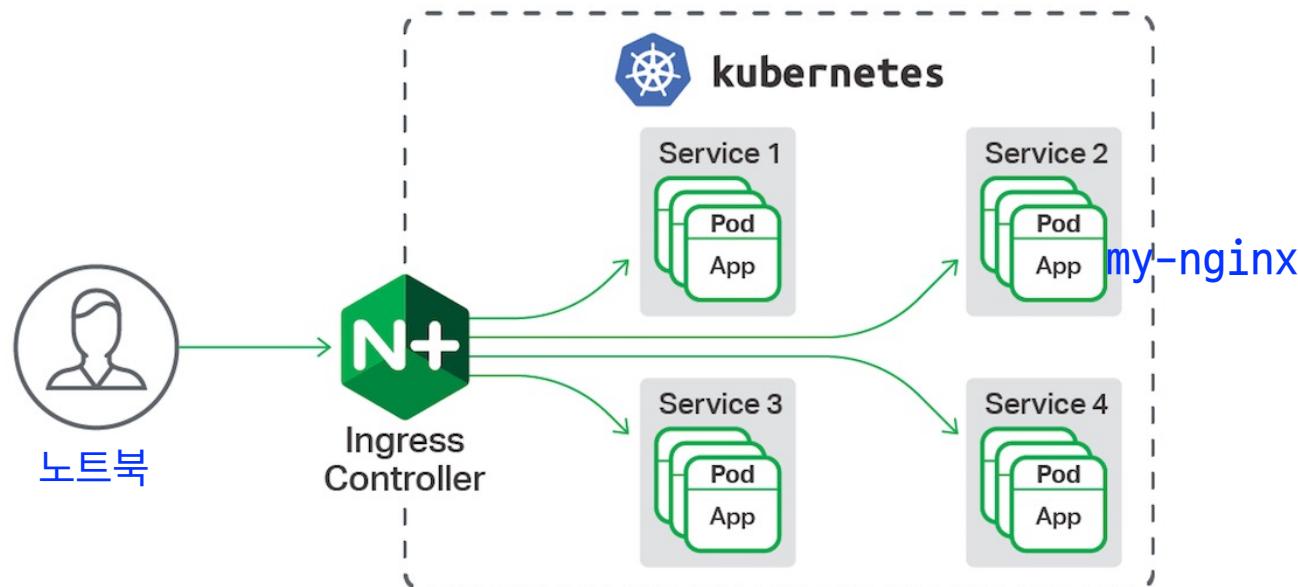
```
[centos@osk-master-01 ~]$ kubectl run --image=alpine dns-test -it -- /bin/sh
If you don't see a command prompt, try pressing enter.
/ # nslookup kubernetes
Name: kubernetes.default.svc.cluster.local # 모든 서비스는 생성시 <service name>.<namespace>.svc.cluster.local
Address: 10.96.0.1
생략
/ # nslookup my-nginx
Name: my-nginx.default.svc.cluster.local
Address: 10.104.164.89
/ # cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local kr-central-1.c.kakaoi.io kr-central-1.c.internal
options ndots:5
```

# Ingress

- 클러스터 내의 서비스에 대한 외부 접근을 관리하는 API 오브젝트이며, 일반적으로 HTTP를 관리함.
- (시나리오) 앞서 Service 과정에서 만든 my-nginx는 ClusterIP 서비스이기에 클러스터 내에서는 접속 가능하지만 외부에서는 접속 불가  
[centos@osk-master-01 ~]\$ kubectl describe svc my-nginx

|       |           |
|-------|-----------|
| Name: | my-nginx  |
| Type: | ClusterIP |

- 외부에 load balancing을 지원하는 Ingress를 통해 외부에서도 nginx 서비스에 접속 가능토록 실습



# Ingress

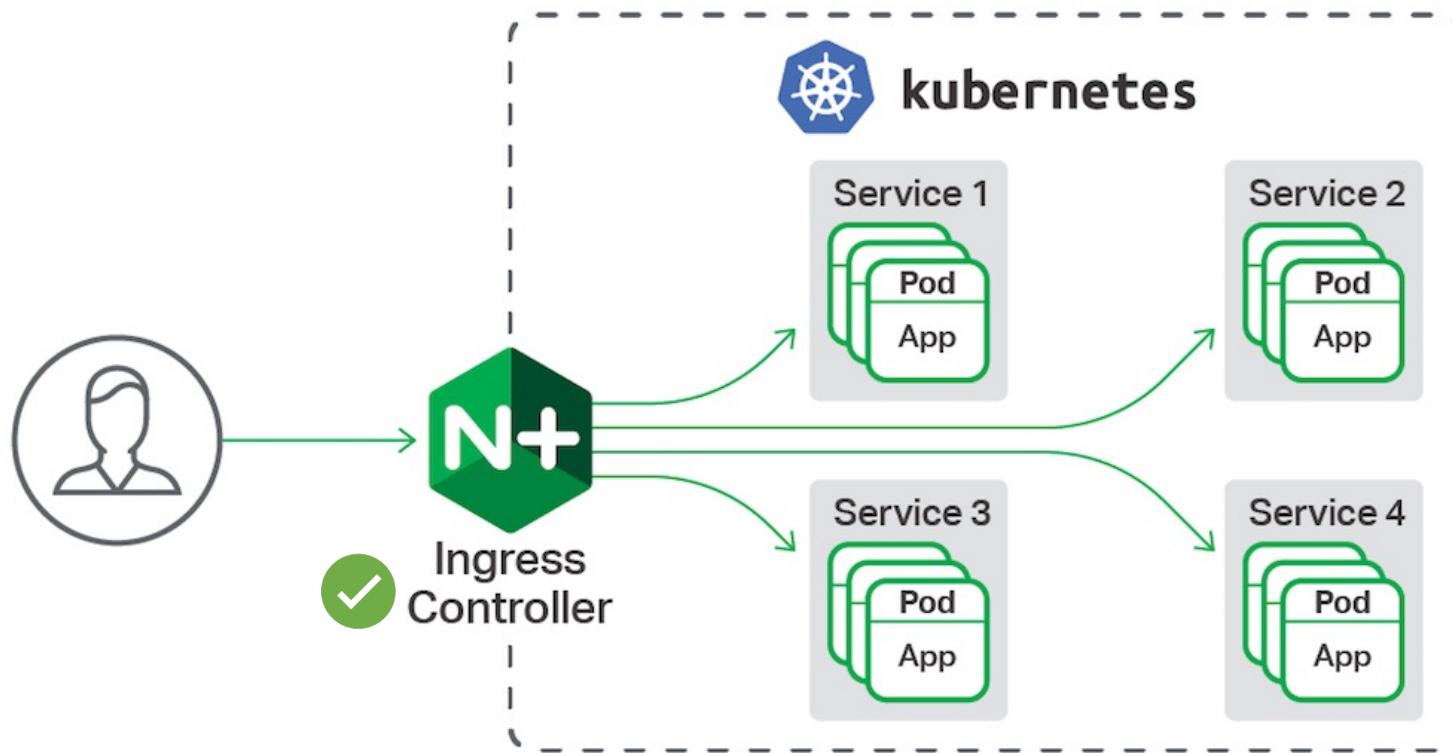
- <https://kubernetes.io/ko/docs/concepts/services-networking/ingress/#인그레스-리소스> 참고

```
[centos@osk-master-01 ~]$ wget https://raw.githubusercontent.com/kubernetes/website/main/content/ko/examples/service/networking/minimal-ingress.yaml
```

```
[centos@osk-master-01 ~]$ cat minimal-ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: my-nginx           → vi로 수정
            port:
              number: 80
```

# Ingress Controller

- 앞서 생성한 Ingress는 로드밸런싱에 필요한 정보 일뿐, 실제 로드 밸런싱을 수행할 프로그램이 있어야함(인그레스 컨트롤러)
- 종류는 다양하며, 쿠버네티스 프로젝트에서는 AWS/GCE/nginx 지원 <https://kubernetes.io/ko/docs/concepts/services-networking/ingress-controllers/>



# Ingress Controller

- Ingress Controller 생성 <https://kubernetes.github.io/ingress-nginx/deploy/#bare-metal>

```
[centos@osk-master-01 ~]$ kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.48.1/deploy/static/provider/baremetal/deploy.yaml
namespace/ingress-nginx created
serviceaccount/ingress-nginx created
configmap/ingress-nginx-controller created
clusterrole.rbac.authorization.k8s.io/ingress-nginx created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created
role.rbac.authorization.k8s.io/ingress-nginx created
rolebinding.rbac.authorization.k8s.io/ingress-nginx created
service/ingress-nginx-controller-admission created
service/ingress-nginx-controller created
deployment.apps/ingress-nginx-controller created
validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created
serviceaccount/ingress-nginx-admission created
clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created
clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
role.rbac.authorization.k8s.io/ingress-nginx-admission created
rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created
job.batch/ingress-nginx-admission-create created
job.batch/ingress-nginx-admission-patch created
```

# Ingress Controller

- Ingress 생성

```
[centos@osk-master-01 ~]$ kubectl apply -f minimal-ingress.yaml
ingress.networking.k8s.io/minimal-ingress created
```

```
[centos@osk-master-01 ~]$ kubectl get service -n ingress-nginx
```

| NAME                               | TYPE      | CLUSTER-IP   | EXTERNAL-IP | PORT(S)                    | AGE  |
|------------------------------------|-----------|--------------|-------------|----------------------------|------|
| ingress-nginx-controller           | NodePort  | 10.109.83.45 | <none>      | 80:30189/TCP,443:32660/TCP | 3h8m |
| ingress-nginx-controller-admission | ClusterIP | 10.98.63.210 | <none>      | 443/TCP                    | 3h8m |



