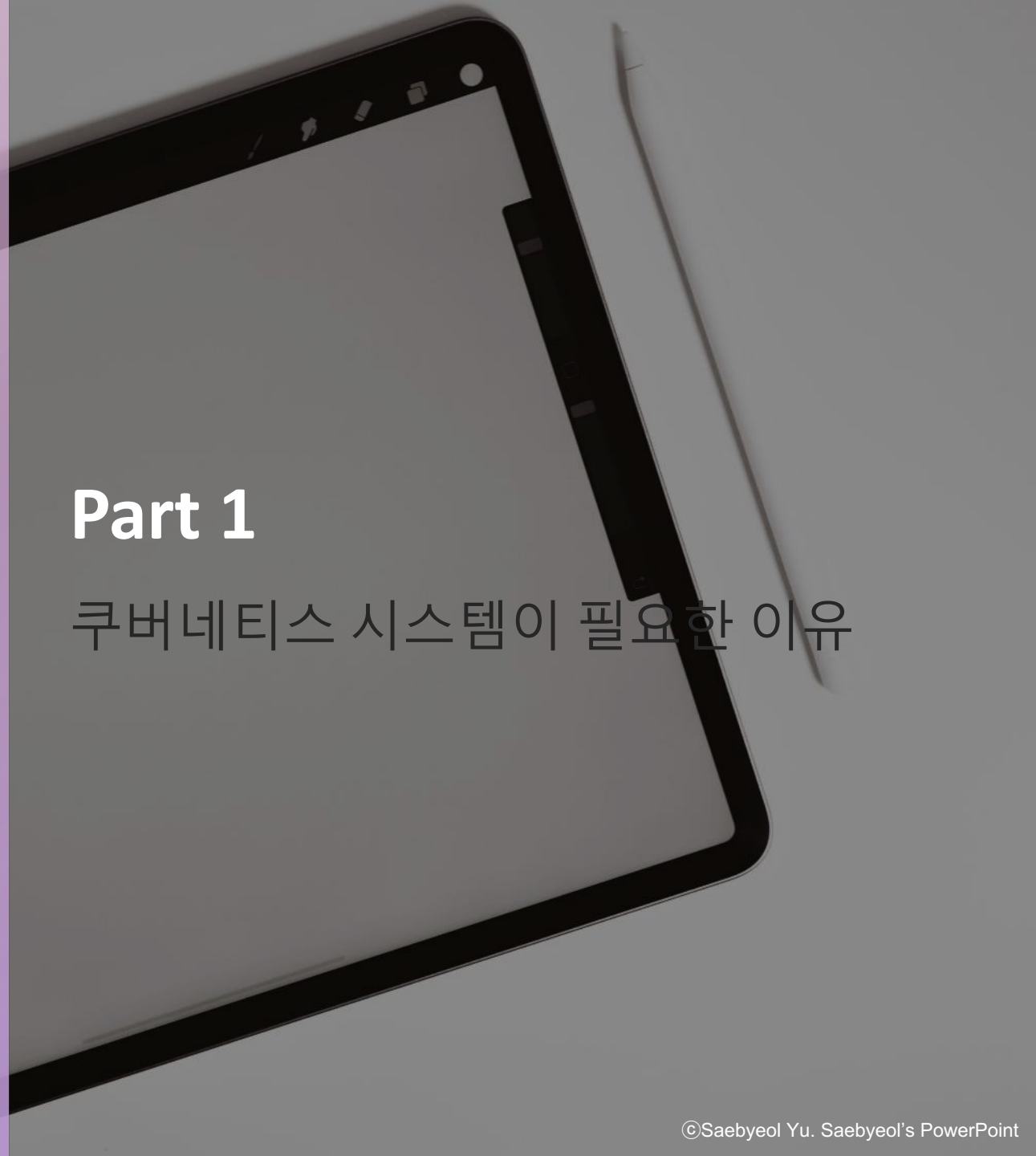
쿠버네티스 소개

김수민

목차

table of contents

2022-2학기
ooooooo 발표



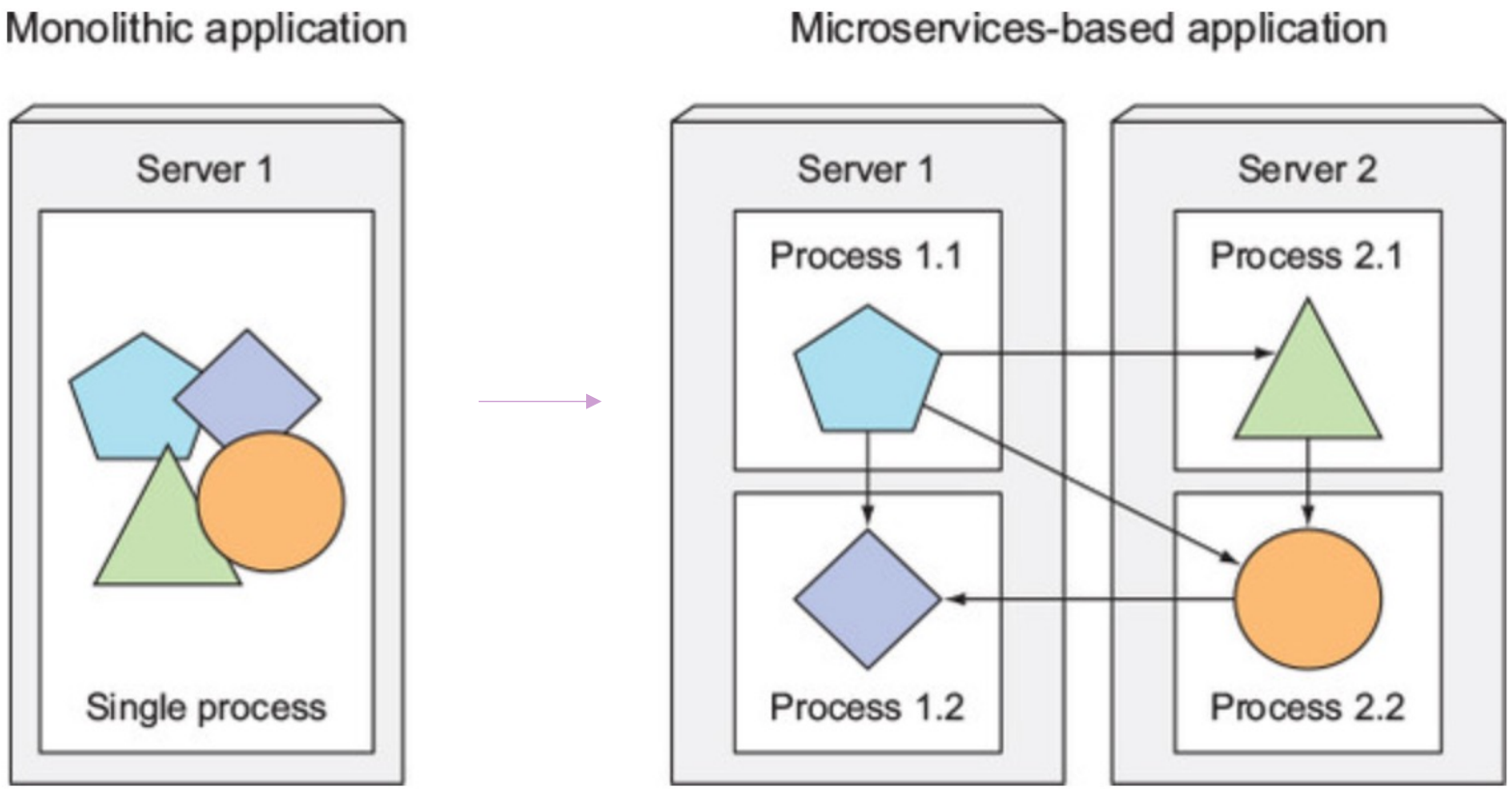
Part 1

쿠버네티스 시스템이 필요한 이유

Part 1

현대 애플리케이션 구조

과거 현대

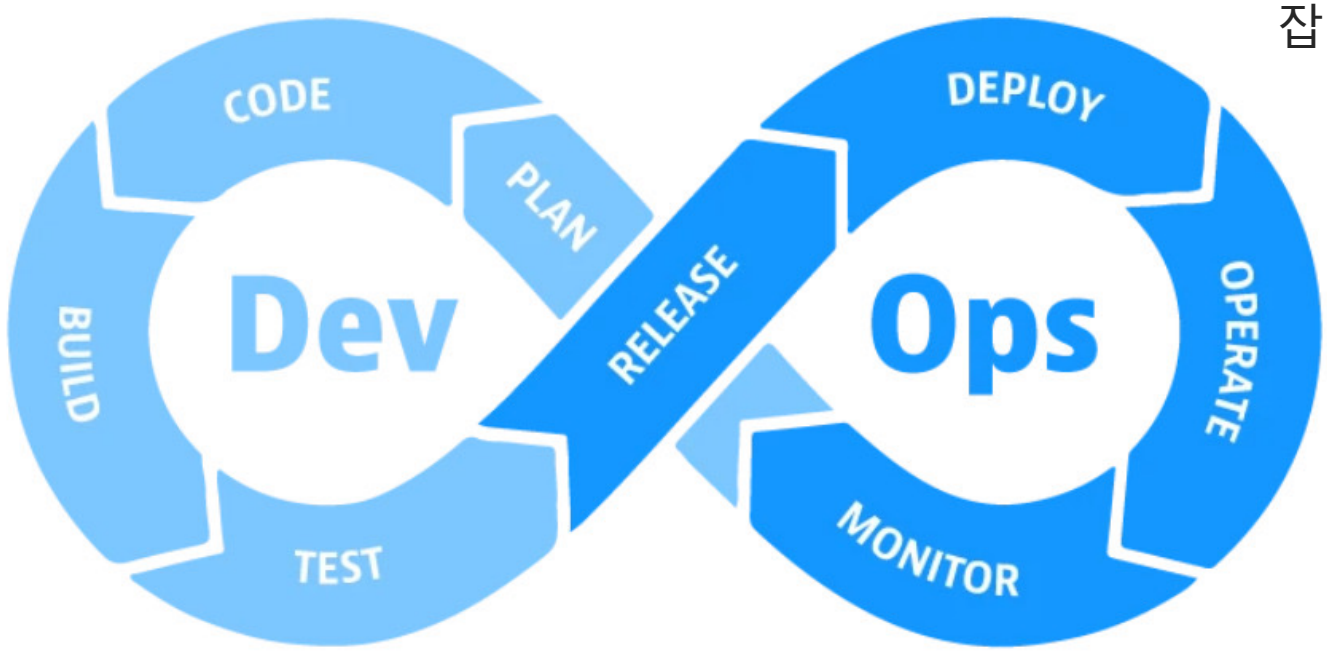


물론 모놀리식 구조도 있지만 현대 애플리케이션은
마이크로서비스 구조로 바뀌고 있는 추세

Part 1

DevOps 개념의 등장

대기업에선 협업자..
스타트업이나 중소기업에서는
잡부(개발 운영 다하는)..



매번 개발을 하고 운영팀에게 넘겨줘서 운영하도록 하는 과정이 효율적이지 못함 ->
개발도 할 수 있으면서 운영도 할 수 있는 사람...? -> DevOps

Part 2

컨테이너 기술 소개

Part 2

컨테이너란?

