초보를 위한 쿠버네티스 안내서

쿠버네티스 알아보기

KIM CHUNG SUB

subicura

쿠버네티스 시작하기

컨테이너 오케스트레이션

왜 쿠버네티스인가?

어떤걸 배울까?

쿠버네티스 알아보기

쿠버네티스 소개

쿠버네티스 아키텍처

실습

쿠버네티스 실습하기

쿠버네티스 소개

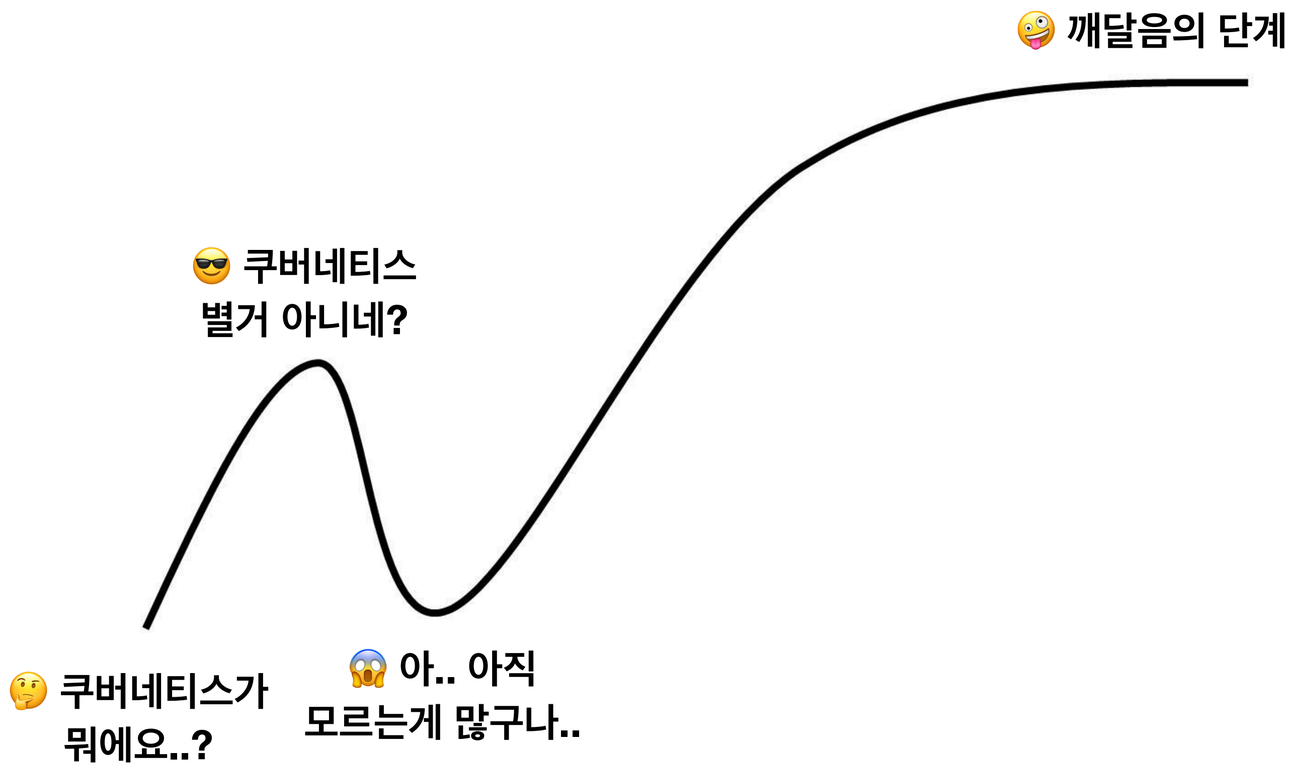
01

발음정리

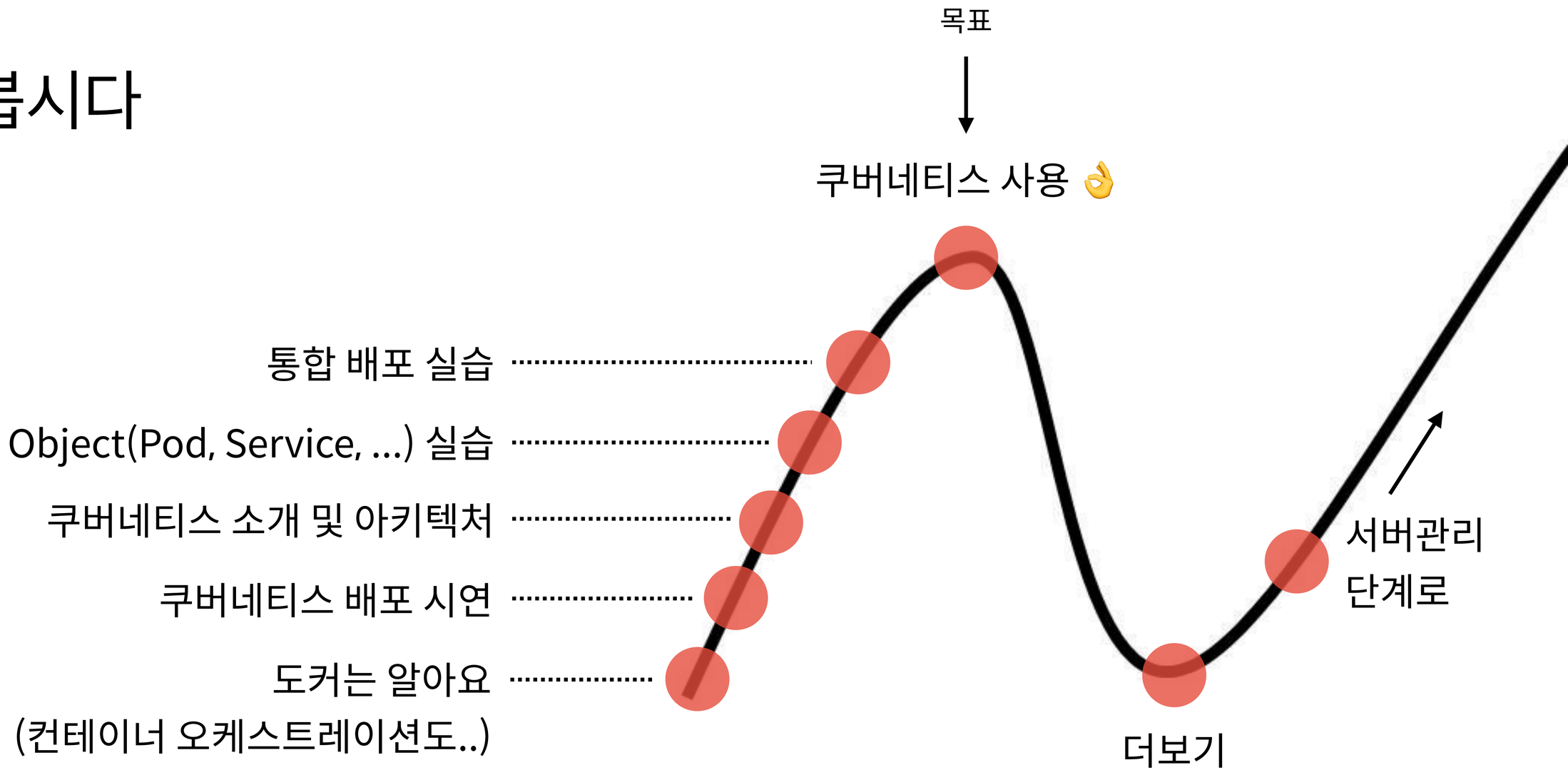
*의미가 통하는게 중요!

용어	발음
master	**마스터**
node	**노드** (구 minion 미니언)
k8s	**쿠버네티스**, 케이에잇츠, 케이팔에스
kubectl	**큐브 컨트롤**, 큐브 시티엘, 큐브커들
etcd	**엣지디**, 엣시디, 이티시디
pod	**팟**, 파드, 포드
istio	**이스티오**
helm	**헬름**, 햄, 햄
knative	**케이 네이티브**

Learning Curve



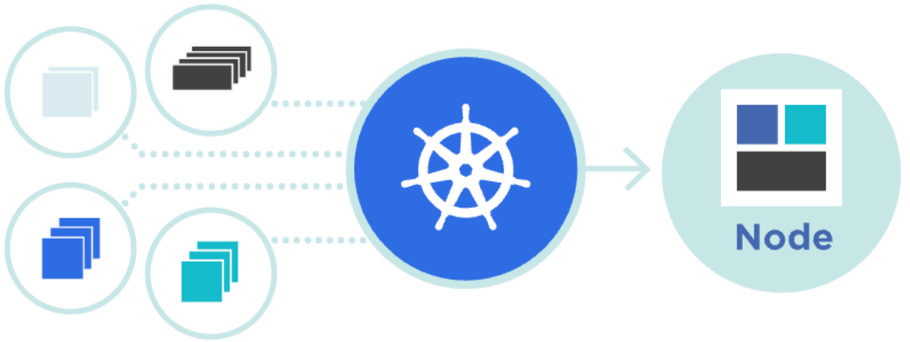
배워봅시다


목표
쿠버네티스 사용 👌
통합 배포 실습
Object(Pod, Service, ...) 실습
쿠버네티스 소개 및 아키텍처
쿠버네티스 배포 시연
도커는 알아요
(컨테이너 오케스트레이션도..)
더보기
서버관리
단계로

쿠버네티스 소개

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

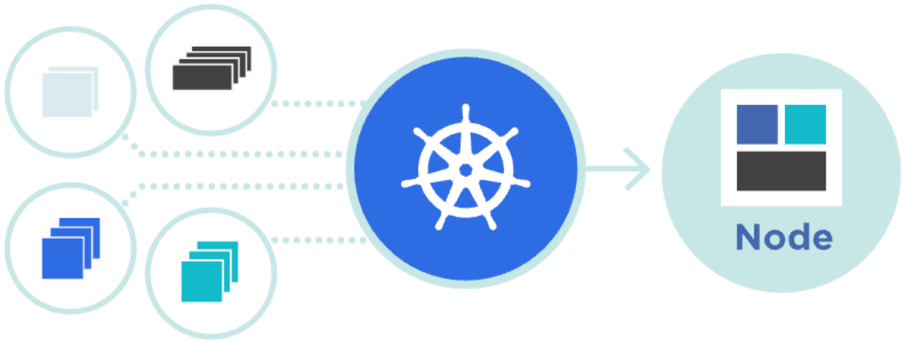


컨테이너화된 애플리케이션을 자동으로 배포, 스케일링 및 관리

쿠버네티스 소개

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

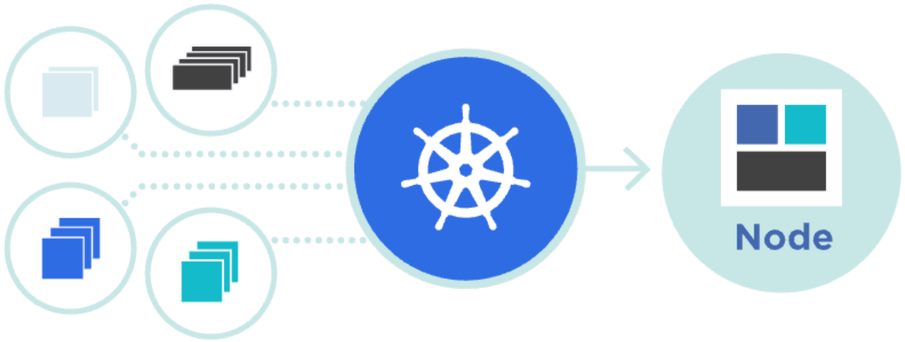


컨테이너를 쉽게 관리하고 연결하기 위해 논리적인 단위로 그룹화

쿠버네티스 소개

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

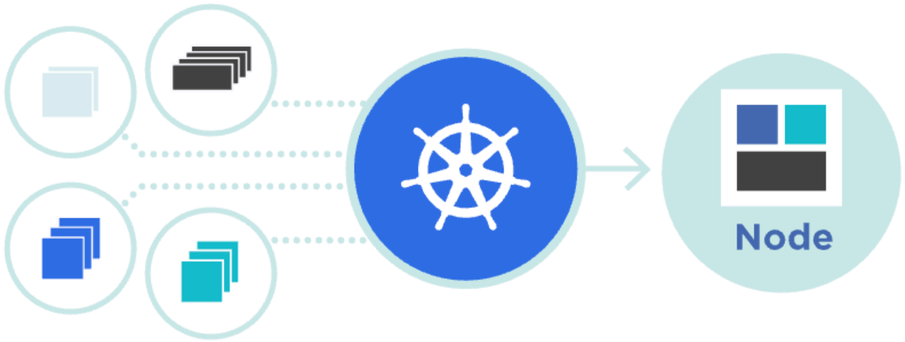


Google에서 15년간 경험을 토대로 최상의 아이디어와 방법들을 결합

쿠버네티스 소개

Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.



kubernetes

자동화

Pod	Node
ReplicaSet	Namespace
Deployment	Endpoint
StatefulSet	Volume
DaemonSet	Service
Job	Ingress...

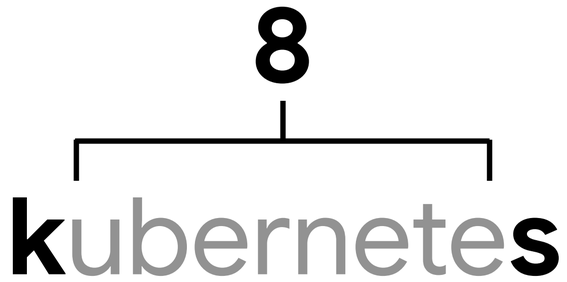
논리적인 단위

Microsoft IBM intel Red Hat

노하우

The name **Kubernetes** originates from Greek,
meaning **helmsman** or **pilot**.

쿠버네티스는 그리스어로 조타수 또는 조종사에서 따왔습니다



쿠버네티스는 줄여서 K8s라고도 합니다

kubernetes

쿠베 또는 큐브라고도 합니다

We are a CNCF graduated project

CLOUD NATIVE
COMPUTING FOUNDATION

클라우드 이전

리소스를 한 땀 한 땀 직접 관리

SERVER

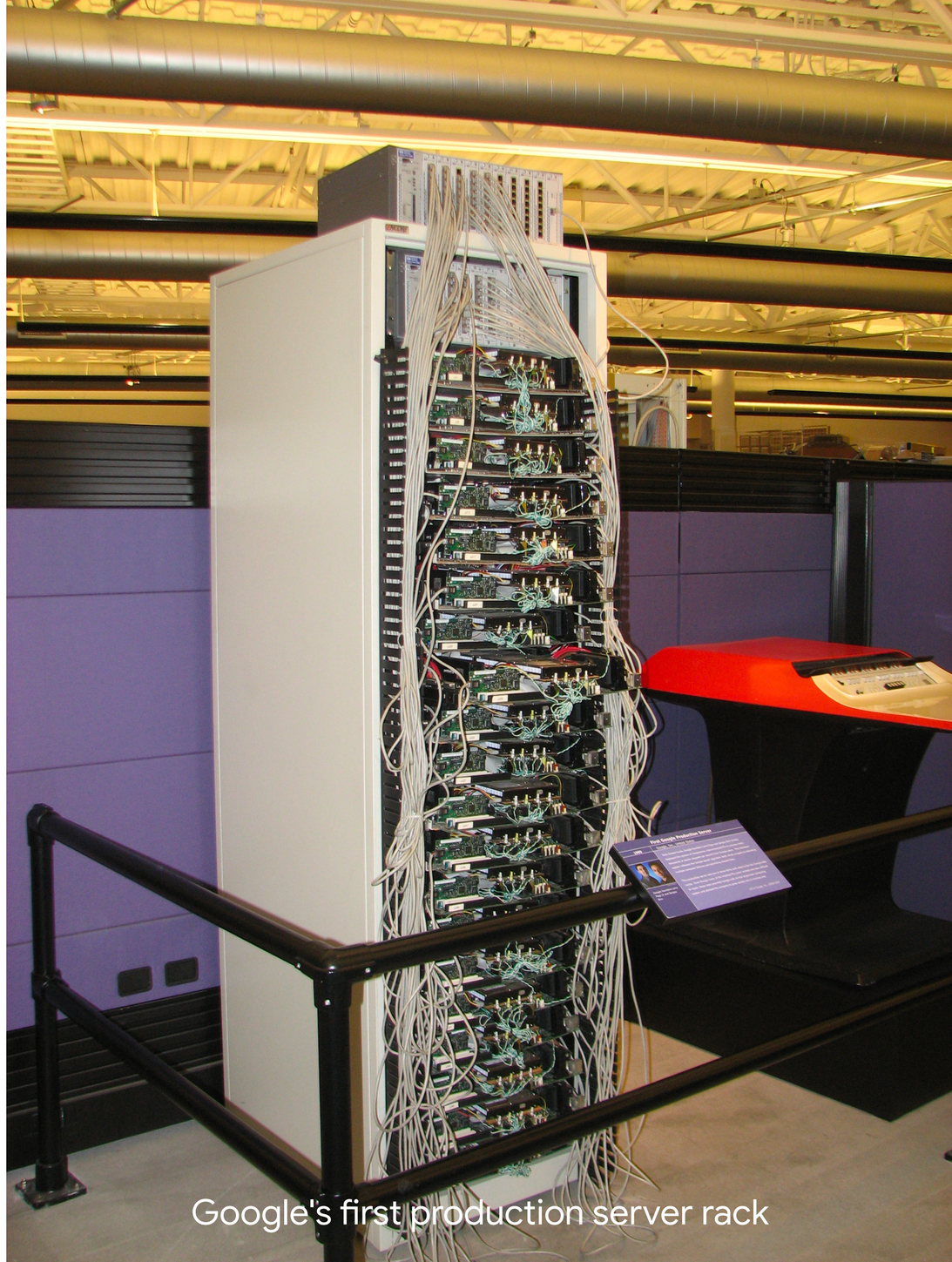
WEB-PROD
192.168.0.101
web

SERVER

API-PROD
192.168.0.102
api

SERVER

DB-PROD
192.168.0.103
db

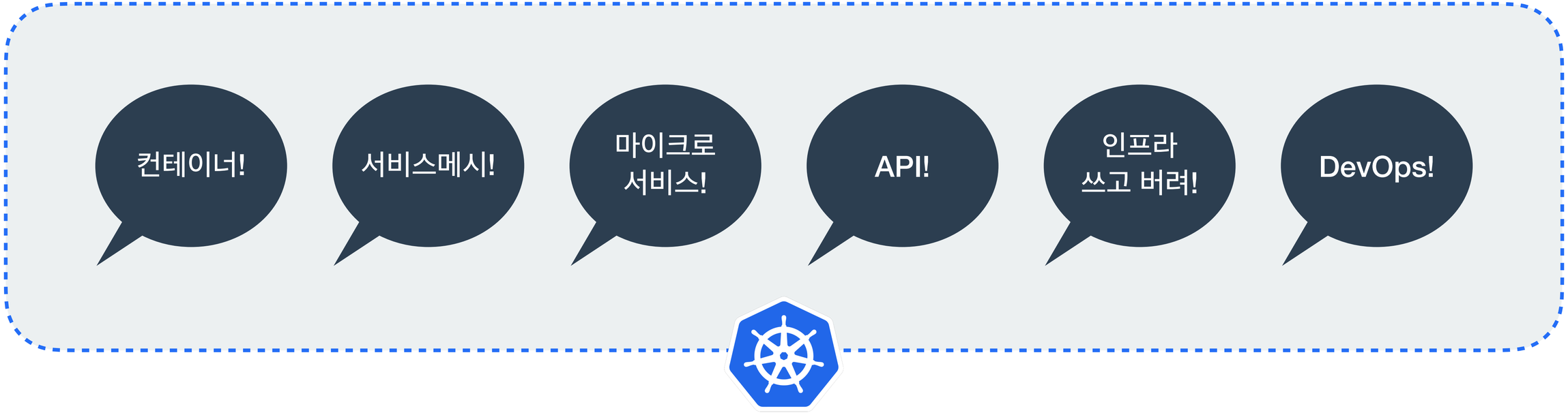


Google's first production server rack

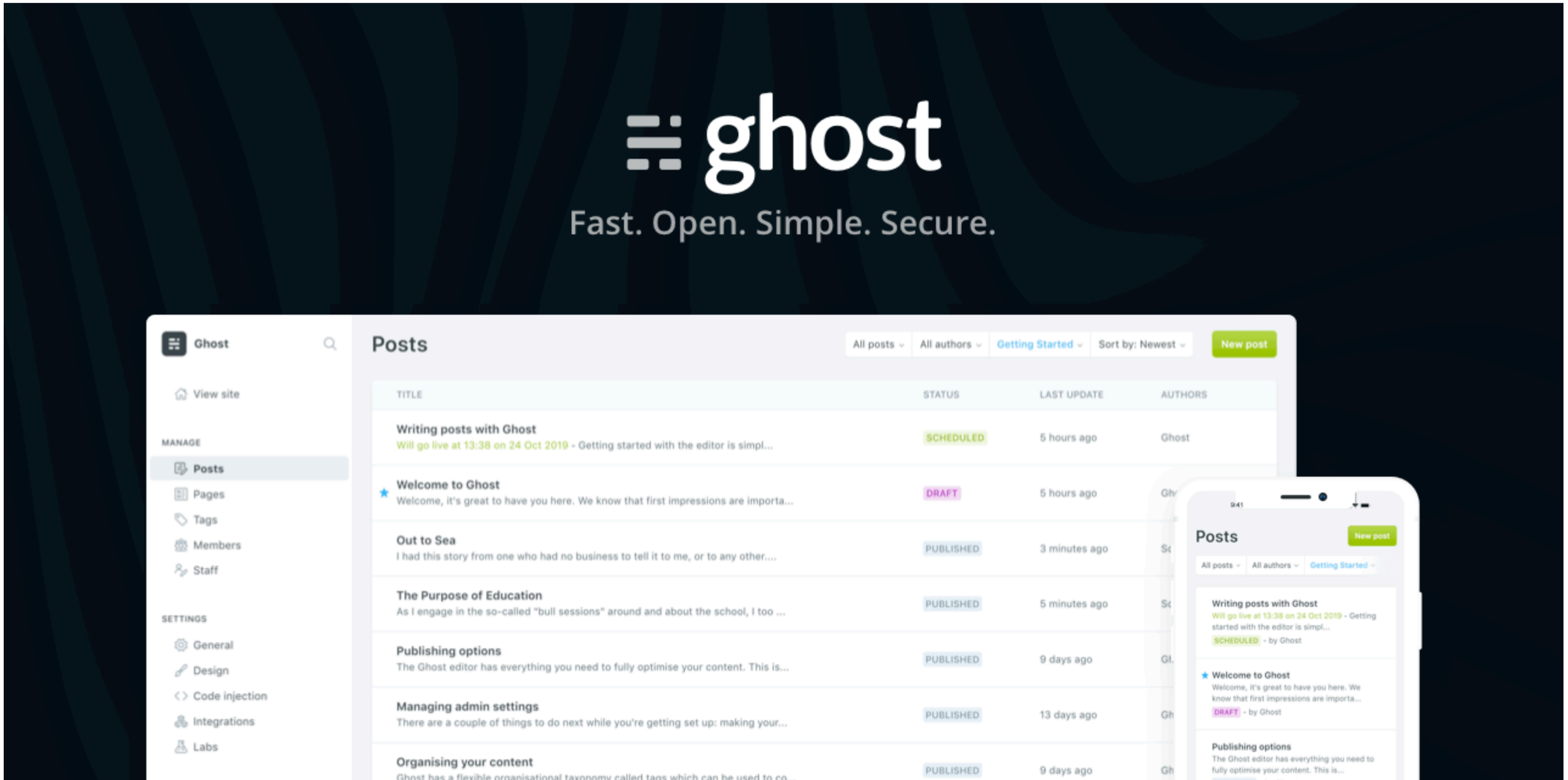
클라우드 환경에서 **어떻게 애플리케이션을 배포**하는게 좋은걸까?

CloudNative!