**404 Not Found**

nginx



**Welcome to nginx!**

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org). Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

- Ingress Class 관련 : <https://kubernetes.github.io/ingress-nginx/#i-have-only-one-instance-of-the-ingress-nginx-controller-in-my-cluster-what-should-i-do> 128

# Ingress Rule

- 1. 선택적 호스트
- 2. 경로목록
- 3. 백엔드

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: simple-fanout-example
spec:
  rules:
    - host: foo.bar.com
      http:
        paths:
          - path: /foo
            pathType: Prefix
            backend:
              service:
                name: service1
                port:
                  number: 4200
          - path: /bar
            pathType: Prefix
            backend:
              service:
                name: service2
                port:
                  number: 8080
```

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
    - host: foo.bar.com
      http:
        paths:
          - pathType: Prefix
            path: "/"
            backend:
              service:
                name: service1
                port:
                  number: 80
    - host: bar.foo.com
      http:
        paths:
          - pathType: Prefix
            path: "/"
            backend:
              service:
                name: service2
                port:
                  number: 80
```

# [Operation] Logging & Troubleshooting, Backup, Upgrade

# Logging & Troubleshooting

- 구글의 사이트 신뢰성 엔지니어링(<https://sre.google>)

Google Site Reliability Engineering     Home     Spotlight     Resources     Books     Mobaas     Classroom     Careers

## What is Site Reliability Engineering (SRE)?

SRE is what you get when you treat operations as if it's a software problem. Our mission is to protect, provide for, and progress the software and systems behind all of Google's public services — Google Search, Ads, Gmail, Android, YouTube, and App Engine, to name just a few — with an ever-watchful eye on their availability, latency, performance, and capacity.

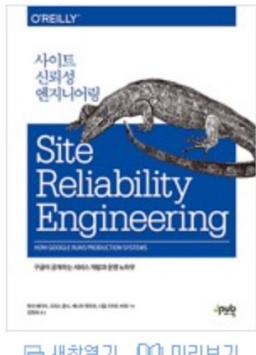
### What is SRE?

Since 2004, SRE has evolved to become the industry-leading practice for service reliability.

Hear from key figures about the history of SRE and what's next for the SRE community.







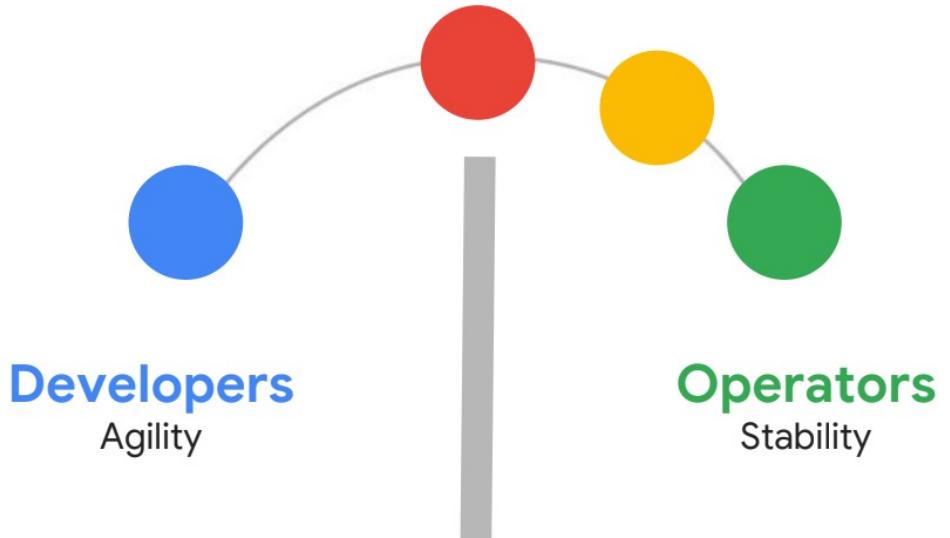
[국내도서] [사이트 신뢰성 엔지니어링](#) - 구글이 공개하는 서비스 개발과 운영 노하우  
벳시 베이어, 크리스 존스, 제니퍼 펫오프, 니얼 리처드 머피 (지은이), 장현희 (옮긴이) | 제이펍 | 2018년 1월  
36,000원 → **32,400원** (10% 할인), 마일리지 1,800원 (5% 적립)  
세일즈포인트 : 1,069

**양단자배송** 오후 8시 **퇴근후 배송**  
(중구 중림동) [지역변경](#)

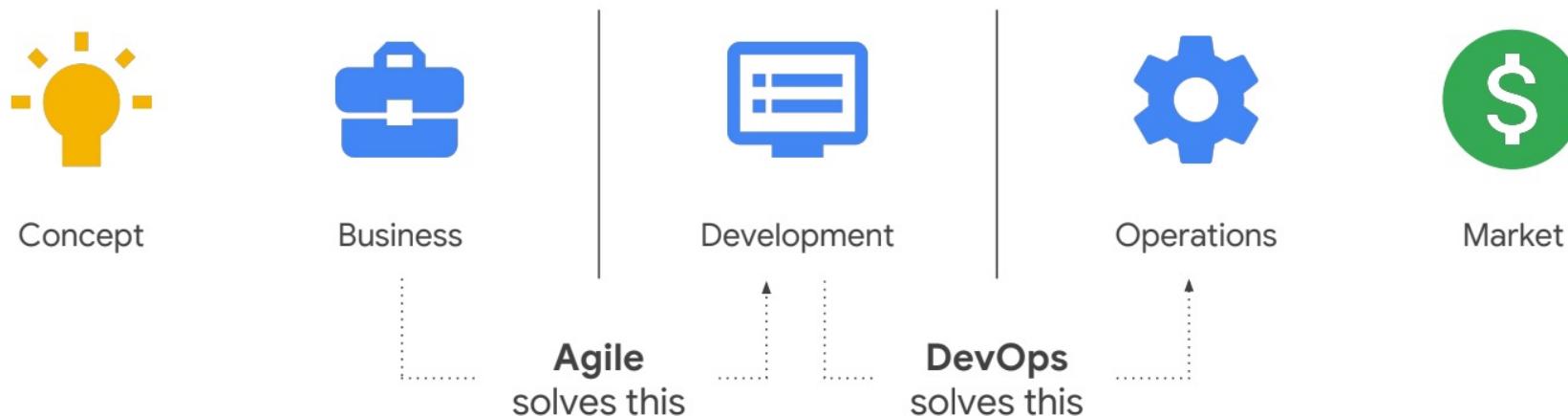
**스프링분철 서비스 이용이 가능한 도서입니다.** [자세히보기](#)

# Logging & Troubleshooting

- 개발자와 운영자는 추구 목표가 다름



- 아이디어가 마켓까지 오는 시간이 점점 빨라지고 있음



# Logging & Troubleshooting

- SRE들은 DevOps를 구현
  - DevOps는 IT의 Silo, Ops, Network, Security 등을 허무는 방식, 가이드라인, 문화의 집합
  - SRE는 그동안 찾은 작업 방식, 이러한 사례로 구체화하는 신념, 그리고 역할에 대한 집합

## class SRE implements DevOps

### DevOps

is a set of **practices**,  
**guidelines** and **culture**  
designed to break down  
silos in IT, ops, networks,  
security, etc.

### Site Reliability Engineering

is a set of **practices** we've  
found to work, some **beliefs**  
that animate those practices,  
and **a job role**.



# Logging & Troubleshooting

- 메트릭&모니터링 / 용량 관리 / 변화 관리 / 긴급 대응 / 문화
- 메트릭&모니터링 : 모니터링 지표를 정의하고, 정의된 지표를 모니터링 시스템으로 구성.
  - 인사이트를 통해 시스템이 안정적인 상황과 또는 장애가 나는 지표는 무엇인지, 왜인지? 어떻게 해결할지 고민.
  - 기본적으로 SRE에서 가장 중요한 부분은 모든 것을 데이터화하고, 의사결정을 데이터 기반으로 하는 것

## Summary of SRE practices



### Metrics & Monitoring

- SLOs
- Dashboards
- Analytics



### Capacity Planning

- Forecasting
- Demand-driven
- Performance



### Change Management

- Release process
- Consulting design
- Automation



### Emergency Response

- On-call
- Analysis
- Postmortems



### Culture

- Toil management
- Engineering alignment
- Blamelessness

# Logging & Troubleshooting

- 모니터링 : 쿼리 카운트와 종류, 에러 카운트와 종류, 프로세싱 타임, 서버의 라이프타임과 같은 수치를 실시간으로 수집/처리/집계/보여주는
  - 메트릭 : 특정 시스템에서 리소스, 응용 프로그램 작업 또는 비즈니스 특성이 특정 시점에서의 수치로 표현되는 것. 보통 키-밸류(key-value) 형태로 수집된 숫자가 일반적
  - 로깅(Logging) : 로깅은 메트릭보다 훨씬 많은 데이터를 포함하는 시스템이나 애플리케이션의 이벤트로 나타나며, 이러한 이벤트에 의해 생성되는 모든 정보를 포함
  - 트레이스(Tracing) : 특정 고유한 식별자가 모든 시스템에 걸쳐 전체 수명 주기 동안 추적될 수 있도록 제공하는 로깅의 특별한 경우