동일한 host 시스템 내에서 여러 개의 서비스를 동시에 서로 다른 환경에서 실행할 수 있도록 만들어주는 기술.

하나의 os 커널을 공유

리눅스의 namespace 기술 이용

리눅스의 cgroups 기술 이용

Part 2

컨테이너란?

Name space

•NS namespace - 파일 시스템의 마운트 지점을 분할 격리

•PID namespace - process ID를 분할 관리

•NET namespace - 네트워크 인터페이스, iptables 등의 네트워크 리소스와 관련된 정보를 분할

•IPC namespace - IPC리소스를 분할 격리

•UTS namespace - 호스트와 도메인 네임 별로 격리

• USER namespace - user와 group id를 분할하고 격리

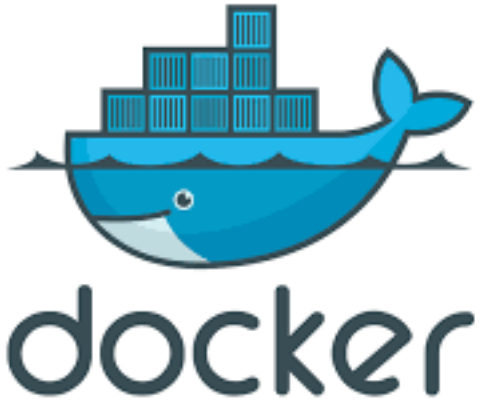
동일한 시스템에서 별개의 독립된 공간을 격리된 환경에서 운영하는 가상화 기술 -> 이름을 붙여줘서 분리하는 기술로 특정 리소스 그룹을 격리하는 데 사용됨.