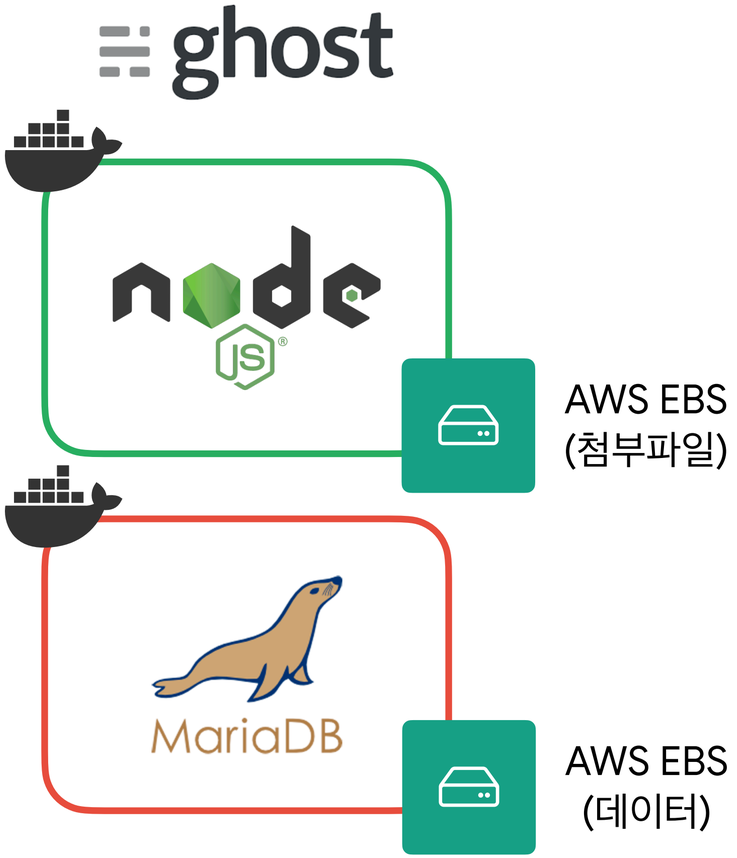
CloudNative 소개

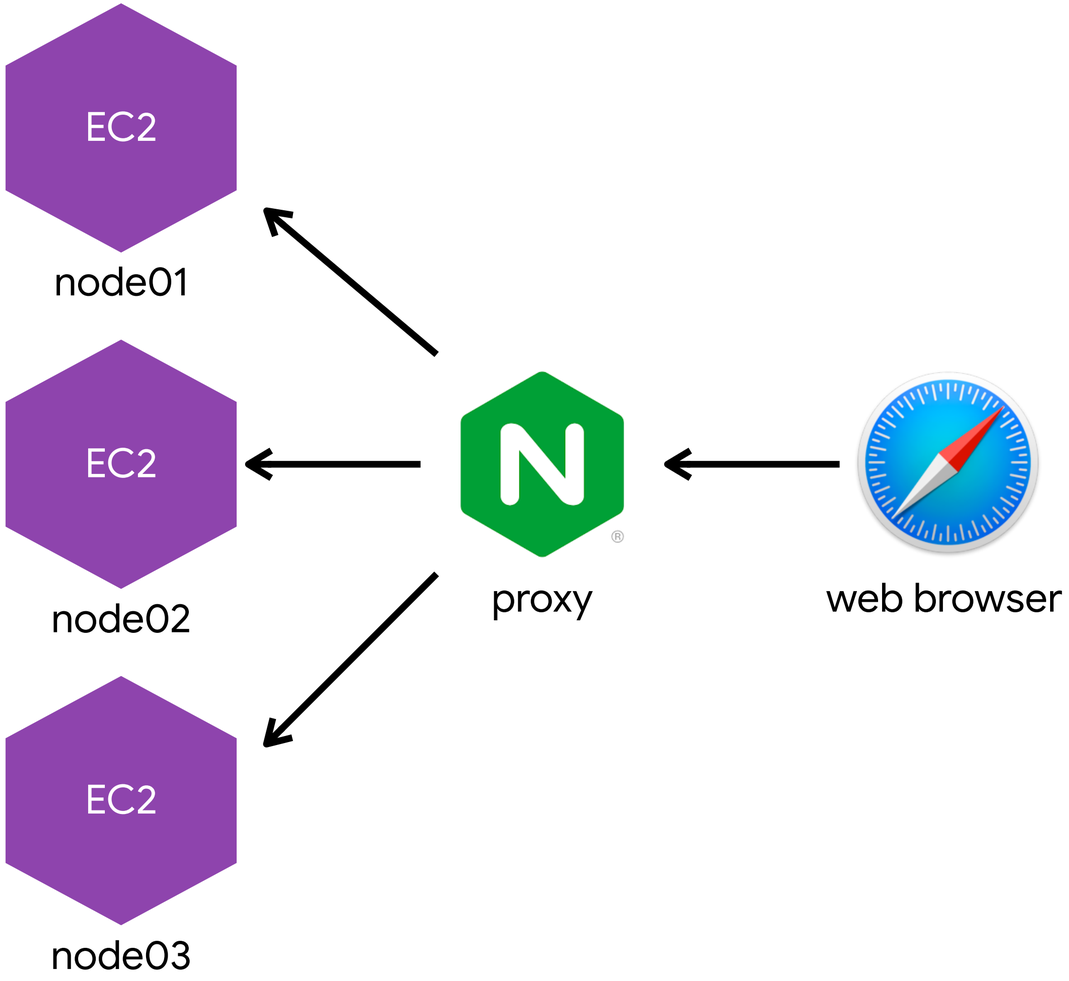


AWS 클라우드에 Ghost 블로그 배포

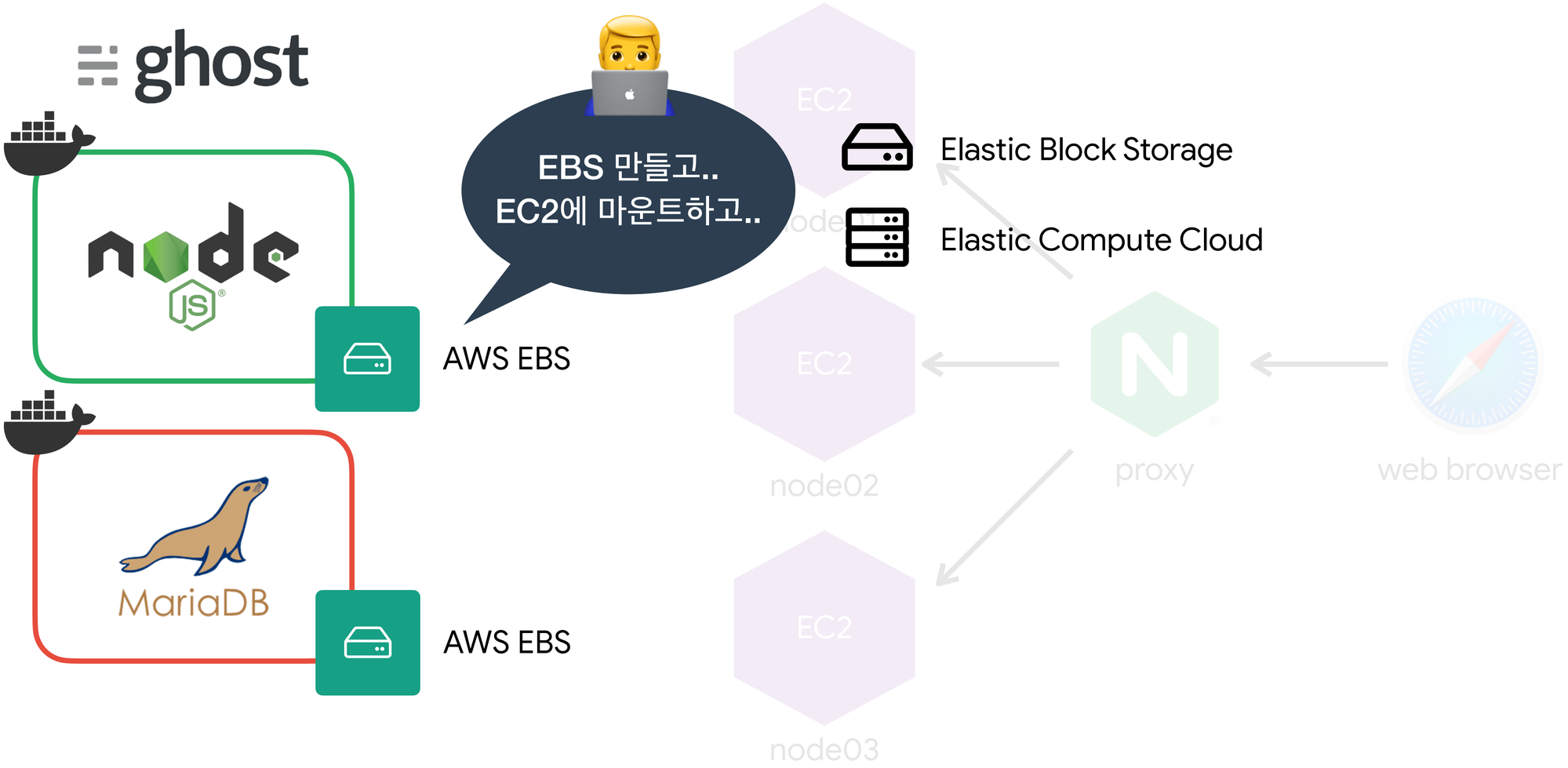


쿠버네티스 데모

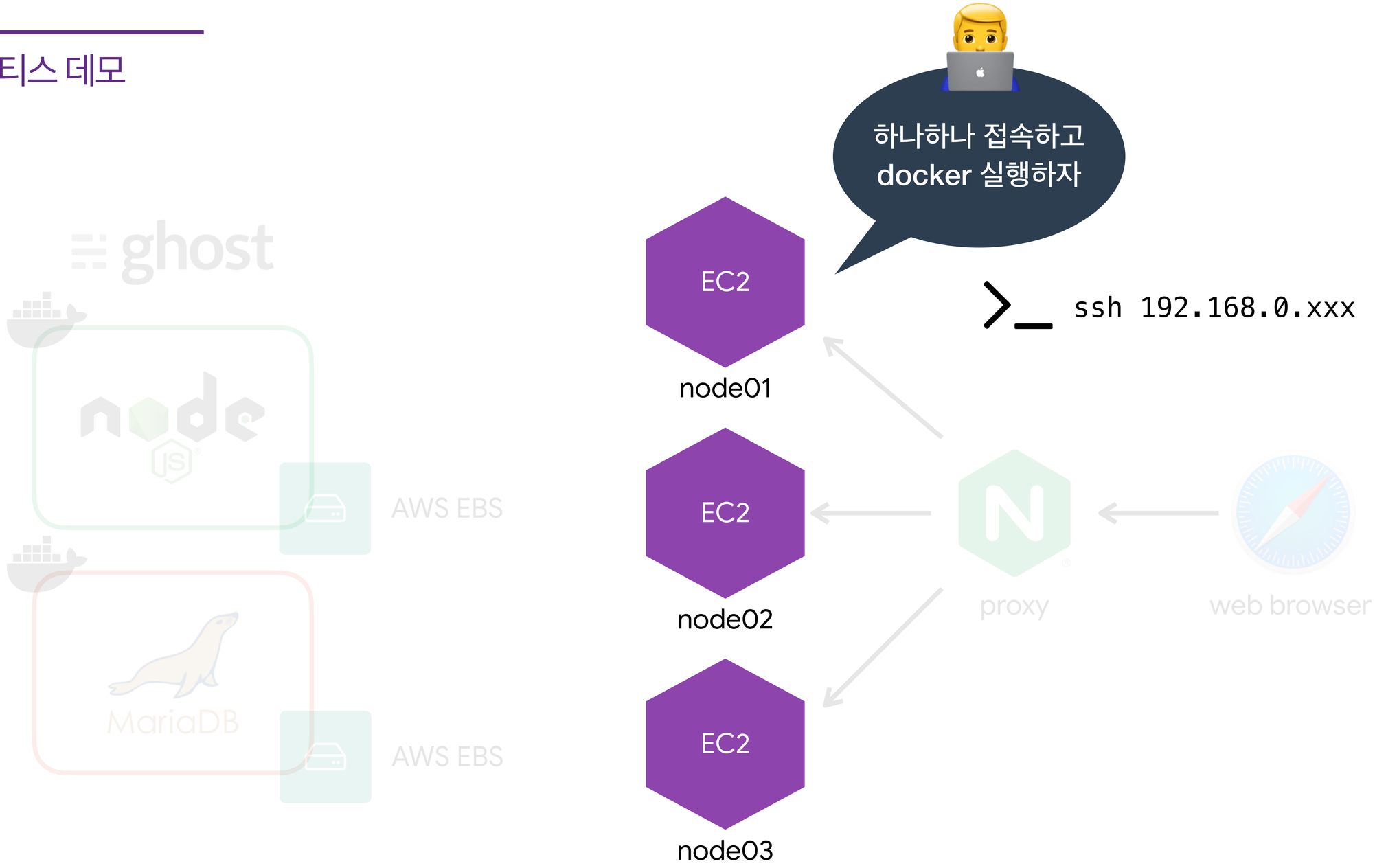

ghost
node
JS
MariaDB
AWS EBS
(첨부파일)
AWS EBS
(데이터)


EC2
node01
EC2
node02
EC2
node03
proxy
web browser

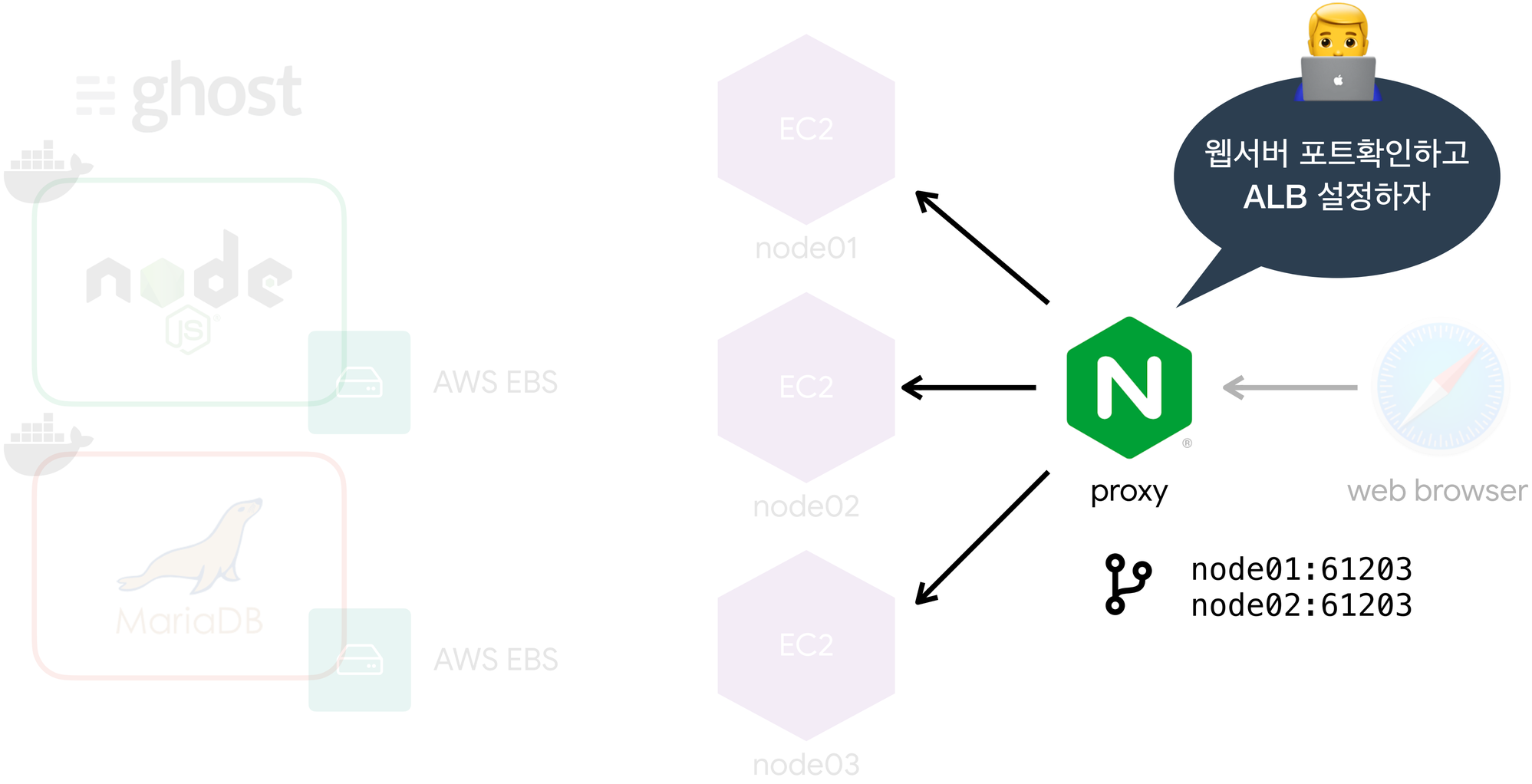
쿠버네티스 데모


ghost
node
JS
MariaDB
AWS EBS
AWS EBS
EBS 만들고..
EC2에 마운트하고..
EC2
EC2
EC2
node02
node03
Elastic Block Storage
Elastic Compute Cloud
proxy
web browser

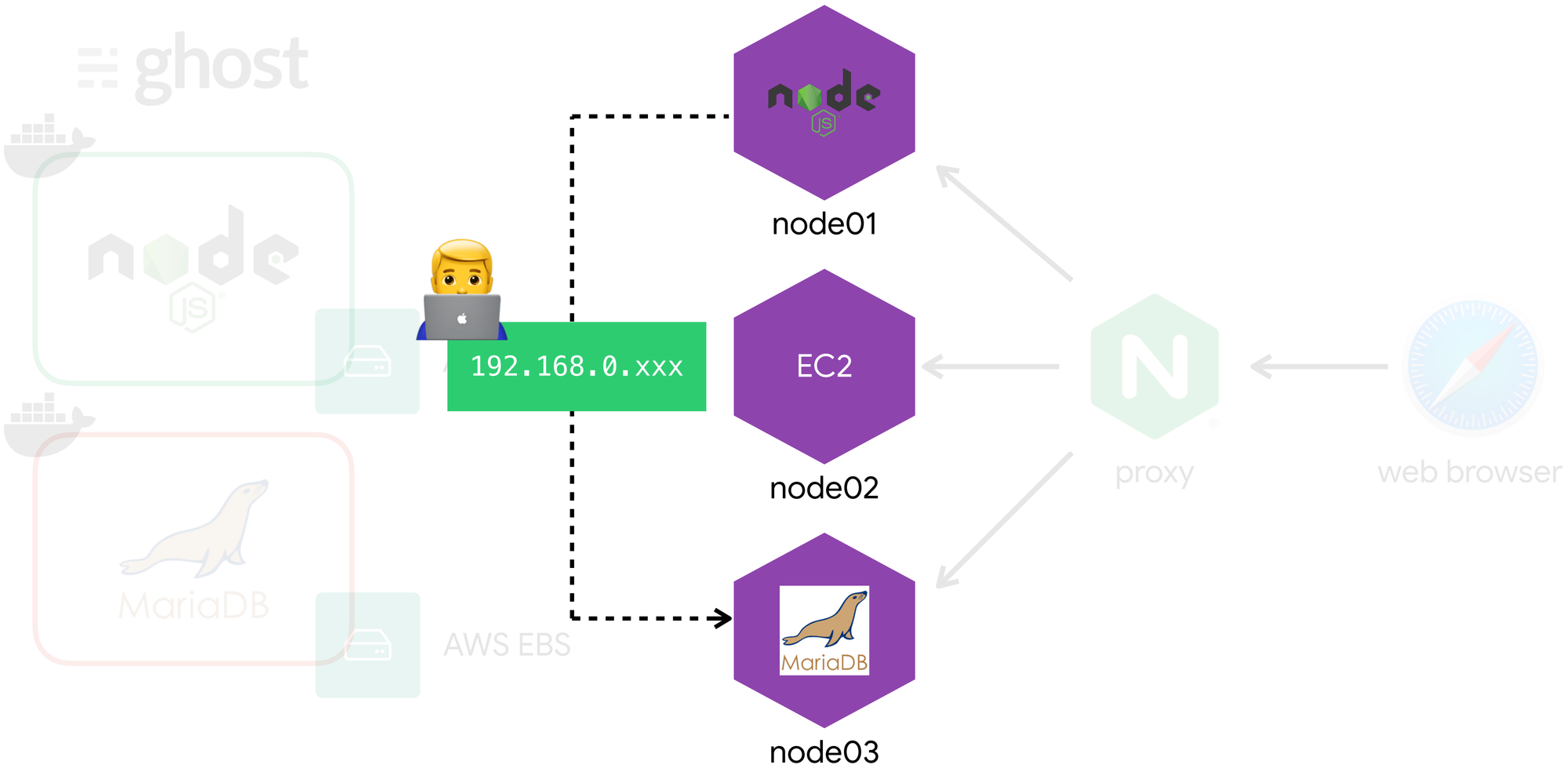
쿠버네티스 데모

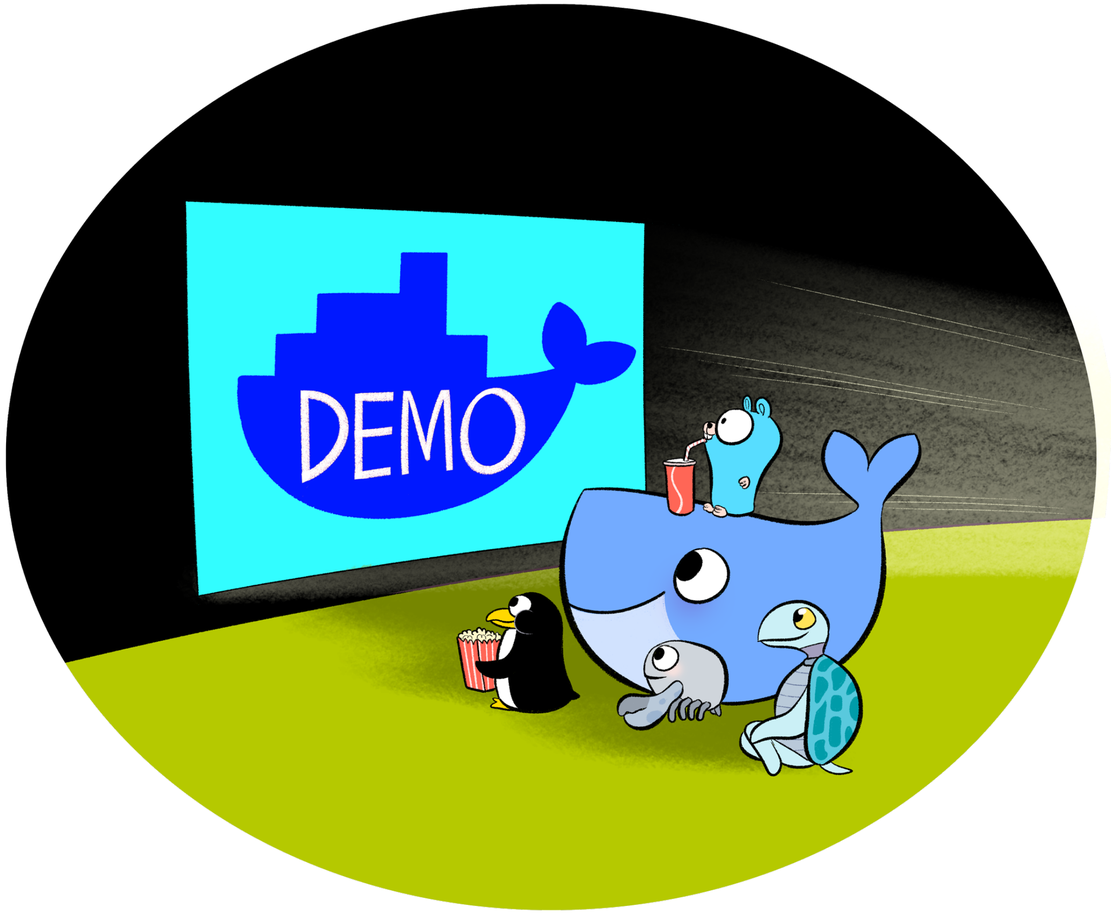

하나하나 접속하고
docker 실행하자
EC2
node01
EC2
node02
EC2
node03
>_ ssh 192.168.0.xxx
proxy
web browser
ghost
node
AWS EBS
MariaDB
AWS EBS

쿠버네티스 데모


ghost
node
MariaDB
AWS EBS
AWS EBS
EC2
node01
EC2
node02
EC2
node03
웹서버 포트확인하고
ALB 설정하자
proxy
web browser
node01:61203
node02:61203

쿠버네티스 데모


ghost
node
MariaDB
AWS EBS
192.168.0.xxx
node01
EC2
node02
MariaDB
node03
proxy
web browser


DEMO

쿠버네티스 아키텍처

02

구성/설계 Architecture

오브젝트 Objects

API 호출

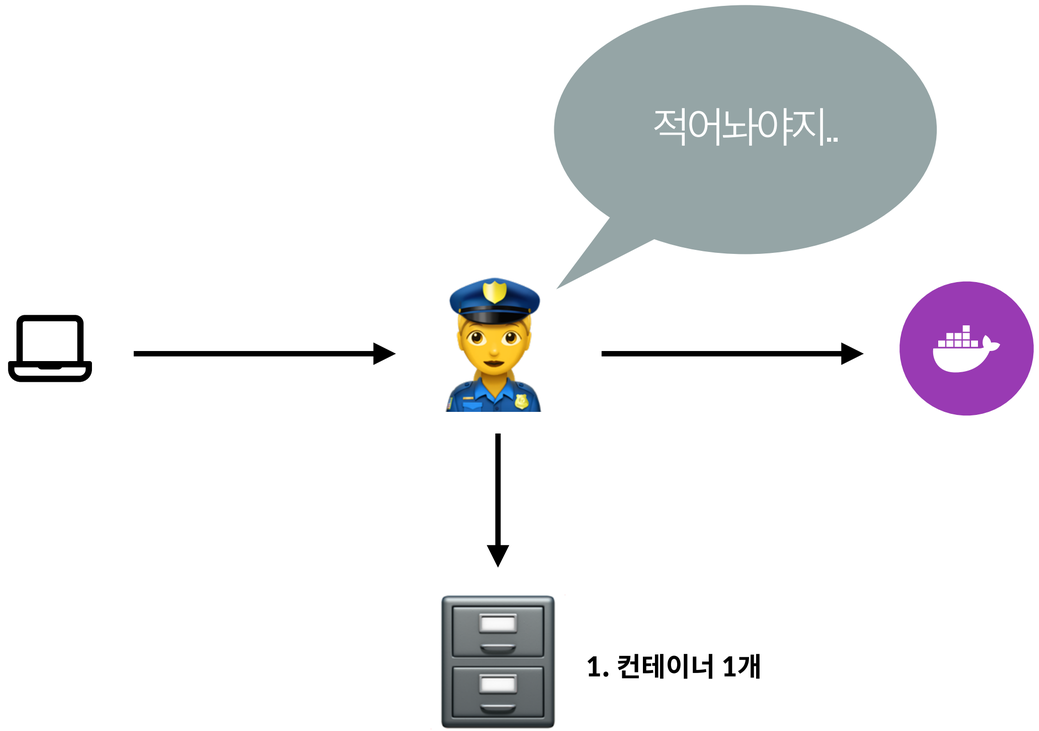
쿠버네티스 아키텍처

쿠버네티스 아키텍처

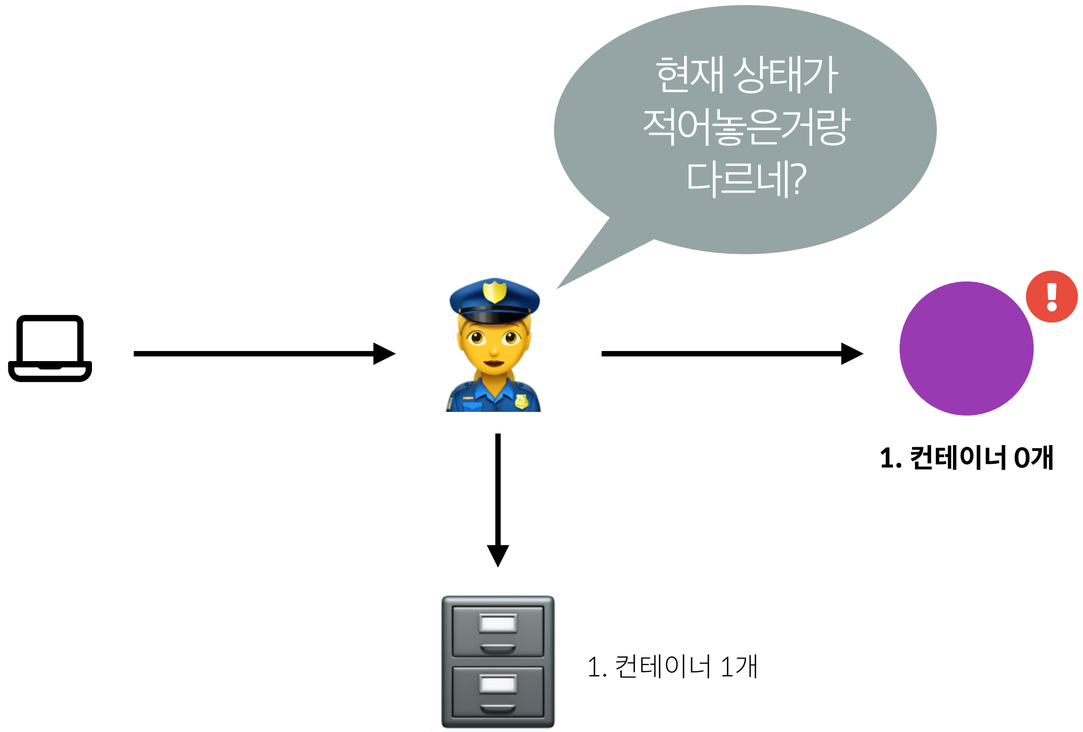
쿠버네티스 아키텍처

쿠버네티스 아키텍처

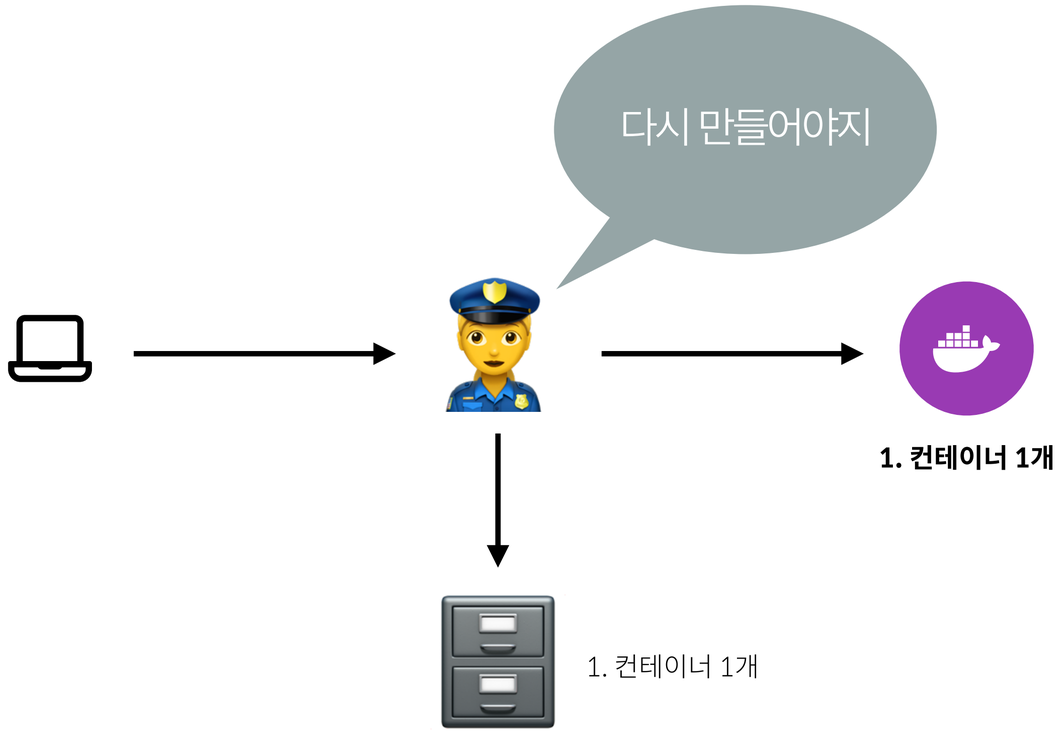
쿠버네티스 아키텍처



쿠버네티스 아키텍처



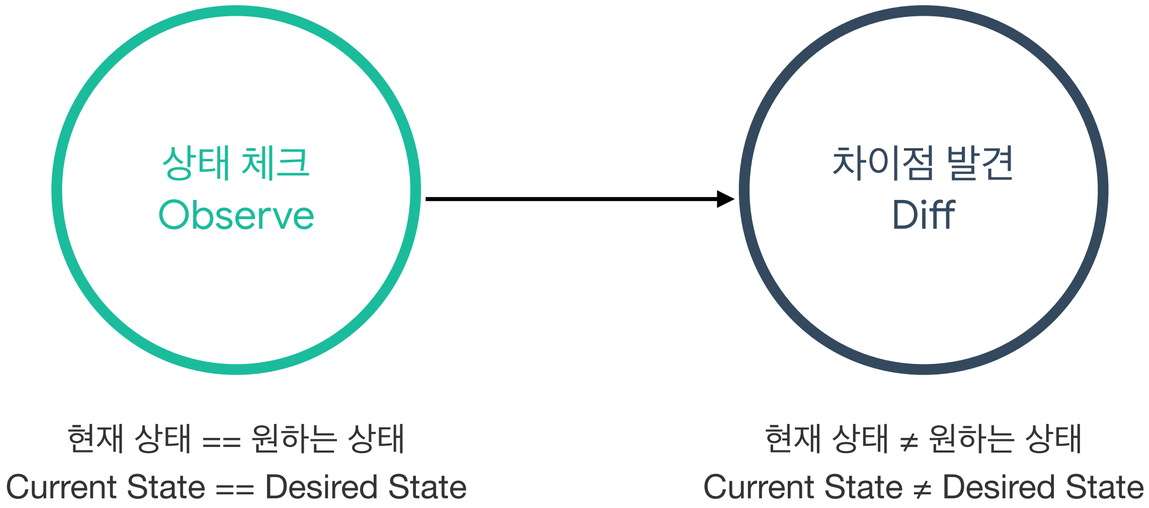
쿠버네티스 아키텍처



Desired State

현재 상태 == 원하는 상태
Current State == Desired State

Desired State

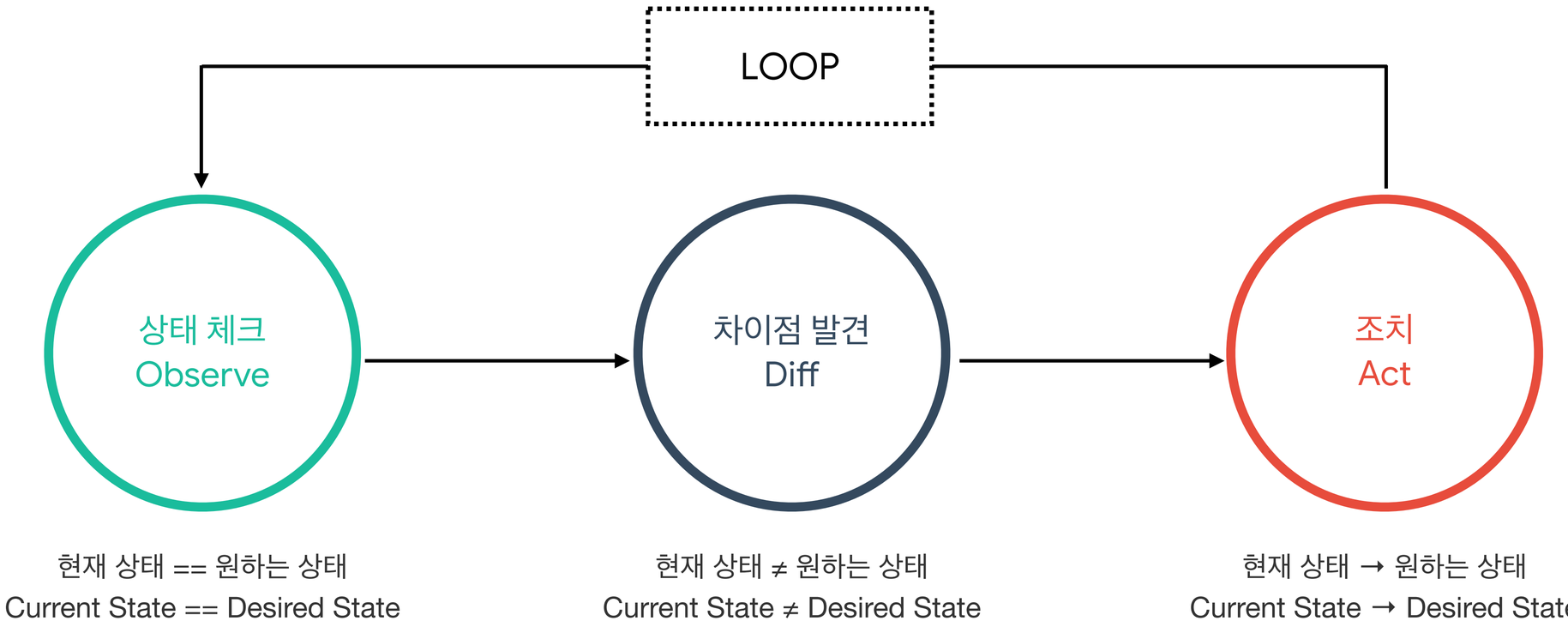


현재 상태 == 원하는 상태
Current State == Desired State

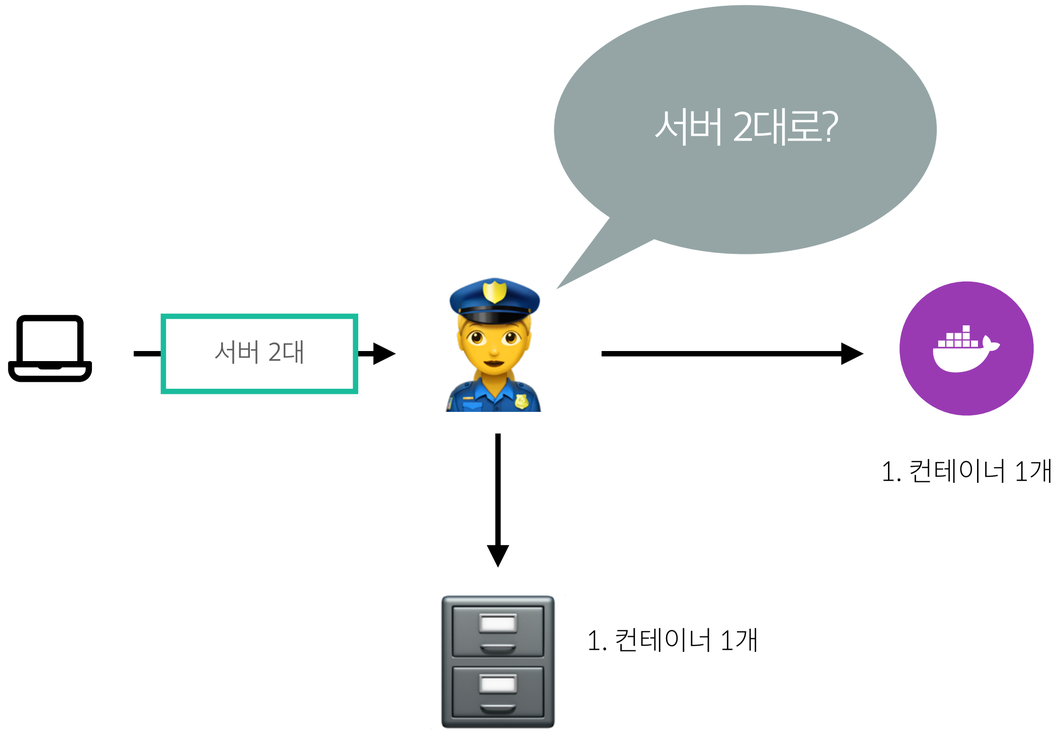
현재 상태 ≠ 원하는 상태
Current State ≠ Desired State