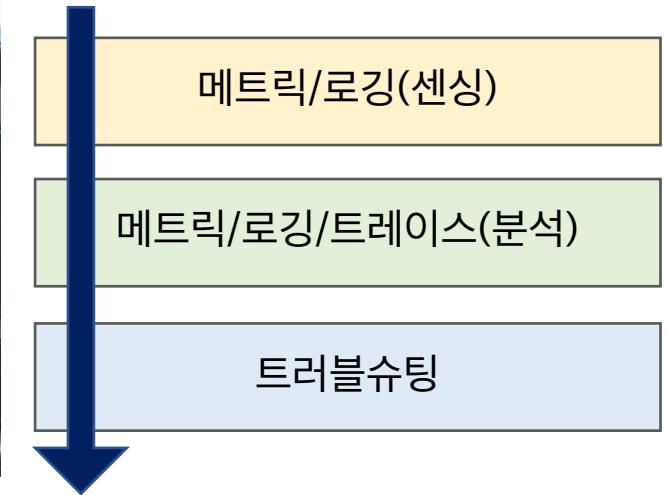
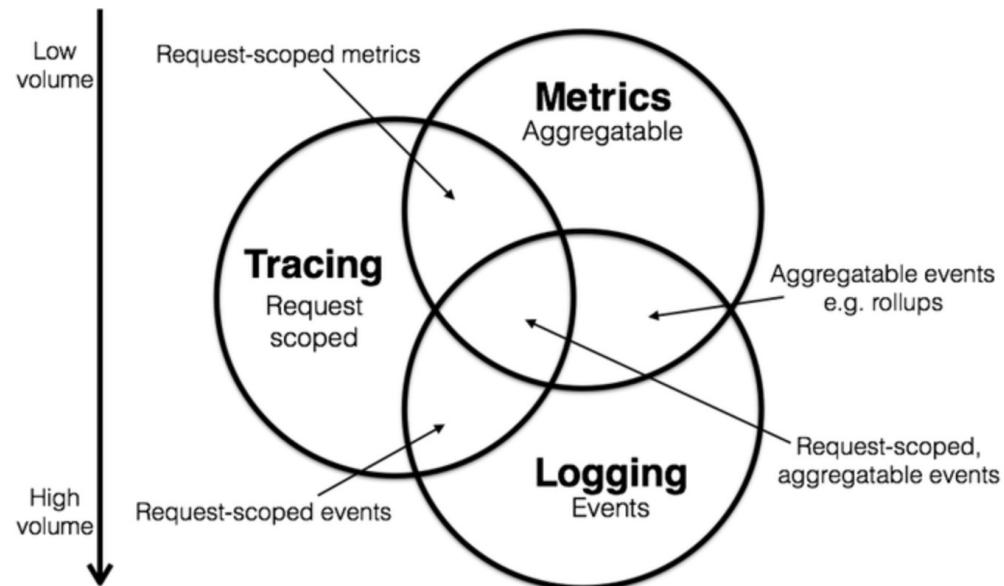


그림 출처 :<https://peter.bourgon.org/blog/2017/02/21/metrics-tracing-and-logging.html>  
[https://cdn.emetro.co.kr/pdf/2015/08/05/kr\\_metro\\_1\\_0805\\_14.pdf](https://cdn.emetro.co.kr/pdf/2015/08/05/kr_metro_1_0805_14.pdf)

# Logging & Troubleshooting

- 쿠버네티스에서의 기본 로깅 (<https://kubernetes.io/ko/docs/concepts/cluster-administration/logging/#쿠버네티스의-기본-로깅>)

```
apiVersion: v1
kind: Pod
metadata:
  name: counter
spec:
  containers:
  - name: count
    image: busybox
    args: [/bin/sh, -c,
            'i=0; while true; do echo "$i: $(date)"; i=$((i+1)); sleep 1; done']
```

```
[centos@osk-master-01 ~]$ kubectl apply -f https://k8s.io/examples/debug/counter-pod.yaml
pod/counter created

[centos@osk-master-01 ~]$ kubectl logs counter
0: Sat Aug 21 03:18:01 UTC 2021
1: Sat Aug 21 03:18:02 UTC 2021
2: Sat Aug 21 03:18:03 UTC 2021
3: Sat Aug 21 03:18:04 UTC 2021
4: Sat Aug 21 03:18:05 UTC 2021
```

# Logging & Troubleshooting

- 쿠버네티스에서의 기본 로깅 (<https://kubernetes.io/ko/docs/concepts/cluster-administration/logging/#쿠버네티스의-기본-로깅>)

```
[centos@osk-master-01 ~]$ kubectl logs counter
```

```
0: Sat Aug 21 03:18:01 UTC 2021
```

- 이 로그의 실제 위치는?

```
[root@osk-worker-02 containers]# ls -lrt /var/log/containers/counter_default_count-b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190.0.log
lrwxrwxrwx. 1 root root 78 Aug 21 03:18 /var/log/containers/counter_default_count-b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190.log
log -> /var/log/pods/default_counter_657461bd-1677-4384-bf7c-15dc29271837/count/0.log
```

```
[root@osk-worker-02 containers]# ls -lrt /var/log/pods/default_counter_657461bd-1677-4384-bf7c-15dc29271837/count/0.log
```

```
lrwxrwxrwx. 1 root root 165 Aug 21 03:18 /var/log/pods/default_counter_657461bd-1677-4384-bf7c-15dc29271837/count/0.log -> /var/lib/docker/containers/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190-json.log
```

```
[root@osk-worker-02 containers]# ls -lrt /var/lib/docker/containers/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190-b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190-json.log
```

```
-rw-r-----. 1 root root 215303 Aug 21 03:52 /var/lib/docker/containers/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190-json.log
```

```
[root@osk-worker-02 containers]# file /var/log/pods/default_counter_657461bd-1677-4384-bf7c-15dc29271837/count/0.log
```

```
/var/log/pods/default_counter_657461bd-1677-4384-bf7c-15dc29271837/count/0.log: symbolic link to `/var/lib/docker/containers/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190-json.log'
```

```
[root@osk-worker-02 containers]# file /var/lib/docker/containers/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190-json.log
```

```
/var/lib/docker/containers/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190-json.log: ASCII text
```

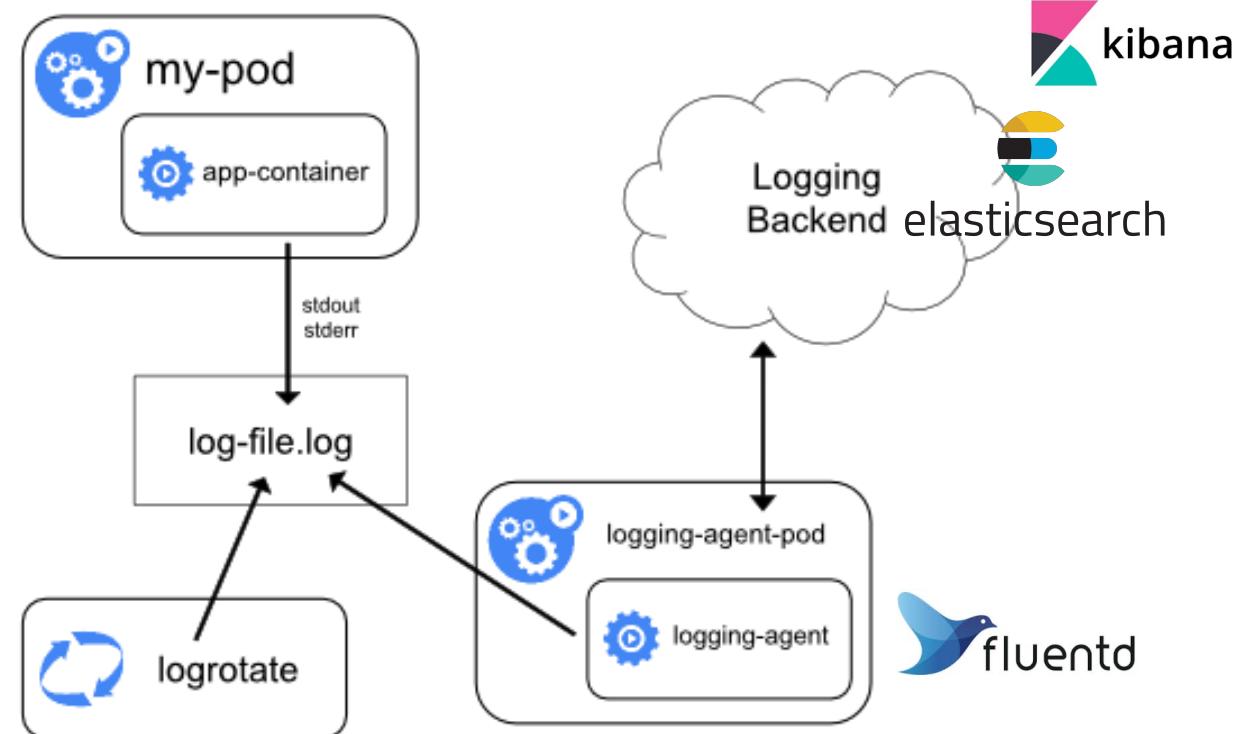
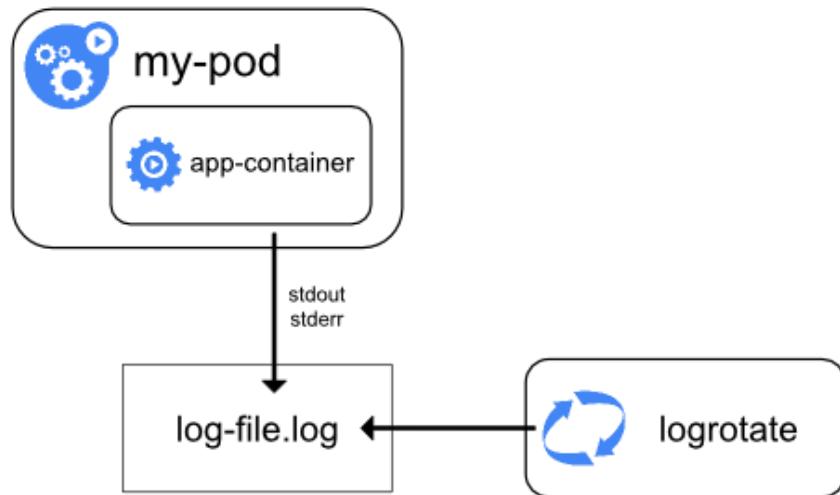
```
[root@osk-worker-02 containers]# head -3 /var/lib/docker/containers/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190/b7efee0b84250fd890a4e5275e6078c0e553e509ac56f100368bfa3650673190-json.log
```

```
{"log":"0: Sat Aug 21 03:18:01 UTC 2021\n","stream":"stdout","time":"2021-08-21T03:18:01.679793924Z"}
```

```
{"log":"1: Sat Aug 21 03:18:02 UTC 2021\n","stream":"stdout","time":"2021-08-21T03:18:02.681347872Z"}
```