cgroups

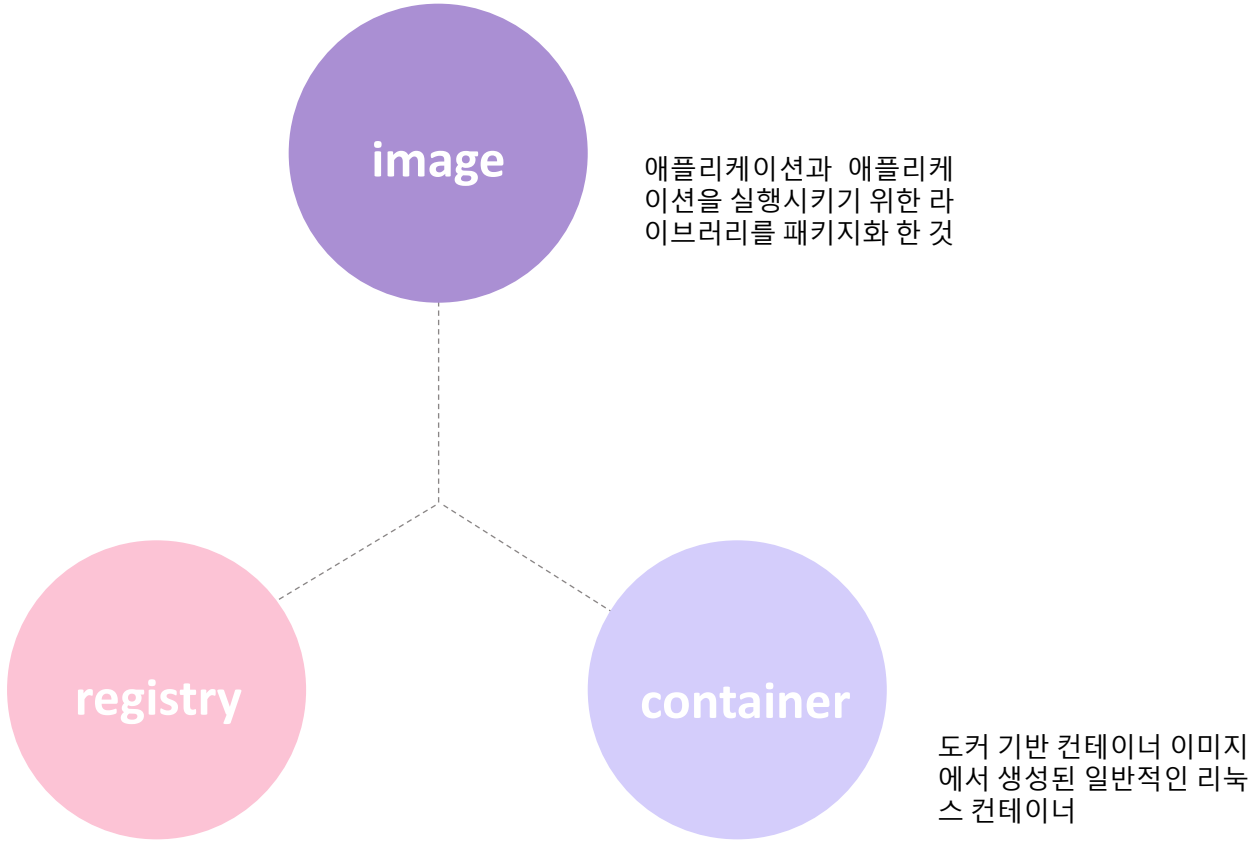
프로세스의 리소스 사용을 제한하는 기술

Part 2

Docker 소개

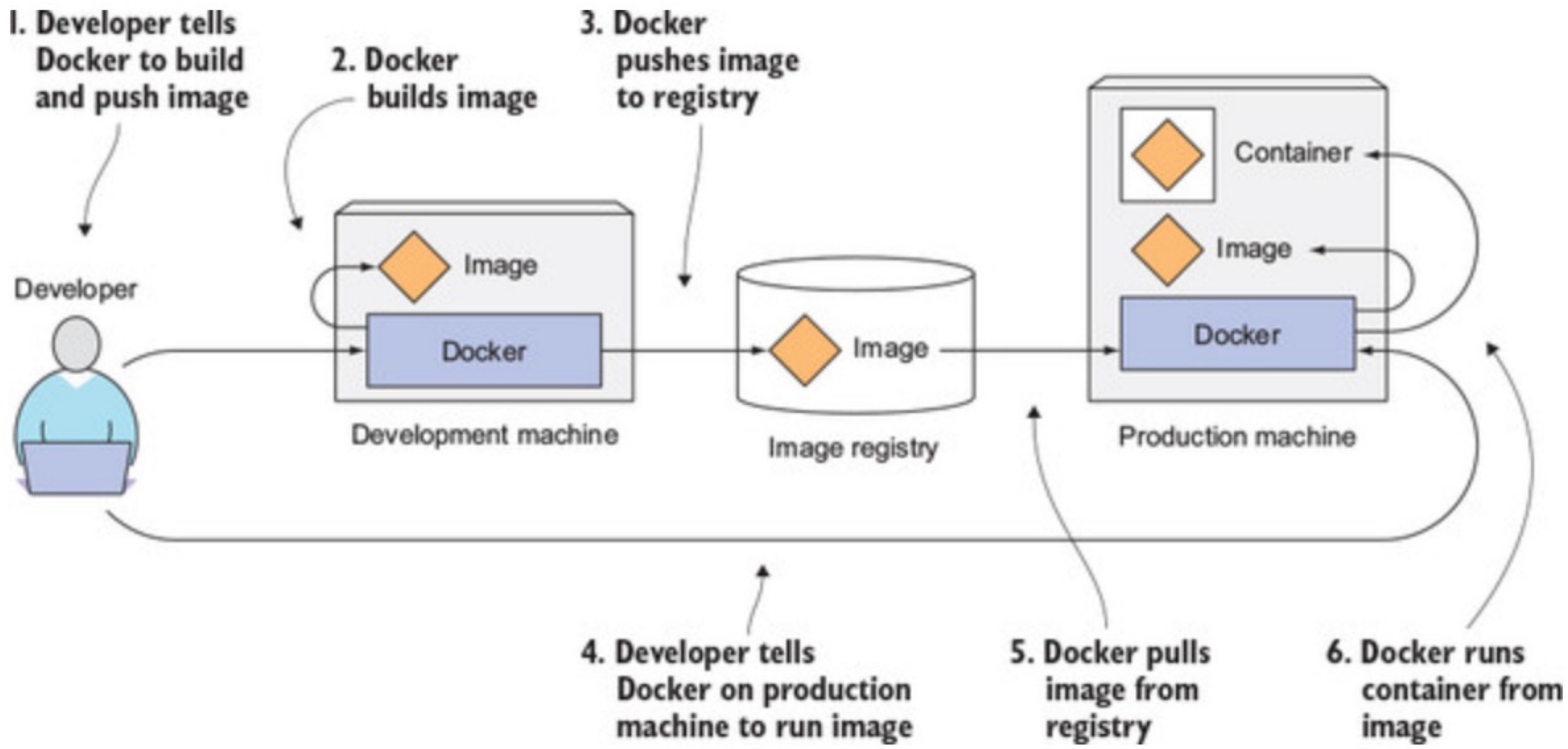


컨테이너를 이미지화해서 간편한 이식 가능한 패키지로 패키징하는 컨테이너 플랫폼

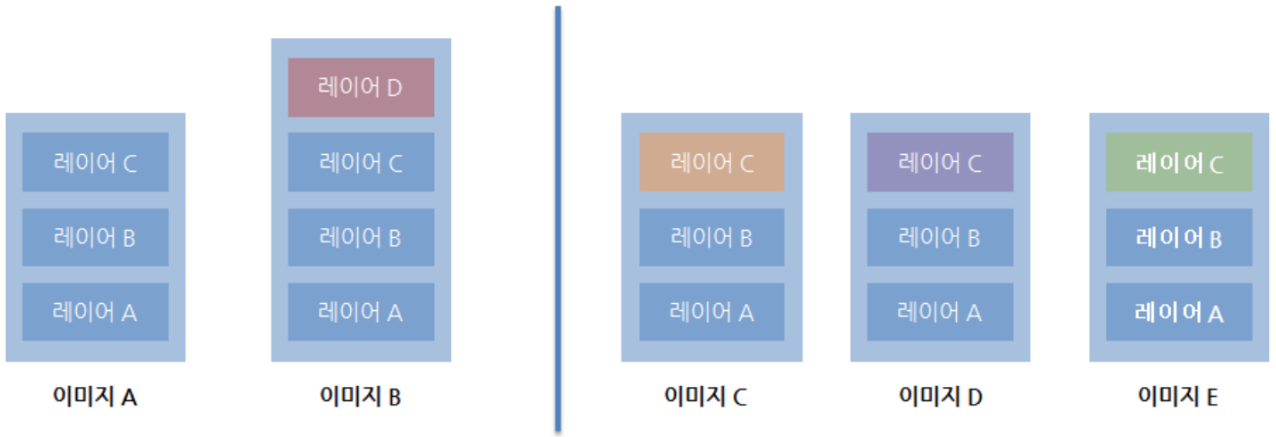


Part 2

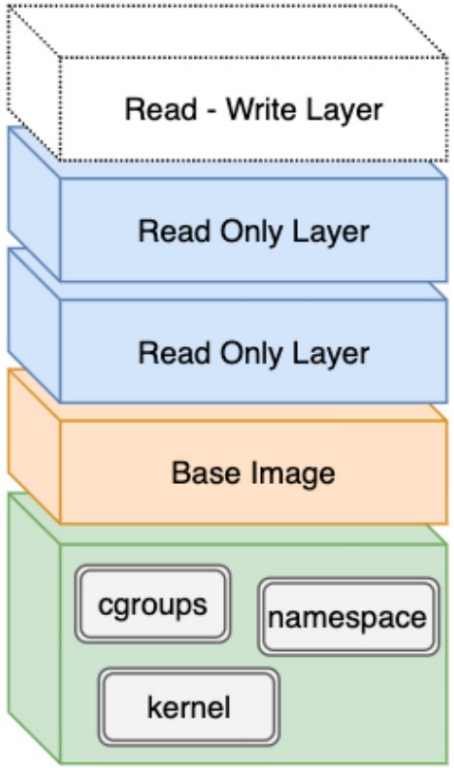
Docker 소개



Part 2 Docker 소개



이미지는 여러 개의 레이어로 구성 ->
같은 레이어가 이미 로컬에 있으면 저
장 시 다른 레이어만 가져옴.



컨테이너가 실행될 때 읽기 전용 위에 새로운
쓰기 가능한 레이어가 만들어짐.

Container Runtimes

> **Note:** Dockershim has been removed from the Kubernetes project as of release 1.24. Read the Dockershim Removal FAQ for further details.

You need to install a container runtime into each node in the cluster so that Pods can run there. This page outlines what is involved and describes related tasks for setting up nodes.

Kubernetes 1.26 requires that you use a runtime that conforms with the Container Runtime Interface (CRI).

See CRI version support for more information.

This page provides an outline of how to use several common container runtimes with Kubernetes.