Desired State

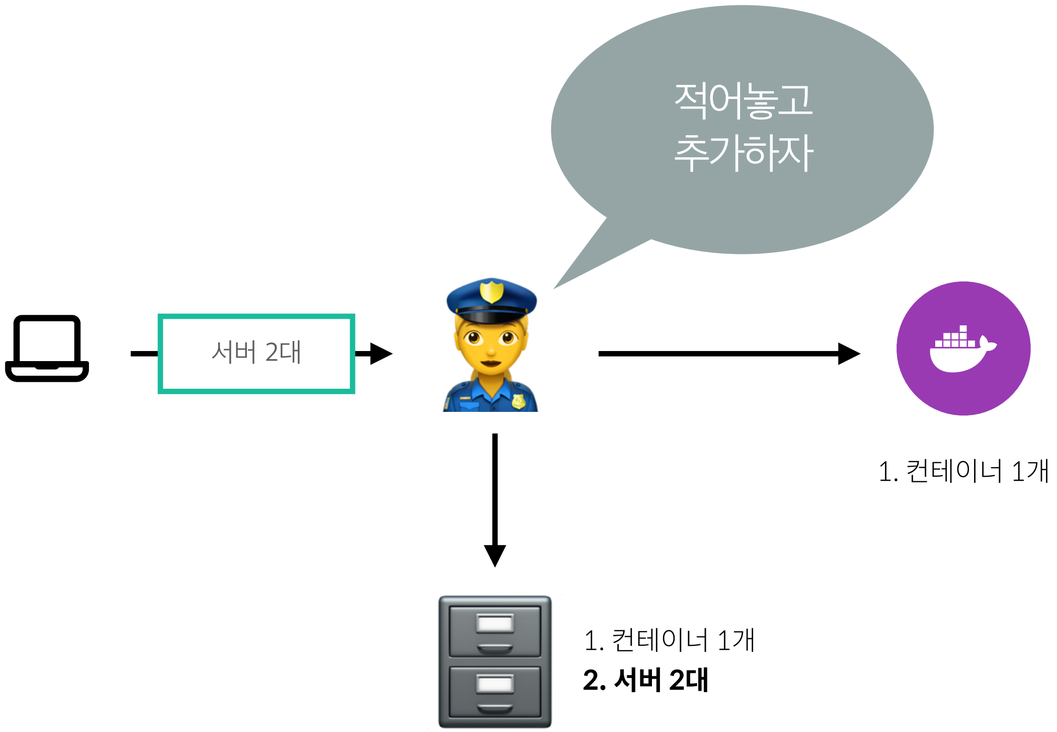
Desired State



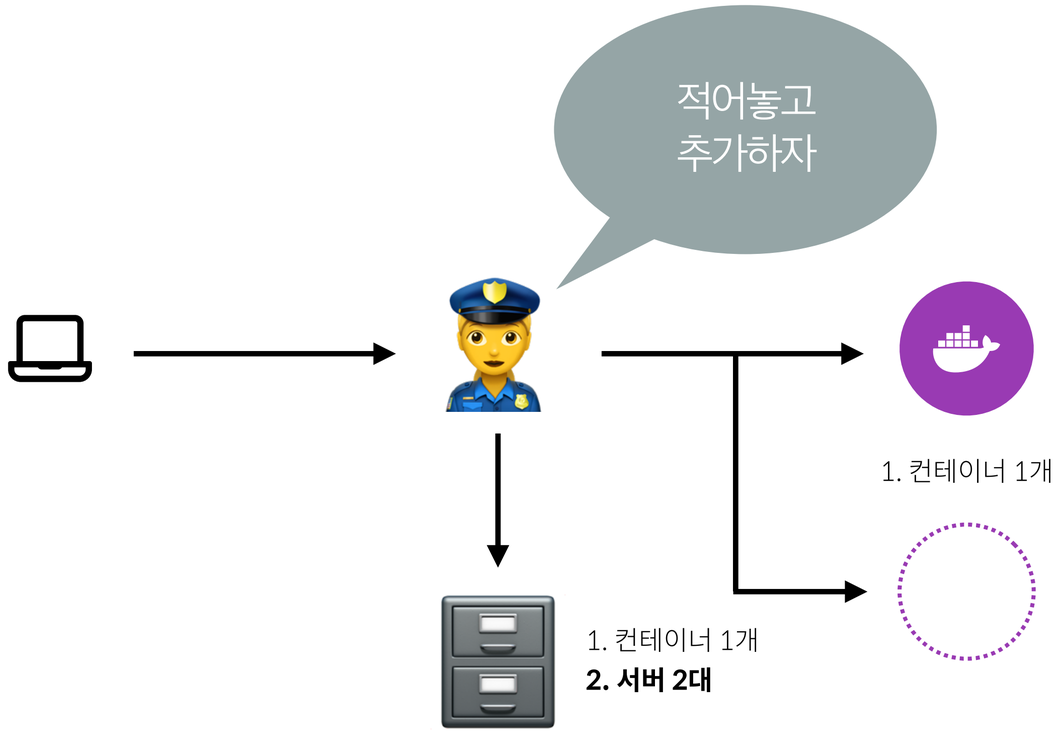
쿠버네티스 아키텍처



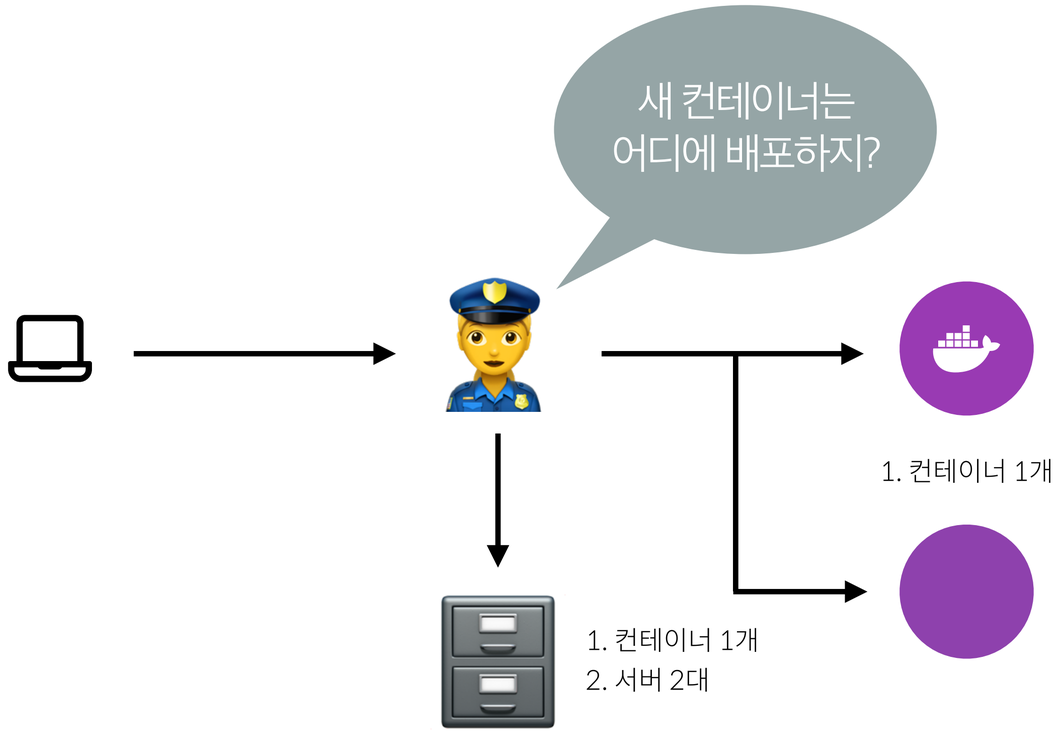
쿠버네티스 아키텍처



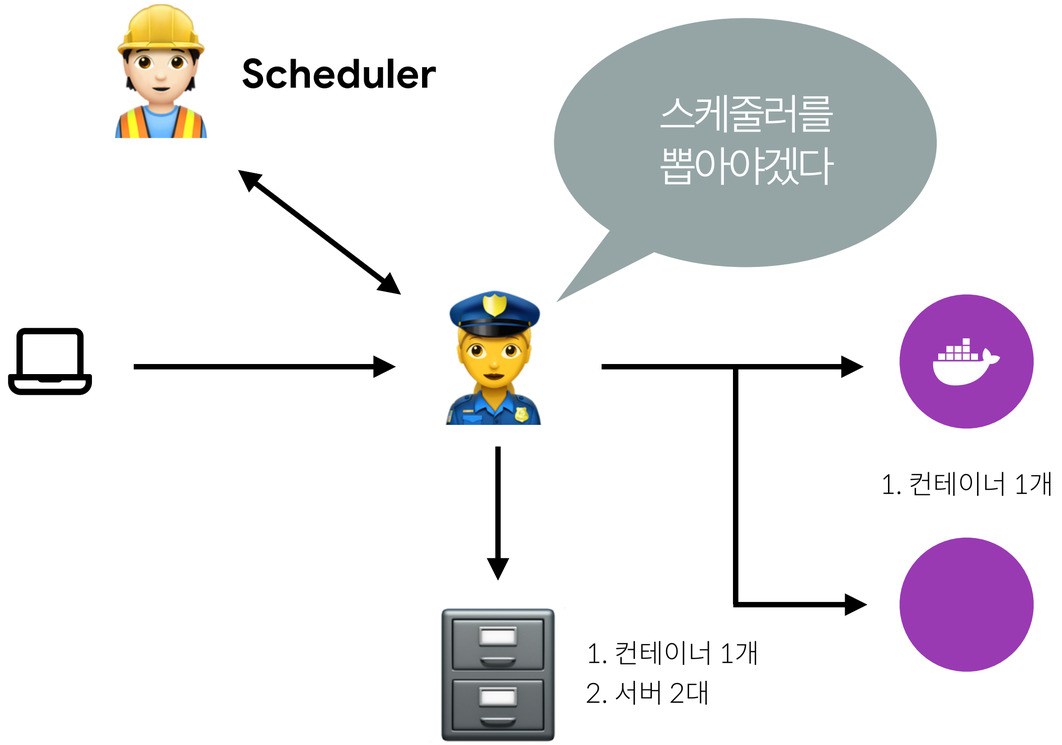
쿠버네티스 아키텍처



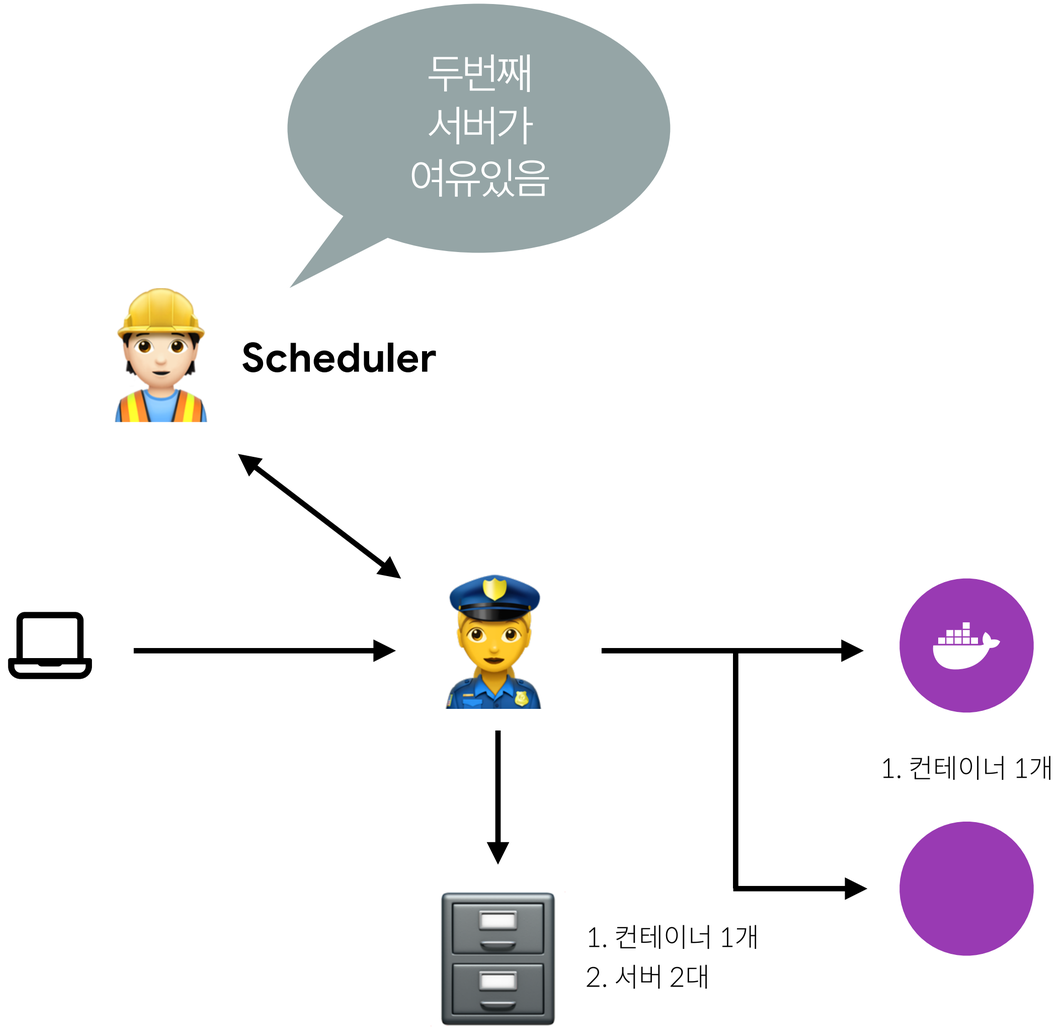
쿠버네티스 아키텍처



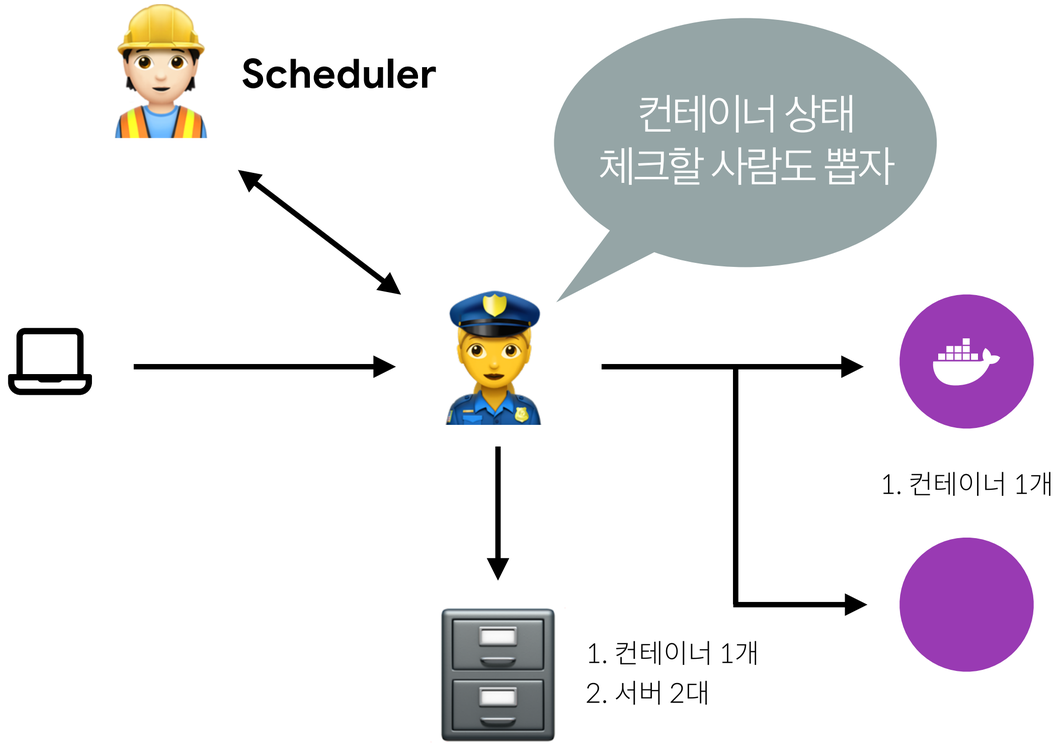
쿠버네티스 아키텍처



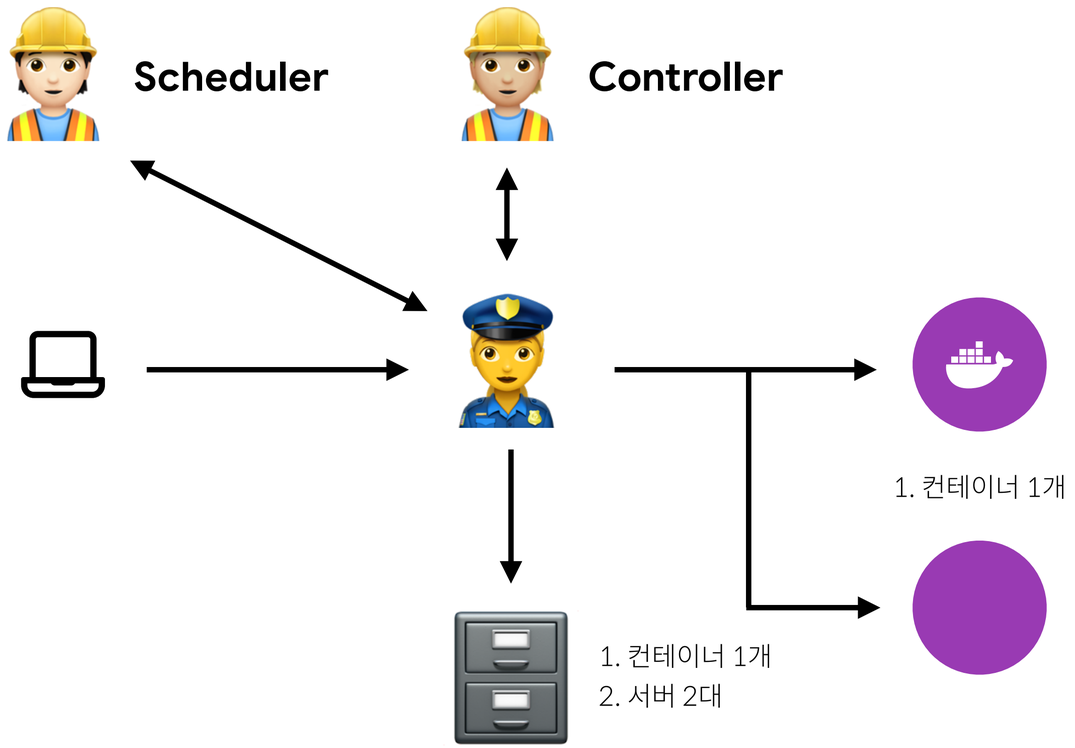
쿠버네티스 아키텍처



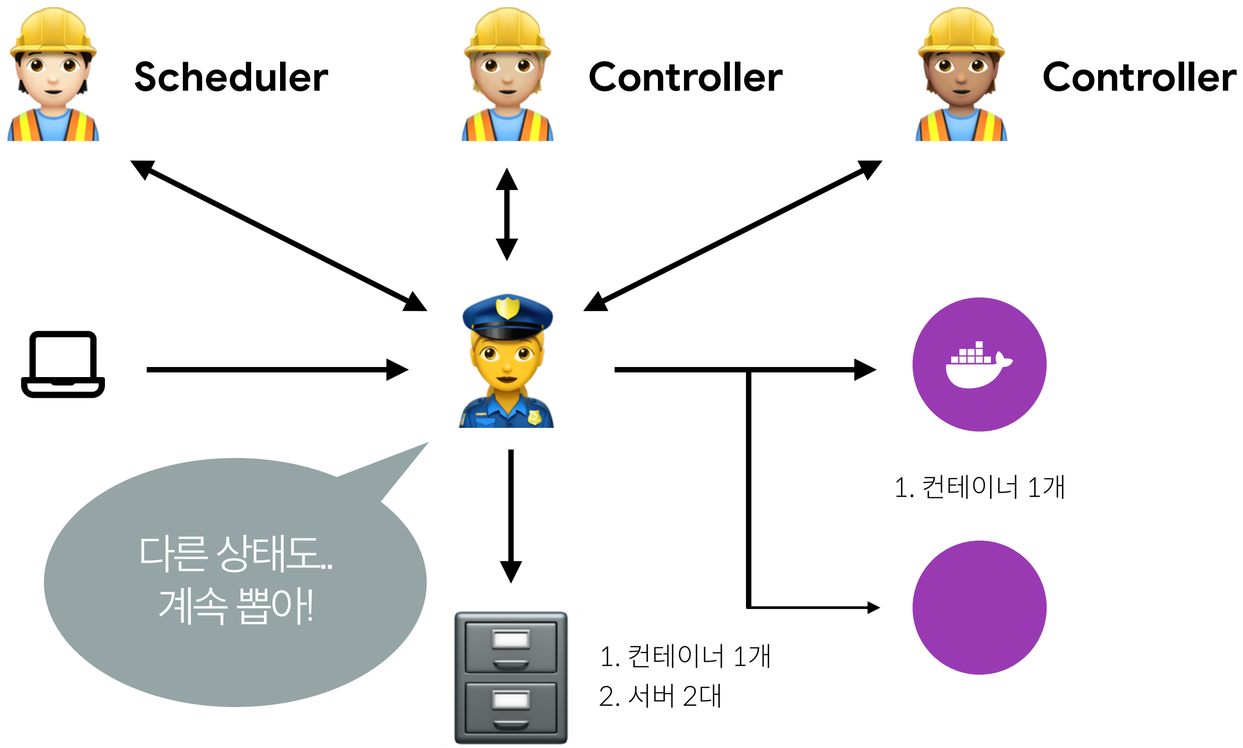
쿠버네티스 아키텍처



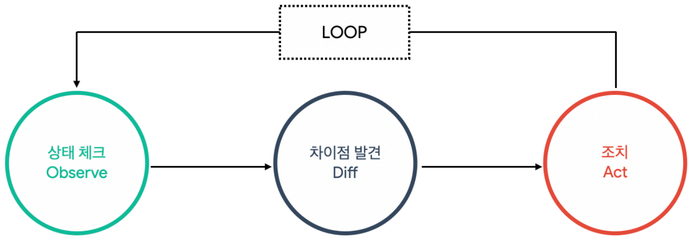
쿠버네티스 아키텍처



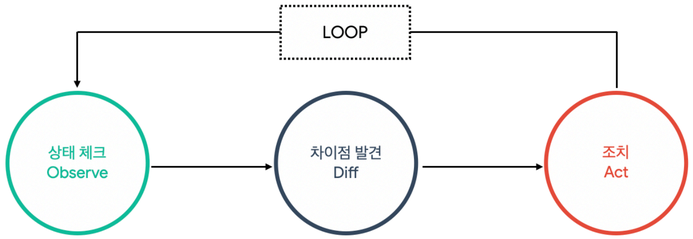
쿠버네티스 아키텍처


Scheduler
Controller
Controller
다른 상태도..
계속 뽑아!
1. 컨테이너 1개
1. 컨테이너 1개
2. 서버 2대

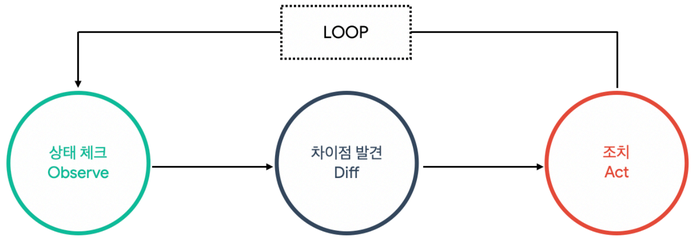
Desired State



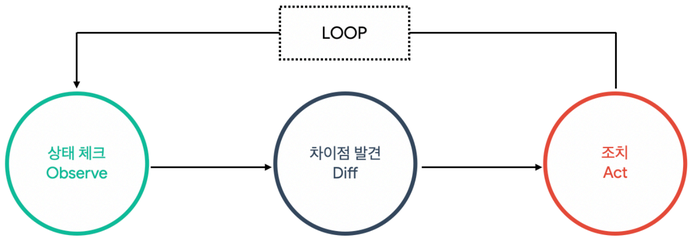
Replication Controller



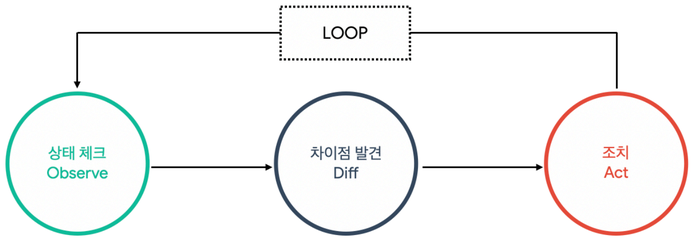
Endpoint Controller



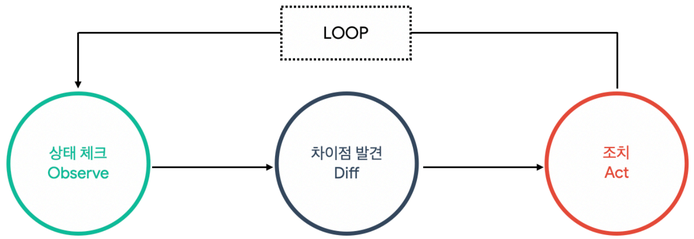
Namespace Controller



Custom Controller

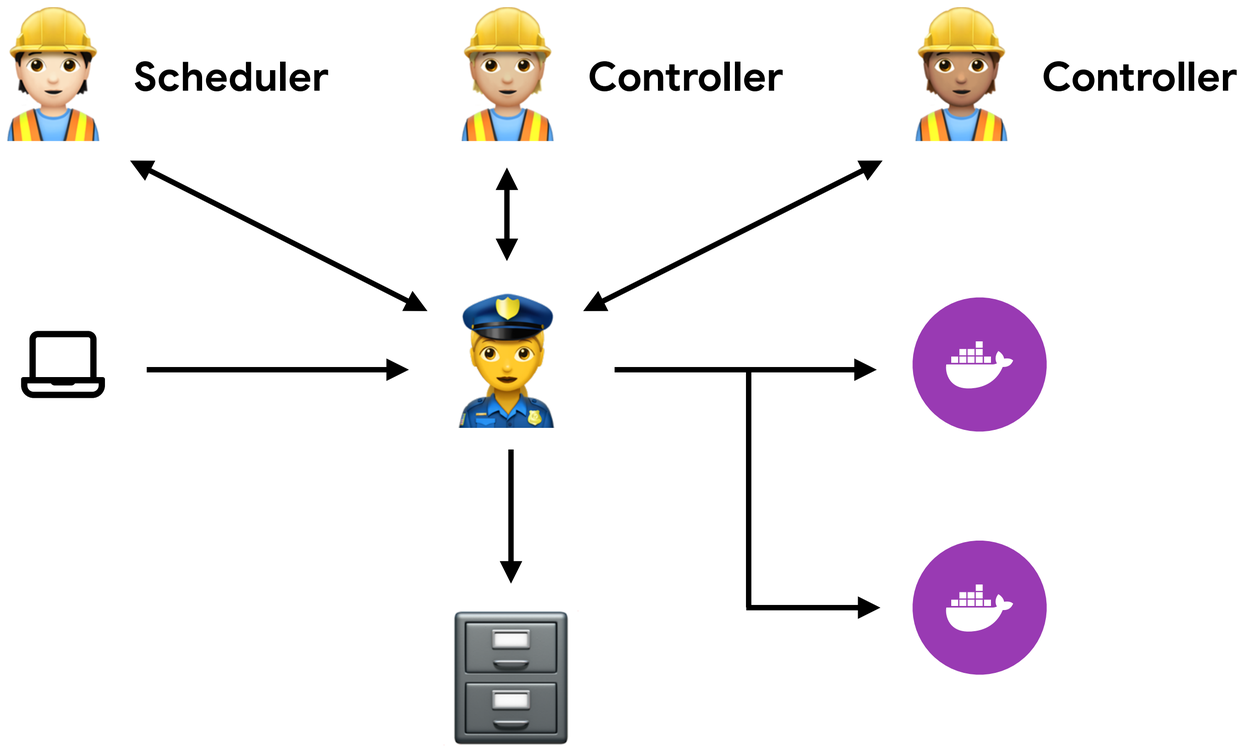


ML Controller

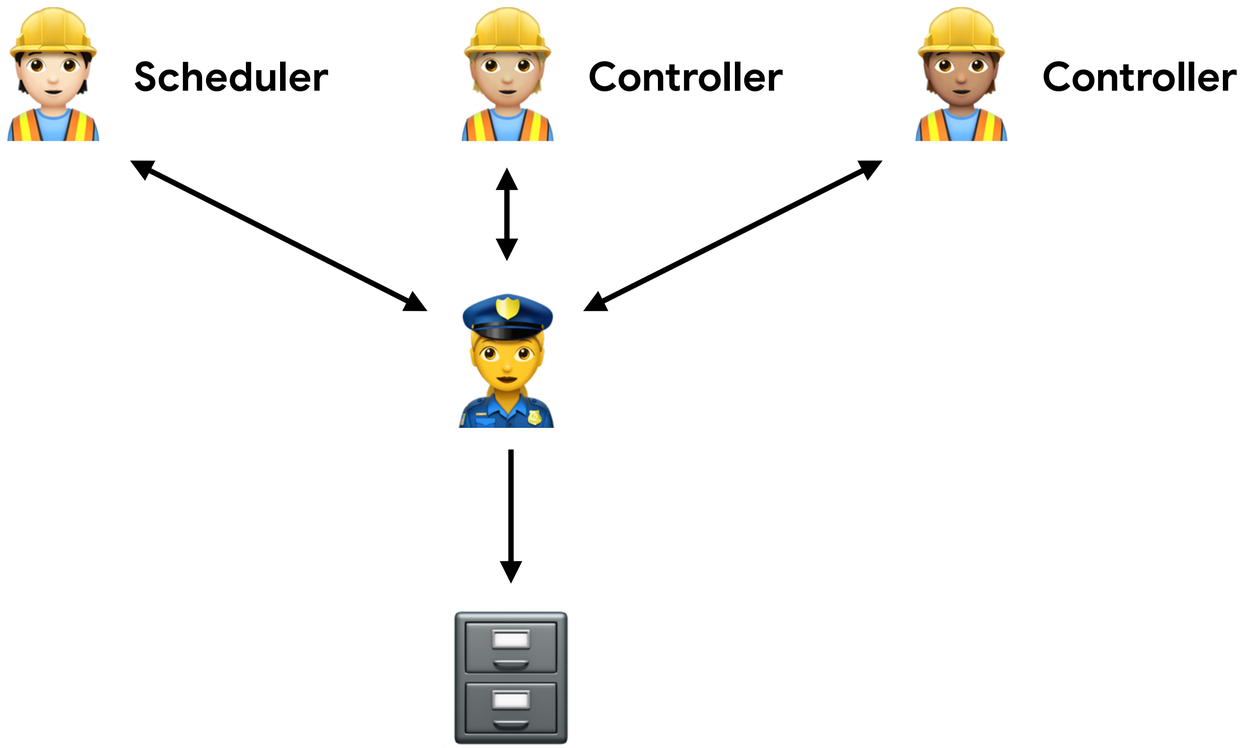


CI/CD Controller

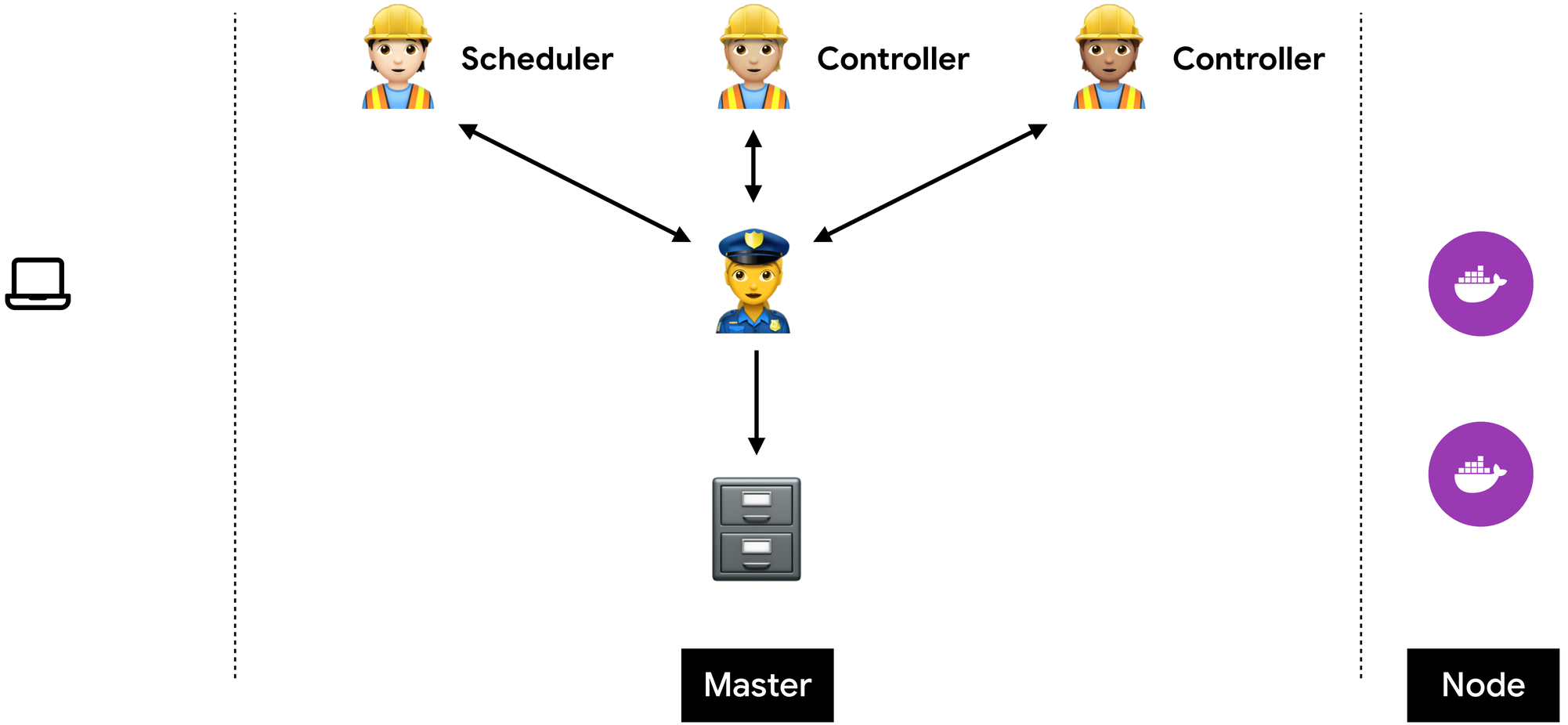
쿠버네티스 아키텍처



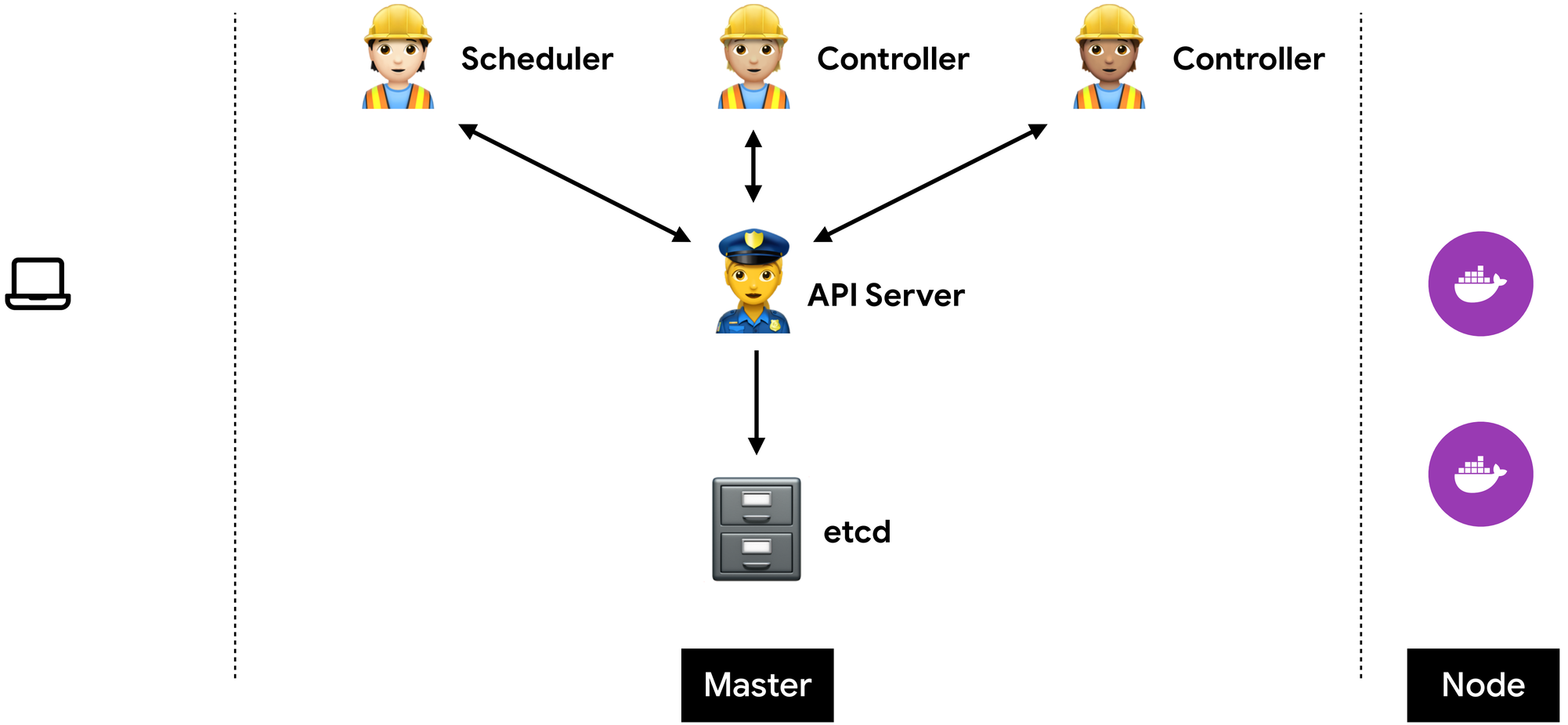
쿠버네티스 아키텍처



쿠버네티스 아키텍처



쿠버네티스 아키텍처


Scheduler
Controller
Controller
API Server
etcd
Master
Node

Master 상세 - etcd

모든 상태와 데이터를 저장

분산 시스템으로 구성하여 안전성을 높임 (고가용성)

가볍고 빠르면서 정확하게 설계 (일관성)

Key(directory)-Value 형태로 데이터 저장