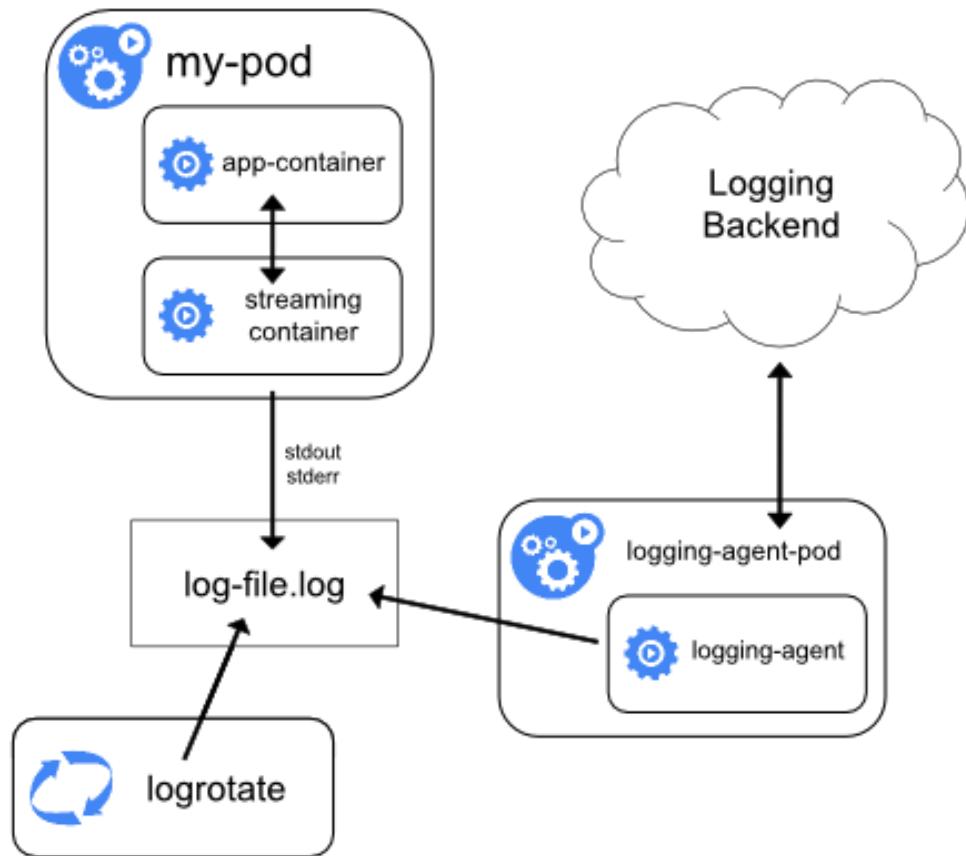
# Logging & Troubleshooting

- 쿠버네티스에서의 다른 로깅 아키텍쳐
- 노드 레벨에서의 로깅
- 클러스터 레벨에서의 로깅(노드 로깅 에이전트 사용)

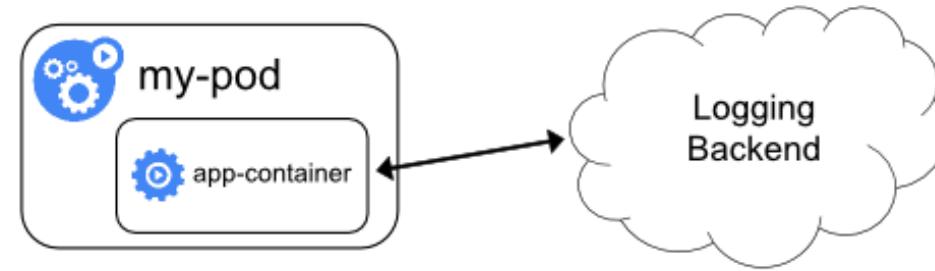


# Logging & Troubleshooting

- 쿠버네티스에서의 다른 로깅 아키텍쳐
- 로깅 에이전트와 함께 사이드카 컨테이너 사용
- 로깅 에이전트가 있는 사이드카 컨테이너



- 애플리케이션에서 직접 로그 노출



# Troubleshooting 절차/Debug 순서/참고 자료

과제

- Trouble Shooting 파일 전체 노드에 다운로드 및 실행
  - Pod 생성이 가능토록 조치하고, Pod가 생성된 화면 캡쳐 및 원인 / 해결방법 강사 메일로 제출 ([ohsk@kakao.com](mailto:ohsk@kakao.com))
- Trouble Shooting 절차/Debug 순서 : 로그를 확인, 쿠버네티스의 구조/특징을 생각
- 참고 자료 : 본 자료 앞 부분 쿠버네티스 구조

[ master노드 1~3, worker 노드 1~2 모두 ]

```
# wget http://172.30.5.154/troubleshooting  
# ./troubleshooting
```

```
[centos@osk-master-01 ~]$ ./troubleshooting
```

Create a pod. If you can't create pods, trouble shooting.

After the trouble shooting, capture the screen that created the pod and send an e-mail to [ohsk@kakao.com](mailto:ohsk@kakao.com) along with the cause and solution.

```
[centos@osk-worker-01 ~]$ ./troubleshooting
```

Create a pod. If you can't create pods, trouble shooting.

After the trouble shooting, capture the screen that created the pod and send an e-mail to [ohsk@kakao.com](mailto:ohsk@kakao.com) along with the cause and solution.

# 모니터링 클러스터 컴포넌트

- #kubectl top node
- #kubectl top pod
- (참고) 리눅스 TOP 명령어 (table of process) : CPU/메모리 정보 표시

```
[root@osk-master-01 ~]# kubectl top --help
Display Resource (CPU/Memory) usage.
```

The top command allows you to see the resource consumption for nodes or pods.  
This command requires Metrics Server to be correctly configured and working on the server.

Available Commands:

```
node      Display Resource (CPU/Memory) usage of nodes
pod      Display Resource (CPU/Memory) usage of pods
```

Usage:

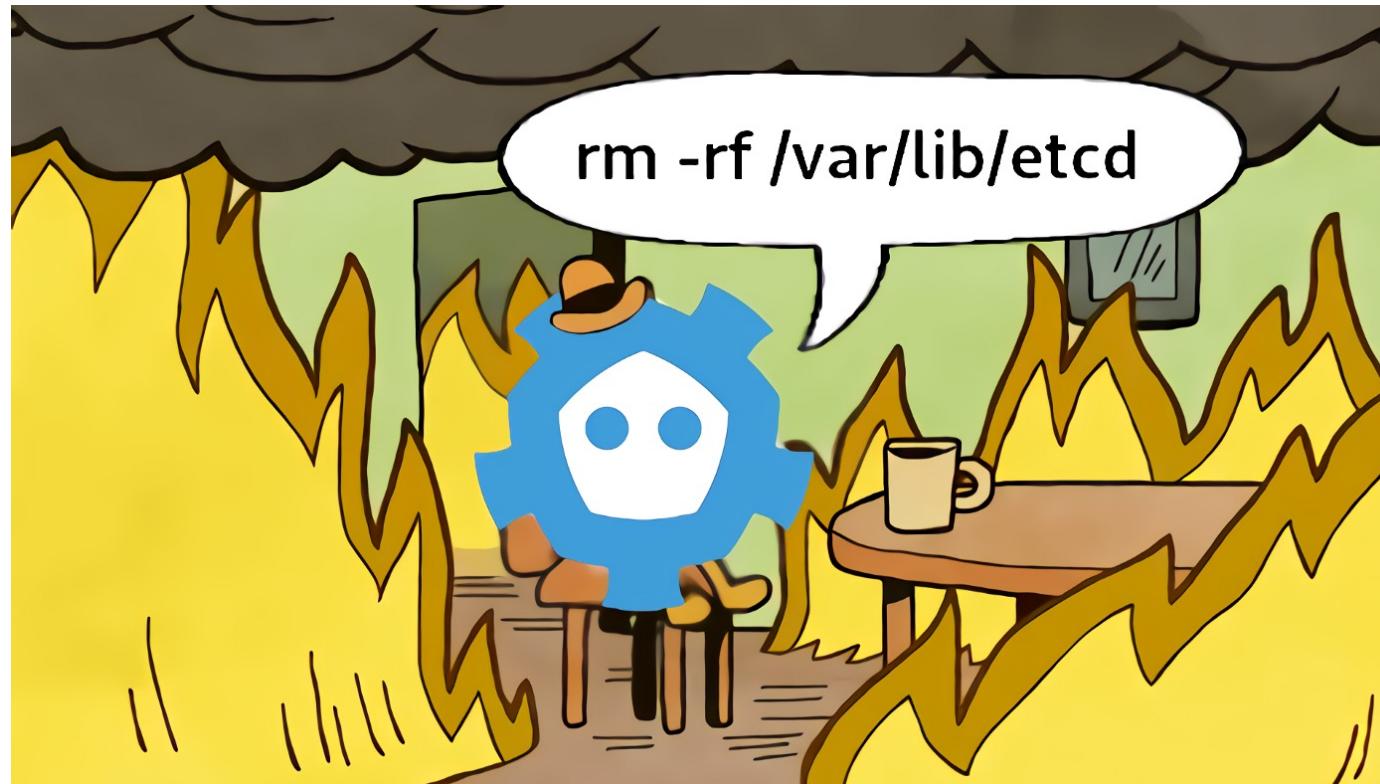
```
kubectl top [flags] [options]
```

Use "kubectl <command> --help" for more information about a given command.

Use "kubectl options" for a list of global command-line options (applies to all commands).

# 백업 및 복구

- 쿠버네티스 클러스터의 정보들을 갖고 있는 etcd.
- Etcd 문제 발생 시, 쿠버네티스 클러스터 전체가 영향을 받게 됨



# 백업 및 복구

- Pod로 구동된 etcd 버전 확인 후 해당 etcdctl 설치 (<https://github.com/etcd-io/etcd/releases>)

```
[centos@osk-master-01 ~]$ kubectl describe -n kube-system pod etcd-osk-master-01.kr-central-1.c.internal
```

Name: etcd-osk-master-01.kr-central-1.c.internal

Namespace: kube-system

생략

Controlled By: Node/osk-master-01.kr-central-1.c.internal

Containers:

etcd:

Container ID: docker://f8a0505c83155284b02973a4e2bcb2f10fa9c6d5017871ed5ec821a3c594b33a

Image: k8s.gcr.io/[etcd:3.5.0-0](#)

```
[centos@osk-master-01 ~]# ETCD_VER=v3.5.0
```

```
[centos@osk-master-01 ~]# wget https://storage.googleapis.com/etcd/${ETCD_VER}/etcd-${ETCD_VER}-linux-amd64.tar.gz
```

```
[centos@osk-master-01 ~]# tar xzvf etcd-${ETCD_VER}-linux-amd64.tar.gz
```

```
[centos@osk-master-01 ~]# sudo mv etcd-${ETCD_VER}-linux-amd64/etcdctl /usr/local/bin/etcdctl
```

```
[centos@osk-master-01 ~]# etcdctl version
```

etcdctl version: 3.5.0

API version: 3.5

# 백업 및 복구



- etcd backup/restore 명령어 북마크!
  - <https://discuss.kubernetes.io/t/etcd-backup-and-restore-management/11019/11>
- 백업    ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt \ --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key \ snapshot save /tmp/snapshot-pre-boot.db
- 복구    ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt \ --name=master \ --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key \ --data-dir /var/lib/etcd-from-backup \ --initial-cluster=master=https://127.0.0.1:2380 \ --initial-cluster-token etcd-cluster-1 \ --initial-advertise-peer-urls=https://127.0.0.1:2380 \ snapshot restore /tmp/snapshot-pre-boot.db

# 백업 및 복구



- etcd backup/restore 명령어 북마크!
  - <https://discuss.kubernetes.io/t/etcd-backup-and-restore-management/11019/11>
- 백업    ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key snapshot save /tmp/snapshot-pre-boot.db
  - 백업 명령어
  - 백업을 저장할 경로/이름
  - 각 master node ip
  - 현재 구동된 etcd에서 확인  
(인증서/키 경로)
- 복구    ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --name=master --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key --data-dir /var/lib/etcd-from-backup --initial-cluster=master=https://127.0.0.1:2380 --initial-cluster-token etcd-cluster-1 --initial-advertise-peer-urls=https://127.0.0.1:2380 snapshot restore /tmp/snapshot-pre-boot.db
  - 복구 명령어
  - 복구할 파일이 저장된 경로/이름
  - 기존 백업시에 설정되었던 값  
(or 문제에서 주어진 값)<sup>145</sup>

# 백업 및 복구



- etcd backup/restore 명령어 북마크!
  - <https://discuss.kubernetes.io/t/etcd-backup-and-restore-management/11019/11>
- 백업    ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key  
snapshot save /tmp/snapshot-pre-boot.db
  - (etcd 데이터를 저장할 경로  
(기존 데이터 덮어쓰지 않게 다른 공간 추천))
- 복구    ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --name=master --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key --data-dir /var/lib/etcd-from-backup --initial-cluster=master=https://127.0.0.1:2380 --initial-cluster-token etcd-cluster-1 --initial-advertise-peer-urls=https://127.0.0.1:2380 snapshot restore /tmp/snapshot-pre-boot.db

# 백업 및 복구



- etcd backup/restore 명령어 북마크!
  - <https://discuss.kubernetes.io/t/etcd-backup-and-restore-management/11019/11>
- 백업    ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key  
snapshot save /tmp/snapshot-pre-boot.db
- 복구    ETCDCTL\_API=3 etcdctl --endpoints=https://[127.0.0.1]:2379 --cacert=/etc/kubernetes/pki/etcd/ca.crt --name=master --cert=/etc/kubernetes/pki/etcd/server.crt --key=/etc/kubernetes/pki/etcd/server.key --data-dir /var/lib/etcd-from-backup --initial-cluster=master=https://127.0.0.1:2380 --initial-cluster-token etcd-cluster-1 --initial-advertise-peer-urls=https://127.0.0.1:2380 snapshot restore /tmp/snapshot-pre-boot.db

Etcd는 static Pod로 구성되어 있음.

Static Pod는 어떻게 구성되었는지 떠올리기!

etcd 데이터를 저장할 경로  
(기존 데이터 덮어쓰지 않게 다른 공간 추천)

# 백업 및 복구

과제

- 마스터 노드에 snapshot 파일 다운로드 및 복구
  - 복구 된 후 Pod 상태를 조회하고, 화면 캡쳐 하여 강사 메일로 제출 ([ohsk@kakao.com](mailto:ohsk@kakao.com))
  - 3개 master 노드를 모두 복구 하지 않고 1개만 복구 완료하여도 Pass로 인정
- Hint1 : Static Pod의 Volumes 수정
- Hint2 : 현재 각자 환경에 설치되어 실행중인 etcd config 준용

[ master노드 ]

```
# wget http://172.30.5.154/snapshot-pre-boot.db
```

```
[root@dyjs-master-01 manifests]# kubectl get pod
```

| NAME                                         | READY | STATUS  | RESTARTS | AGE |
|----------------------------------------------|-------|---------|----------|-----|
| capture-the-screen-and-send-an-email-to-ohsk | 1/1   | Running | 0        | 12m |
|                                              | 1/1   | Running | 0        | 12m |
|                                              | 1/1   | Running | 0        | 12m |

복구 완료 되면 어떤 Pod가 생기는지 확인



그동안 수고 많으셨습니다



# (참고) Kakao Kubernetes Engine 사용하기

( 참고: [https://console.kakaoi.io/docs/posts/k8se/k8se\\_htg/2021-05-31-k8se\\_htg\\_settingKubectl/k8se\\_htg\\_settingKubectl](https://console.kakaoi.io/docs/posts/k8se/k8se_htg/2021-05-31-k8se_htg_settingKubectl/k8se_htg_settingKubectl))

The screenshot shows a web browser displaying the Kakao Kubernetes Engine User Guide. The URL in the address bar is [https://console.kakaoi.io/docs/posts/k8se/k8se\\_htg/2021-05-31-k8se\\_htg\\_settingKubectl/k8se\\_htg\\_settingKubectl](https://console.kakaoi.io/docs/posts/k8se/k8se_htg/2021-05-31-k8se_htg_settingKubectl/k8se_htg_settingKubectl). The page title is "kubectl 제어 설정하기". The left sidebar has a navigation tree: Home, Virtual Machine, VPC, Kubernetes Engine (expanded), Overview, Concepts, How-to guides (expanded), Cluster 만들기, Cluster 관리하기, **kubectl 제어 설정하기** (highlighted in blue), Ingress Controller 설정하기, Load Balancer 생성하기. The main content area has two sections: "kubectl 설치 및 설정하기" and "kubectl 클라이언트 설치". The "kubectl 설치 및 설정하기" section contains a note about installing the client and a link to the official guide. The "kubectl 클라이언트 설치" section contains a note about installing the client and a link to the official guide.