- containerd
- CRI-O
- Docker Engine
- Mirantis Container Runtime

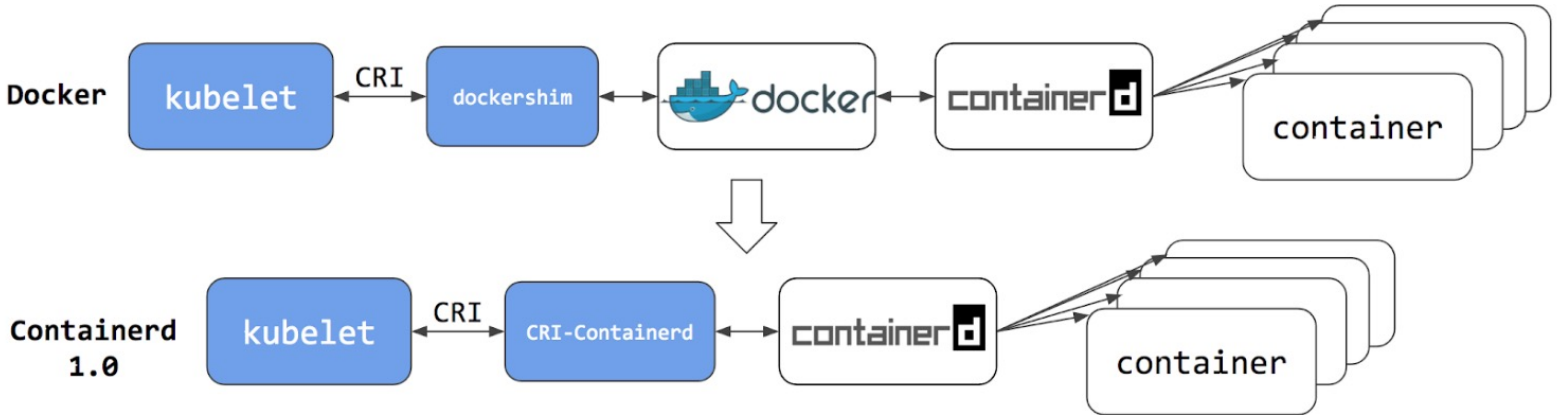
OCI(open container initiatice) – 컨테이너 런타임에 대한 표준으로 컨테이너를 실행하기 위한 저수준 런타임

CRI(container runtime interface) – 쿠버네티스 측에서 제공하는 컨테이너 런타임 추상화 계층

CRI-O - CRI+OCI에서 유래되었고 컨테이너 실행을 목적으로 도커보다 경량화되어 있고 컨테이너 생성인 run과 이미지 빌드를 제공하지 않음.(이미지 빌드 툴 필요)

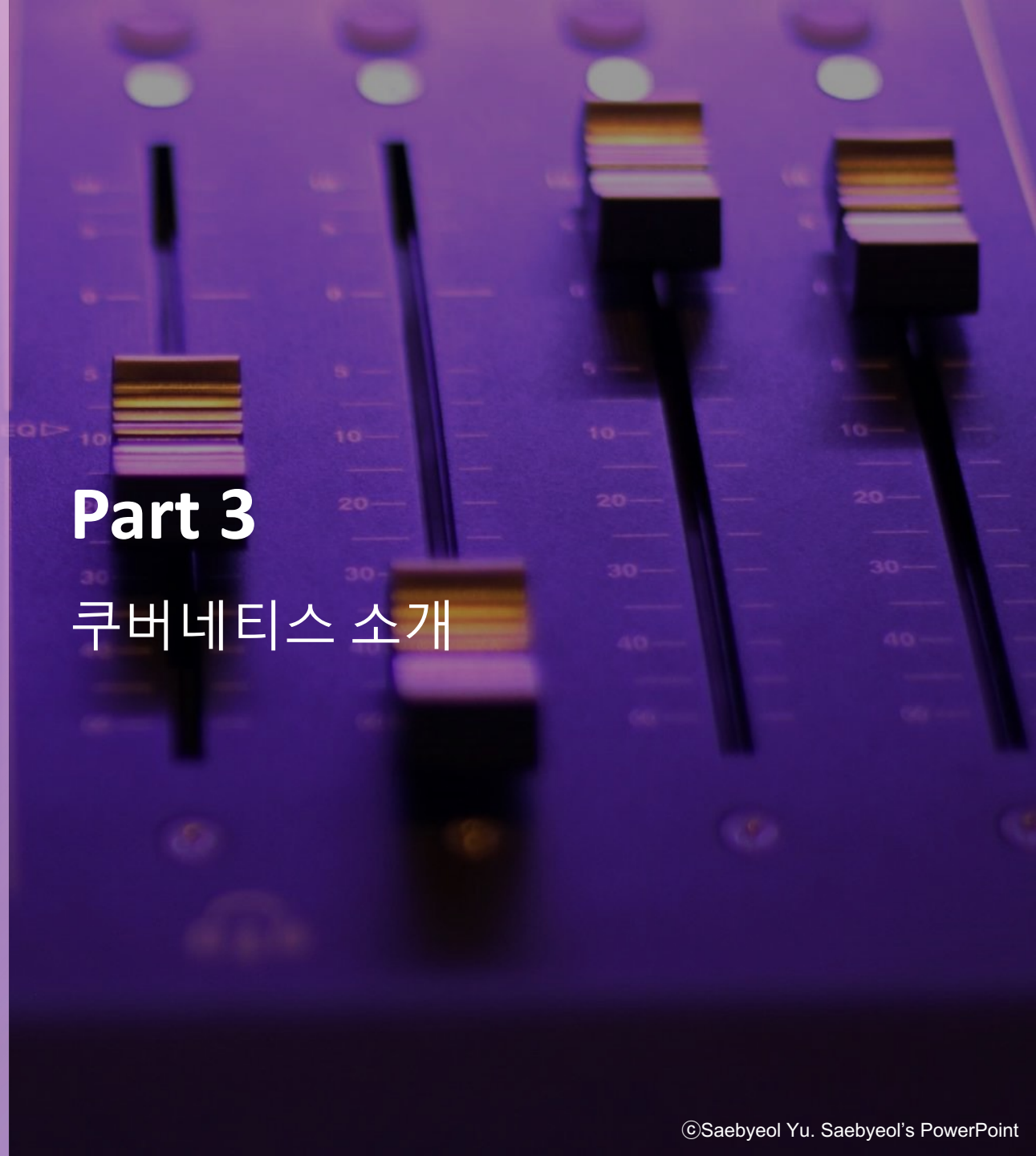
Part 2

Docker 소개



쿠버네티스는 컨테이너 런타임과 통신할 때 CRI라는 표준 인터페이스 API를 사용 ->
도커는 CRI 지원되지 않기 때문에 둘을 잇는 도구로 Dockershim이 사용 ->
갈수록 배포 속도도 느리고 유지 관리도 어려운 문제 ->
컨테이너 런타임으로 Containerd가 새롭게 채택

2022-2학기
ooooooo 발표



Part 3

쿠버네티스 소개

Part 3 kubernetes 소개

Large-scale cluster management at Google with Borg

Abhishek Verma[†] Luis Pedrosa[‡] Madhukar Korupolu
David Oppenheimer Eric Tune John Wilkes
Google Inc.

Abstract

Google's Borg system is a cluster manager that runs hundreds of thousands of jobs, from many thousands of different applications, across a number of clusters each with up to tens of thousands of machines.