TTL(time to live), watch같은 부가 기능 제공

백업은 필수!

Master 상세 - API server

상태를 바꾸거나 조회

etcd와 유일하게 통신하는 모듈

REST API 형태로 제공

권한을 체크하여 적절한 권한이 없을 경우 요청을 차단

관리자 요청 뿐 아니라 다양한 내부 모듈과 통신

수평으로 확장되도록 디자인

Master 상세 - Scheduler

새로 생성된 Pod을 감지하고 실행할 노드를 선택

노드의 현재 상태와 Pod의 요구사항을 체크

- 노드에 라벨을 부여
- ex) a-zone, b-zone 또는 gpu-enabled, ...

Master 상세 - Controller

논리적으로 다양한 컨트롤러가 존재

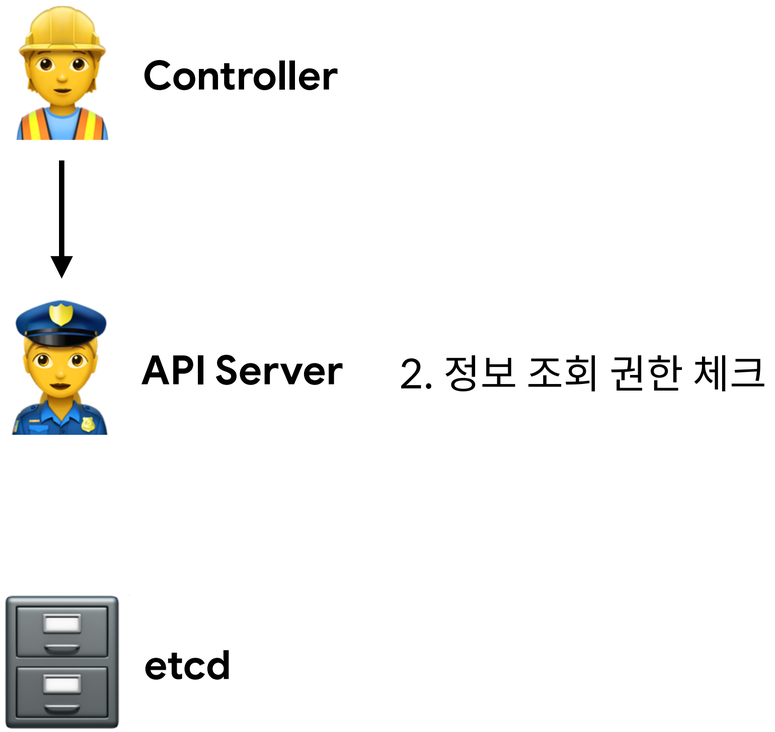
- 복제 컨트롤러
- 노드 컨트롤러
- 엔드포인트 컨트롤러...

끊임 없이 상태를 체크하고 원하는 상태를 유지

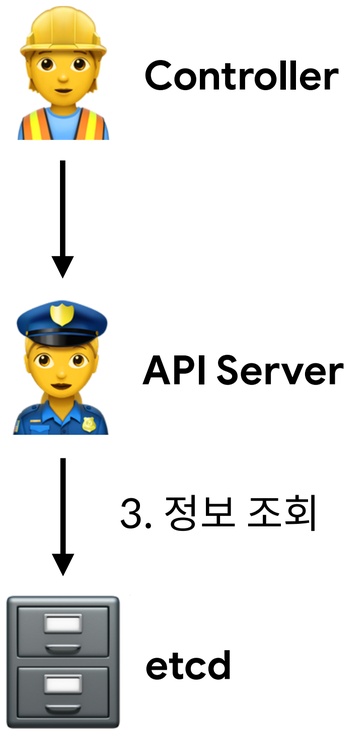
복잡성을 낮추기 위해 하나의 프로세스로 실행

Master 상세 - 조회흐름

Master 상세 - 조회흐름

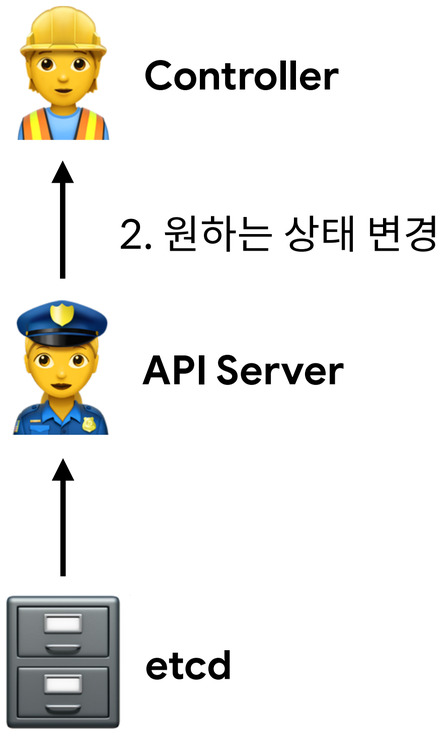


Master 상세 - 조회흐름

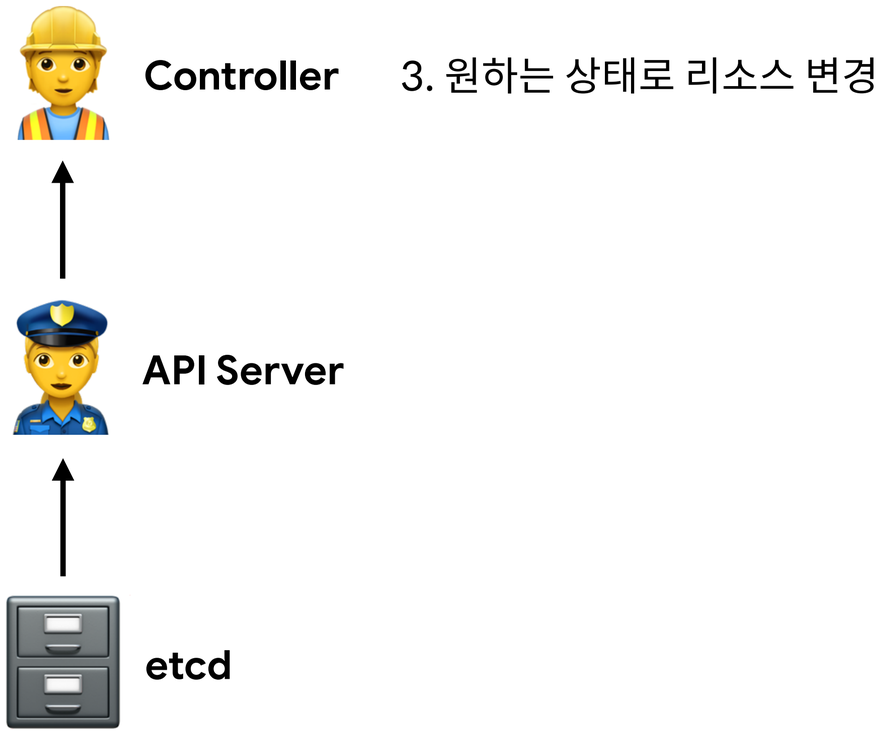


Master 상세 - 기본흐름

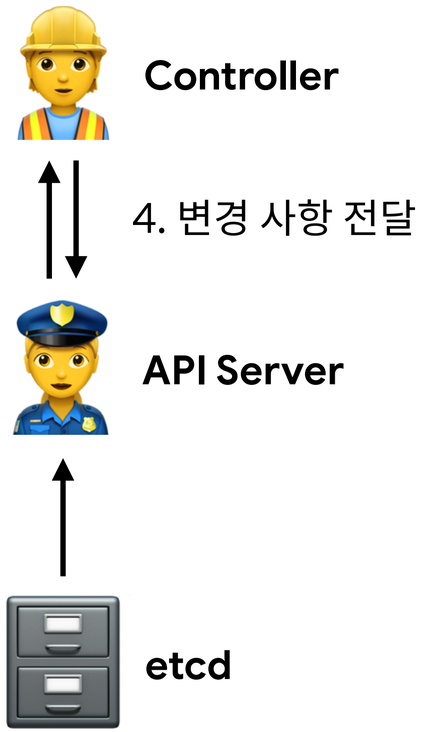
Master 상세 - 기본흐름



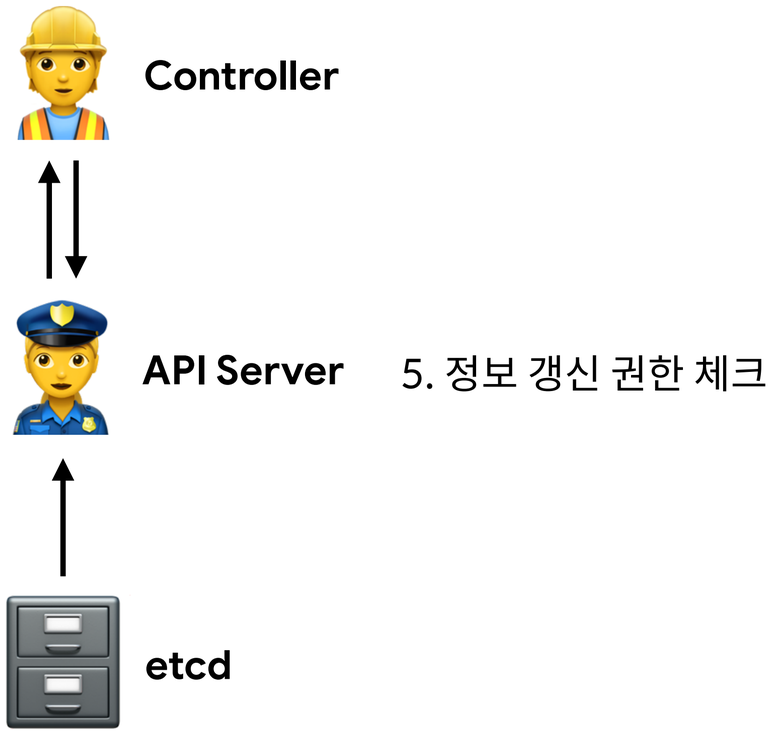
Master 상세 - 기본흐름



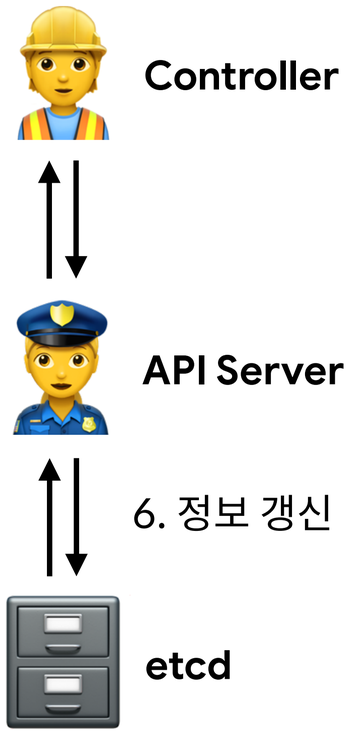
Master 상세 - 기본흐름



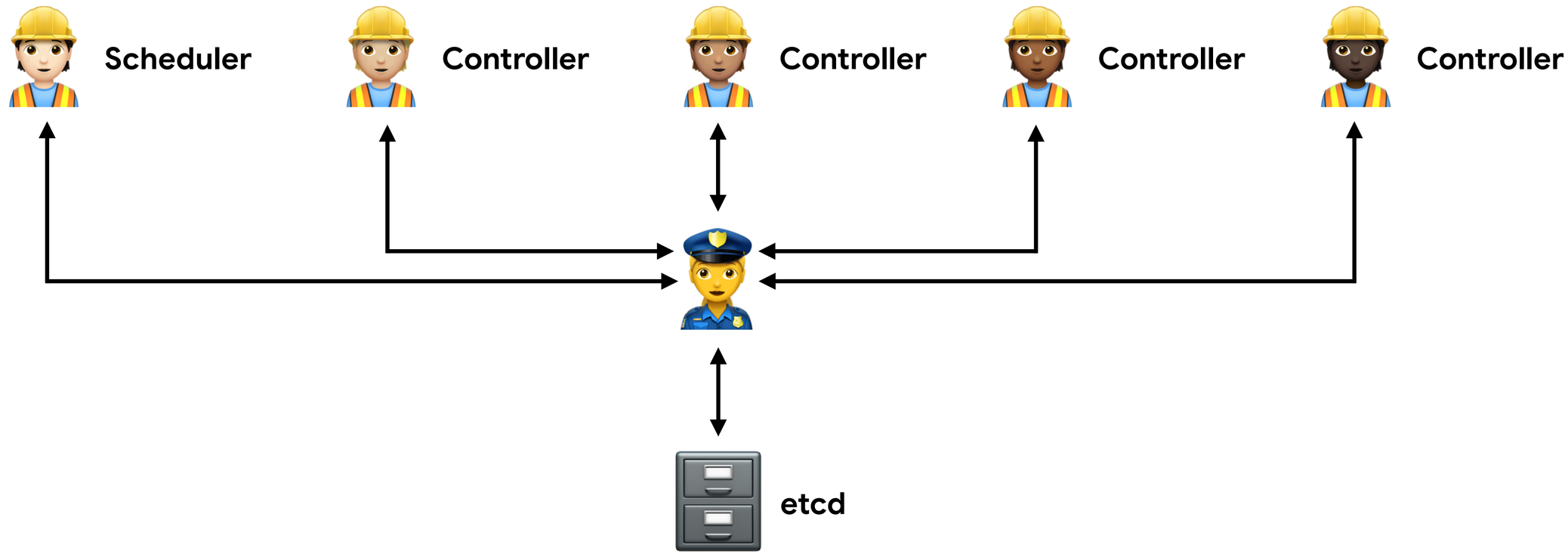
Master 상세 - 기본흐름



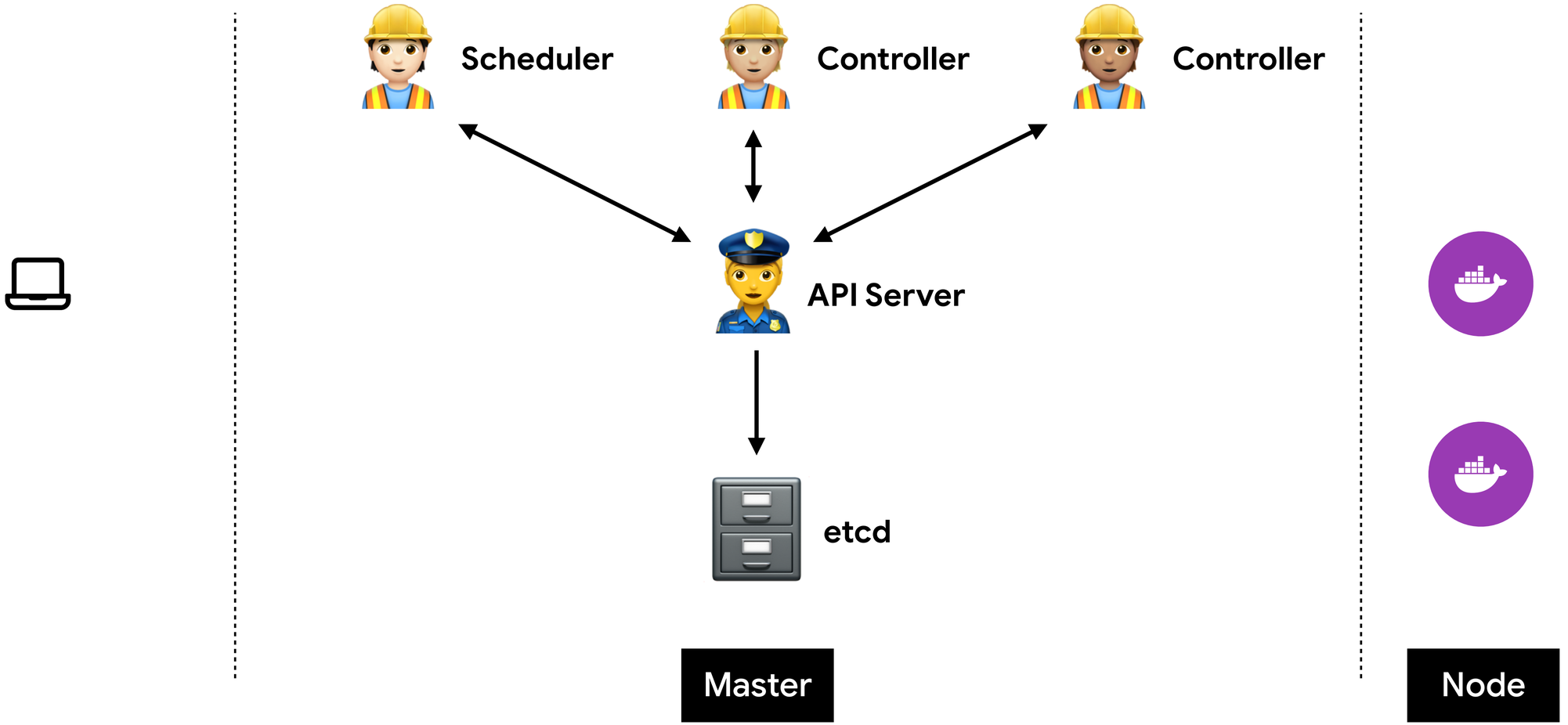
Master 상세 - 기본흐름



Master 상세 - API Server 통신



쿠버네티스 아키텍처


Scheduler
Controller
Controller
API Server
etcd
Master
Node

쿠버네티스 아키텍처

API Server

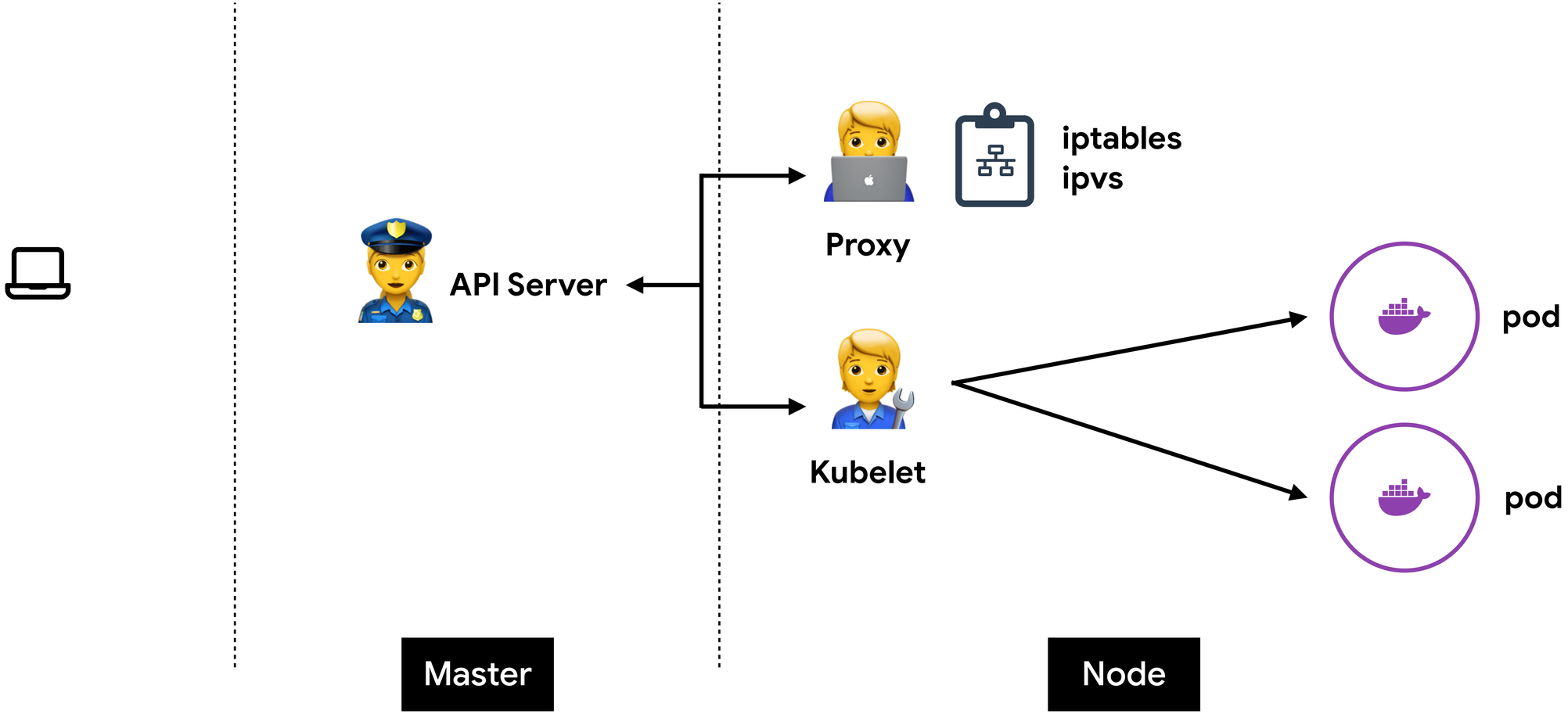
Master

Proxy

Kubelet

Node

쿠버네티스 아키텍처



Node 상세 - kubelet

각 노드에서 실행

Pod을 실행/중지하고 상태를 체크

CRI (Container Runtime Interface)

- docker
- Containerd
- CRI-O
- ...