Home

> Virtual Machine

> VPC

▼ Kubernetes Engine

- Overview
- Concepts
- How-to guides
  - 클러스터 만들기
  - 클러스터 관리하기
  - **kubectl 제어 설정하기**
  - 인그레스 컨트롤러 설정하기
  - 로드밸런서 생성하기

Kubernetes Engine - 카카오의 노하우가 집약된 클라우드

사용자 가이드 - kubectl 제어 설정하기

kakao i cloud | User Guide

Search

## kubectl 제어 설정하기

클러스터를 제어하기 위한 Kubernetes 커맨드 라인 도구인 **kubectl** 제어 설정 방법을 안내합니다. 여기에는 **kubectl** 클라이언트 설치 방법과 클러스터 접근 설정을 위한 **kubeconfig** 파일 다운로드 방법이 포함됩니다.

## kubectl 설치 및 설정하기

### kubectl 클라이언트 설치

Kubernetes 공식 가이드 문서를 따라 kubectl 클라이언트를 설치합니다. 사용자의 OS 환경에 따라 설치 방법을 확인 후 설치를 진행해주세요.

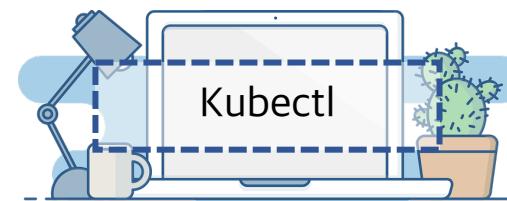
### kubectl 제어 설정

kubectl 클라이언트를 설치한 후, 다음을 따라 클러스터에 대한 kubectl 제어 설정을 진행합니다.

# (참고) Kakao Kubernetes Engine 사용하기 - PC환경설정

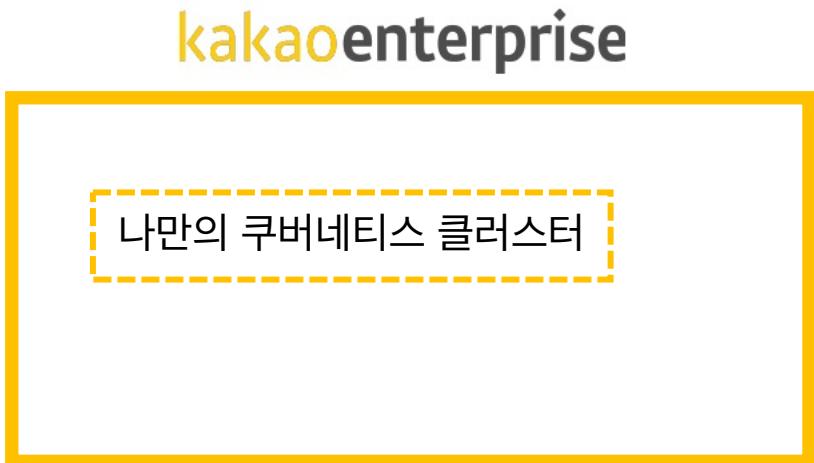
- 쿠버네티스 클러스터에 명령어를 내릴 수 있는 CLI 도구
- 애플리케이션 배포/ 클러스터 리소스 검사, 관리, 로그 확인 등 가능
  - ✓ 리눅스에 kubectl 설치하기 : <https://kubernetes.io/ko/docs/tasks/tools/install-kubectl-linux/>
  - ✓ macOS에 kubectl 설치하기 : <https://kubernetes.io/ko/docs/tasks/tools/install-kubectl-macos/>
  - ✓ 윈도우에 kubectl 설치하기 : <https://kubernetes.io/ko/docs/tasks/tools/install-kubectl-windows/>

kakaoenterprise



# (참고) Kakao Kubernetes Engine 사용하기 - 클러스터 배포

- Kubernetes Engine > 새 클러스터 만들기
  - ✓ 클러스터 이름 : 수강생 간 구분될 수 있게 가급적 이름 약자를 포함해서 작성
  - ✓ 노드 풀 : lkln
  - ✓ 인스턴스타입 : a1.4c16m
  - ✓ 노드수 : 3



새 클러스터 만들기

| 클러스터 정보  | 클러스터 이름  |
|----------|----------|
| osk-test | osk-test |

| 네트워크 구성 | 네트워크                |
|---------|---------------------|
| 관리 페이지  | likelion-private-01 |
| 서브넷     | likelion-private-01 |

| 노드 풀     | 노드 풀 이름  |
|----------|----------|
| likelion | likelion |

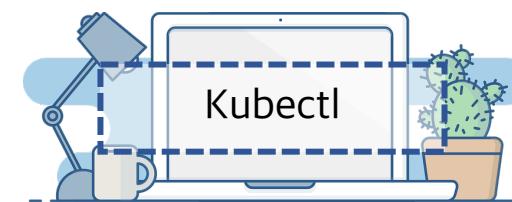
| 노드 풀 설명 (선택) | 60자 이내로 작성 |
|--------------|------------|
|              |            |

| 인스턴스 타입 | a1.4c16m (4vCPU 16GB 메모리) |
|---------|---------------------------|
|         |                           |

| 볼륨 타입 / 크기 | SSD | 50 |
|------------|-----|----|
|            |     |    |

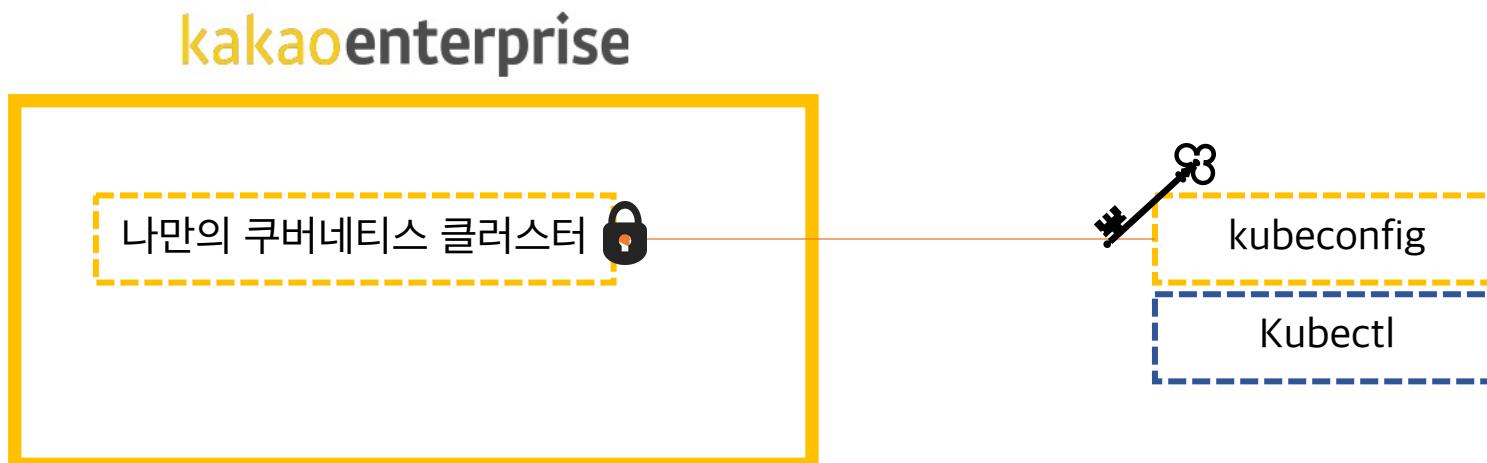
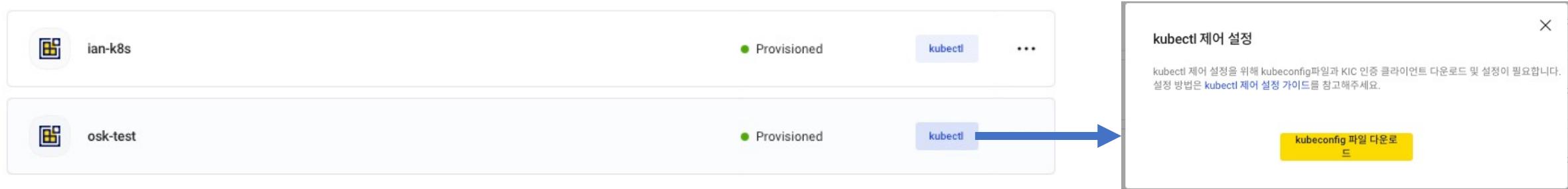
| 노드 수 | 3 |
|------|---|
|      |   |

| 키페어 | kakao-osk |
|-----|-----------|
|     |           |



# (참고) Kakao Kubernetes Engine 사용하기 - 클러스터 배포

- Kubectl을 클릭하여 클러스터 접근 정보가 담긴 Kubeconfig 파일 다운로드 (파일명 : kubeconfig-{clusterName})