It achieves high utilization by combining admission control, efficient task-packing, over-commitment, and machine sharing with process-level performance isolation. It supports high-availability applications with runtime features that minimize fault-recovery time, and scheduling policies that reduce the probability of correlated failures. Borg simplifies life for its users by offering a declarative job specification language, name service integration, real-time job monitoring, and tools to analyze and simulate system behavior.

We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

1. Introduction

The cluster management system we internally call Borg ad-

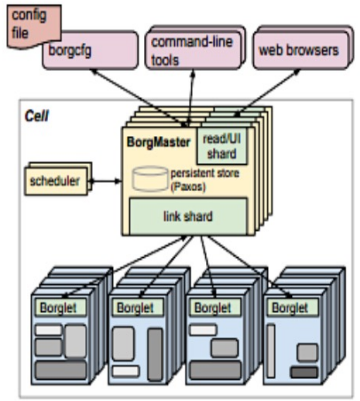


Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

cluding with a set of qualitative observations we have made from operating Borg in production for more than a decade.

2. The user perspective

Omega: flexible, scalable schedulers for large compute clusters

Malte Schwarzkopf[†*] Andy Konwinski[‡*] Michael Abd-El-Malek[§] John Wilkes[§]
[†]University of Cambridge Computer Laboratory [‡]University of California, Berkeley [§]Google, Inc.
[†]ms705@cl.cam.ac.uk [‡]andyk@berkeley.edu [§]{mabdelmalek,johnwilkes}@google.com

Abstract

Increasing scale and the need for rapid response to changing requirements are hard to meet with current monolithic cluster scheduler architectures. This restricts the rate at which new features can be deployed, decreases efficiency and utilization, and will eventually limit cluster growth. We present a novel approach to address these needs using parallelism, shared state, and lock-free optimistic concurrency control.

We compare this approach to existing cluster scheduler designs, evaluate how much interference between schedulers occurs and how much it matters in practice, present some techniques to alleviate it, and finally discuss a use case highlighting the advantages of our approach – all driven by real-life Google production workloads.

Categories and Subject Descriptors D.4.7 [*Operating Systems*]: Organization and Design—Distributed systems; K.6.4 [*Management of computing and information systems*]: System Management—Centralization/decentralization

Keywords Cluster scheduling, optimistic concurrency control

1. Introduction

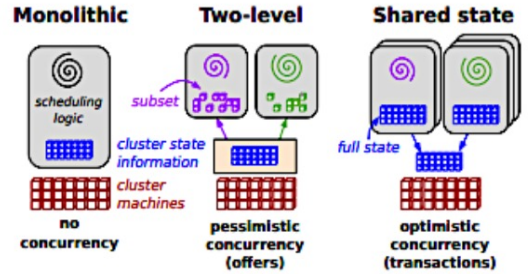


Figure 1: Schematic overview of the scheduling architectures explored in this paper.

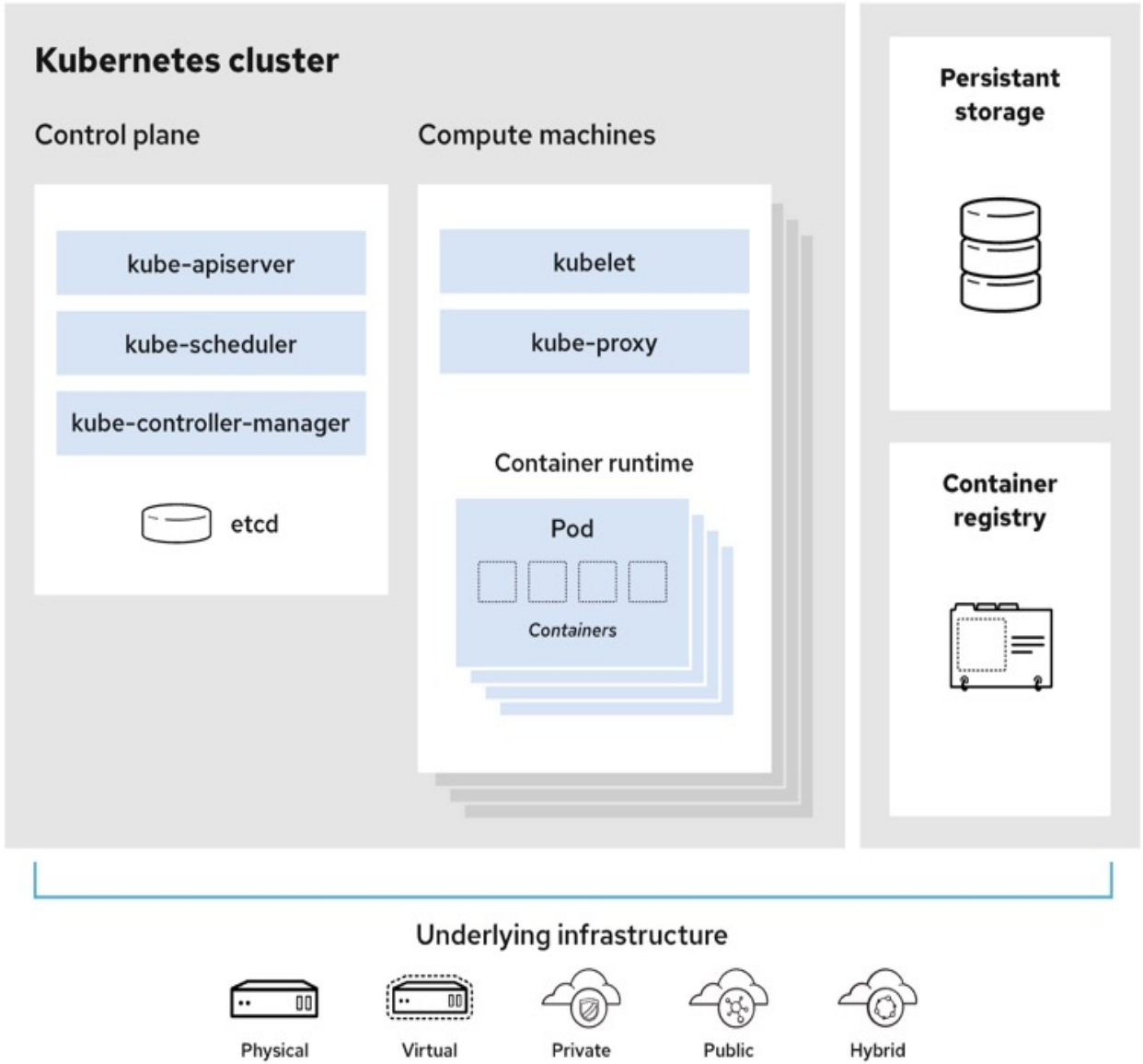
and policies have to be taken into account. Meanwhile, clusters and their workloads keep growing, and since the scheduler's workload is roughly proportional to the cluster size, the scheduler is at risk of becoming a scalability bottleneck.