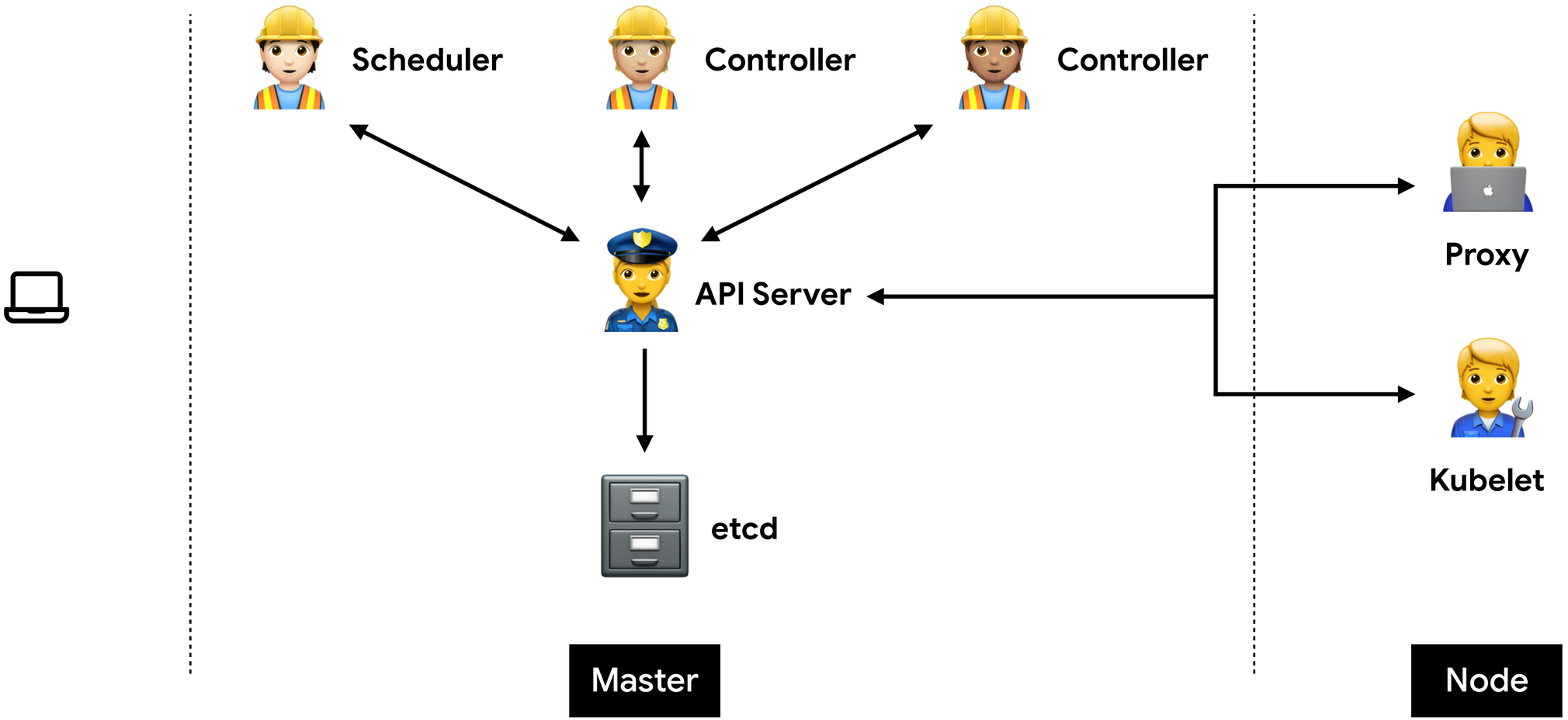
Node 상세 - proxy

네트워크 프록시와 부하 분산 역할

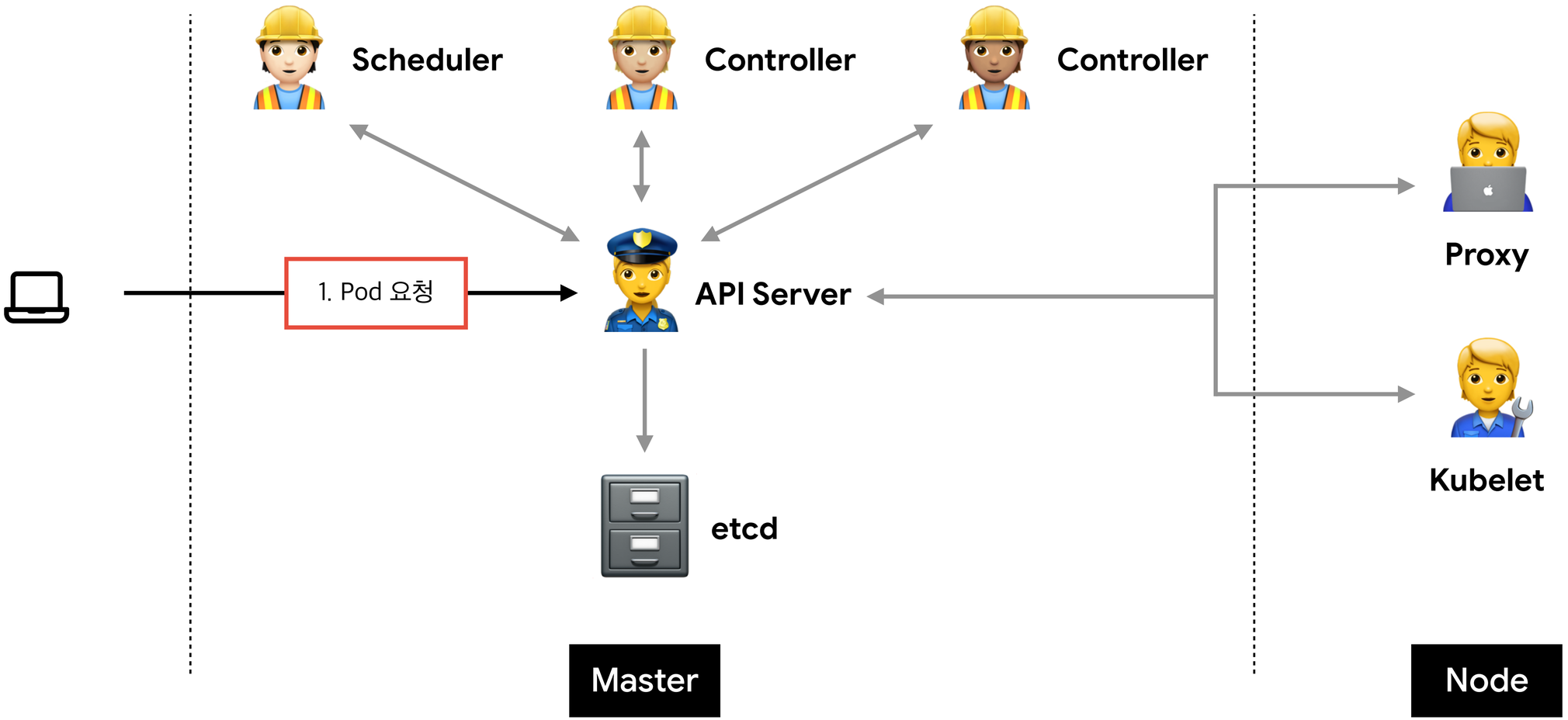
성능상의 이유로 별도의 프록시 프로그램 대신

iptables 또는 IPVS를 사용 (설정만 관리)

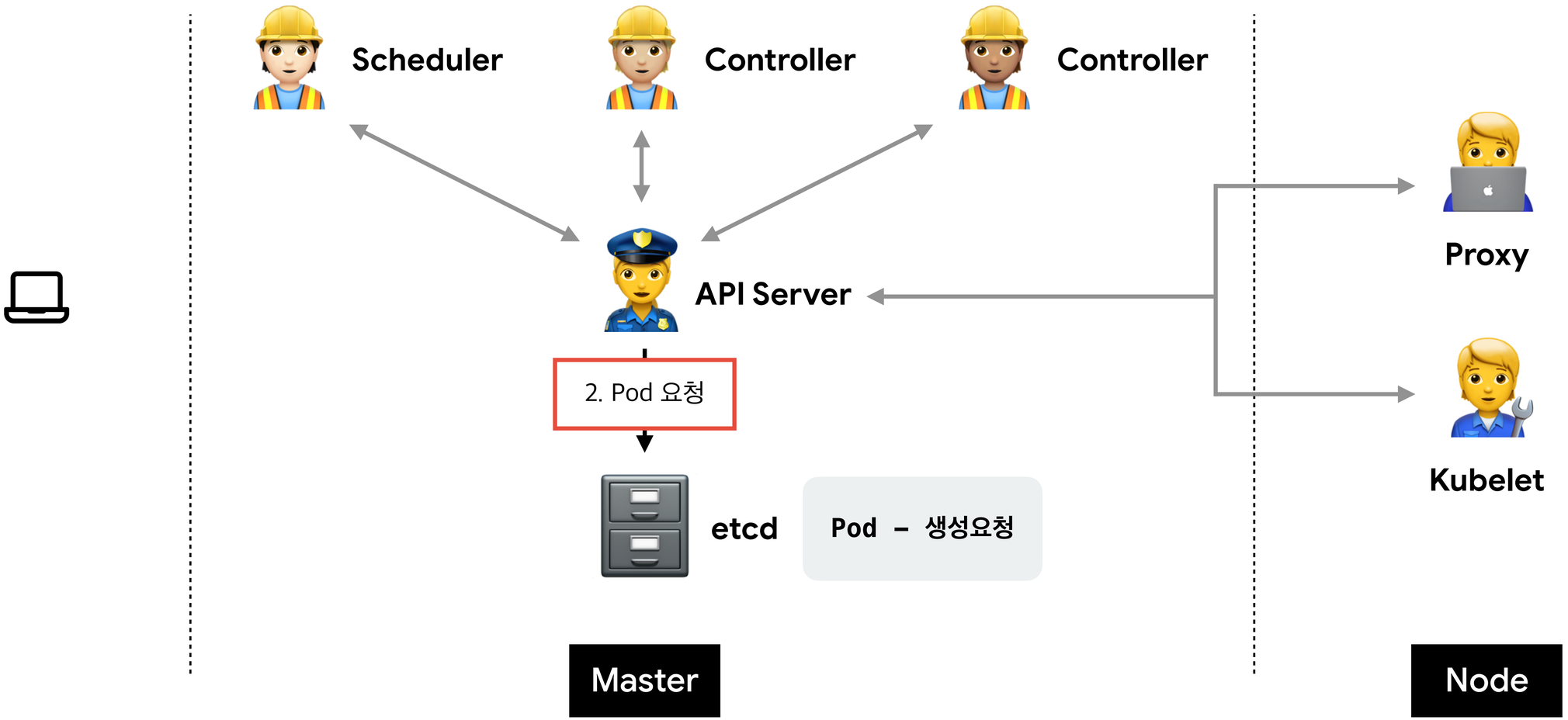
쿠버네티스 아키텍처


Scheduler
Controller
Controller
API Server
etcd
Proxy
Kubelet
Master
Node

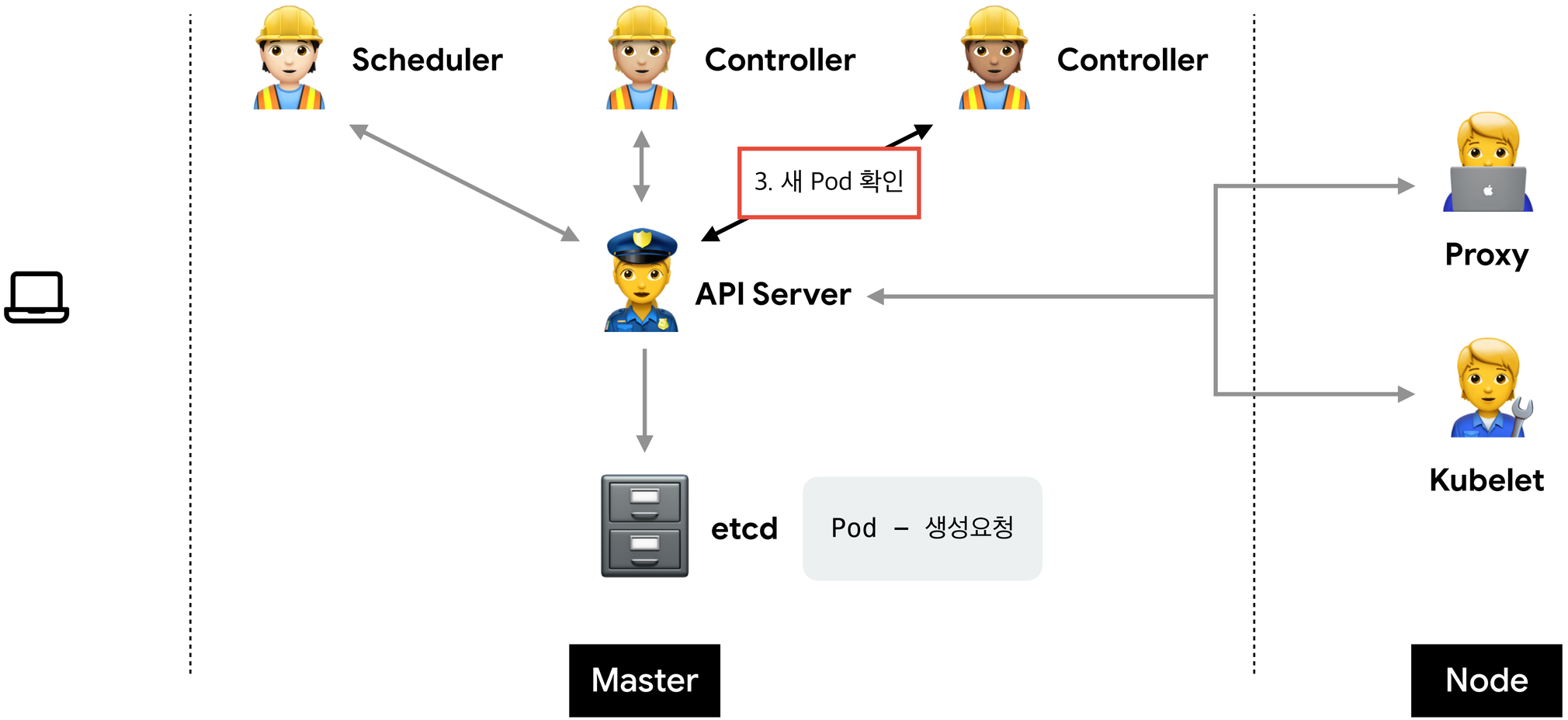
쿠버네티스 흐름


Scheduler
Controller
Controller
1. Pod 요청
API Server
Proxy
Kubelet
etcd
Master
Node

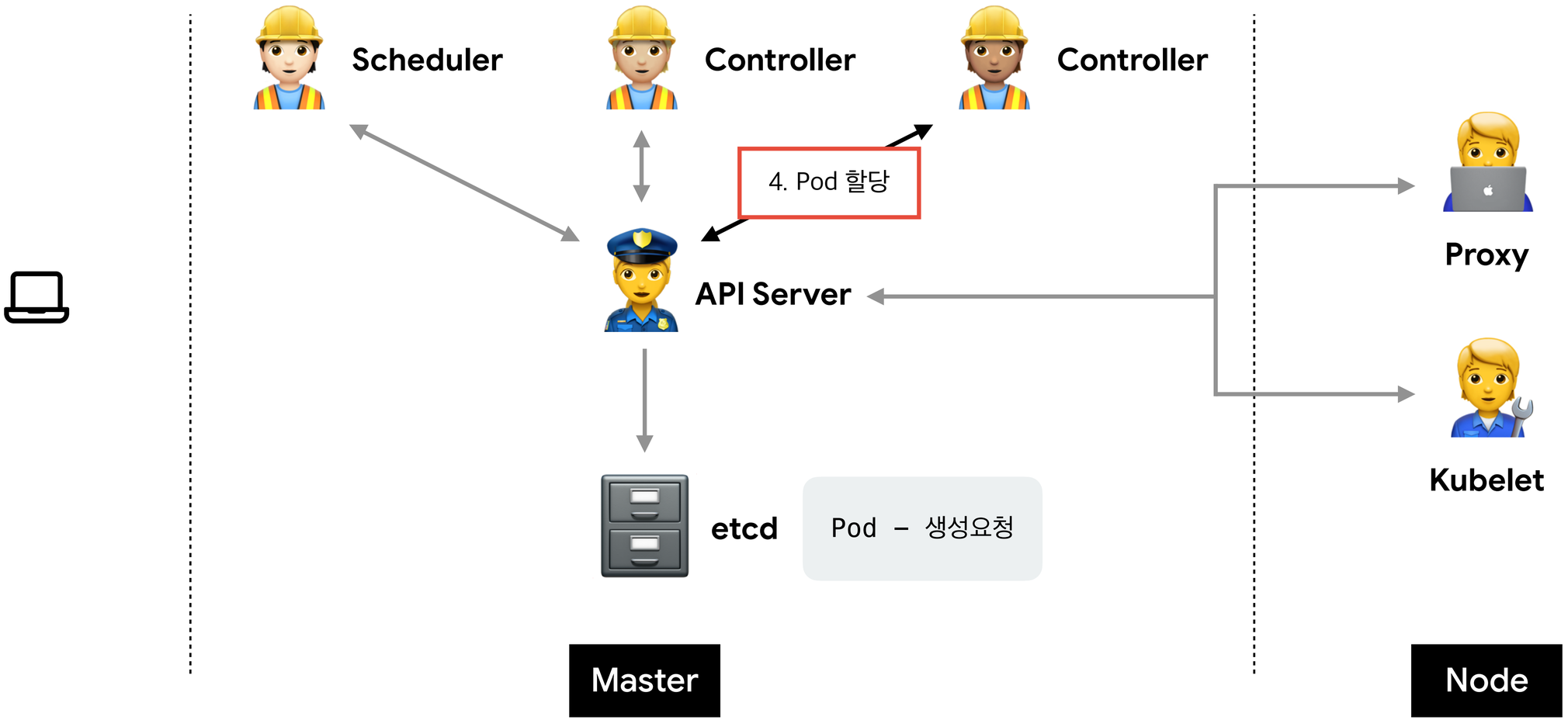
쿠버네티스 흐름


Scheduler
Controller
Controller
API Server
Proxy
Kubelet
2. Pod 요청
etcd
Pod – 생성요청
Master
Node

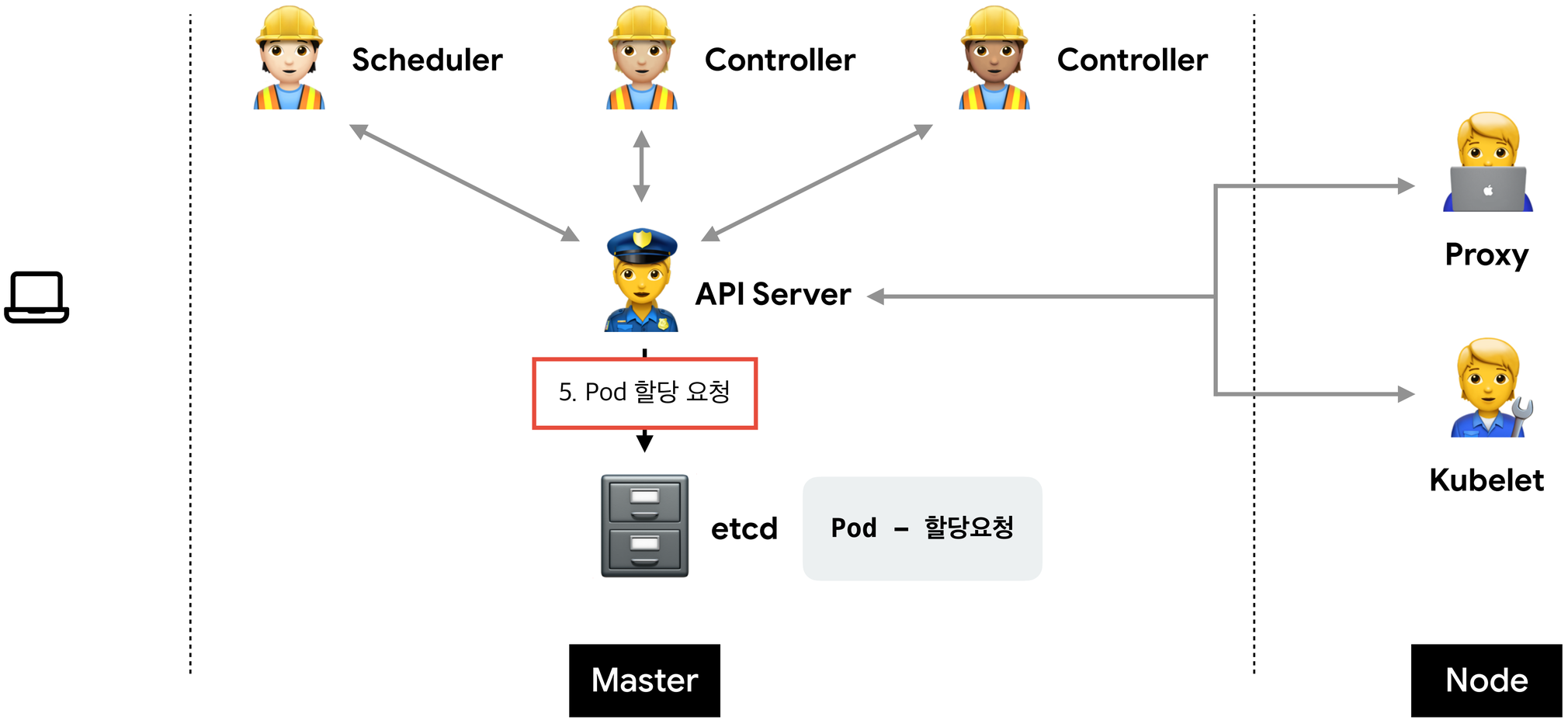
쿠버네티스 흐름


Scheduler
Controller
Controller
3. 새 Pod 확인
API Server
Proxy
Kubelet
etcd
Pod – 생성요청
Master
Node

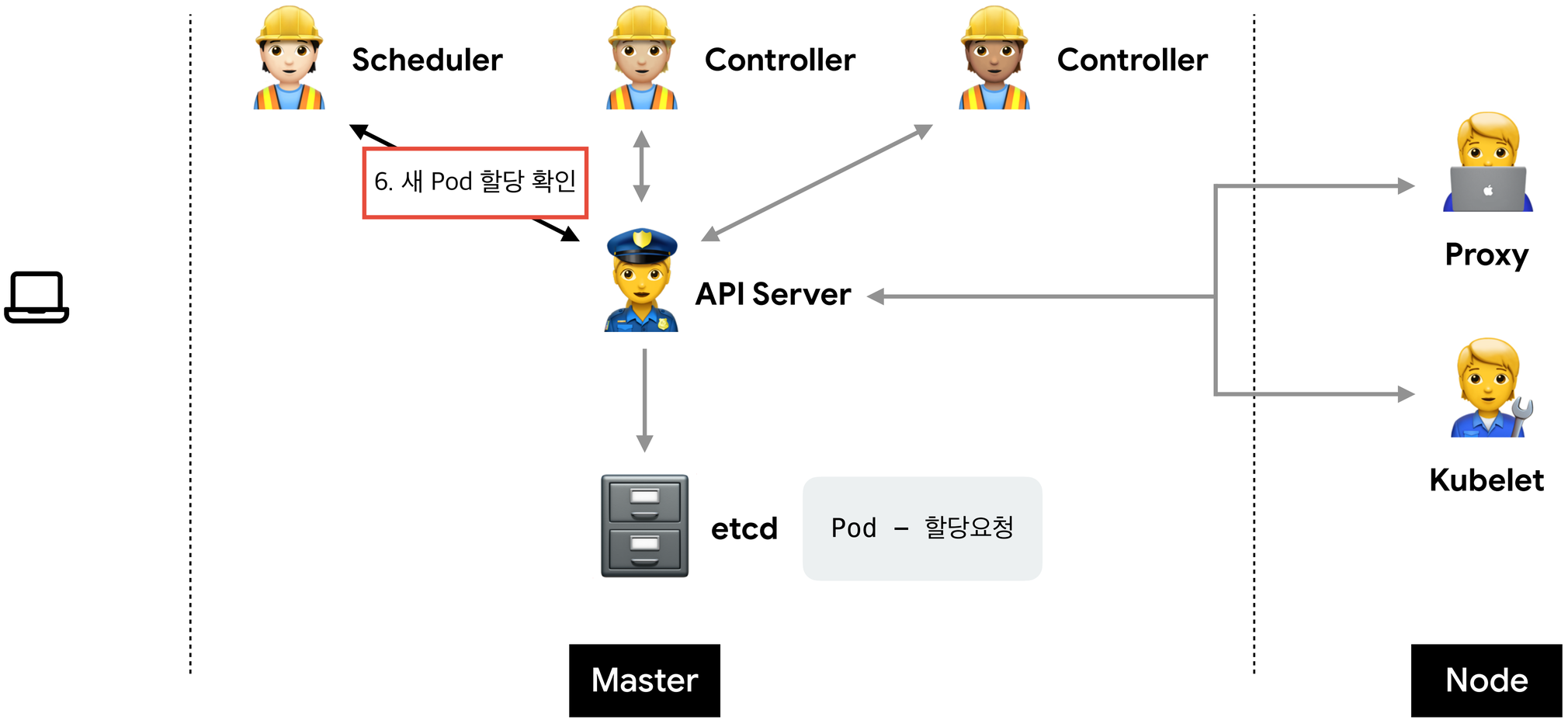
쿠버네티스 흐름


Scheduler
Controller
Controller
4. Pod 할당
API Server
Proxy
Kubelet
etcd
Pod – 생성요청
Master
Node

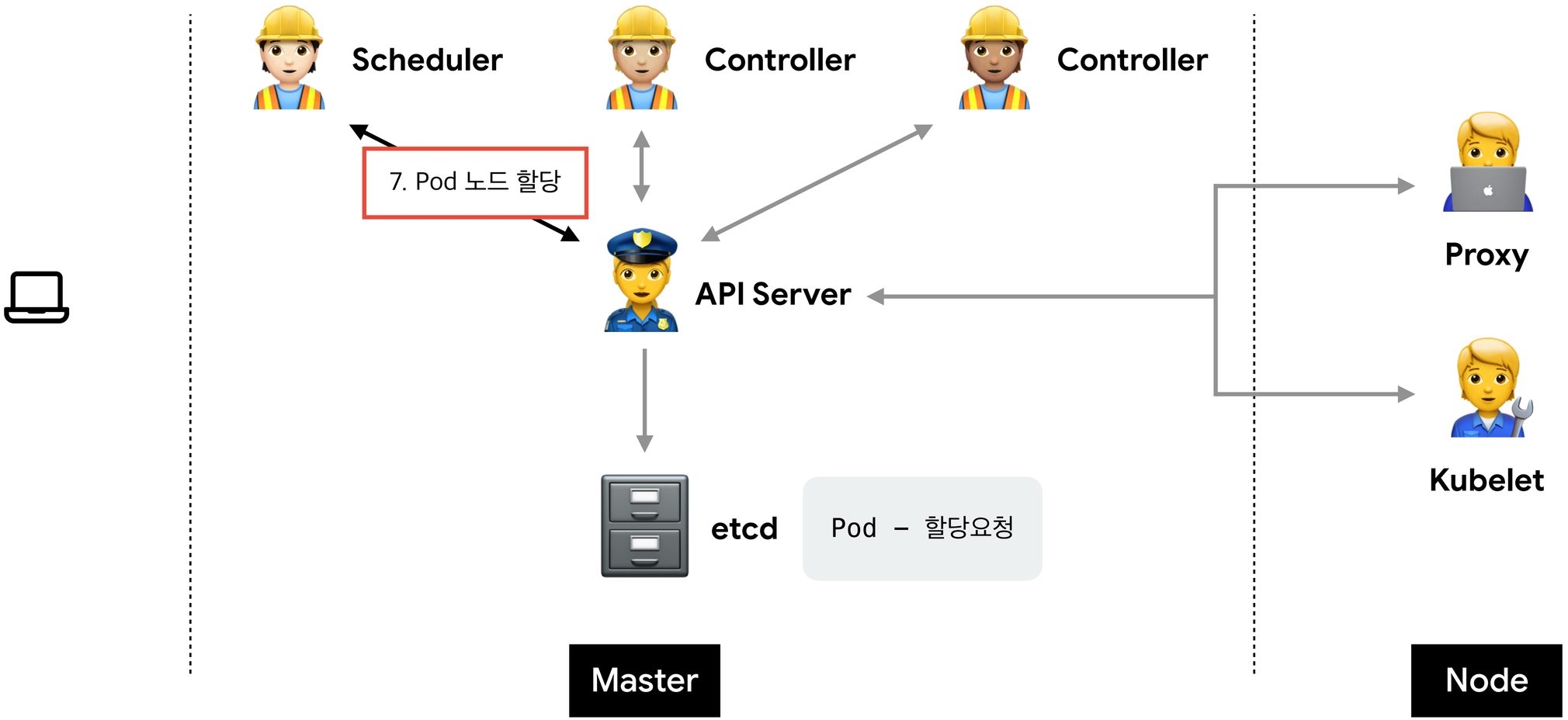
쿠버네티스 흐름


Scheduler
Controller
Controller
API Server
Proxy
Kubelet
5. Pod 할당 요청
etcd
Pod – 할당요청
Master
Node

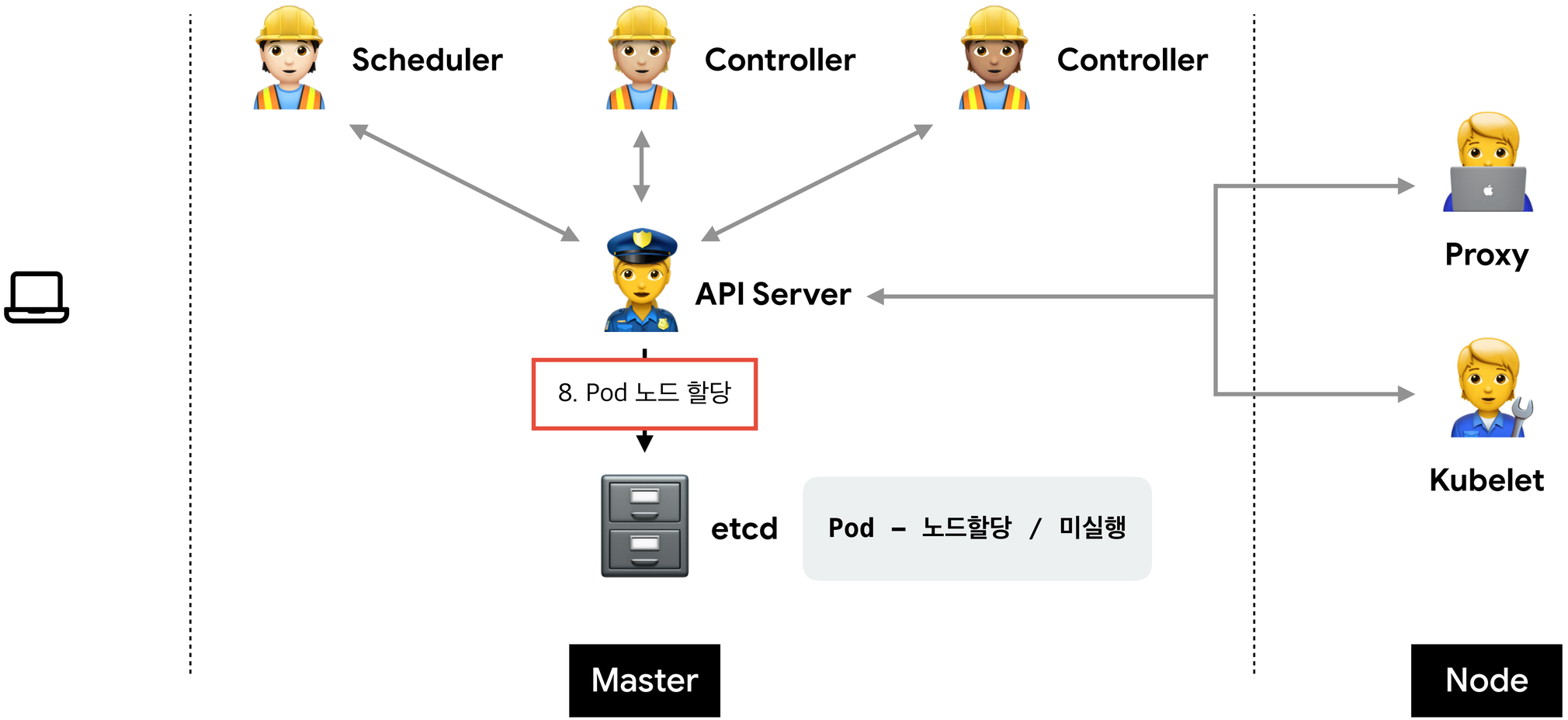
쿠버네티스 흐름


Scheduler
Controller
Controller
6. 새 Pod 할당 확인
API Server
Proxy
Kubelet
etcd
Pod – 할당요청
Master
Node