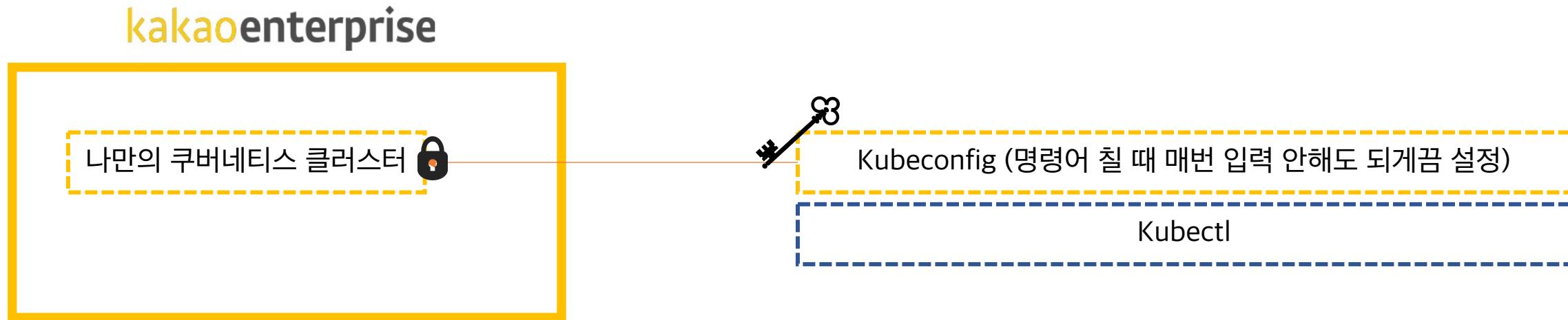


```
osk-MacBook-Pro:lkln-kakao osk$ cat kubeconfig-osk-test.yaml
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCkI
  XcwQkFRc0ZBREFWTVJNjd0VRWURUVFRXdwcmRXSmwKY201bGRHVnpNnjRyRFRJeE1EY3p
  ZURVRN0kVHOTFVRQpBeE1LYTNwVaVpYSnVawfJsjY3pDQ0fTSXdeUV1KS29aSwH2Y5BUUVCC
  wdm05Mut1QXh1TGws3ljMG1BamiaS0NzWGthY0tZTX1PeFNERUg420@0ekdzac5cj1KNv
  RE1oSznR0jZ3bkxJcWhuk256bdQRQL1mFA1Z21XVFRLSmNaZQorVmNx0DF2bXYrRUTPMV:
  @hEbWrsREVQD1@0N9aw1mCld@0Ux@0dwFKM05VRTVqczdub212Um@CemVnbwC1khjdzl
  @hMwVR2mt3k0JISGVn9yeTZBZ01PZFgTTZpcWY2QJFRU1HZ0U2dj1SMXRSL1Vi0dxB
  3RFuFYU1tTUNRd0RnWURWU1BQQVFI0JBURBZ0tTJUJR0ExxWRF@VCCi93U01NOV1CQV
  RWJEWk4xbvRyNzRIS1b1VEM0TRUaFQKTG82VXFZGg4eGovU61Ujh3KzdVUFJuOVNUnt
  WdEUopMaUk0Zn1xYf2ejB3bEtMZXJZZKRISnycOHNZSDBZWUs5akV5L0N1L05xwFY5dVht
  51R2dPQUp1VG9ndG1iL230MmlutTZRcjhx2d9nbGpjdkNrP2xzWF1NWES301mTGEKRxdtK
  rb1d4ZzditT2U4em16M2V4Q2fDeWNRajZyV2VylwpV0jZPbnhdUFFocUZTTWpuWVNsSj0cna
  RVJFRStsaTA9C10tLS0tRU5IEFU1RJK1DQVRFLS0tLS0K
    server: https://10.183.67.34:6443
    name: osk-test
  contexts:
  - context:
    cluster: osk-test
    user: osk-test-admin
    name: osk-test-admin@osk-test
  current-context: osk-test-admin@osk-test
  kind: Config
  preferences: {}
  users:
  - name: osk-test-admin
    user:
      exec:
        apiVersion: client.authentication.k8s.io/v1beta1
        args: null
        command: kic-keystone-auth
        env: null
osk-MacBook-Pro:lkln-kakao osk$ _
```

# (참고) Kakao Kubernetes Engine 사용하기 - PC 환경설정

- PC에서 kubectl 명령어를 쉽게 입력할 수 있도록, 다운받은 Kubeconfig 을 활용한 환경설정 진행
- ✓ 맥북(리눅스) : export KUBE\_CONFIG="{DownloadPath}/{kubeconfigFile}"
- ✓ 윈도우 : set KUBE\_CONFIG={DownloadPath}\{kubeconfigFile}

```
osk@osk-MacBook-Pro ~ % export KUBE_CONFIG=/Users/osk/Desktop/lkln-kakao/kubeconfig-osk-test.yaml
```



# (참고) Kakao Kubernetes Engine 사용하기 - PC 환경설정

- (추가 편의) 재부팅 마다 설정 하지 않도록 추가 세팅

✓ 맥북(리눅스) : export \$KUBE\_CONFIG >> ~/.zshrc

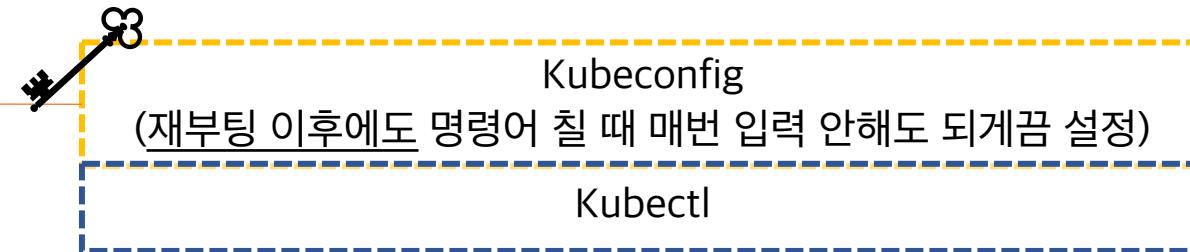
```
osk@osk-MacBook-Pro ~ % echo $KUBE_CONFIG  
/Users/osk/Desktop/lkln-kakao/kubeconfig-osk-test.yaml  
osk@osk-MacBook-Pro ~ % echo $KUBE_CONFIG >> ~/.zshrc
```

[맥북용 카카오 인증 파일은 zsh에서만 구동됨]

```
osk-MacBook-Pro:~ osk$  
osk-MacBook-Pro:~ osk$ echo $SHELL  
/bin/bash  
osk-MacBook-Pro:~ osk$ chsh -s /bin/zsh  
Changing shell for osk.  
Password for osk:  
osk-MacBook-Pro:~ osk$ zsh  
osk@osk-MacBook-Pro ~ %
```

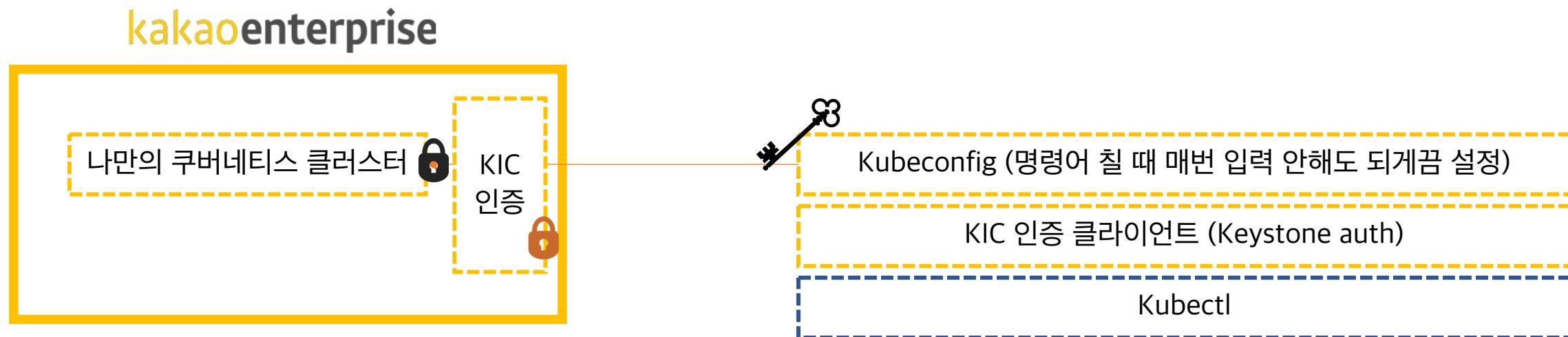
kakaoenterprise

나만의 쿠버네티스 클러스터 



# (참고) Kakao Kubernetes Engine 사용하기 - PC 환경설정

- KIC(Kakao i Cloud) 인증 클라이언트 다운로드 및 설정
- 사용자 인증을 위해 KIC 인증 클라이언트를 다운로드
  - 맥북 : [https://objectstorage.kr-central-1.kakaoi.io/v1/c901636667fe4668ae745ea7de035ae9/kic-keystone-auth-client/darwin\\_amd64/kic-keystone-auth](https://objectstorage.kr-central-1.kakaoi.io/v1/c901636667fe4668ae745ea7de035ae9/kic-keystone-auth-client/darwin_amd64/kic-keystone-auth)
  - 윈도우 : [https://objectstorage.kr-central-1.kakaoi.io/v1/c901636667fe4668ae745ea7de035ae9/kic-keystone-auth-client/windows\\_amd64/kic-keystone-auth.exe](https://objectstorage.kr-central-1.kakaoi.io/v1/c901636667fe4668ae745ea7de035ae9/kic-keystone-auth-client/windows_amd64/kic-keystone-auth.exe)
  - 리눅스 : [https://objectstorage.kr-central-1.kakaoi.io/v1/c901636667fe4668ae745ea7de035ae9/kic-keystone-auth-client/linux\\_amd64/kic-keystone-auth](https://objectstorage.kr-central-1.kakaoi.io/v1/c901636667fe4668ae745ea7de035ae9/kic-keystone-auth-client/linux_amd64/kic-keystone-auth)