Google's production job scheduler has experienced all of this. Over the years, it has evolved into a complicated, sophisticated system that is hard to change. As part of a rewrite of this scheduler, we searched for a better approach.

We identified the two prevalent scheduler architectures

구글은 Borg와 Omega의 경험을 바탕으로 쿠버네티스를 개발

Part 3 kubernetes 소개

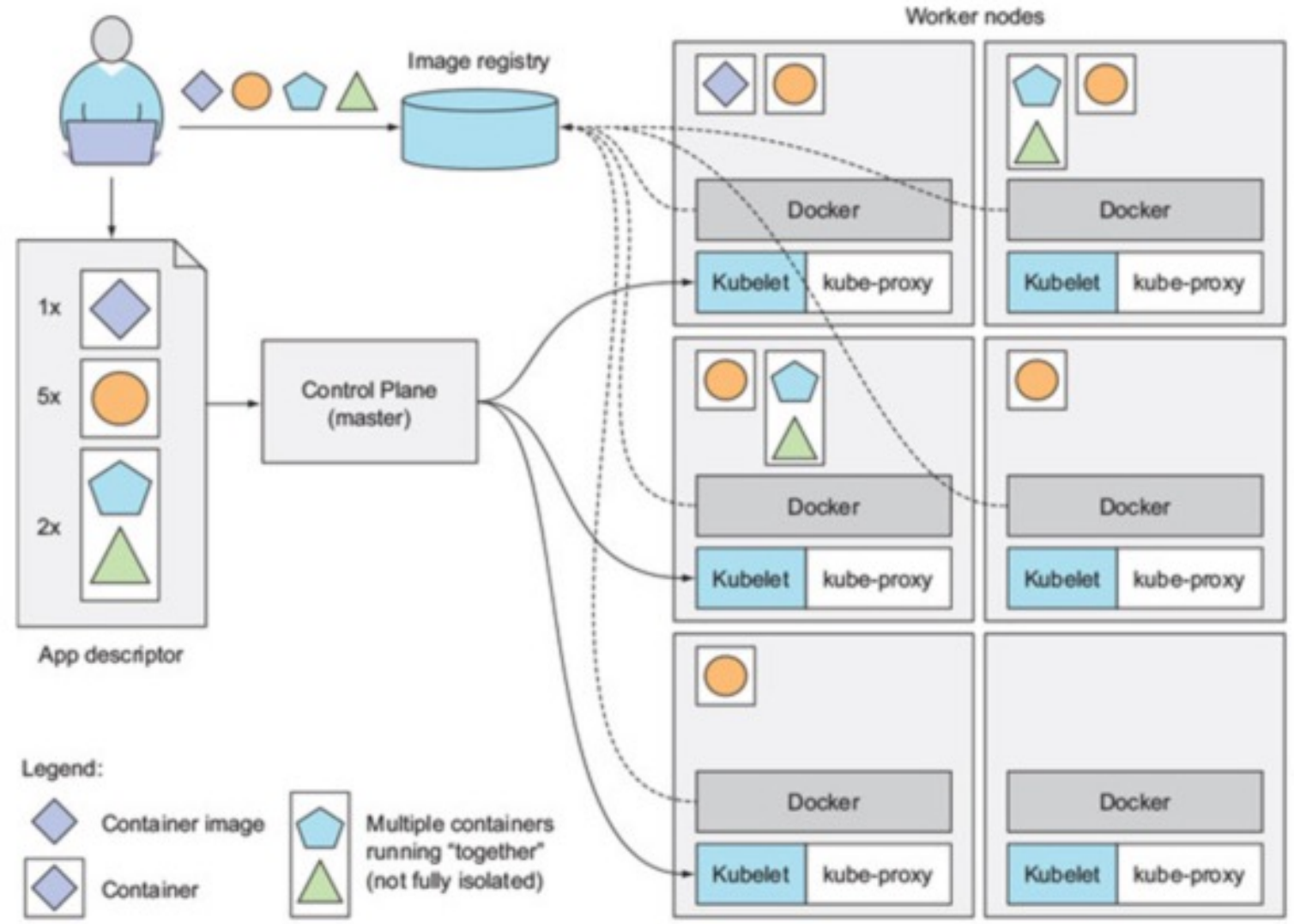


Control plane은 api서버(사용자, 컨트롤 플레인 구성 요소와 통신하는 역할)와 스케쥴러(애플리케이션의 배포를 담당), 컨트롤러 매니저(구성 요소 복제본, 워커 노드 추적, 노드 장애 처리 등과 같은 클러스터단의 기능 수행), etcd(클러스터 구성을 지속적으로 저장하는 신뢰할 수 있는 분산 데이터 저장소)로 구성

노드는 컨테이너 런타임(컨테이너를 실행하는), kubelet(api 서버와 통신하고 노드의 컨테이너를 관리하는 역할), kube-proxy (애플리케이션 구성 요소 간에 네트워크 트래픽을 로드밸런싱하는 역할)로 구성

Part 3

kubernetes 소개



2022-2학기
ooooooo 발표

Part 4

기타

Part 4

실제 적용 사례

Results for move kubernetes

Stories | People | Publications | Topics

Tinder in Tinder Tech Blog · Apr 17, 2019

Tinder's move to Kubernetes

Written By: Chris O'Brien, Engineering Manager |Chris Thomas, Engineering Manager| Jinyong Lee, Senior Software Engineer | Edited By: Cooper Jackson, Software Engineer Why Almost two years ago,...

Terraform 11 min read

Zenhub · Aug 21, 2020

ZenHub's move to Kubernetes

ZenHub recently migrated its entire production and CI/CD infrastructure to Kubernetes. This blog post documents the history of containers at ZenHub, why we eventually went with Kubernetes, how...