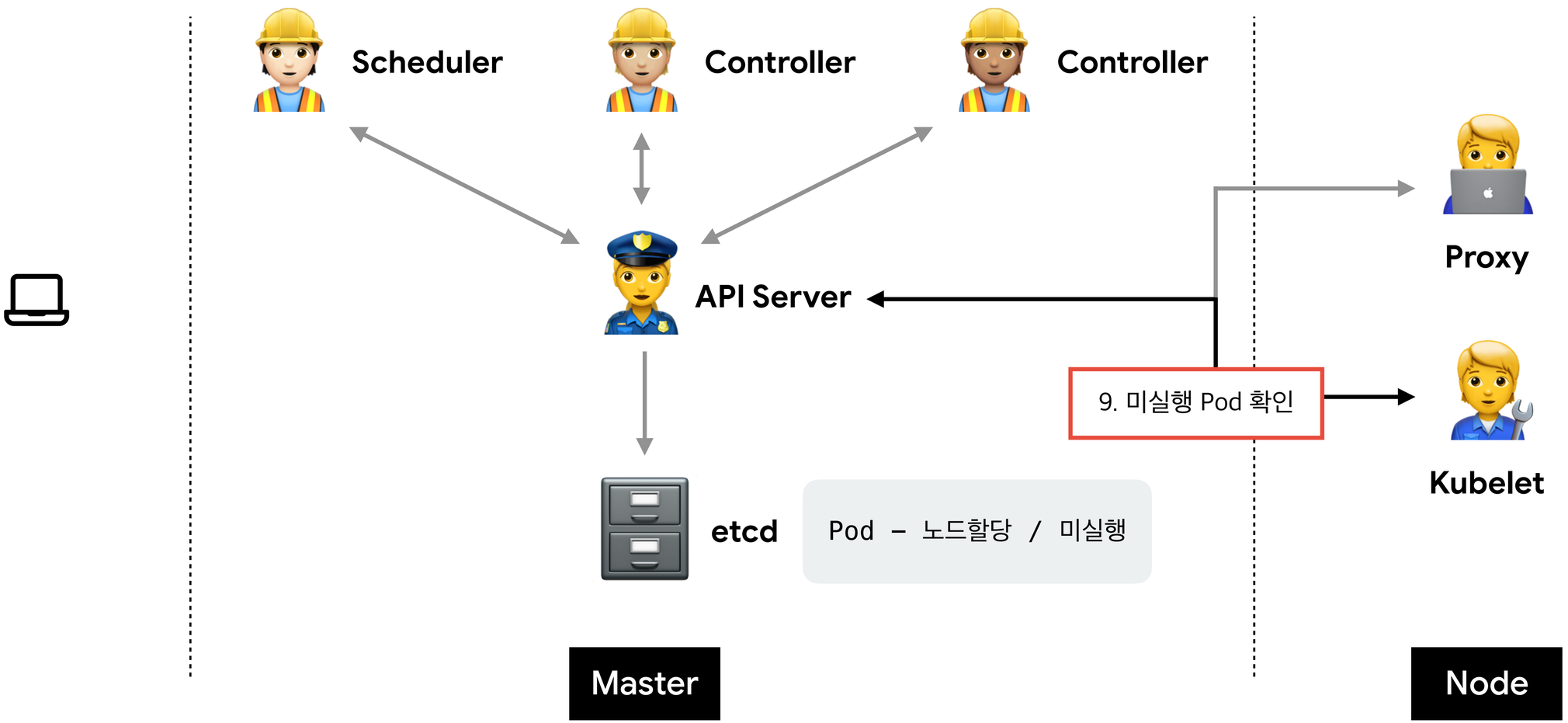
쿠버네티스 흐름


Scheduler
Controller
Controller
7. Pod 노드 할당
API Server
Proxy
Kubelet
etcd
Pod – 할당요청
Master
Node

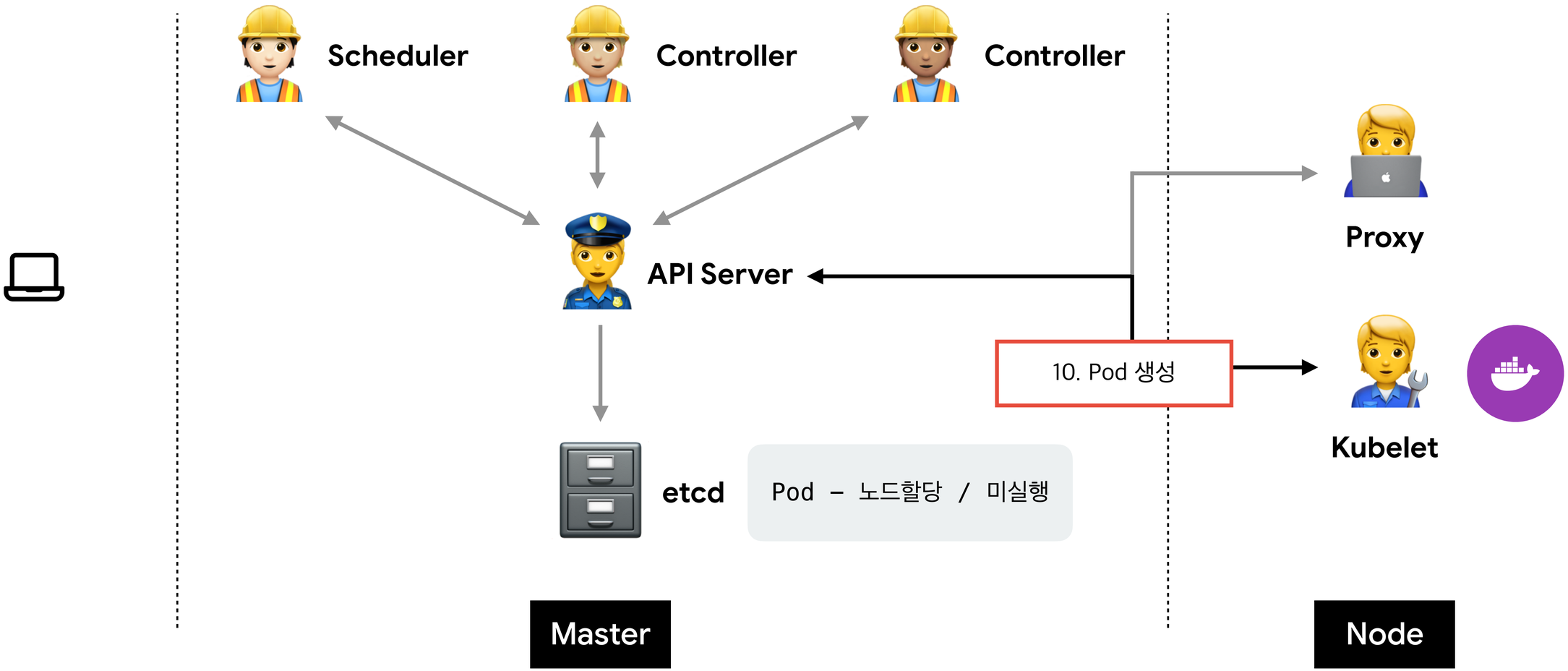
쿠버네티스 흐름


Scheduler
Controller
Controller
API Server
8. Pod 노드 할당
etcd
Pod – 노드할당 / 미실행
Proxy
Kubelet
Master
Node

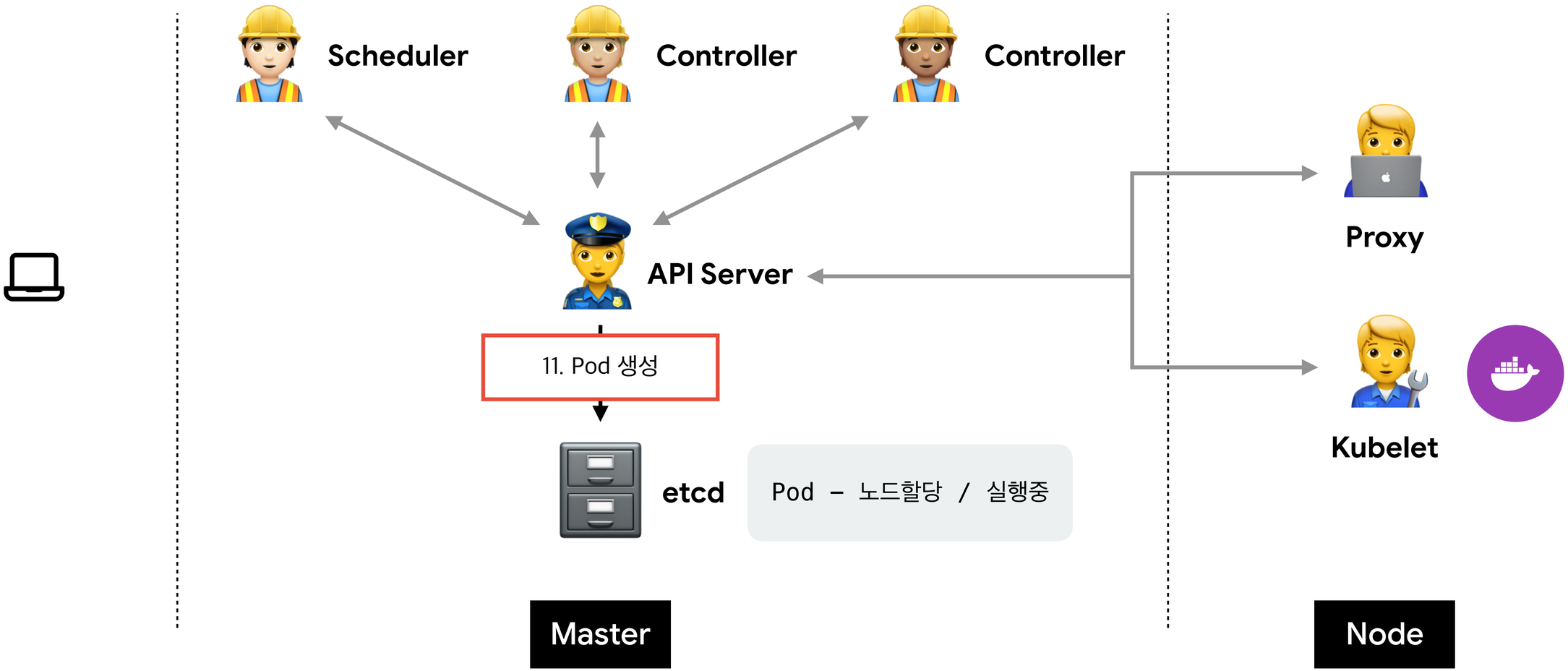
쿠버네티스 흐름


Scheduler
Controller
Controller
API Server
Proxy
9. 미실행 Pod 확인
Kubelet
etcd
Pod – 노드할당 / 미실행
Master
Node

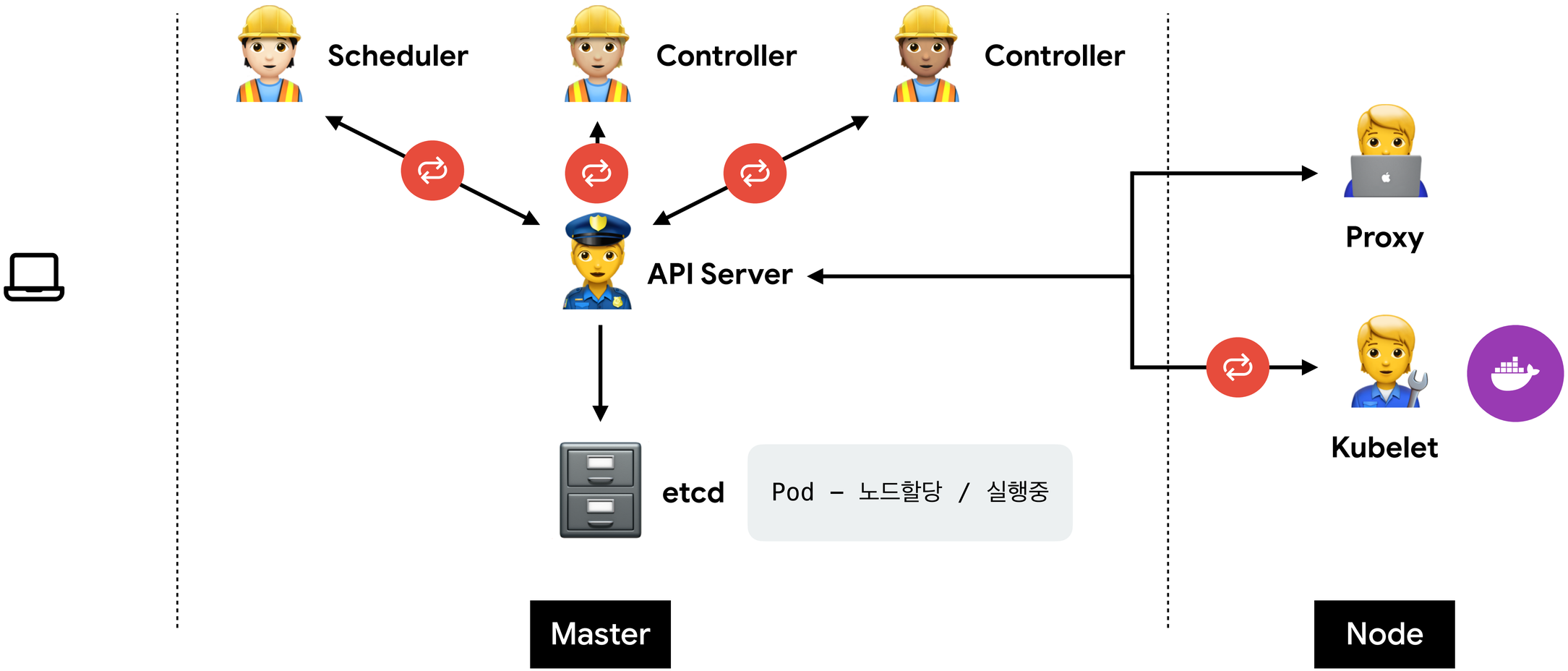
쿠버네티스 흐름


Scheduler
Controller
Controller
API Server
Proxy
10. Pod 생성
Kubelet
etcd
Pod – 노드할당 / 미실행
Master
Node

쿠버네티스 흐름


Scheduler
Controller
Controller
API Server
Proxy
Kubelet
11. Pod 생성
etcd
Pod – 노드할당 / 실행중
Master
Node

쿠버네티스 흐름


Scheduler
Controller
Controller
API Server
Proxy
Kubelet
etcd
Pod – 노드할당 / 실행중
Master
Node

Addons

CNI (네트워크)

DNS (도메인, 서비스 디스커버리)

대시보드 (시각화)

구성/설계 Architecture

오브젝트 Objects

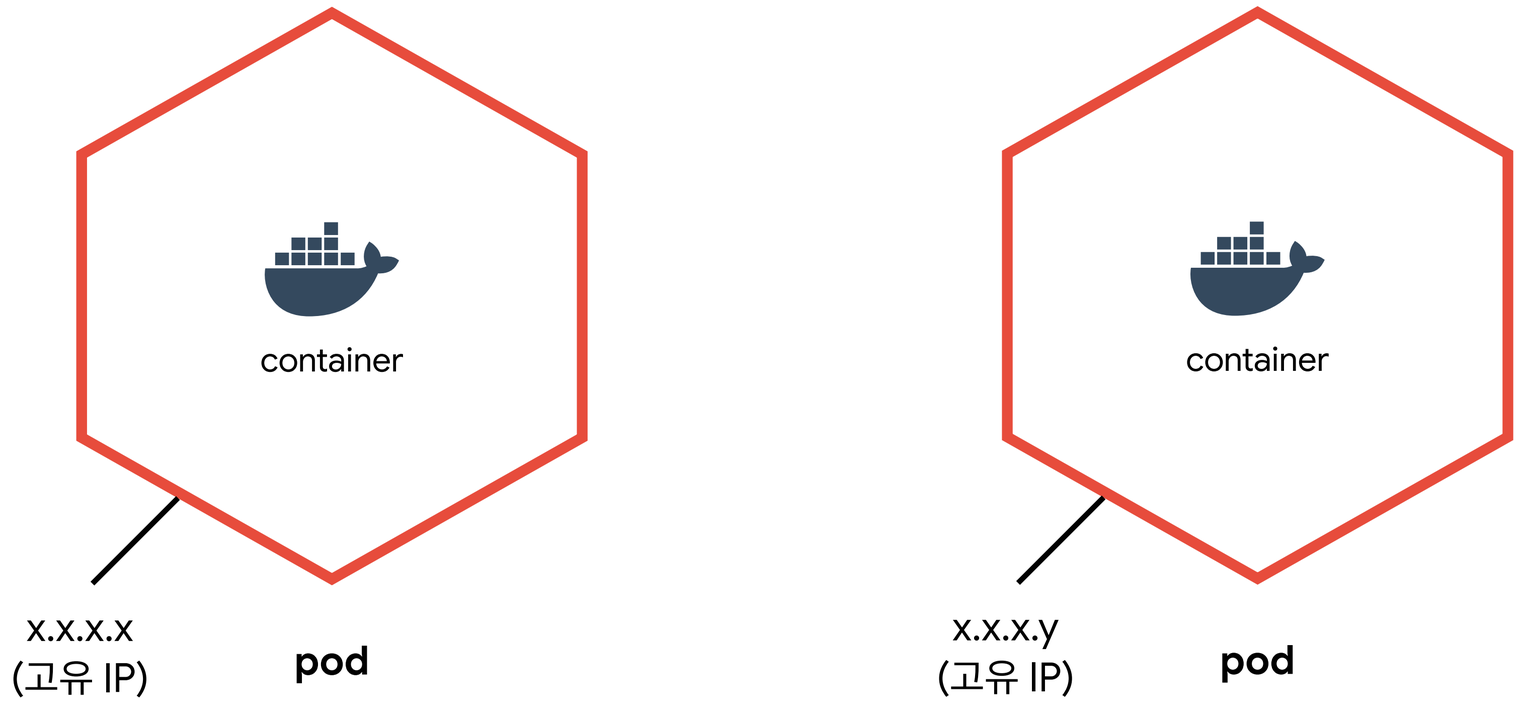
API 호출

Pod

가장 작은 배포 단위

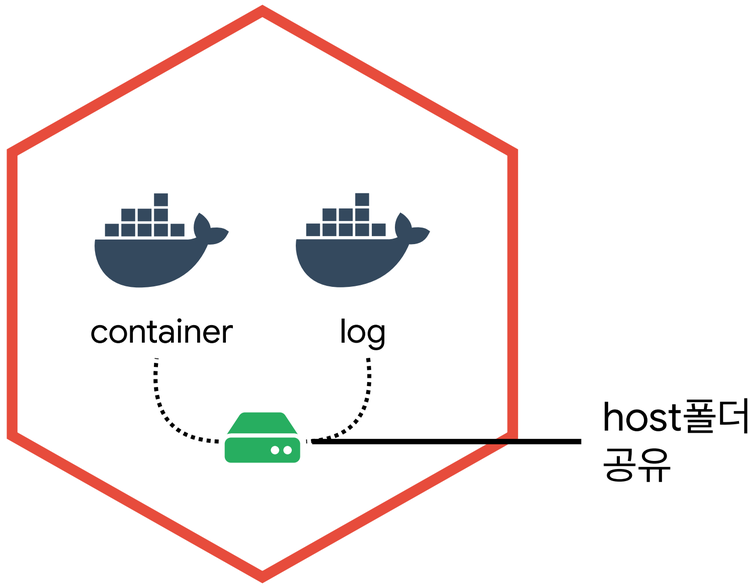
Pod

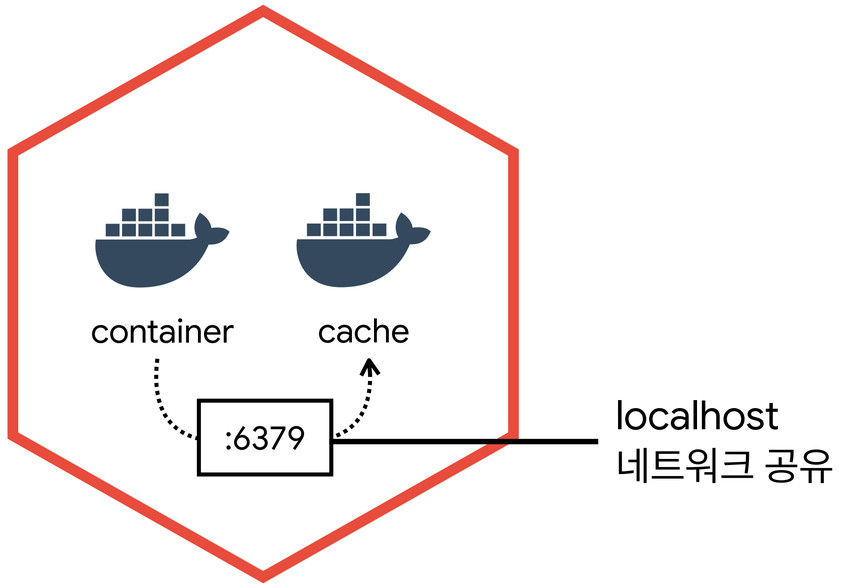
전체 클러스터에서 고유한 IP를 할당



Pod

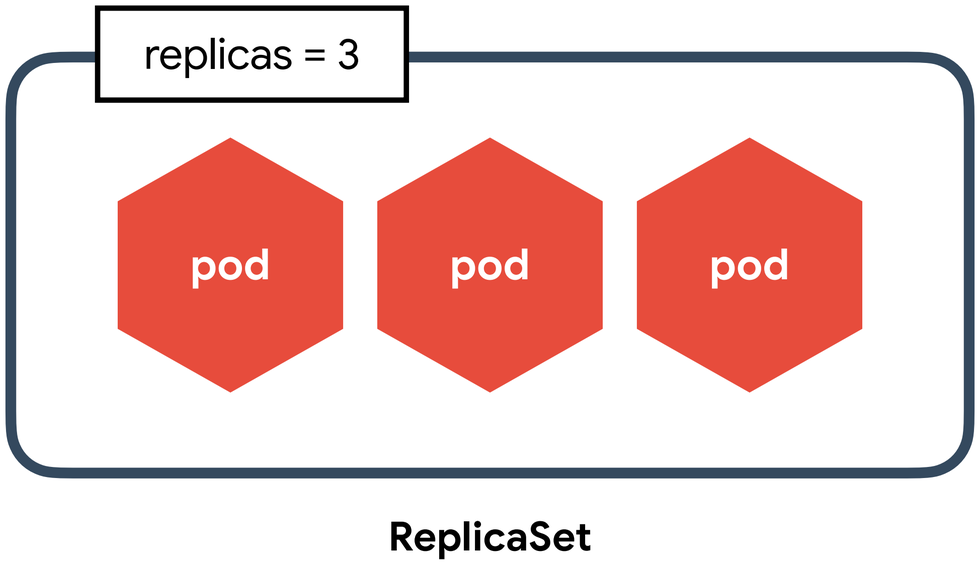
여러개의 컨테이너가 하나의 Pod에 속할 수 있음





ReplicaSet

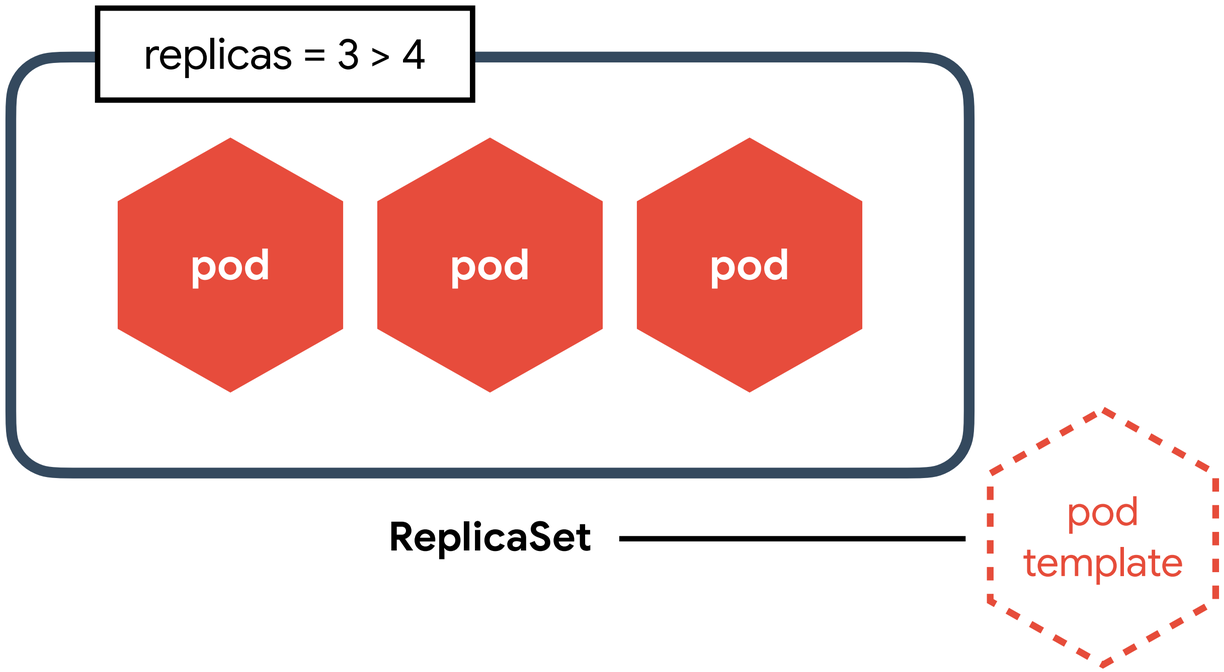
여러개의 Pod을 관리



ReplicaSet

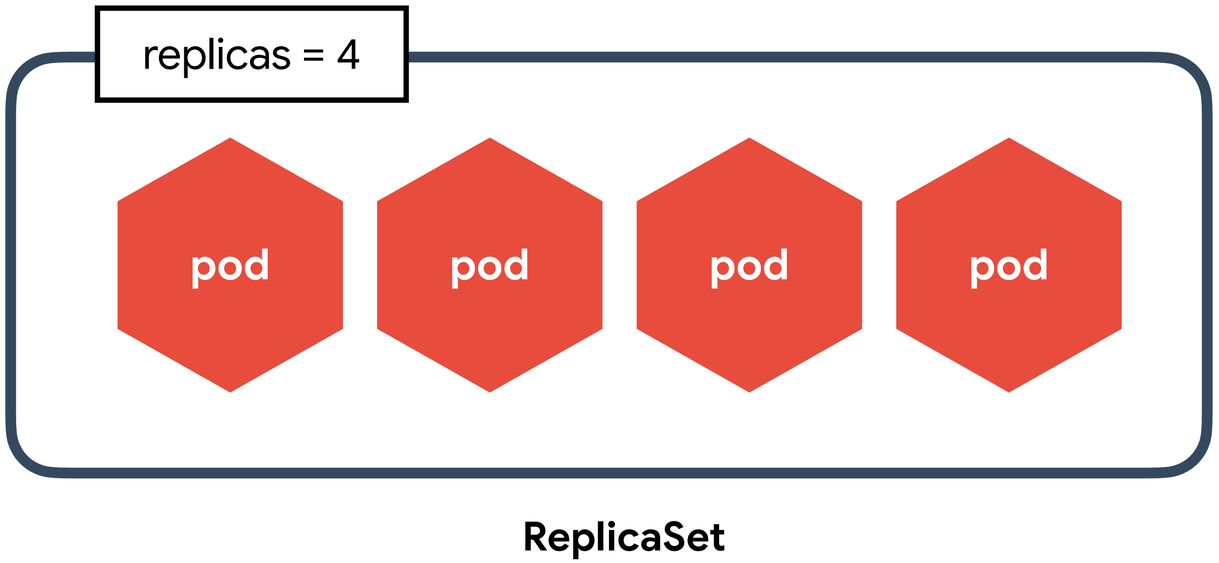
ReplicaSet

새로운 Pod은 Template을 참고하여 생성



ReplicaSet

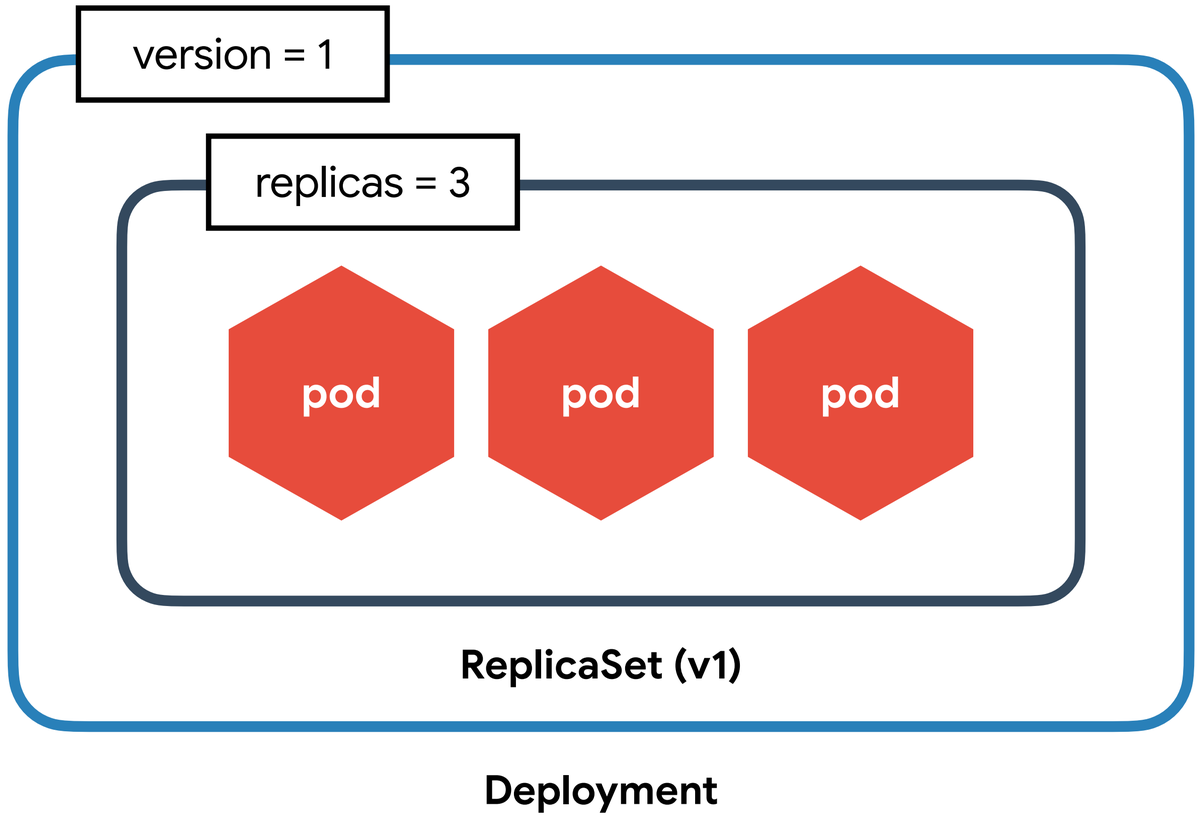
신규 Pod을 생성하거나 기존 Pod을 제거하여 원하는 수(Replicas)를 유지