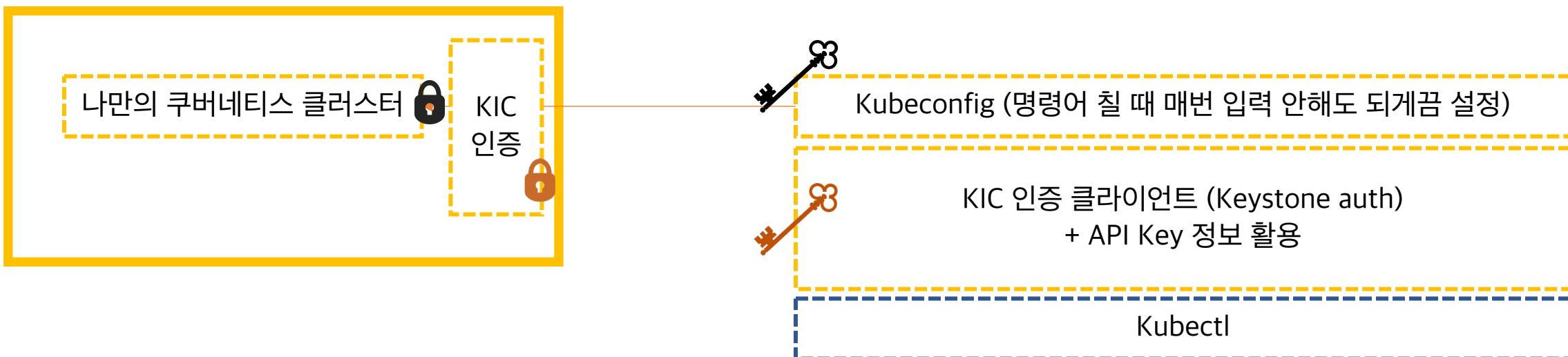
# (참고) Kakao Kubernetes Engine 사용하기 - API 키/인증 설정

- API 키 만들기

<https://console.kakaoi.io/settings/key>

The screenshot shows the 'User API Key' section of the Kakao Kubernetes Engine console. It displays a list of API keys for the 'osk-test' cluster. The first key listed is 'ike-cluster-member-osk-test-likelion-icloud', which is highlighted with a blue background. A tooltip for this key states: 'osk-test 클러스터가 사용하는 API 키입니다. 삭제 시 클러스터의 노드 제거가 제한됩니다.' (This is the API key used by the osk-test cluster. Deleting it will limit node removal). The interface includes tabs for 'User Settings' and 'User API Key'. A note at the top right says: '발급한 프로젝트의 역할이 삭제 시, 사용자 API 키는 자동으로 만료됩니다. 프로젝트 역할이 변경된 경우, 생성 시점과 일치하는 권한만 부여됩니다. [사용자 API 키](#) 사용자 가이드를 참고하세요.' (When a project role is deleted, the user API key is automatically expired. If the project role is changed, only the same permissions as the creation time will be granted. Refer to the [User API Key](#) user guide for more information).

kakaoenterprise

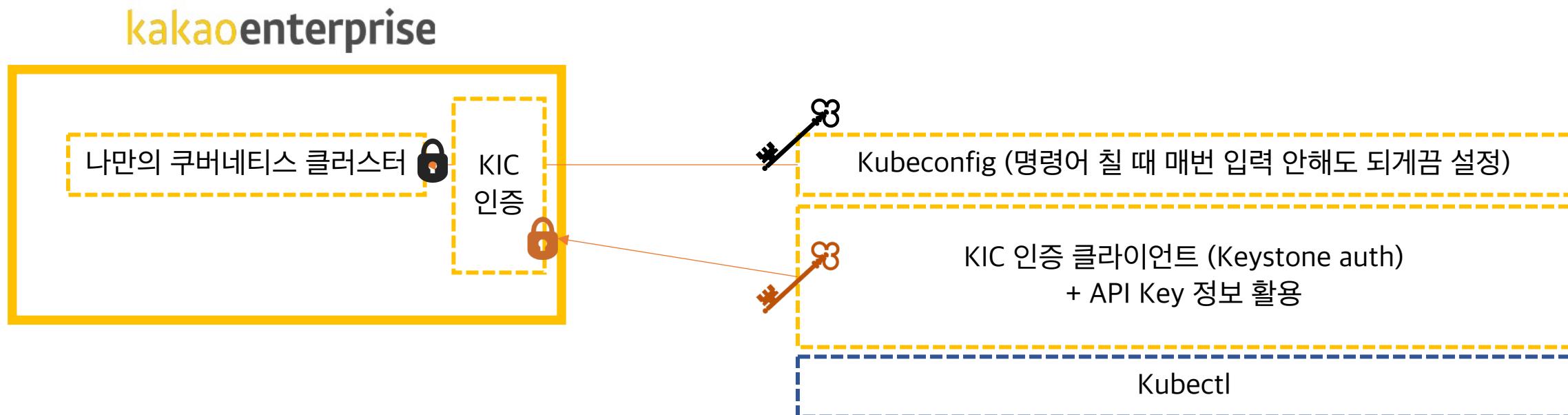


# (참고) Kakao Kubernetes Engine 사용하기 - API 키/인증 설정

- 만들어진 API Key 설정하기

```
osk@osk-MacBook-Pro lklm-kakao % export OS_AUTH_URL=https://keystone.kr-central-1.kakaoi.io/v3  
osk@osk-MacBook-Pro lklm-kakao % export OS_AUTH_TYPE=v3applicationcredential  
osk@osk-MacBook-Pro lklm-kakao % export OS_APPLICATION_CREDENTIAL_ID=각자 웹에서 확인  
osk@osk-MacBook-Pro lklm-kakao % export OS_APPLICATION_CREDENTIAL_SECRET=각자 웹에서 확인 (한번밖에 안보이므로 잘 기입)  
osk@osk-MacBook-Pro lklm-kakao % export OS_REGION_NAME=kr-central-1  
osk@osk-MacBook-Pro lklm-kakao %  
osk@osk-MacBook-Pro lklm-kakao % vi ~/.zshrc
```

(재부팅시 편의를 위해 설정하는 단계임. .zshrc 파일을 열고 위에 export 부터 값까지 5줄 모두 복붙)



# (참고) Kakao Kubernetes Engine 사용하기 - API 키/인증 설정

- 다운로드 인증 클라이언트에 실행 권한 부여 → (어떤 위치에서든 쉽게 실행가능토록) PATH 경로로 복사

```
osk@osk-MacBook-Pro lkln-kakao % pwd
```

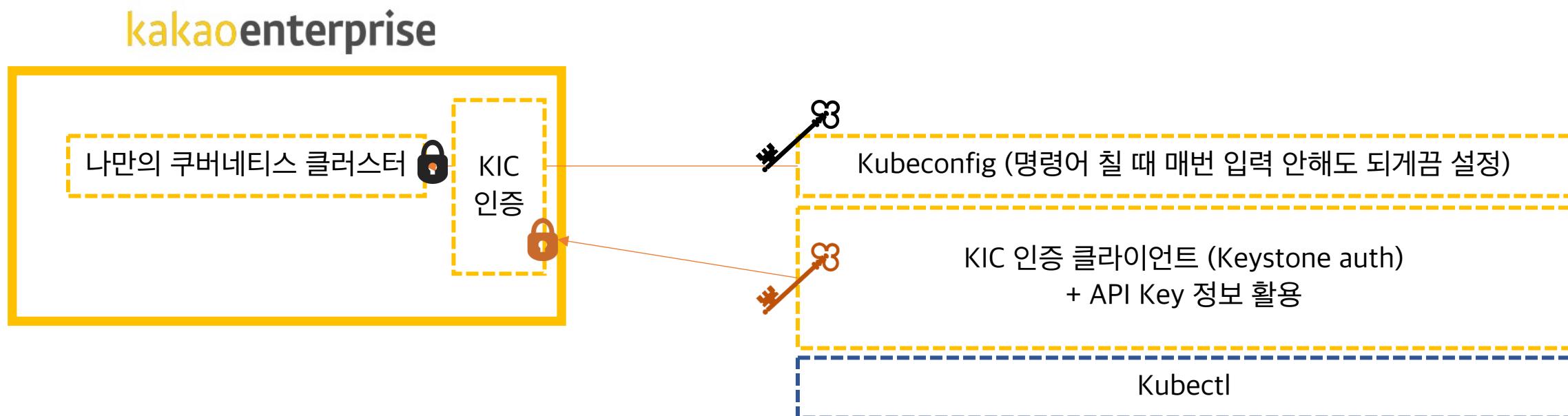
```
/Users/osk/Desktop/lkln-kakao
```

```
osk@osk-MacBook-Pro lkln-kakao % chmod +x kic-keystone-auth
```

```
osk@osk-MacBook-Pro lkln-kakao %
```

```
osk@osk-MacBook-Pro lkln-kakao % cp kic-keystone-auth /usr/local/bin
```

```
osk@osk-MacBook-Pro lkln-kakao %
```



# (참고) Kakao Kubernetes Engine 사용하기 - API 키/인증 설정

※ 맥에서 실행이 어려우면 Finder에서 Control 누르고 클릭

- <https://support.apple.com/ko-kr/guide/mac-help/mh40616/mac>



kakaoenterprise