Software Development 6 min read

Alibaba Cloud in DataDrivenInvestor · Aug 13, 2019

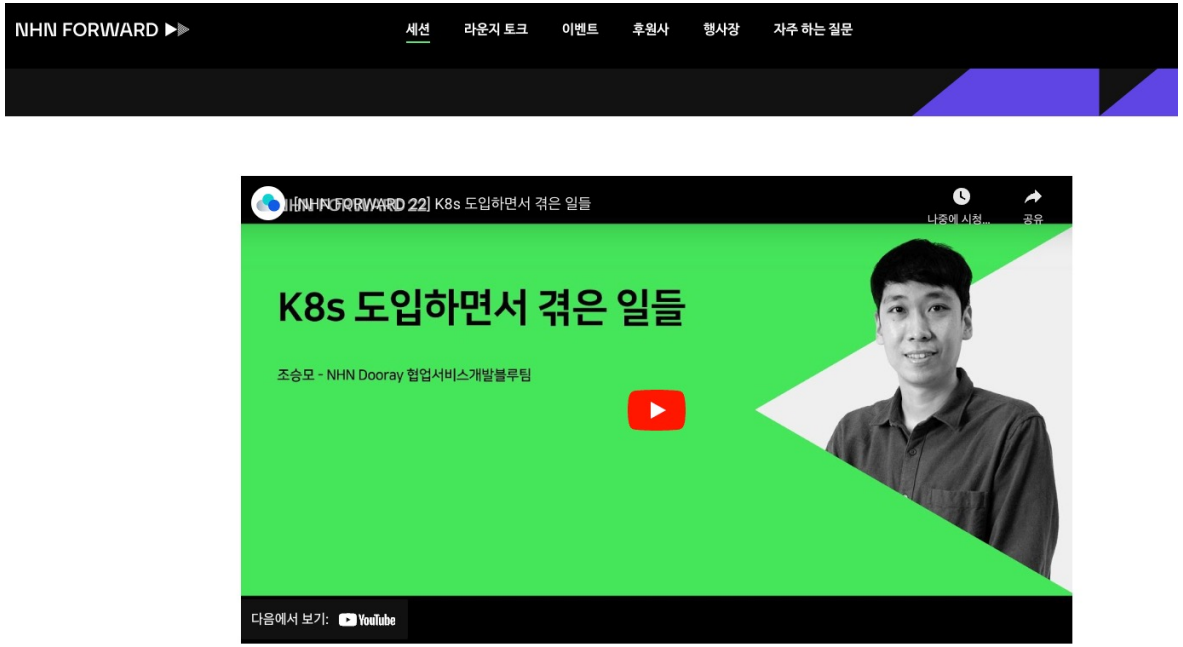
What Can We Learn from Twitter's Move to Kubernetes

This blog outlines how Twitter's switch from Mesos to Kubernetes. Kubernetes is the industry-wide standard when it comes to containers.

https://medium.com/tinder/tinders-move-to-kubernetes-cda2a6372f44

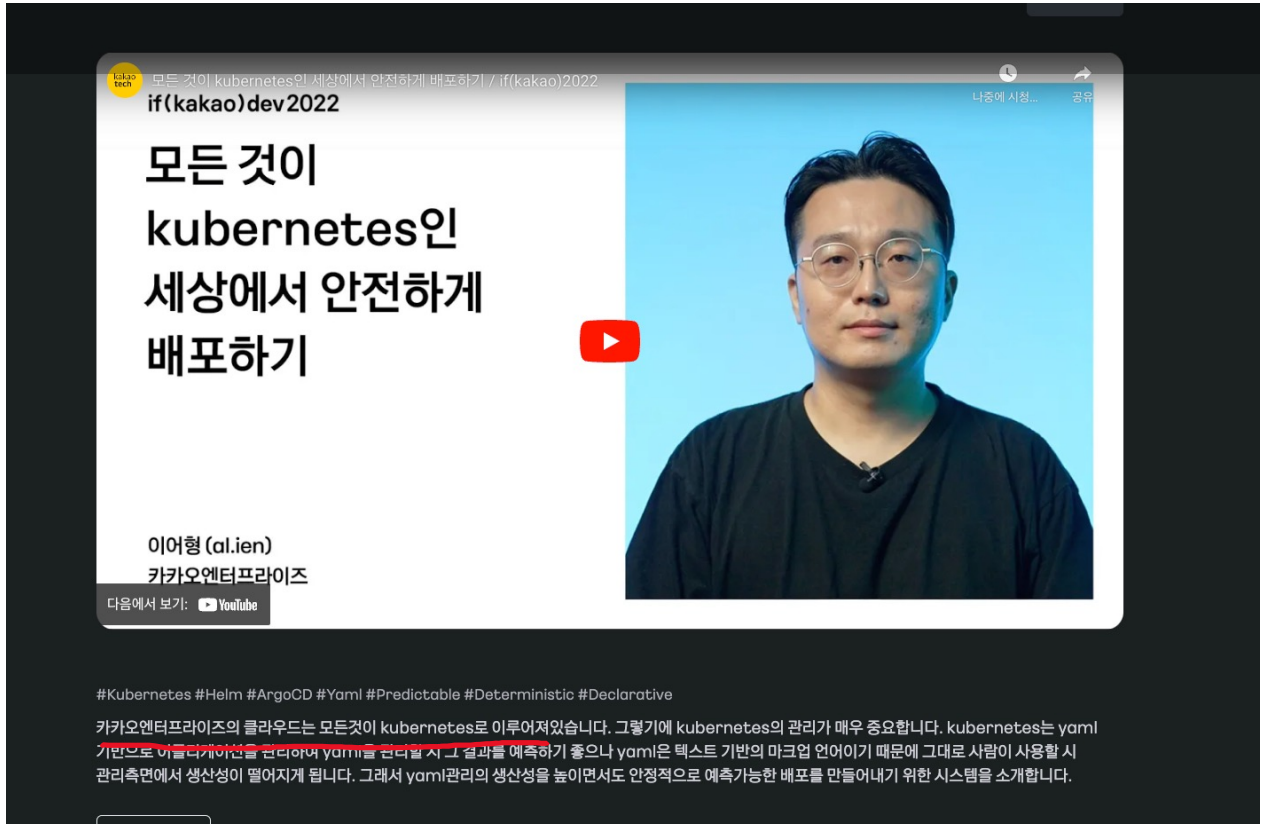
Part 4

실제 적용 사례



15:00~15:40 | 트랙 7 | 초급

K8s 도입하면서 겪은 일들



https://if.kakao.com/2022/session/59
https://forward.nhn.com/2022/sessions/29