ReplicaSet

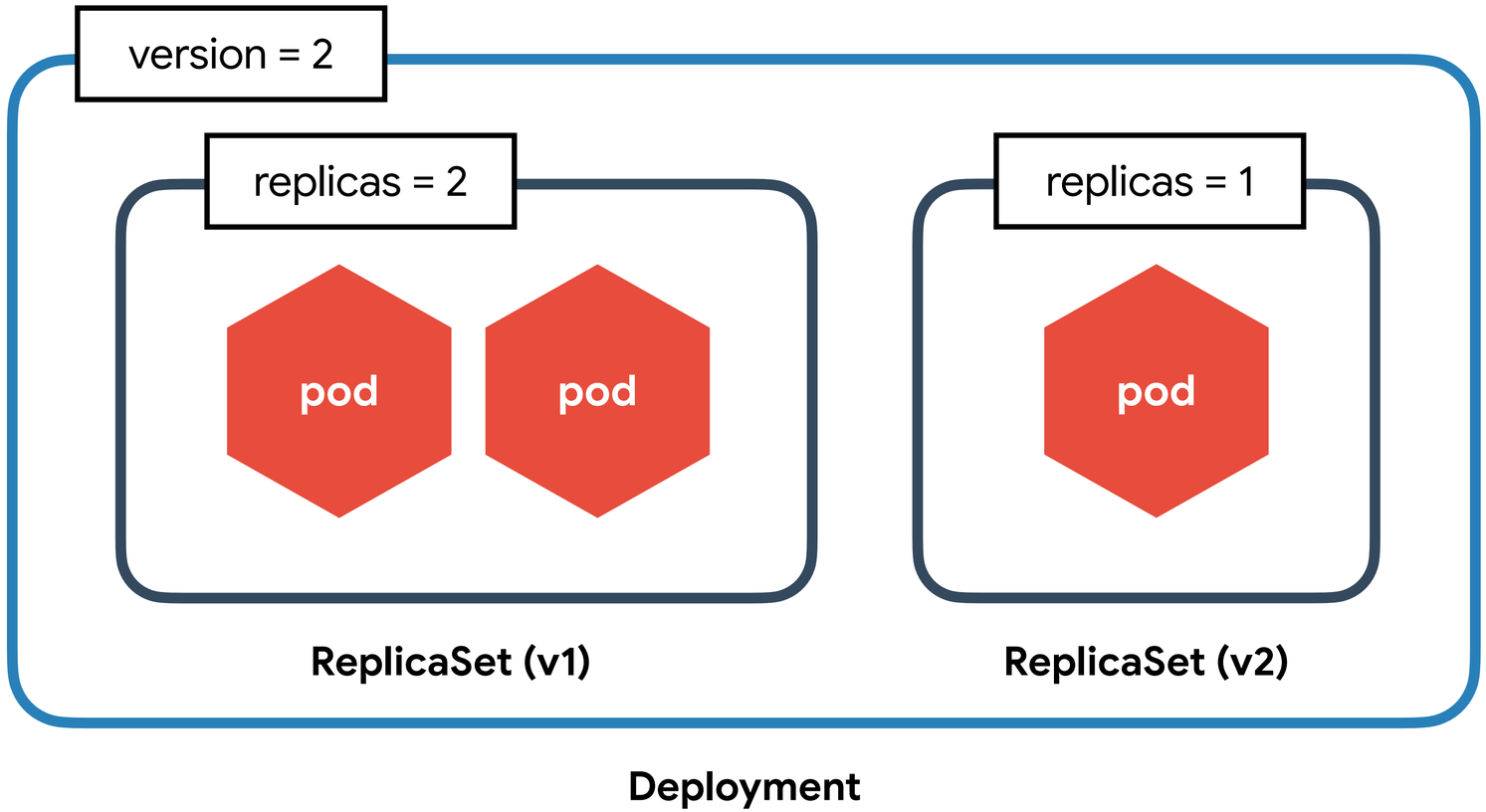
Deployment

배포 버전을 관리



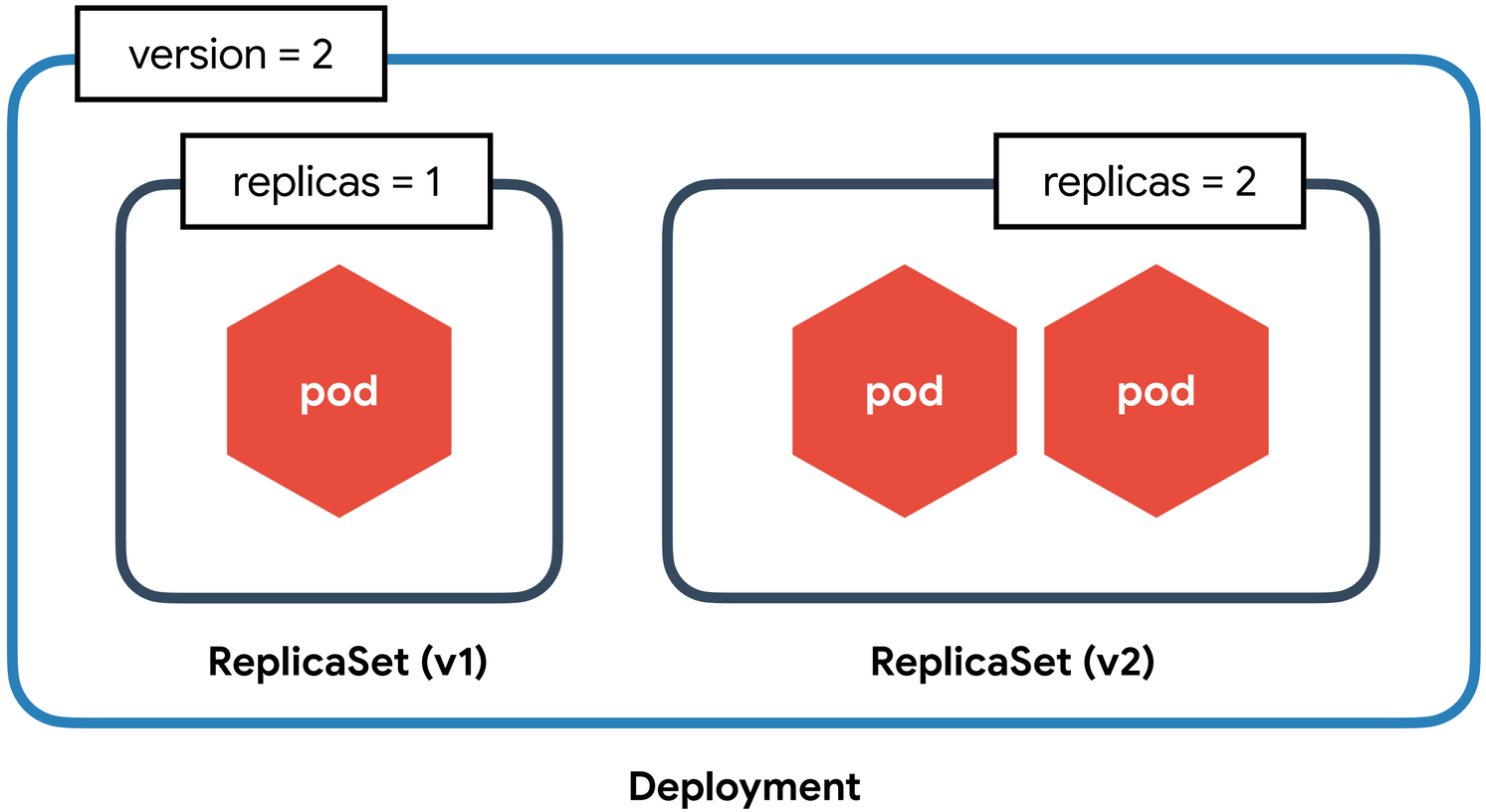
Deployment

내부적으로 ReplicaSet을 이용



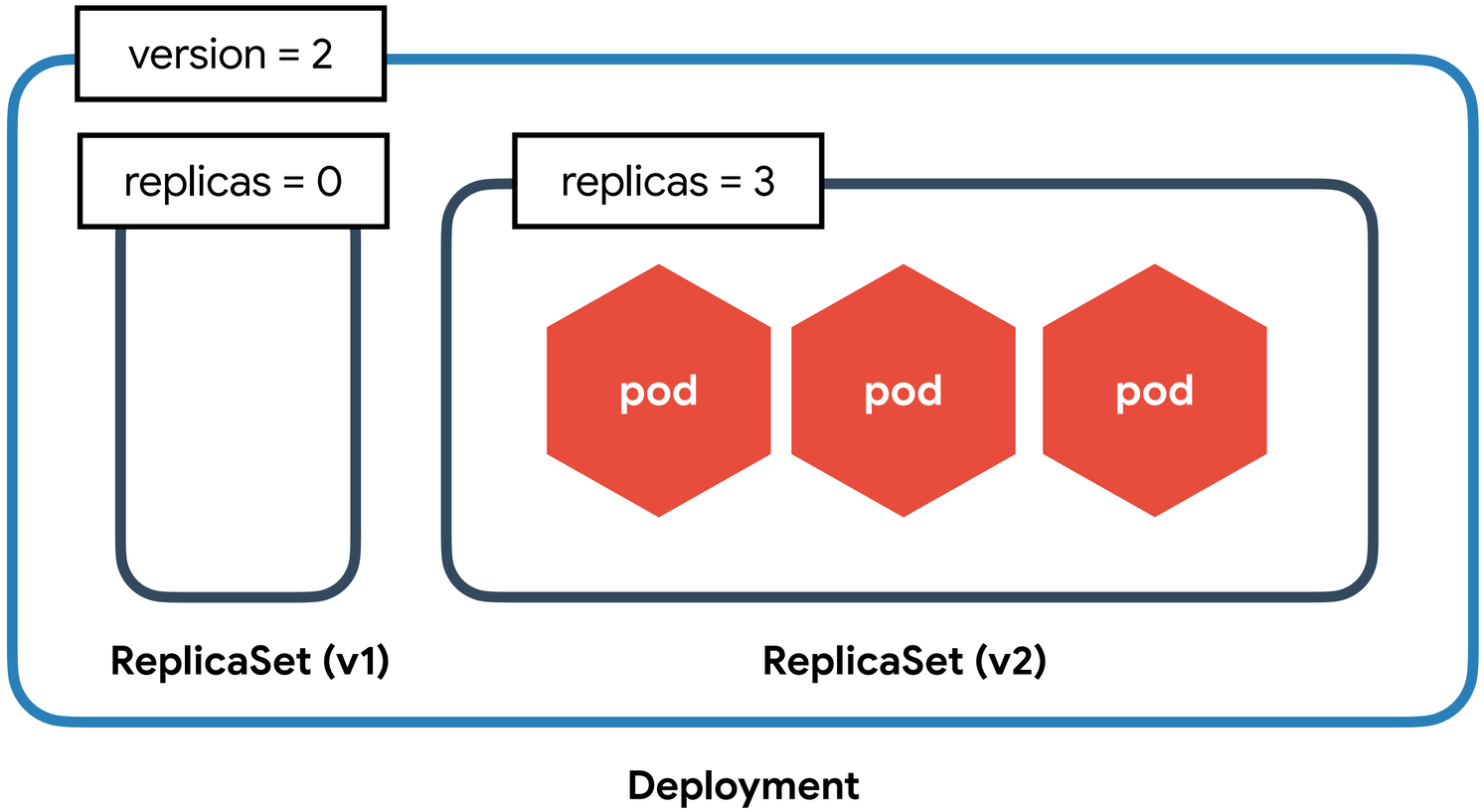
Deployment

내부적으로 ReplicaSet을 이용

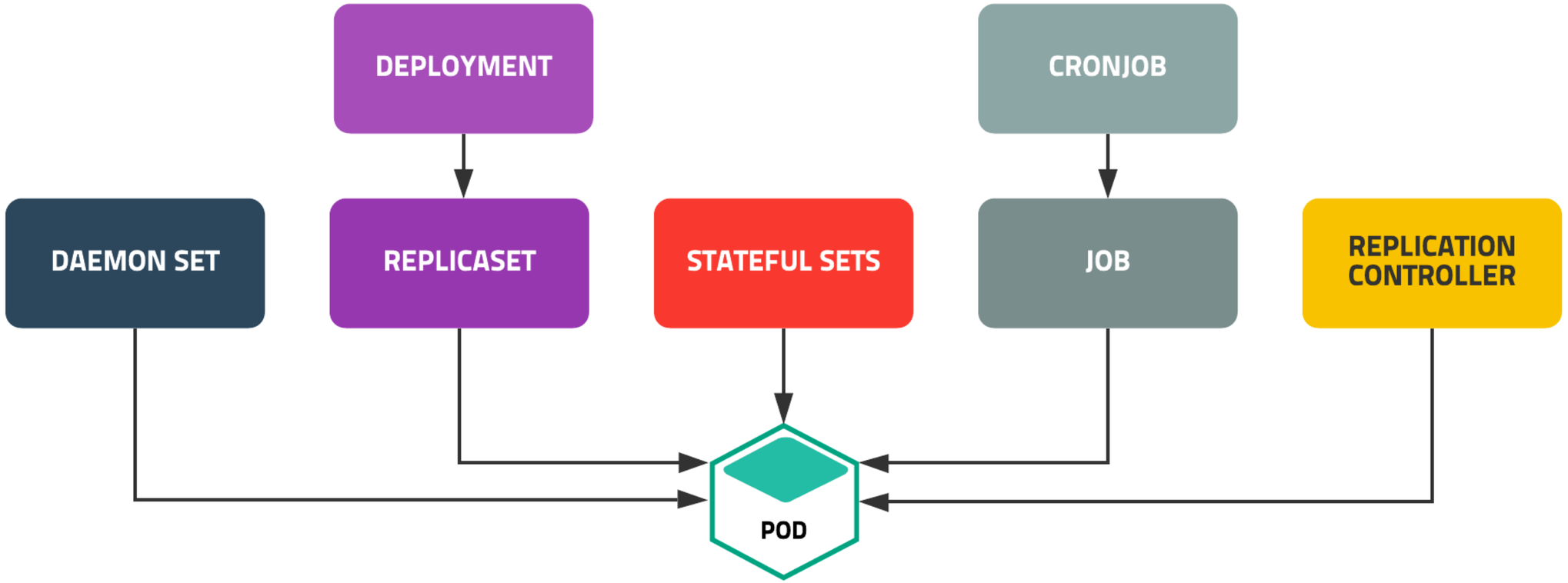


Deployment

내부적으로 ReplicaSet을 이용

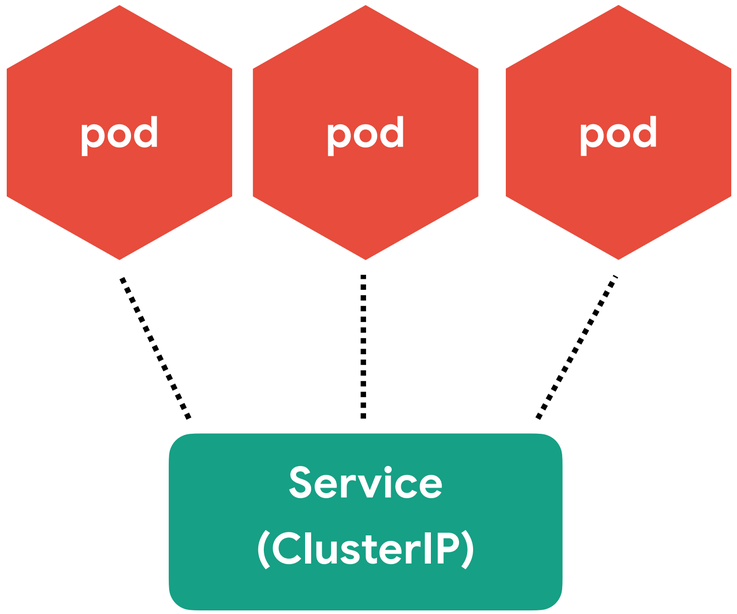


다양한 Workload



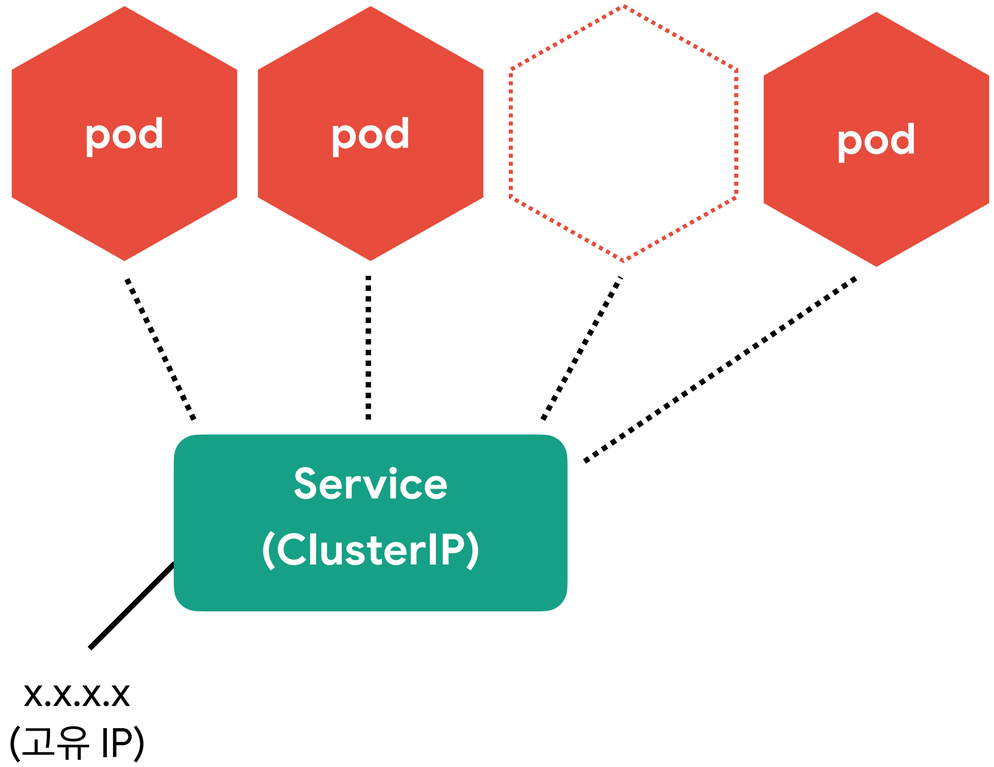
Service - ClusterIP

클러스터 내부에서 사용하는 프록시



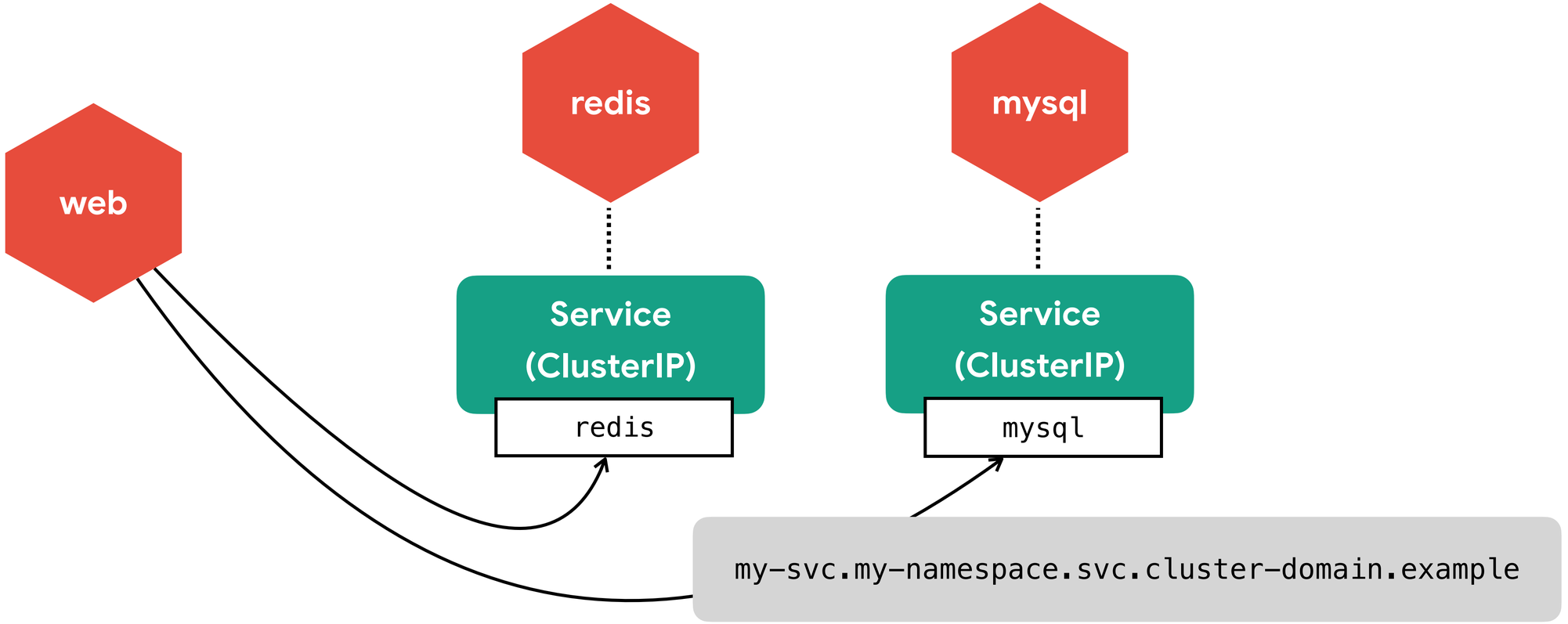
Service - ClusterIP

Pod은 동적이지만 서비스는 고유 IP를 가짐



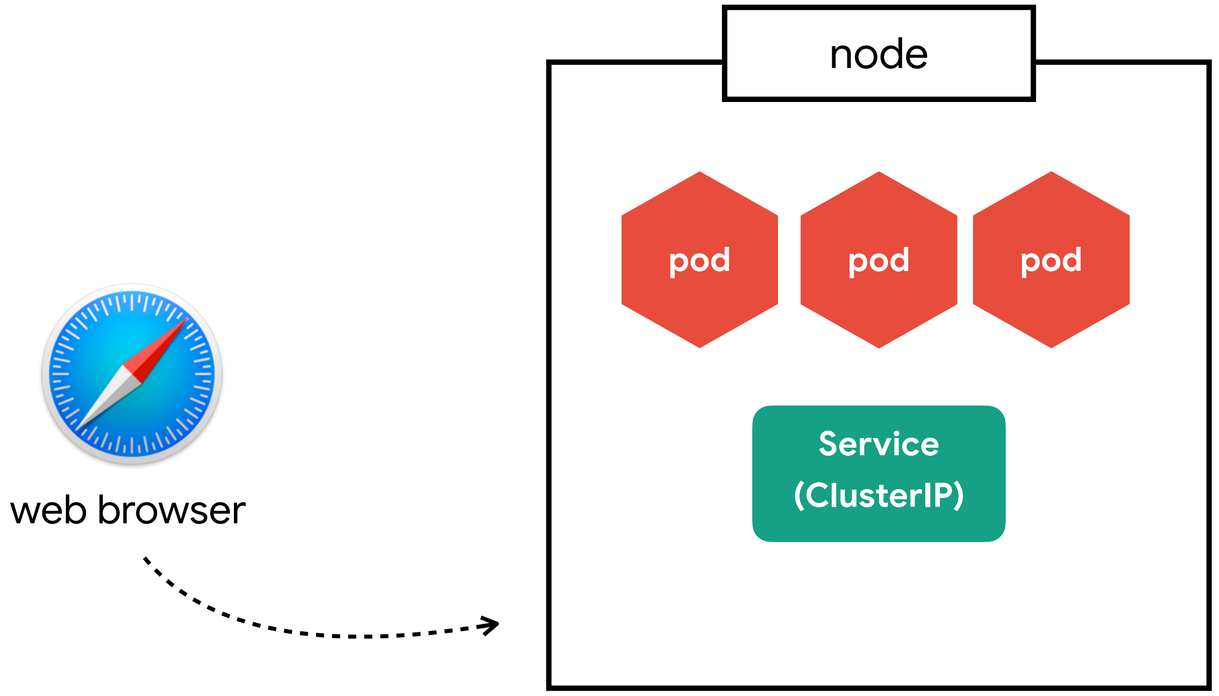
Service - ClusterIP

클러스터 내부에서 서비스 연결은 DNS를 이용



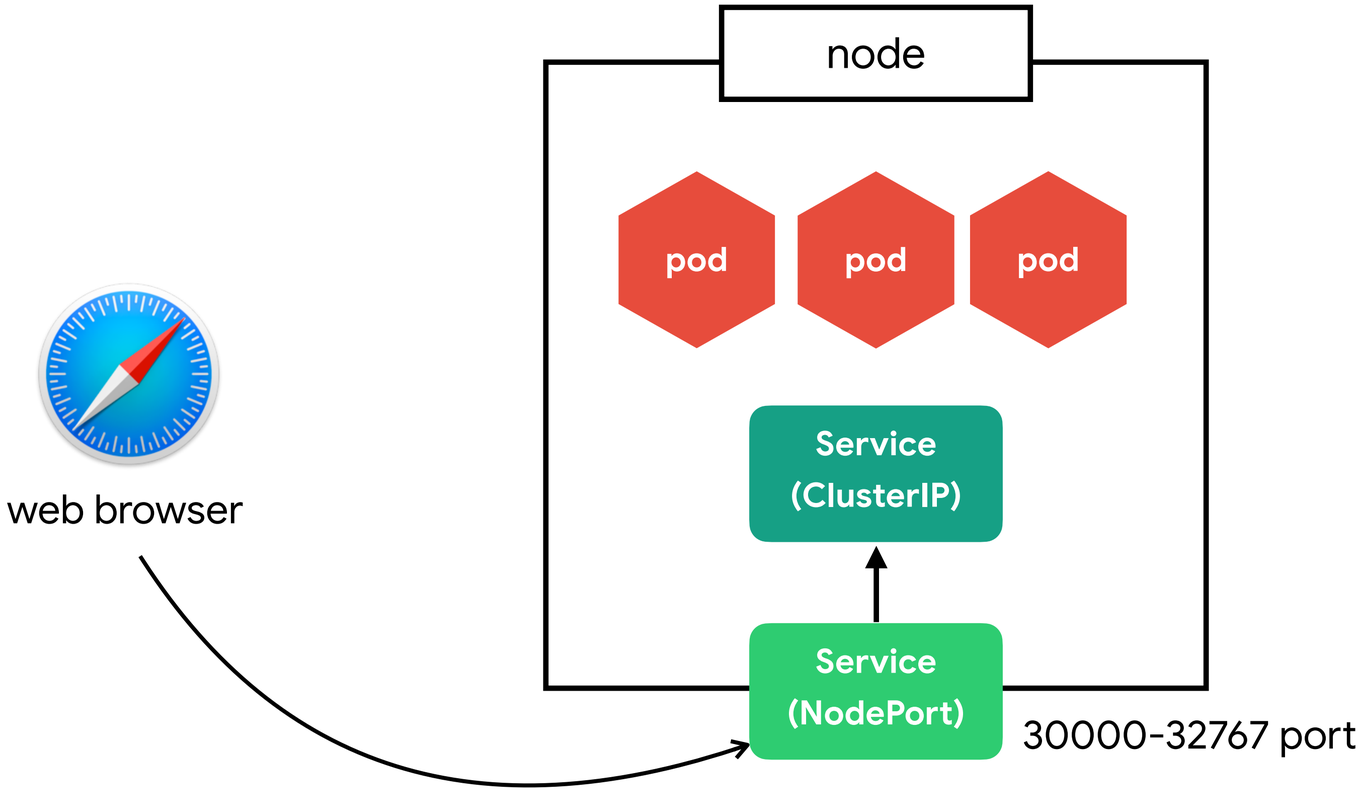
Service - NodePort

노드(host)에 노출되어 외부에서 접근 가능한 서비스



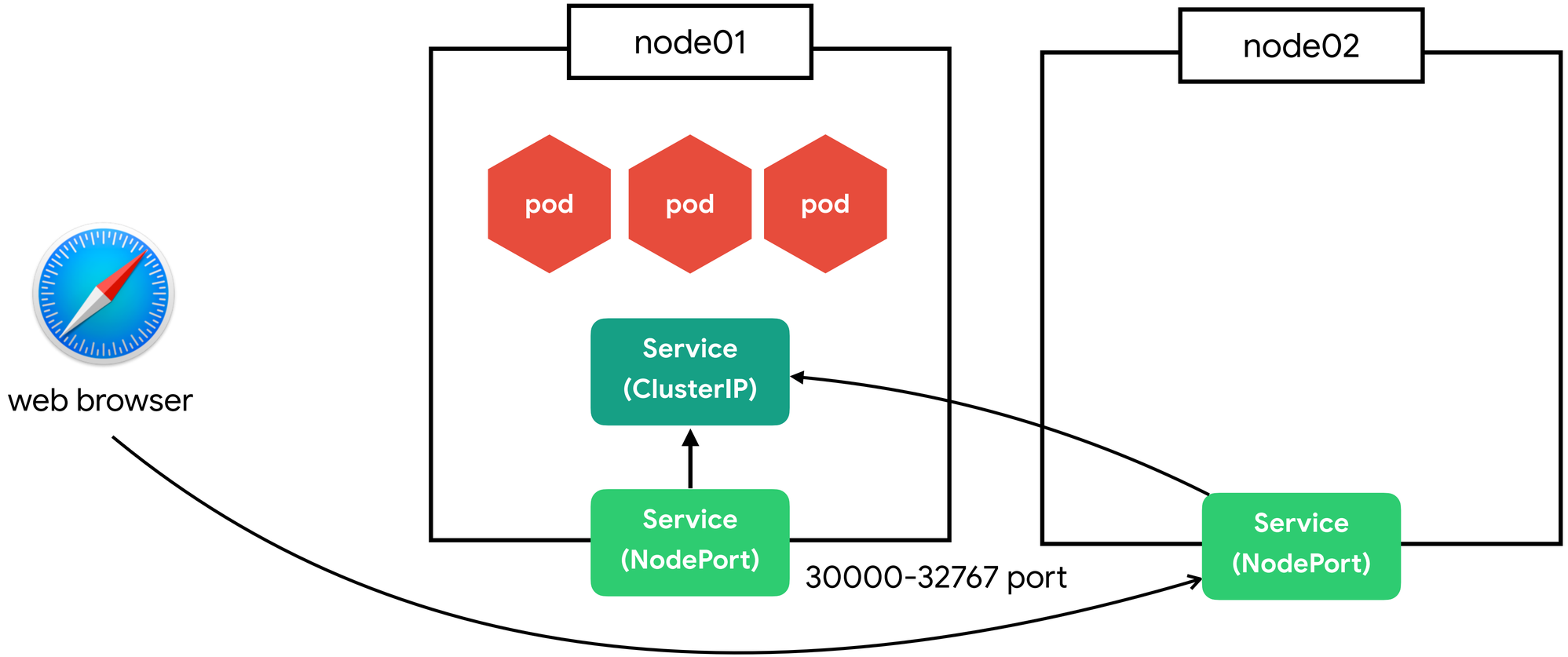
Service - NodePort

노드(host)에 노출되어 외부에서 접근 가능한 서비스



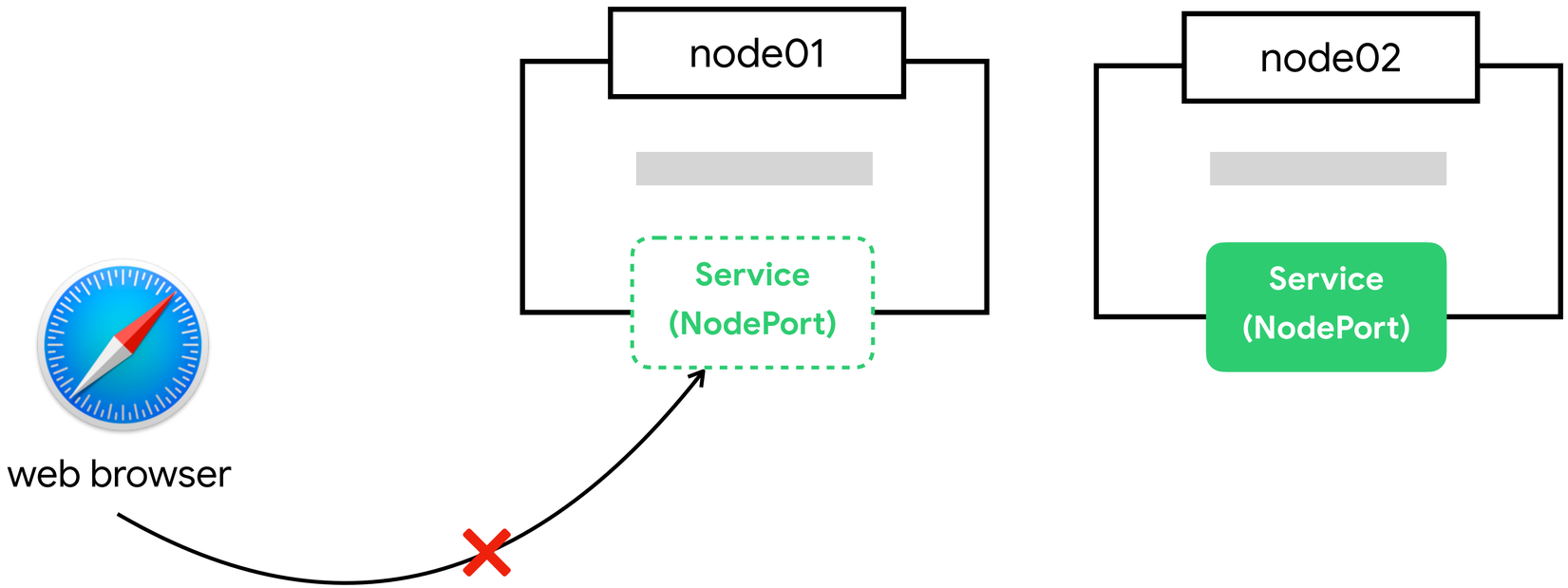
Service - NodePort

모든 노드에 동일한 포트로 생성



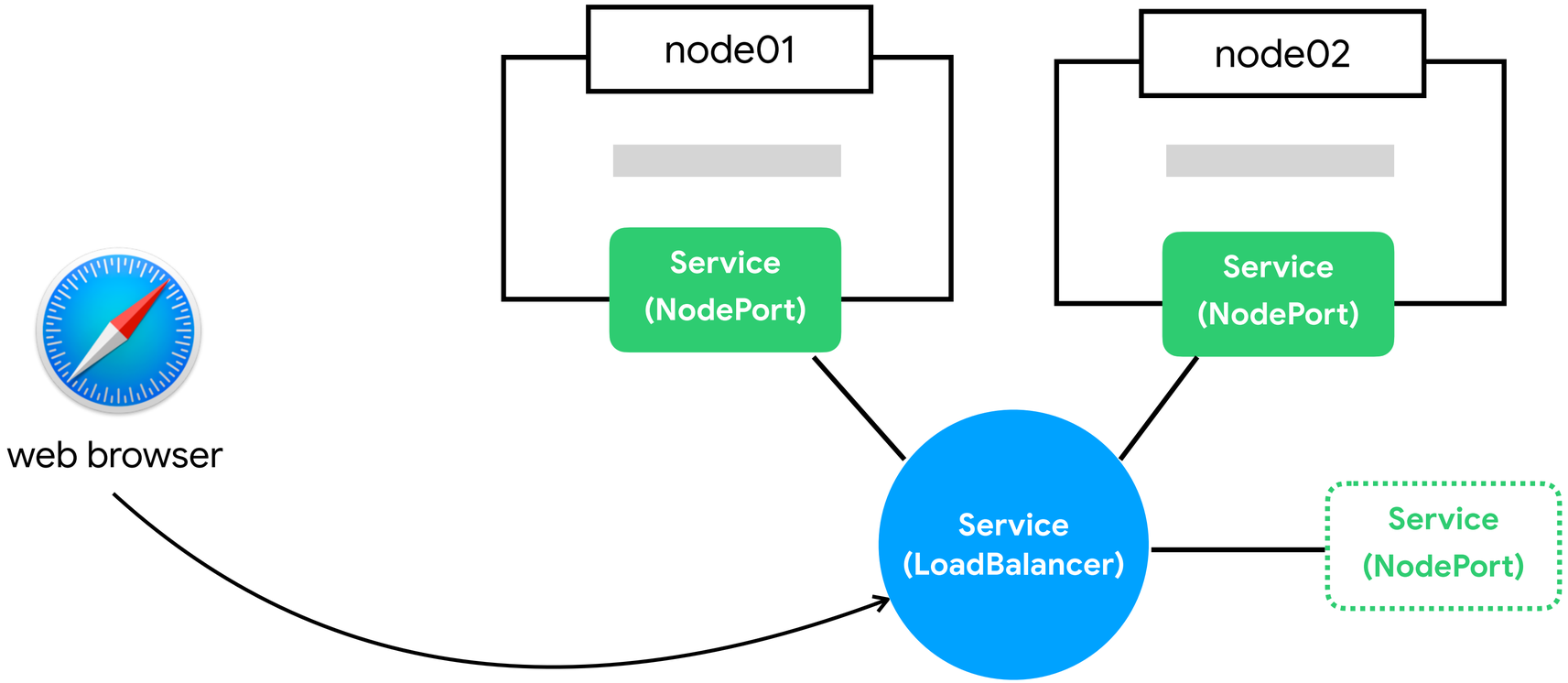
Service - LoadBalancer

하나의 IP주소를 외부에 노출



Service - LoadBalancer

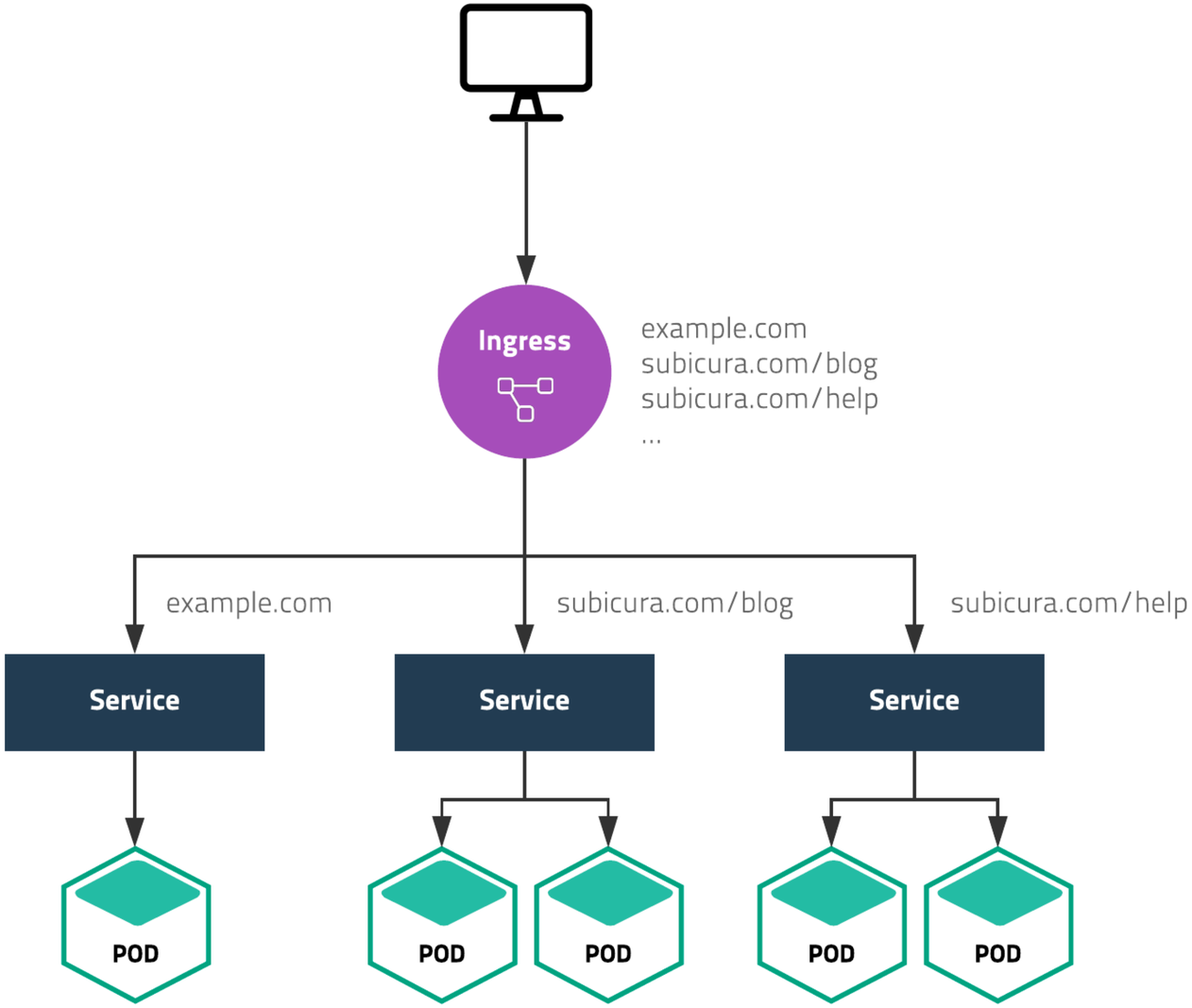
하나의 IP주소를 외부에 노출



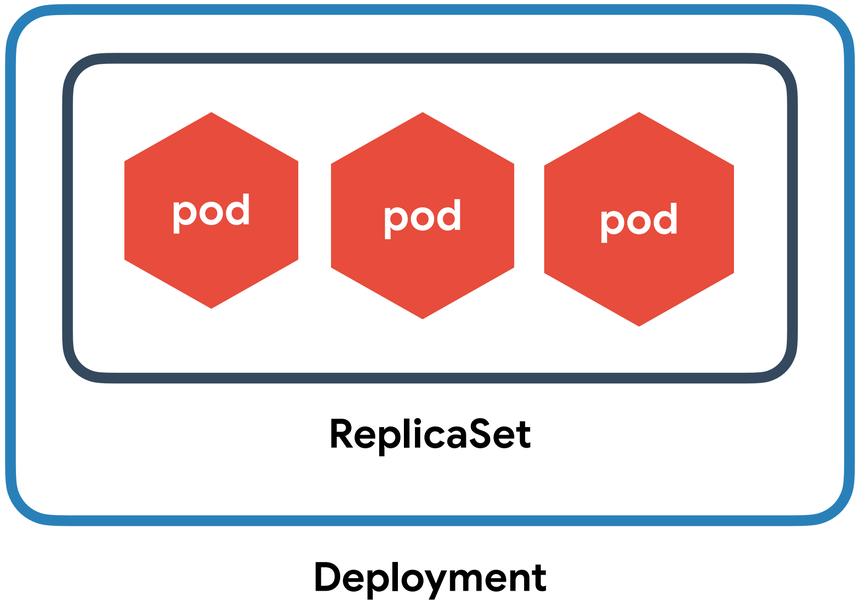
Ingress

도메인 또는 경로별 라우팅

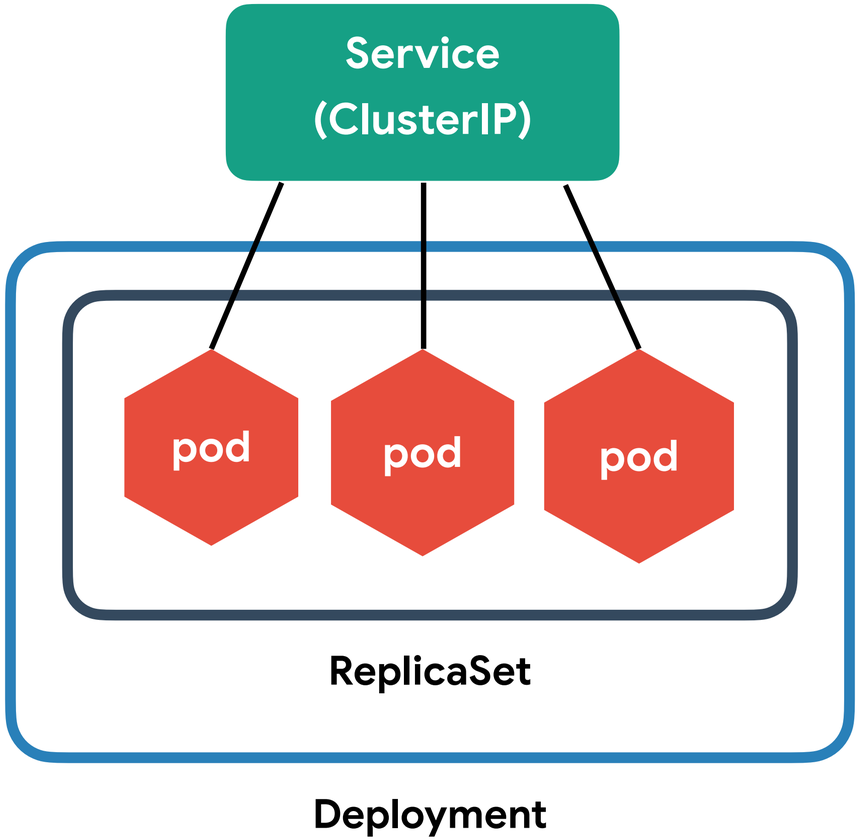
- Nginx, HAProxy, ALB, ...



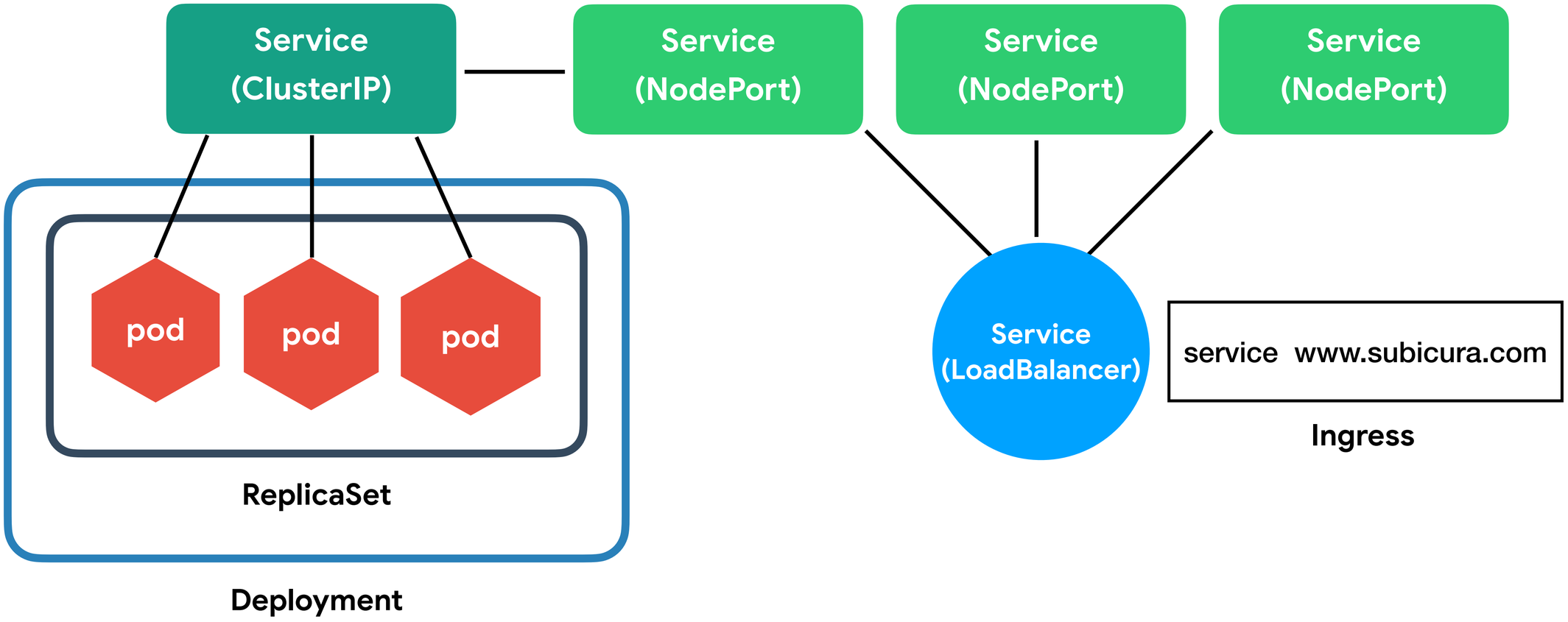
일반적인 구성



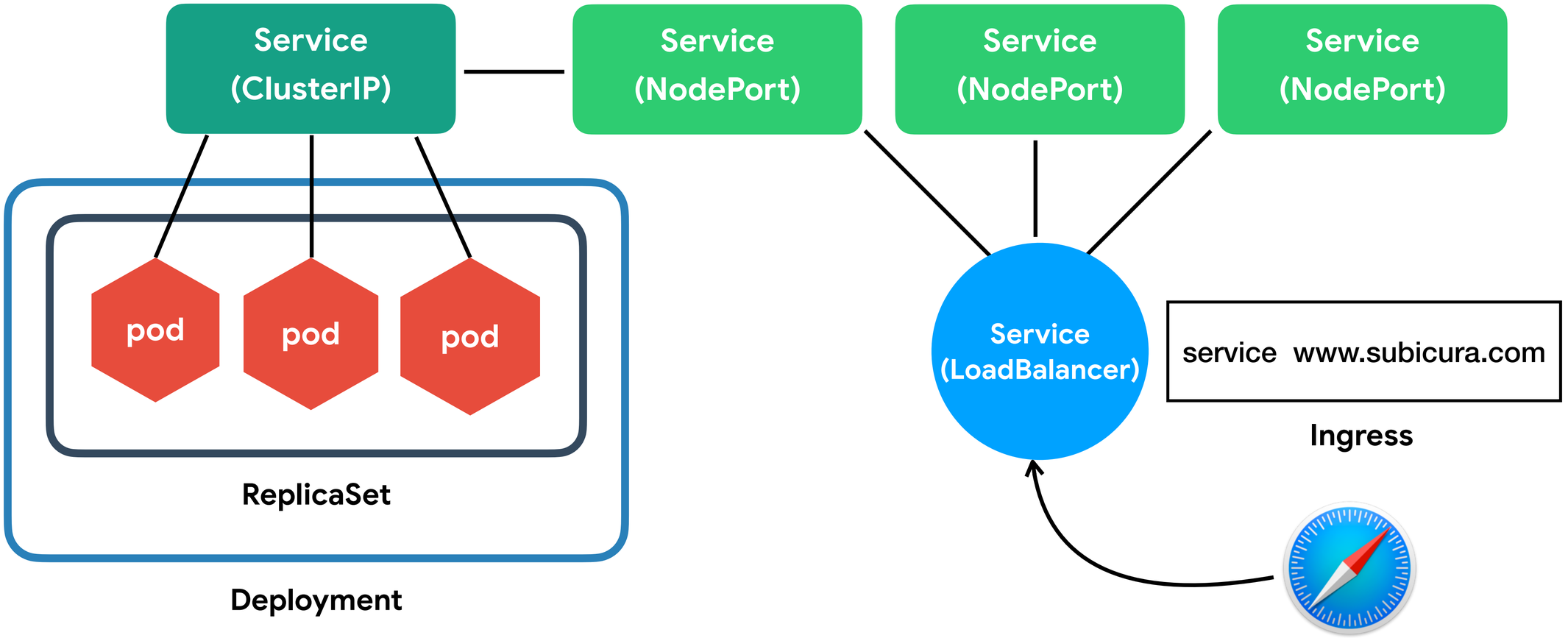
일반적인 구성



일반적인 구성


Service
(ClusterIP)
Service
(NodePort)
Service
(NodePort)
Service
(NodePort)
pod
pod
pod
ReplicaSet
Deployment
Service
(LoadBalancer)
service www.subicura.com
Ingress

일반적인 구성


Service
(ClusterIP)
Service
(NodePort)
Service
(NodePort)
Service
(NodePort)
pod
pod
pod
ReplicaSet
Deployment
Service
(LoadBalancer)
service www.subicura.com
Ingress

그 외 기본 오브젝트

`Volume` - Storage (EBS, NFS, ...)

`Namespace` - 논리적인 리소스 구분

`ConfigMap/Secret` - 설정

`ServiceAccount` - 권한 계정

`Role/ClusterRole` - 권한 설정 (get, list, watch, create, ...)

...

구성/설계 Architecture

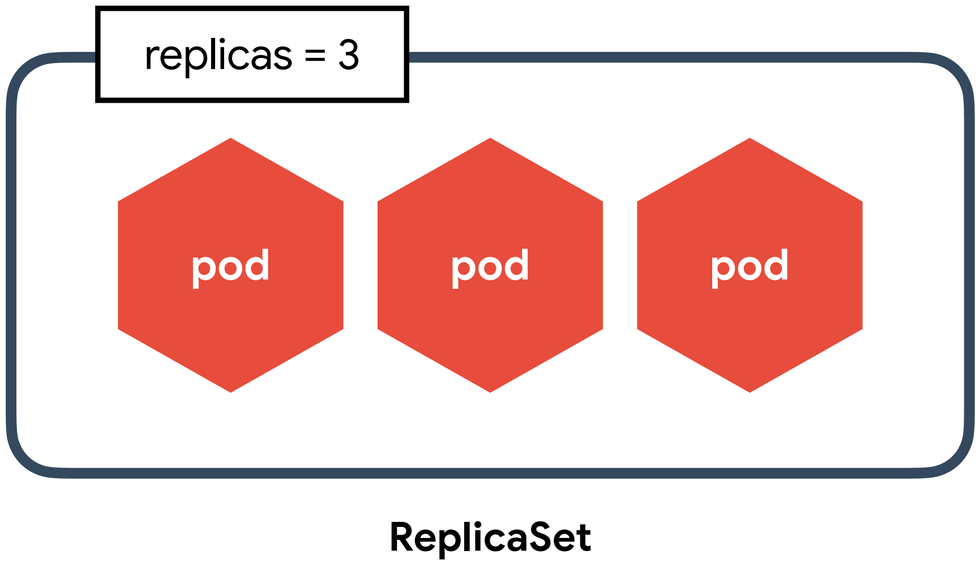
오브젝트 Objects

API 호출

Object Spec - YAML


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: example
spec:
  containers:
  - name: busybox
    image: busybox:1.25
```

Object Spec - YAML



ReplicaSet

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: web
        image: image:v1
```

Object Spec - YAML

ArgoCD (Custom Resource)

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL:
https://github.com/argoproj/argocd-exa
mple-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    server:
https://kubernetes.default.svc
    namespace: guestbook
```

Object Spec

apiVersion

- apps/v1, v1, batch/v1, networking.k8s.io/v1, ...

kind

- Pod, Deployment, Service, Ingress, ...

metadata

- name, label, namespace, ...

spec

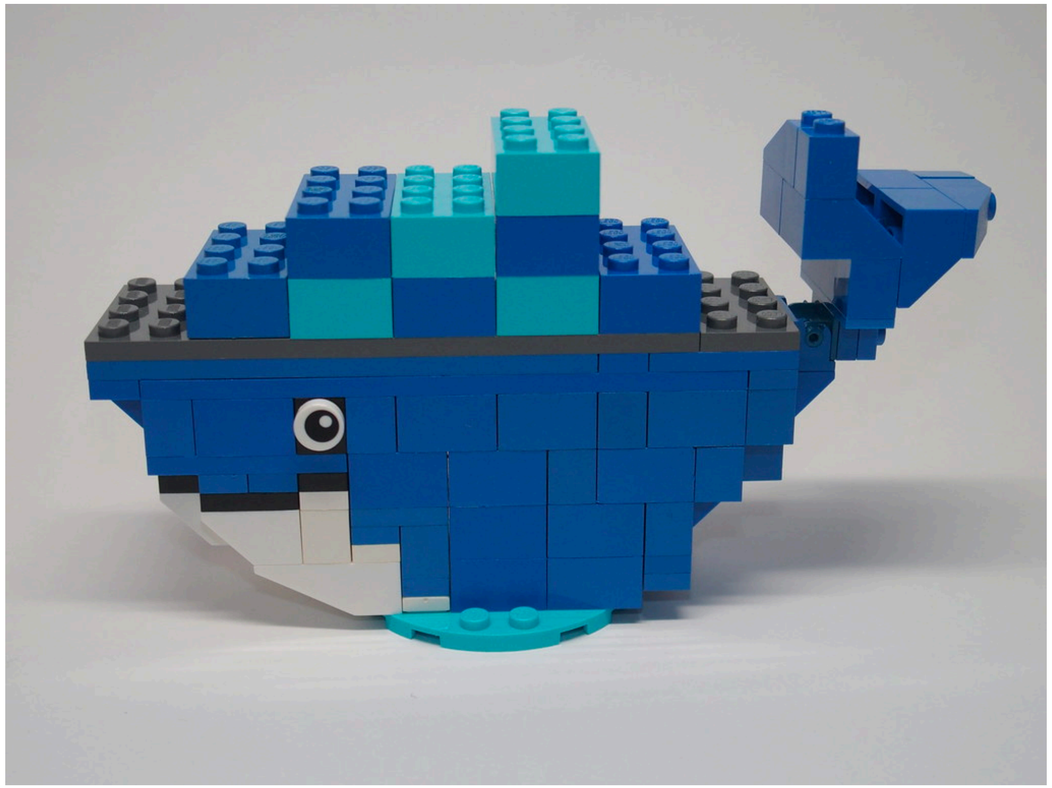
- 각종 설정 (https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.18)

status(read-only)

- 시스템에서 관리하는 최신 상태

API 호출하기

원하는 상태(desired state)를 다양한 오브젝트(object)로 정의(spec)하고 API 서버에 yaml형식으로 전달



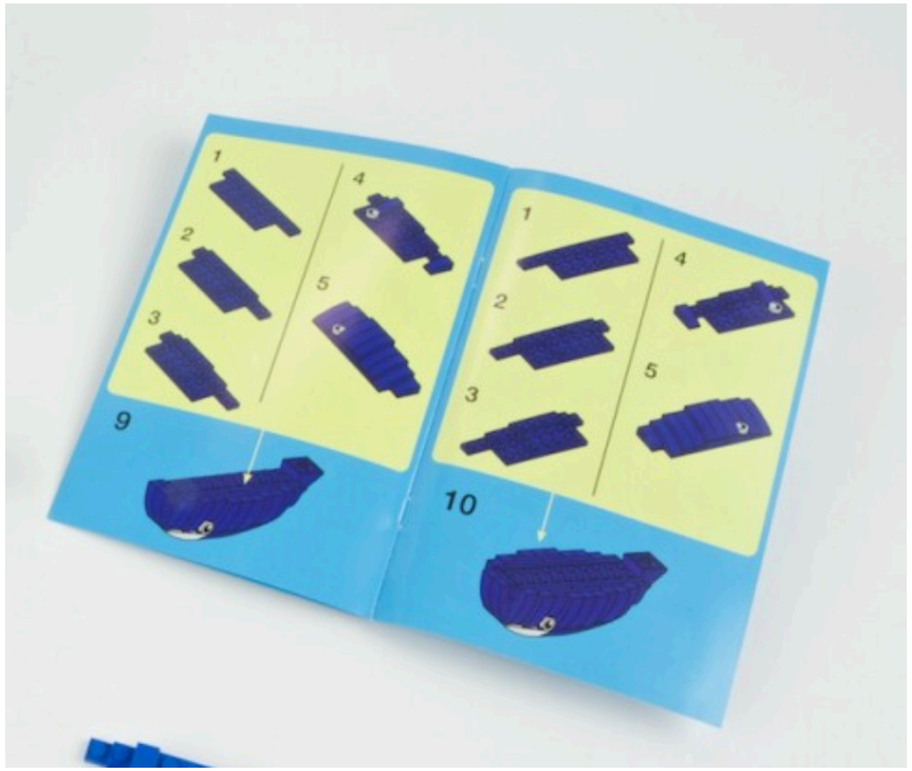
API 호출하기

원하는 상태(desired state)를 **다양한 오브젝트(object)**로 정의(spec)하고 API 서버에 yaml형식으로 전달

@Michael Mattsson

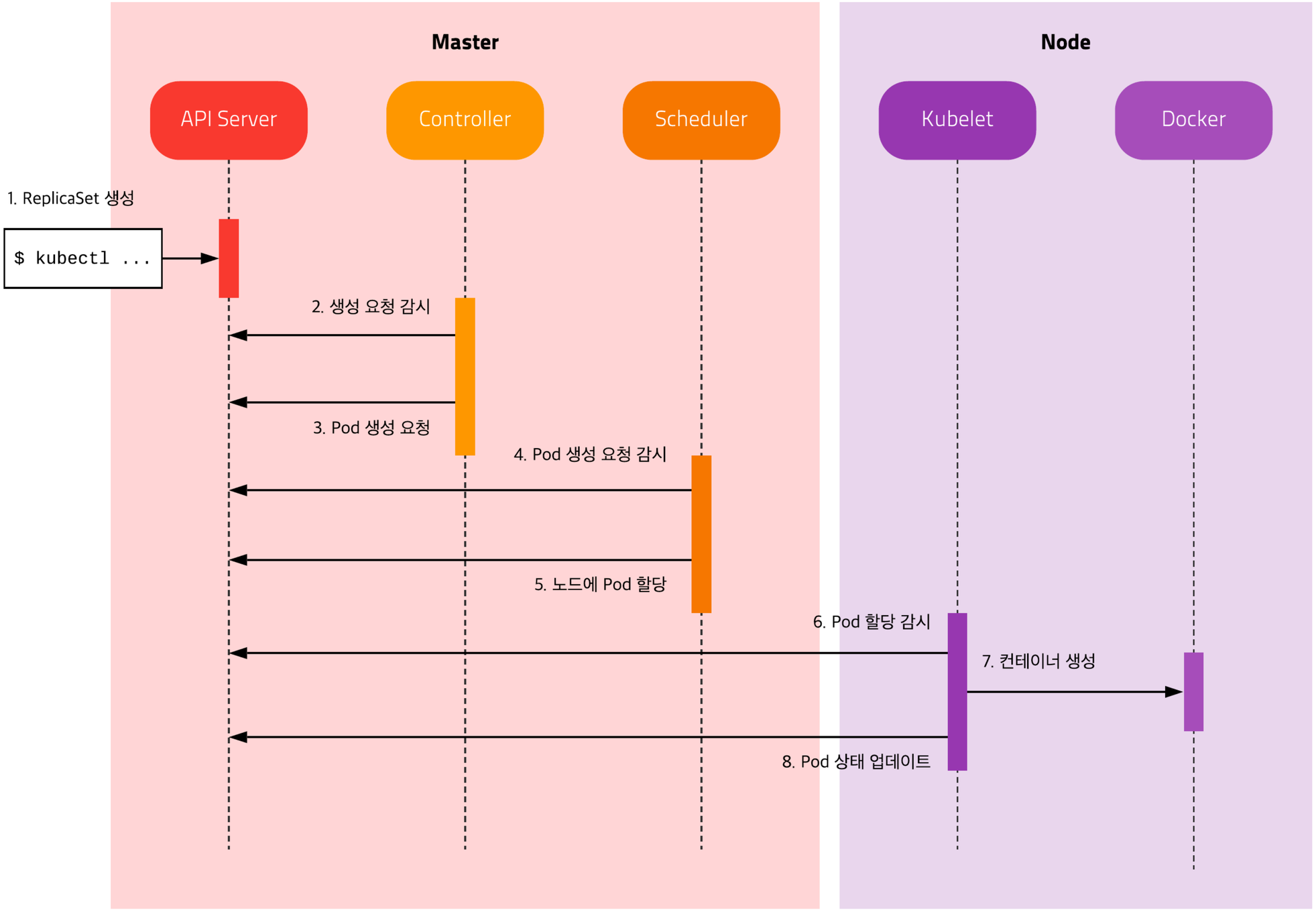
API 호출하기

원하는 상태(desired state)를 다양한 오브젝트(object)로 **정의(spec)**하고 API 서버에 yaml형식으로 전달



API 호출하기

원하는 상태(desired state)를 다양한 오브젝트(object)로 정의(spec)하고 **API 서버에 yaml형식으로 전달**


Master
Node
API Server
Controller
Scheduler
Kubelet
Docker
1. ReplicaSet 생성
$ kubectl ...
2. 생성 요청 감시
3. Pod 생성 요청
4. Pod 생성 요청 감시
5. 노드에 Pod 할당
6. Pod 할당 감시
7. 컨테이너 생성
8. Pod 상태 업데이트

잠시 생각해보자

Thanks!